Technische Universität München
Fakultät für Informatik
Lehrstuhl für Wirtschaftsinformatik (I 17)
Univ.-Prof. Dr. Helmut Krcmar

Performance-Modellierung und Simulation eines SAP-ERP-Systems

Stephan Gradl

Vollständiger Abdruck der von der Fakultät für Informatik der Technischen Universität München zur Erlangung des akademischen Grades eines

Doktors der Naturwissenschaften (Dr. rer. nat.)

genehmigten Dissertation.

Vorsitzender: Univ.-Prof. Dr. Hans Michael Gerndt

Prüfer der Dissertation:

1. Univ.-Prof. Dr. Helmut Krcmar
2. Univ.-Prof. Dr. Martin Bichler

Die Dissertation wurde am 06.06.2012 bei der Technischen Universität München eingereicht und durch die Fakultät für Informatik am 13.08.2012 angenommen.

Zusammenfassung

Ziel: Ziel dieser Arbeit ist die Simulation der technischen Performance eines SAP-Enterprise-Resource-Planning-Systems (ERP-Systems) unter Verwendung einer steigenden Anzahl von parallelen Benutzern. Aufbauend auf der aktuellen Literatur zu Einflussfaktoren auf die Performance von ERP-Systemen wird ein „Layered Queueing Network"-Modell (LQN-Modell) und dessen Parametrisierung für eine diskrete Ereignissimulation erstellt. Das Modell soll die bestehenden Modellierungs-Ansätze um Tabellenpufferung und wechselseitigen Zugriff auf geschützte Systemressourcen innerhalb eines SAP-ERP-Systems erweitern. Fokus der Betrachtung liegt dabei auf dem SAP „Web Application Server" ABAP (WebAS-ABAP).

Methode: Die Arbeit basiert auf einem gestaltungsorientierten Ansatz nach Hevner et al. Ausgehend von den komplexitätsreduzierenden Grundannahmen bestehender Modelle werden systeminhärente Einflussfaktoren auf die Performance des Systems analysiert. Das Modell der Systemkomponenten sowie dessen Parametrisierung stellt das Artefakt dieser Arbeit dar. Das auf Mess- und Warteschlangentheorie basierende Modell wird durch einen Vergleich der Simulationsergebnisse mit gemessenen Performance-Werten eines real existierenden Systems im Labor evaluiert. Dieser Vergleich fokussiert verschiedene Szenarien unterschiedlicher (technischer) Komponenten des SAP-ERP-Systems, um deren Modellierung zu evaluieren.

Resultate: Aus dem erstellten Artefakt und der Umsetzung innerhalb einer Fallstudie lassen sich folgende Ergebnisse ableiten: Durch die Erweiterung bestehender Ansätze konnte ein Modell erstellt werden, das durch die Erweiterung der Modellierung um Puffer- und Sperrverwaltungsmechanismen genauere Vorhersagen der Antwortzeiten unter einer steigenden Anzahl von parallelen Benutzern ermöglicht. Für die Parametrisierung des Modells wurden Mess- und Analysewerkzeuge erstellt. Die Evaluation der Ergebnisse zeigt, dass dies besonders für den Niedrig- und Mittellastbereich relevant ist. Abweichende Simulationsergebnisse für den Hochlastbereich konnten auf externe, derzeit nicht modellierte Faktoren, zurückgeführt werden. Lastbereiche mit unvorhersehbarem, nicht deterministischem System-Verhalten können durch die Simulation des Modells prognostiziert und damit abgegrenzt werden. Als zusätzliches Ergebnis der Systemanalyse zur Erstellung des Simulationsmodells konnten die benötigten Pufferressourcen bestimmt und ABAP-Programme hinsichtlich ihrer Invalidierung(en) gepufferter Inhalte analysiert werden.

Auswirkungen auf die Praxis: Auswirkungen veränderter Workloads in Folge neu entwickelter Programme oder einer steigenden Anzahl paralleler Anfragen ist in der Praxis ein häufig auftretendes Problem. Das in der vorliegenden Arbeit entwickelte Modell stellt einen neuartigen, methodischen Lösungsansatz für die Performance-Analyse in diesen konkreten Fällen bereit. Zusätzlich zu der Vorhersage der Antwortzeiten können die Konfiguration des Systems vor dessen operativen Inbetriebnahme überprüft und Performance-Engpässe in einzelnen Komponenten identifiziert werden. Neben der Unterstützung eines störungsfreien Systembetriebs, besteht ein zusätzlicher Mehrwert für eine a priori Investitions- und Einsatzplanung benötigter Hardware-Ressourcen. Aufgrund der starken Verbreitung von SAP-ERP-Systemen in der Praxis und deren starker Integration in die Geschäftsprozesse ist dieses Artefakt besonders relevant.

Inhaltsverzeichnis

Abbildungsverzeichnis

Tabellenverzeichnis

Abkürzungsverzeichnis

ABAP	Advanced Business Application Programming
DDIC	SAP-Data-Dictionary
DES	Discrete Event Simulation
DIAG	Dynamic Information and Action Gateway
ERP	Enterprise-Resource-Planning
ESM	Export/Import Shared Memory
FB	Finanzbuchhaltung
FCFS	First-Come-First-Serve
FTAB	Field-Description-Puffer
HOL	Head-of-line priority
IREC	Initial-Record-Layouts-Puffer
ISR	Information Systems Research
LCLS	Last-Come-Last-Serve
LO	Logistik Allgemein
LQN	Layered Queueing Networks
lqns	Layered Queueing Network Solver
lqsim	Layered Queueing Network Simulator
LRU	Last-Recently-Used-Verfahren
LUW	Logical Unit of Work
MM	Materialwirtschaft
NFS	Network File Systems
NTAB	Nametab-Puffer
OTR	Online Text Repository
PEER	Performance Evaluation Cockpit for ERP-Systems
PP	Produktionsplanung- und Steuerung
PPR	priority, preemptive, resum
PS	Processor Sharing
PS	Projektabwicklung
PXA	Program Execution Area
QPN	Queueing Petri Nets
RAM	Random-Access-Memory
RAND	Random scheduling

RFC	remote function calls
RR	Round-Robin
SAPS	SAP-Application-Performance-Standard
SAP-WebAS-ABAP	SAP-„Web Application Server“-ABAP
SAP-WebAS-JAVA	SAP-„Web Application Server“-JAVA
SD	Vertriebssystem
SIRO	Service-In-Random-Order
SLA	Service Level Agreements
SNTAB	Short-Nametab-Puffers
SPEX	Software Performance Experimenter
SPN	Stochastic Petri Nets
SRVN	Stochastische Rendezvous Netzwerke
TABL	Generic Key Table Buffer
TABLP	Single Record Buffer
TCO	Total Cost of Ownership
TPC	Transaction Processing Performance Council
TPN	Timed Petri Nets
TTAB	Table-Definitions-Puffer
WI	Wirtschaftsinformatik

1. Einleitung

ERP-Systeme unterstützen die Verwaltung der Ressourcen eines Unternehmens. Eine große Anzahl geschäftskritischer Prozesse sind durch ERP-Systeme unterstützt bzw. in diesen Systemen implementiert. Deshalb sind die Verfügbarkeit und die Performance dieser Systeme äußerst wichtig für den Erfolg eines Unternehmens. Mit dem Wachstum und der Weiterentwicklung eines Unternehmens ändern sich die Anforderungen an das jeweilige Informationssystem. Durch Integration neuer Geschäftsbereiche oder einer Erhöhung der Mitarbeiterzahlen ändern sich oftmals die Anfragen, die an ein ERP-System gestellt werden, in Anzahl und Art. Für den wirtschaftlichen Erfolg eines Unternehmens wird es immer wichtiger, schnell auf neue Anforderungen reagieren zu können. Für den Betrieb von ERP-Systemen resultiert diese Entwicklung in kleiner werdenden Implementierungszyklen. Damit steht nach der Implementierung entsprechend weniger Zeit für die Erstellung und Evaluation einer dem neuen Workload angemessenen Konfiguration bereit. Da herkömmliche Methoden oft mit einer sehr langen Vorbereitungszeit einhergehen, besteht der Bedarf an einem strukturierten Vorgehen für die Performance-Analyse eines SAP-ERP-Systems unter einer wachsenden Anzahl von Benutzern, welche definierte Geschäftsprozesse in einem System durchführen.

1.1. Problemstellung und Motivation

Durch eine wachsende Anzahl von Benutzer- und/oder Geschäftsprozessen steigen auch die Anforderungen an die Leistung eines ERP-Systems. Bei Änderungen der Funktionalitäten eines Systems besteht, bei nicht ausreichenden Informationen über deren Auswirkungen, das Risiko einer nicht ausreichenden Performance oder gar der Nicht-Verfügbarkeit des ganzen Systems. Informationen über die Leistung eines Systems werden von Überwachungswerkzeugen gesammelt. Die Leistungsfähigkeit wird dabei jedoch häufig von Administratoren und Anwendungsbetreuern abgeschätzt und letztendlich vom Endbenutzer getestet. Um den maximalen Workload zu bestimmen, den ein System verarbeiten kann, wird häufig die Last schrittweise erhöht, während das System überwacht wird. Da Lasttests Daten im System erzeugen, muss das produktive System dafür nachgebildet werden. Dies ist jedoch sehr kostspielig. Wird das System nicht auf das Verhalten unter einer hohen Last überprüft, werden Leistungsprobleme oder Systemausfälle riskiert. Um derartige Probleme zu verhindern, ist es nötig, ein Verfahren zu entwickeln, um mögliche Probleme und Nebeneffekte vor der Implementierung neuer Funktionalitäten zu analysieren.

Da SAP der Marktführer in der Klasse der ERP-Systeme ist, fokussiert sich die vorliegende Arbeit auf SAP-ERP-Systeme, welche auf den klassischen ERP-Funktionalitäten basieren. Nach Jain (1991) existieren verschiedene klassische Ansätze für die Kapazitäts-Planung und Performance-Evaluation von Computer-Systemen. Dazu zählen Messung (Benchmarking, Lasttests), Simulation und analytische Modellierung. Ändert sich ein System, so müssen beiden Formen der Messung angepasst werden. Aufgrund der Fülle möglicher Konfigurations-Kombinationen ist dies zusätzlich äußerst kostspielig. Eine Alternative stellt die Simulation der Performance nach möglichen Änderungen am System dar. Obwohl das Erstellen und Validieren eines Simulationsmodells ebenfalls sehr zeitaufwendig sind, so können die identifizierten und modellierten Komponenten bei Systemänderungen häufig wieder verwendet werden (Xu et al. 2005). Es ist das Ziel der vorliegenden Arbeit, ein Model

für die Performance-Vorhersage eines SAP-ERP-Systems unter einer steigenden Anzahl von parallelen Benutzern zu erstellen und zu validieren.

SAP-Systeme haben eine interne Architektur, auf der jedes Programm arbeitet. „Advanced Business Application Programming“, kurz ABAP (SAP 2012a), ist die Programmiersprache der SAP und basiert auf der bereits angesprochenen internen Architektur. Daher ist es nötig, ein Modell dieser Architektur zu entwickeln. Dieses kann als Vorlage für die Modellierung und die Vorhersage der technischen Performance von Geschäftsprozessen auf einem SAP-ERP-System verwendet werden.

1.2. Ziele

Wie bereits in den vorhergegangenen Abschnitten dargestellt, ist es das Ziel der Arbeit, ein Modell für die Vorhersage der Performance eines SAP-ERP-Systems zu erstellen und ein Konzept für dessen Parametrisierung zu erarbeiten. Dabei stellen sich die folgenden Anforderungen an ein solches Artefakt:

- Unterstützung bei der Evaluation zusätzlicher ABAP-Programme bzgl. deren Auswirkung auf die Performance
- Unterstützung bei der Ermittlung benötigter und zu konfigurierender Systemressourcen
- Vermeidung von zeit- und kostenintensiven Tests durch reale Erzeugung des zu untersuchenden Workloads
- Unterstützung einer flexiblen Anpassung des Systems an sich ändernde Anforderungen

1.2.1. Erweiterung bestehender Ansätze

Im Zusammenhang der Performance-Modellierung und -Vorhersage eines SAP-ERP-Systems existiert eine Veröffentlichung von Rolia et al. (2009a). Dieser zeigt auf, dass sich „Layered Queuing“-Modelle, welche auf dem Ansatz der „Method of Layers“ (Rolia/Servcik 1995) basieren, für die Modellierung der Antwortzeit dieser Klasse von Systemen eignen. Das vorgestellte Modell beinhaltet die Abbildung der Benutzer, der Dialog-Work-Prozesse, der verschiedenen Update-Prozesse und der Datenbank. Es wird dargestellt, dass die Eigenschaften des LQM- und LQN-Ansatzes, wie z.B. prioritätsbasiertes Scheduling und asynchrone Aufrufe, die Genauigkeit der Performance-Modellierung erhöhen.

Weitere, in den folgenden Kapiteln noch im Detail vorgestellte, Komponenten des SAP-ERP-Systems, welche signifikanten Einfluss auf die Performance haben können, werden dabei jedoch nicht berücksichtigt. Aufgrund dieser Abstraktion benötigt der vorgestellte Ansatz eine Messdauer von ca. 40 Minuten, um einen konstanten Zustand identifizieren zu können. Dies kann darauf zurückgeführt werden, dass weder Puffer- noch Sperrverwaltung explizit modelliert ist. Gleichzeitig ist es bei diesem Ansatz nötig, den zu untersuchenden Workload bereits im operativen Betrieb zu vermessen, da die Ausführungszeiten der Sperr- und der Pufferverwaltung nur implizit in der Ausführungszeit der modellierten Komponenten

abgebildet werden. Dies verhindert eine ex-ante-Untersuchung der Performance eines Workloads vor der operativen Inbetriebnahme.

Die Erstellung eines Artefakts, das sowohl die Ziele dieser Arbeit als auch die genannten Anforderungen erfüllt, gliedert sich dabei in mehrere Komponenten mit zu erreichenden Zielen auf.

Das erste Ziel ist die Identifikation der Komponenten der Antwortzeit und der dafür verantwortlichen Systemkomponenten. Dazu müssen das System und dessen Komponenten analysiert werden. Diese Informationen werden benötigt, um die Komponenten des Modells spezifizieren zu können.

Um nun das Verhalten der einzelnen Komponenten so zu modellieren, dass sie nicht nur das Verhalten unter der Last während der Vermessung widerspiegeln, sondern auch für die Simulation einer variablen Anzahl von Benutzern tragfähig ist, müssen die internen Abläufe innerhalb der identifizierten Komponenten analysiert werden. Dabei ist ein angemessenes Abstraktionsniveau zu finden, welches einerseits die Abläufe detailliert genug abbildet, um die Simulation einer variablen Anzahl von Benutzern zu ermöglichen als auch noch die Vermessung der Detailschritte ermöglicht. Damit kann die Anforderung nach der Vermeidung von zeit- und kostenintensiven Systemtests unter einer real erzeugten Last erfüllt werden, da die einzelnen Lastschritte nur einmal für die Modellierung vermessen und analysiert werden müssen. Des Weiteren kann durch die Modellierung der internen Abläufe die Evaluation von ABAP-Programmen bzgl. deren Auswirkungen auf die Performance unterstützt werden.

Um den Ansatz nicht nur theoretisch diskutieren zu können, müssen bestehende Werkzeuge analysiert und ggf. weiterentwickelt werden, um Messdaten für die Modellierung, Parametrisierung und Evaluation bereitzustellen.

Ein weiteres Ziel besteht in der Modellierung der Antwortzeit. Aufgrund von Einschränkungen des gewählten Modellierungskonzepts ist eine Umsetzung zu identifizieren, welche die internen Abläufe soweit abstrahiert, dass die zur Verfügung stehenden Modellkomponenten verwendet werden können.

Ein weiteres Ziel dieser Arbeit ist es, aus den durch die identifizierten Messwerkzeuge bereitgestellten Daten die für die Parametrisierung benötigten Informationen zu ermitteln. Dabei müssen die Messdaten passend zu dem in der Modellierung gewählten Abstraktionsniveau aggregiert werden.

Zusammenfassend können durch die Ergebnisse der Simulation und der Analyse der für die Simulation erzeugten Daten neben der Vorhersage der Performance und die Identifikation von möglichen Flaschenhälsen die Anpassung der Konfiguration und damit des Systems flexibler gestaltet werden.

1.3. Verknüpfung mit anderen Arbeiten im Forschungsbereich Performance-Evaluation von SAP-ERP-Systemen

Die vorliegende Arbeit ist eingebettet in eine Reihe von Forschungsarbeiten zum Thema „Performance-Evaluation von SAP-ERP-Systemen“. In diesem Abschnitt sollen die

vorangegangenen und begleitenden Forschungen dieser Arbeit vorgestellt werden. Grundsätzlich teilen sich die Arbeiten, wie in Abbildung 1-1 dargestellt, vertikal in die Betrachtung ABAP- und J2EE-basierter Systeme. Horizontal unterscheiden sich die Arbeiten durch die zu untersuchenden Aspekte und der daraus resultierenden Wahl der Methode. In den bereits abgeschlossenen Arbeiten von Jehle (2010) und (Boegelsack 2010) wurden die Auswirkungen der Virtualisierung auf die Performance des jeweiligen Systemtyps mittels eines kontrollierten Software-Experiments auf Basis von Vergleichsmessungen ermittelt. Die Forschung im Bereich der Performance-Modellierung und Simulation des SAP-WebAS-ABAP, welche in dieser Arbeit beschrieben wird, und SAP-WebAS-Java bauen dabei auf den vorangegangenen Arbeiten auf. Dies zeigt sich durch die Verwendung der Messmethode in Form von Monitoren, Messvorgängen und Metriken. Die identifizierte Messmethode wird durch den zusätzlichen Bedarf an Performance Werten für Analyse, Modellierung, Parametrisierung und Evaluation in den Arbeiten über Performance-Modellierung und Simulation um entsprechende Monitore erweitert. Der bei (Boegelsack 2010) verwendete synthetische Benchmark, initiiert durch den Zachmanntest, fokussiert vollständig auf Erzeugung von Hauptspeicheroperationen. Aus diesem Grund wurde in der vorliegenden Arbeit ein zusätzlicher Lastgenerator entwickelt, welcher die Infrastruktur des SAP-Kernel in Form von Puffer- und Sperrverwaltung adressiert. Das in Jehle (Jehle 2010) konzipierte und implementierte „Performance Evaluation Cockpit for ERP Systems“ (PEER) beinhaltet die „Konfiguration und Steuerung der Ausführung einzelner Testfälle“ Jehle (Jehle 2010). Im Wesentlichen stellt es eine Integrationsbasis für Lastinjektoren und Analyseprogramme in Form von definierten Schnittstellen bereit. Die in der vorliegenden Arbeit prototypisch entwickelten Lastgenerator- und Analysekomponenten unterstützen die Integration in das PEER-Tool.

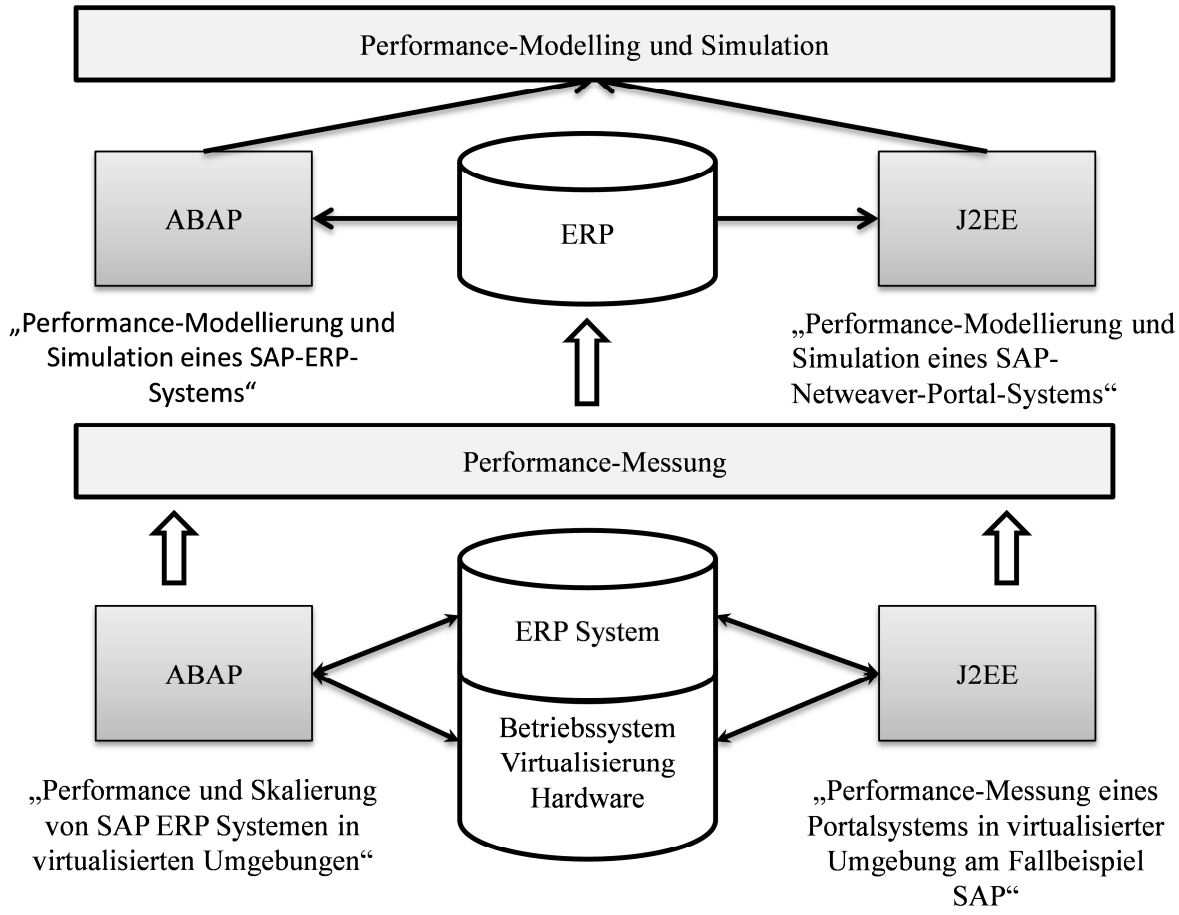


Abbildung 1-1: Verknüpfung mit anderen Arbeiten im Forschungsbereich Performance-Evaluation von SAP-ERP-Systemen
Quelle: Eigene Darstellung

1.4. Forschungsfragen

Um das Verhalten der technischen Performance eines SAP-ERP-Systems unter einer steigenden Anzahl von parallelen Benutzern mittels Simulation zu bestimmen, ist es nötig, die typische Art und Ausprägung der Last zu untersuchen. Darauf aufbauend müssen (Jain 1991) geeignete Metriken und statistische Methoden identifiziert werden, um die Performance unter der spezifizierten Last zu beschreiben. Nach der Analyse der internen Abläufe kann das Artefakt der vorliegenden Arbeit, ein Performance-Modell und eine geeignete Parametrisierung, erstellt werden. Die Evaluation des Artefakts wird durch Vergleich der Simulationsergebnisse mit Messdaten eines real existierenden Systems im Labor durchgeführt. Dazu wird ein Workload entwickelt, welcher in verschiedenen Szenarien vermessen wird und damit eine Basis für den Vergleich mit den Simulationsergebnissen bietet. Der Grundgedanke der vorliegenden Arbeit besteht in der identischen Architektur aller SAP-ERP-Systeme. Trotz einer Vielzahl von Programmen, durch welche einzelne Systeme speziell an ein Unternehmen und dessen Geschäftsprozesse angepasst werden, nutzen alle diese Programme die gleiche Infrastruktur, welche durch den SAP-Kernel bereitgestellt wird.

Die Durchführung der in der vorliegenden Arbeit beschriebenen Forschung wird anhand von 3 zentralen Forschungsfragen durchgeführt.

Um geeignete Modellierungs- und Simulationswerkzeuge für die Prognose der Performance auswählen zu können, müssen die internen Vorgänge eines SAP-ERP-Systems bzgl. ihrer Einflüsse auf die Performance analysiert werden. Aufgrund des großen Einflusses auf die Performance soll in der vorliegenden Arbeit ein besonderer Fokus auf die Analyse der Mechanismen gelegt werden, welche die Tabellenpufferung und die Sperrverwaltung durchführen. Unabhängig davon müssen alle technischen Komponenten, welche während der Verarbeitung von Benutzeranfragen aktiv sind, analysiert werden.

Forschungsfrage 1:

Welche Systemkomponenten haben Einfluss auf die Performance und wie wirken sich deren internen Vorgänge auf die Performance eines SAP-ERP-Systems aus, welche Performancedaten werden von einem SAP-ERP-System aufgezeichnet und wie können diese für eine externe Analyse extrahiert werden?

Einen weiteren wichtigen Punkt für die Auswahl der Modellierungs- und Simulationswerkzeuge stellen die verfügbaren Informationen über die internen Abläufe und deren Ausführungsdauer dar. Aus diesem Grund müssen nach der Analyse der internen Abläufe Werkzeuge untersucht und gegebenenfalls neu erstellt werden, welche die benötigten Leistungsdaten des SAP-ERP-Systems messen und für die Analyse zur Verfügung stellen.

Das Ergebnis dieser Forschungsfrage stellt eine Metrik dar, nachdem die technische Performance eines SAP-ERP-Systems bewertet werden kann. Zusätzlich wird eine Beschreibung der für die Performance des Systems relevanten Komponenten geliefert. Diese enthält die Analyse der internen Vorgänge, deren Einflussfaktoren auf die Gesamtantwortzeit und ein Vorgehen, wie die einzelnen Schritte vermessen werden können.

Forschungsfrage 2:

Welche technischen und logischen Systemkomponenten müssen modelliert und wie müssen diese parametrisiert werden, um ausreichend genaue Simulationsergebnisse zu erzielen?

Ist auf Basis der ermittelten Informationen die Entscheidung über geeignete Modellierungs- und Simulationswerkzeuge getroffen, kann das Performance-Modell erstellt werden. Dazu müssen die identifizierten Komponenten bzgl. ihres Einflusses auf die Performance analysiert und deren Verhalten modelliert werden. Dabei ist zu beachten, dass oftmals nicht die vollständigen internen Abläufe offengelegt sind, und daher die Modellierung nur eine Annäherung darstellen kann. Dabei ist es nötig, ein angemessenes Abstraktionsniveau zu finden, welches zum einen das Verhalten hinreichend genau abbildet, zum anderen jedoch mit den verfügbaren Informationen parametrisiert werden kann.

Die Abhängigkeiten zwischen den Komponenten des Modells werden dabei iterativ entwickelt und überprüft. Da LQN-Modelle (Woodside 2002) zu komplexeren Modellen kombiniert werden können Xu et al. (2005), startet die Modellierung und damit ebenfalls die Parametrisierung der in Forschungsfrage 1 identifizierten Komponenten auf einem hohen

Abstraktionsniveau. Um die Genauigkeit zu erhöhen, können die Komponenten iterativ durch detaillierte Modelle ersetzt werden. Dieser Ansatz soll die Komplexität des Modells und seiner Parametrisierung verringern.

Das Ergebnis von Forschungsfrage 2 stellt ein Modell samt geeigneter Parametrisierung dar, das die Performance eines SAP-ERP-Systems unter einer steigenden Anzahl von parallelen Benutzern einer Simulation wiedergibt.

> Forschungsfrage 3:
>
> Welche Unterschiede treten zwischen den Simulationsergebnissen und den im Labor gemessenen Werten auf, und was sind die Ursachen hierfür?

Um das Modell evaluieren zu können, wird in dieser Forschungsfrage ein Workload in Form einer Fallstudie identifiziert, welcher die Möglichkeit bietet, die Modellierung und Parametrisierung der einzelnen Komponenten zu überprüfen. Da die Validität des Modells unter einer steigenden Anzahl von parallelen Benutzern untersucht werden soll, müssen entsprechende Messwerte der einzelnen Lastsituationen vorliegen. Daher ist es für einen tragfähigen Vergleich nötig, einen Lastgenerator zu entwickeln, der den Workload reproduzierbar ausführt. Aufgrund der großen Fülle an Messdaten, welche für die Parametrisierung benötigt werden, muss ebenfalls ein Analyse- und Modellierungswerkzeug entwickelt werden, welches ein in Forschungsfrage 2 konzeptioniertes Modell für den Workload der Fallstudie erstellt und parametrisiert.

Ergebnis dieser Forschungsfrage ist die Beurteilung der Simulationsergebnisse im Vergleich zu real gemessenen Performance-Werten eines Systems im Labor. Ebenso wird der Einsatz des entwickelten Performance-Modells sowie der für die Parametrisierung erhobenen Daten in weiteren Analyse-Szenarien dargestellt.

1.5. Forschungsdesign

Die empirische Forschung ist wohl die am weitesten verbreitete Forschungsmethode. Dabei wird die Realität beobachtet und versucht quantitativ oder qualitativ Regeln, oft in Form von Ursache-Wirkungs-Ketten, für ein untersuchtes Phänomen zu identifizieren. Dieses Vorgehen ist vor allem in der angelsächsischen „Information Systems Research“ (ISR)-Disziplin sehr dominierend. In der deutschen „Wirtschaftsinformatik“ (WI) ist dagegen eine andere grundlegende Methode, Forschung durchzuführen, sehr beliebt. Das vorrangige Ziel dieser Methode ist, ein identifiziertes Problem mit Hilfe eines Artefakts in einer Umgebung zu lösen oder zu verringern. Während die strenge empirische Forschung versucht, Beobachtungen zu erklären, fokussiert die „Science of the artificial“ (Simon 1996), allgemein auch „design science“, auf die Erstellung von „things that serve human purposes“ (March/Smith 1995). Nach March/Smith (1995) können diese „Dinge“ unterschiedlich ausgeprägt sein.

- Konstrukte:
  Werden verwendet, um reale Beobachtungen zu beschreiben.

- Modelle:
  Innerhalb von Modellen werden Beziehungen zwischen Konstrukten aufgezeigt.

- Methoden
  Methoden werden erstellt, um eine Überführung eines Zustands in einen anderen Zustand zu beschreiben.

- Implementierungen:
  Die reale Instanziierung eines Konstrukts, Modells oder einer Methode

Unabhängig von der gewählten Forschungsmethode sind die Veröffentlichung und die Diskussion ein wichtiger Aspekt der Forschung. Damit dies möglich ist, muss das Vorgehen intersubjektiv nachvollziehbar sein. So können andere Personen verstehen, wie die Ergebnisse ermittelt wurden und damit die Qualität des Vorgehens sowie der Ergebnisse beurteilen.

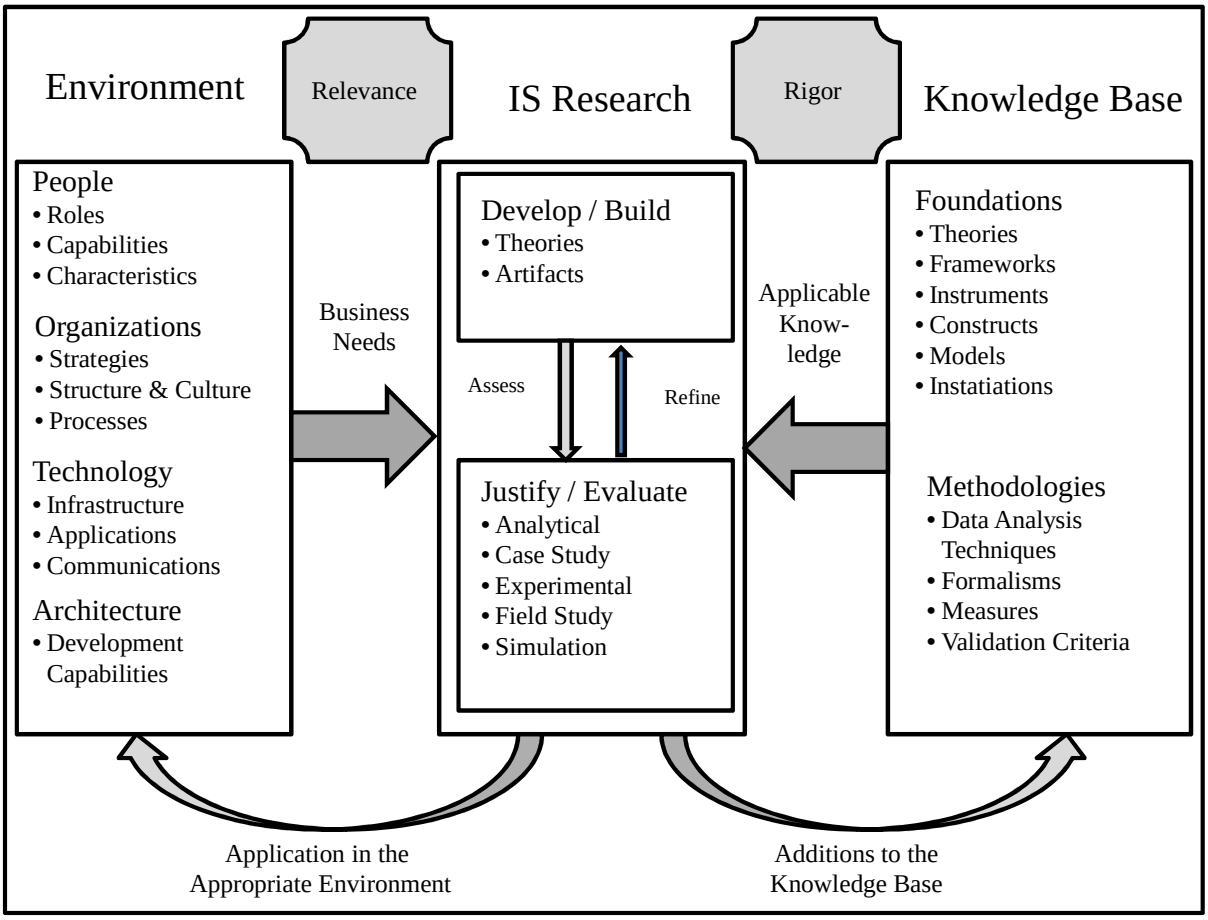


Abbildung 1-2: Design Science Framework nach Hevner et al.
Quelle: In Anlehnung an (Hevner et al. 2004)

Design Science wird verwendet, um eine Klasse von Problemen zu bearbeiten. Die führt oftmals dazu, dass keine oder nur sehr wenige theoretische Grundlagen für die Lösung des Problems und die Konzeption des Artefakts zur Verfügung stehen (Hevner et al. 2004). Obwohl nach Briggs (2006) die Erstellung des Artefakts durch Theorien unterstützt wird, ist es wahrscheinlicher, dass neue Modelle aus der Evaluation der Auswirkungen des Artefakts folgen. Nach (Nunamaker et al. 1991) folgen diese neuen Theorien vor allem aus dem grundlegenderen Verständnis des Kernproblems. Hevner et al. (2004) folgend, stellt sich Design Science vor allem als eine Suche nach einer passenden Lösung eines relevanten Problems und dem Ergebnis in Form eines nützlichen Artefakts dar. Design Science ist

grundsätzlich ein angewandtes Forschungs-Paradigma, dessen Ziel sich aus Bedürfnissen aus der Umgebung, in Abbildung 1-2 Environment genannt, ergibt. Die Umgebung spannt damit den Problemrahmen für gestaltungorientierte Forschung nach Hevner et al. (2004) auf. Diese ist auf einer häufig in der Literatur verwendeten Struktur in „People“, „Organizations“ und “Technology“ aufgeteilt. Nach (Silver/Markus/Beath 1995; Krcmar 2010) stellen diese die sich gegenseitig beeinflussenden Komponenten der ISR dar. Da die Umgebung den Problemrahmen aufspannt, muss das entwickelte Artefakt auch dort bzgl. seines Nutzens überprüft und evaluiert werden. Der Prozess des Suchens einer nützlichen Lösung besteht dabei aus einer Iteration aus der Implementierung oder Weiterentwicklung des Artefakts („Develop/Build“) und dem Evaluieren („Justify/Evaluate“) bzgl. der Nützlichkeit in der Umgebung („Application in the appropriate Environment“). Dieser Prozess wird durch existierendes Wissen aus der Wissensbasis („Knowledge Base“) unterstützt („Applicable Knowledge“), indem anerkannte Methoden verwendet werden. Des Weiteren werden Entscheidungen für das Design des Artefakts mittels vorhandener Theorien getroffen und bereits evaluierte Artefakte für die Erstellung verwendet. Damit baut das Artefakt auf bestehendem, evaluiertem Wissen auf und garantiert die mit „Rigor“ bezeichnete Nachweisbarkeit der Ergebnisse (Hevner et al. 2004). Das damit gewonnene Wissen, z.B. in Form verbesserter Modelle, muss dann in die Wissensbasis zurückgeführt werden („Additions to the Knowledgebase“), um diese zu erweitern.

Für die Umsetzung der in (Hevner et al. 2004) postulierten Anforderungen an eine nachhaltige, gestaltungsorientierte Forschung, orientiert sich die vorliegende Arbeit an dem von Pfeffers et al. (2006) vorgeschlagenen „design science process“. Dieser baut auf einer Analyse der Literatur zu gestaltungsorientierter Forschung auf und schlägt ein Modell, bestehend aus sechs Phasen, vor (Pfeffers et al. 2006):

- Problemidentifikation und Motivation:
  In dieser Phase muss das Problem identifiziert und der Nutzen einer Lösung dargestellt werden. Ebenso muss der Forscher sein Verständnis des Problems kommunizieren und die Durchführung der Forschungsarbeit rechtfertigen. Dazu müssen in dieser Phase bestehende Lösungen analysiert und das Verbesserungspotential aufgezeigt werden.

- Definition der Ziele:
  Aus dem in der vorhergehenden Phase dargestellten Problemverständnis müssen in dieser Phase die Ziele abgeleitet werden, welche das Artefakt erfüllen soll. Gibt es bereits Lösungen in diesem Problemkontext, so müssen diese auf Vor- und Nachteile überprüft werden.

- Konzeption und Entwicklung:
  Aufbauend auf den in der vorhergehenden Phase entwickelten Zielen muss nun das Artefakt zuerst konzeptioniert und dann entwickelt werden. Dies geschieht unter Zuhilfenahme bereits in der Wissensbasis vorhandener Theorien und Methoden.

- Demonstration:
  Ist ein Artefakt entwickelt, so muss in dieser Phase dessen Nützlichkeit innerhalb des definierten Anwendungsszenarios gezeigt und nachgewiesen werden. Dies impliziert

die Darstellung von Informationen über die Anwendung des Artefakts im definierten Problem-Raum.

- Evaluation:
  In dieser Phase werden die nachgewiesenen Ergebnisse bzgl. der bereits formulierten Ziele evaluiert. Dazu schlägt Hevner et al. (2004) verschiedene Methoden vor, um die Leistungsfähigkeit und Wirksamkeit in Bezug auf die entwickelten Ziele zu evaluieren. Das Ergebnis der Evaluation entscheidet, ob die Nützlichkeit in der geforderten Ausprägung erreicht wurde, oder ob der Zyklus aus Entwicklung und Evaluation erneut durchgeführt werden muss.

- Kommunikation:
  Ist die Nützlichkeit nachgewiesen, so muss das Ergebnis der Forschung durch Veröffentlichung in die Wissensbasis überführt werden. Dies beinhaltet neben der exakten Beschreibung des Problemrahmens, das Artefakt, das in dessen Erstellung erlangte Wissen, den erwarteten oder bereits nachgewiesenen Beitrag sowie die Konsistenz des Erstellungsprozesses.

Aufgrund der Durchführung des von Pfeffers et al. (2006) erstellten „Design Science“-Prozesses folgt die vorliegende Arbeit dem von Hevner et al. (2004) vorgeschlagenen „Design Science“-Paradigma. Dazu wird in Abschnitt 1.1 das zu lösende Problem motiviert und identifiziert. Abgeleitet aus den vorhergehenden Abschnitten werden in 1.2 die Ziele definiert. Die Kapitel 2, 3 und 4 stellen die aus der Wissensbasis verwendeten Erkenntnisse für die Konzeption und Entwicklung sowie deren Erweiterungen für die Erstellung des Artefakts dar. Das erstellte Artefakt wird in Kapitel 5 detailliert besprochen. Kapitel 6 enthält die Durchführung der Demonstration der Nützlichkeit sowie die Evaluation anhand eines ausgewählten Workloads, welcher den spezifizierten Problembereich nachbildet. Das zusammenfassende Fazit in Kapitel 7 setzt den letzten Schritt des „Design Science“-Prozesses nach Pfeffers et al. (2006) um, indem es die erlangten Erkenntnisse beschreibt und diese damit in die Wissensbasis überführt.

1.6. Struktur der Arbeit

Die Vorhersage oder Abschätzung der Performance eines SAP-ERP-Systems bei zu erwartender Änderung des Workloads in Form von neuen Programmen und/oder einer erhöhten Anzahl von Benutzern stellt Unternehmen vor große Herausforderungen. In der vorliegenden Arbeit soll ein neuartiger Ansatz für die Lösung dieses Problems vorgeschlagen werden. Wie bereits beschrieben, folgt diese Arbeit dem gestaltungsorientierten Forschungsansatz durch die Umsetzung des von Pfeffers et al. (2006) vorgeschlagenen Prozesses. Dazu werden in Abbildung 1-3 die 8 Kapitel dieser Arbeit den verschiedenen Phasen des „Design Science“-Prozesses nach Pfeffers et al. (2006) zugeordnet. Ebenso wurden die Kapitel in die Forschungsfragen eingeordnet, welche, wie in Abschnitt 1.4 dargestellt, die Erstellung des Artefakts leiten.

Das Kapitel 1 beschreibt zuerst ERP-Systeme, welche Geschäftsprozesse unterstützen, als den Kontext der Arbeit. In dem darauffolgenden Abschnitt wird die Problemstellung dargestellt und der in der vorliegenden Arbeit entwickelte Lösungsansatz motiviert, und die angestrebten Ziele werden definiert. Eine Darstellung des Kontextes der Erstellung dieser Arbeit enthält

das darauffolgende Unterkapitel. Die Art und Weise des Vorgehens und der daraus resultierende Aufbau der Arbeit wird in den Unterkapiteln zu den Forschungsfragen, dem Forschungsdesign und der Struktur der Arbeit beschrieben.

Da in der vorliegenden Arbeit ein Modell für die Performance-Vorhersage eines SAP-ERP-Systems erstellt werden soll, behandelt das Kapitel 2 in den theoretischen Grundlagen zunächst die Performance-Evaluation von Computersystemen. Als Grundlage für den Vergleich von Computersystemen anhand deren Leistung wird in diesem Kapitel dargestellt, welche Anforderungen eine Metrik für die Leistungsbewertung eines Computersystems zu erfüllen hat. Auf dieser Grundlage wird die Auswahl der in der vorliegenden Arbeit verwendeten Metrik dargestellt und begründet. Damit wird in diesem Kapitel die Forschungsfrage 1 bzgl. der für die Analyse, Simulation und Evaluation zu verwendenden Metrik beantwortet. Darauffolgend wird dargestellt, welche Vergleichskriterien in Form von Benchmarks für verschiedene Einsatzszenarien von Computersystemen verwendet werden können. Den Abschluss dieses Kapitels bilden die Grundlagen von LQNs, sowie eine Begründung warum diese für die Modellierung in dieser Arbeit verwendet wurden.

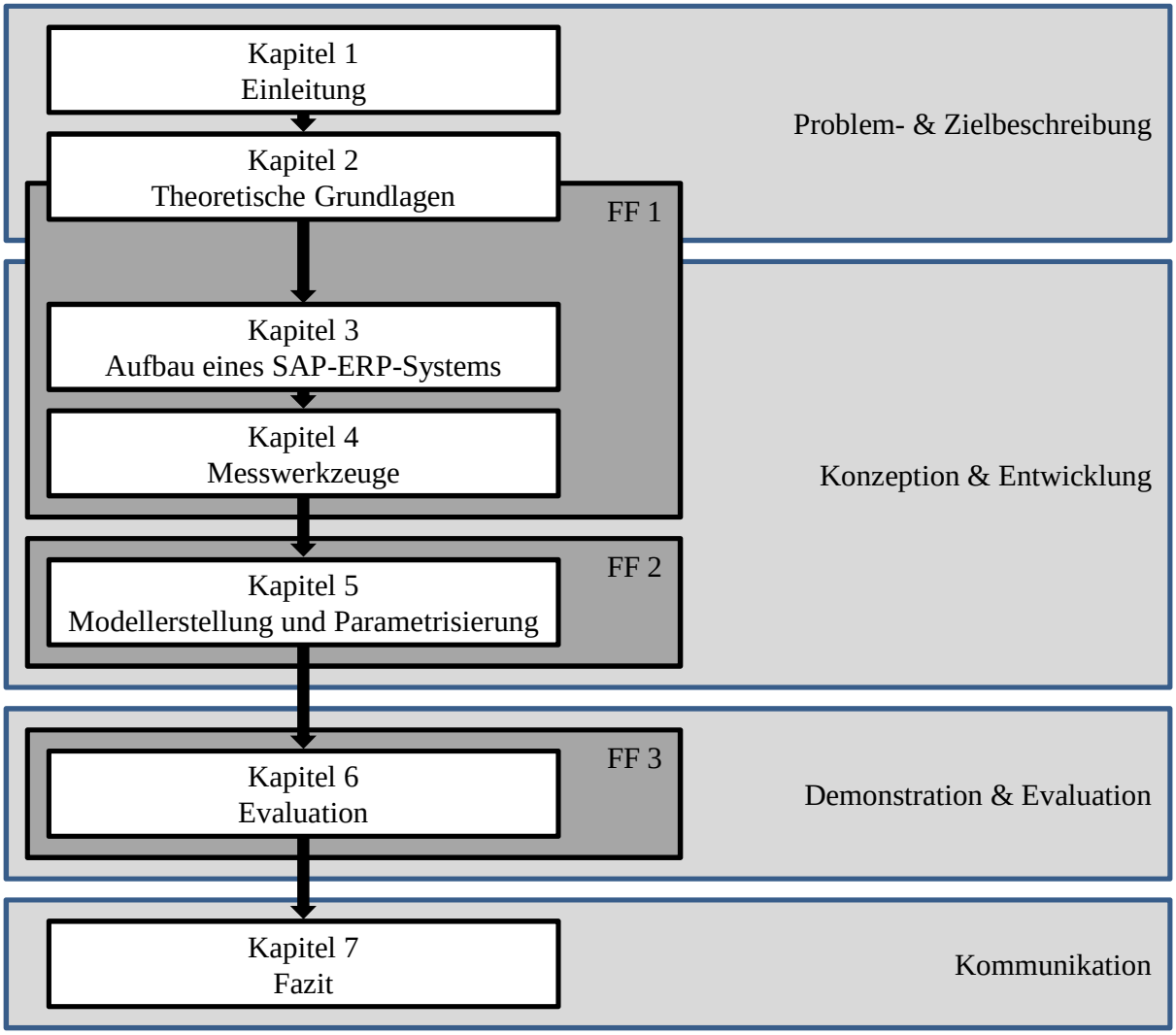


Abbildung 1-3: Einordnung der Kapitel der vorliegenden Arbeit in die Forschungsfragen und den Design-Science- Prozess nach Pfeffer
Quelle: Eigene Darstellung

In Kapitel 3 werden die Architektur und die an der Bearbeitung einer Benutzeranfrage beteiligten Komponenten dargestellt und deren Einfluss auf die Antwortzeit erarbeitet. Die ersten Abschnitte beschreiben daher die in der Architektur verwendeten Komponenten eines SAP-ERP-Systems und wie das Verhalten des Systems über die zentrale Konfiguration angepasst werden kann. Das darauffolgende Unterkapitel zeigt die Verwendung von unterschiedlichen Speicherarten sowie deren Einfluss auf die Performance des Systems auf. Dabei wird mittels eines Unterkapitels besonderer Fokus auf die Pufferung von Datenbanktabellen innerhalb des SAP-WebAS-ABAP gelegt. Um den Einfluss der verschiedenen Komponenten und Bearbeitungsschritte auf die Antwortzeit herauszuarbeiten, wird darauffolgend der Ablauf der Bearbeitung einer Benutzeranfrage und der beteiligten Komponenten im Detail dargestellt. Ein besonderes Augenmerk wird dabei auf die Regelung von wechselseitigen Zugriffen auf geschützte Systemressourcen gelegt. Die in diesem Kapitel identifizierten Einflussfaktoren auf die Antwortzeit werden in dem nachfolgenden Kapitel der offiziellen Beschreibung der Antwortzeitkomponenten zugeordnet. Im letzten Unterkapitel werden die in diesem Kapitel identifizierten Einflussfaktoren bewertet. Damit kann der Teil von Forschungsfrage 1 nach den zu modellierenden Einflussfaktoren auf die Antwortzeit beantwortet werden.

Kapitel 4 beschäftigt sich mit der Beantwortung des letzten Teils von Forschungsfrage 1. Dabei werden die verschiedenen Messwerkzeuge behandelt, die benötigt werden, um zum einen das Modell erstellen und parametrisieren zu können und zum anderen, um die Simulationsergebnisse mit gemessenen Werten eines realen Systems im Labor vergleichen zu können. Das Kapitel gliedert sich dabei in die Betrachtung von Messwerkzeugen auf Ebene des Betriebs- und des SAP-ERP-Systems auf. Während der erste Teil die Aufzeichnung von CPU-, I/O- und Hauptspeicher-Ressourcen mit Standard Unix-Tools beschreibt, werden im zweiten Teil zusätzlich zu den klassisch verwendeten Transaktionen neu entwickelte Werkzeuge und Vorgehen für die detaillierte Vermessung des Workloads innerhalb eines SAP-ERP-Systems eingeführt. Mit Kapitel 4 wird der zweite Teil der ersten Forschungsfrage und diese damit vollständig beantwortet.

Die zweite Forschungsfrage nach der Modellierung und Parametrisierung des Modells wird in Kapitel 5 bearbeitet, welches analog dazu in diese zwei Teile gegliedert ist. Dabei werden zunächst die für die Modellierung identifizierten Systemkomponenten dargestellt, Anforderungen an die Modellierung abgeleitet und die Umsetzung beschrieben. Aus den Modellen der einzelnen Komponenten folgt am Ende des Modellierungsteils dieses Kapitels die Zusammenstellung zum Gesamtmodell der Antwortzeit eines SAP-ERP-Systems. Der zweite Teil dieses Kapitel beschäftigt sich mit der Parametrisierung der Modellkomponenten. Dabei wird beschrieben, welche Daten mit den bereits in Kapitel 4 erläuterten Werkzeugen gemessen werden müssen, und wie daraus die für die Parametrisierung benötigten Werte ermittelt werden können. Damit beantwortet das Kapitel 5 die zweite Forschungsfrage und beschreibt das in der vorliegenden Arbeit erstellte Artefakt.

Die Demonstration und Evaluation des Artefakts ist Inhalt des 6. Kapitels. Dies ist zugleich Gegenstand der 3. Forschungsfrage nach den Unterschieden zwischen den Simulationsergebnissen und den im Labor gemessenen Performance-Werten eines realen Systems. Dabei wird zunächst der für die Evaluation verwendete Workload anhand von allgemein erstellten Anforderungen diskutiert und im Anschluss vorgestellt. Dabei wird

sowohl auf die inhaltlichen als auch auf die technischen Vorgänge im SAP-ERP-System eingegangen und die Implementierung aufgezeigt. Das folgende Unterkapitel beschäftigt sich mit der Lastintensität des Workloads, den Anforderungen und deren Umsetzung bei der Implementierung eines Lastgenerators für die wiederholbare Erzeugung eines realitätsnahen Anfrageverhaltens. Darauffolgend werden die gemessenen Performance-Werte vorgestellt. Dazu wird zuvor auf die Umgebung, die Voraussetzungen und die für die Messung ausgewählten Szenarien eingegangen. Der Entwicklung der Szenarien liegen die identifizierten Einflussfaktoren auf die Performance zugrunde. So werden diese Einflussfaktoren durch die Szenarien möglichst isoliert untersucht. Ein weiteres Unterkapitel behandelt die Durchführung der Simulationsläufe und deren Einteilung in Experimente sowie die aus der Rückgabedatei des Simulators zu extrahierenden Informationen. Der tatsächliche Vergleich der Mess- und Simulationsergebnisse findet im darauffolgenden Unterkapitel statt. Dazu werden die in dem jeweiligen Szenario fokussierten Modellkomponenten den Messergebnissen gegenübergestellt und diskutiert. Aus den dabei gewonnenen Erkenntnissen werden darauffolgend Lastbereiche eines SAP-ERP-Systems identifiziert und der Einsatz von Simulation in diesen diskutiert. Das Kapitel endet mit einer Beschreibung der möglichen Einsatzszenarien für das erstellte Artefakt. Dabei werden Aussagen sowohl aus den Simulationsergebnissen als auch aus den in dieser Arbeit entwickelten Messwerkzeugen dargestellt.

Kapitel 7 erfüllt die Anforderungen des „Design Science“-Prozesses von Pfeffers et al. (2006) nach der Kommunikation der Ergebnisse, in dem diese in Form eines Fazits dargestellt werden. Um das Fazit vollständig und inter-subjektiv nachvollziehbar zu gestalten, werden in diesem Kapitel ebenfalls die den Ergebnissen zu Grunde liegenden Limitationen erläutert. Die Arbeit endet mit einem Ausblick auf mögliche Anwendungsbereiche und weitere Forschungsmöglichkeiten.

2. Grundlagen der Performance-Evaluation

2.1. Performance-Evaluation von Computersystemen

Nach (Claus/Schwill 2003) ist Performance die „Geschwindigkeit und Qualität, mit der ein Auftrag oder eine Menge von Aufträgen von einer Datenverarbeitungsanalage verarbeitet wird“.

2.1.1. Methoden der Performance-Evaluation

Nach Jain (1991) existieren drei Methoden der Performance Evaluation von Computersystemen:

- Messung
- Analytische Modellierung
- Simulation

In den folgenden Absätzen werden diese Methoden kurz dargestellt und diskutiert.

2.1.1.1. Messung

Für die Evaluationsmethode Messungen werden in der Literatur drei Ziele diskutiert (Risse 2006). Der Hauptgrund ist dabei das Sammeln von Informationen über die Performance des Systems, um diese in Zukunft zu verbessern bzw. steuern zu können. Weitere Verwendung finden Messungen, um Informationen für die Modellierung von Workloads zu sammeln und Zusammenhänge im System analysieren und modellieren zu können. Schließlich werden Messergebnisse auch noch, wie in der vorliegenden Arbeit, verwendet, um Performance-Modelle zu validieren.

Um Messungen in Computersystemen durchführen zu können, werden Monitore verwendet, welche nach (Jain 1991) in Software-, Hardware- und hybride Monitore unterteilt werden. Unabhängig davon, welche Metriken durch Monitore gemessen werden, ist es nötig das Messinstrument hinsichtlich seiner Auswirkungen auf das Testobjekt zu untersuchen. In der vorliegenden Arbeit werden ausschließlich Softwaremonitore verwendet. Da sie softwareseitig implementiert sind, benötigen sie Systemressourcen und werden nach (Jain 1991) als intrusiv bezeichnet. Dies beeinflusst die Messung und kann die Ergebnisse verfälschen.

2.1.1.2. Analytische Modellierung

Im Gegensatz zur Messung wird bei der Performance Evaluation mit einem analytischen Modell der System-Performance gearbeitet. Dabei wird die Performance des Systems in einem vereinfachten und idealisierten Modell abgebildet, um die Komplexität zu reduzieren und damit auf die wesentlichen Punkte fokussieren zu können. Die Zusammenhänge der Modellkomponenten werden mathematisch ausgedrückt und können für die Lösung des Modells verwendet werden.

2.1.1.3. Simulation

Analog zu der analytischen Performance-Modellierung wird auch beim Simulationsansatz ein vereinfachtes und idealisiertes Modell der System-Performance entwickelt. Die Auswertung erfolgt dann jedoch nicht durch das Lösen mathematischer Formeln, sondern durch die Ausführung des Modells mit Hilfe eines Simulators. Die wohl am häufigsten verwendete Simulation ist die „Discrete Event Simulation“ (DES) mit den häufig verwendeten und weiterentwickelten Warteschlangennetzwerken, welche in Abschnitt 2.4. näher erläutert werden.

2.1.1.4. Beurteilung und Einsatzkriterien der Methoden

Um für ein Szenario die geeignete Methode wählen zu können, gibt Jain (1991) eine Einschätzung der verschiedenen Methoden bzgl. sieben Kriterien, welche für verschiedene Einsatzszenarien charakteristisch sind.

Criterion	Analytical Modeling	Simulation	Measurement
Stage	Any	Any	Prostprototype
Time required	Small	Medium	Varies
Tools	Analysts	Computer languages	Instrumentation
Accuracy	Low	Moderate	Varies
Trade-off evaluation	Easy	Moderate	Difficult
Cost	Small	Medium	High
Saleability	Low	Medium	High

Tabelle 2-1: Einschätzung der verschiedenen Evaluations-Methoden
Quelle: (Jain 1991)

Wie in Tabelle 2-1 dargestellt, sind die analytische Modellierung und die Simulation bzgl. der Phase des zu untersuchenden Objekts im Gegensatz zur Messung nicht eingeschränkt. Jedoch steigert sich die mögliche Genauigkeit der Modelle und Simulationen, wenn das Testobjekt und damit Performance-Messungen, wenn auch in einem frühen Stadium, verfügbar sind.

Auch die benötigte Zeit beurteilt Jain (1991) bei analytischer Modellierung und Simulation als vorhersagbar gering bzw. mittel, wohingegen die Messung das größte Risiko bzgl. der Zeit birgt.

Bei dem Kriterium „Tools“ wird gezeigt, dass für die verschiedenen Methoden natürlich unterschiedliche Ressourcen und Fähigkeiten benötigt werden, die nicht zwangsweise in ausreichender Menge innerhalb eines Projektteams oder einer Firma vorhanden sind.

Analog zu der benötigten Zeit beurteilt Jain (1991) die Genauigkeit des Ergebnisses der verschiedenen Methoden mit „Gering“ und „Mittelmäßig“ für die Modellierung und Simulation. Die Messung kann potentiell genauer sein, birgt jedoch verschiedene Risiken in sich. So können die Experimentvariablen nicht die Ausprägung der realen Welt treffen. Grundsätzlich weist Jain (1991) ausdrücklich darauf hin, dass alle Methoden große Fehlerquellen in sich bergen und im Idealfall durch eine zweite Methode validiert werden sollten.

Die Abschätzung der Wechselwirkungen zwischen verschiedenen Parameter-Konfigurationen für die Identifikation der optimalen Anpassung beurteilt Jain (1991) bei den analytischen Modellen als einfach, bei der Simulation als gemäßigt und bei der Messung als schwierig. Dies wird damit begründet, dass sich aufgrund der großen Vereinfachung verschiedene Konfigurationen mit Hilfe analytischer Modelle am einfachsten nachstellen und evaluieren lassen. Aufgrund der größeren Detailtreue, ganz abgesehen von der Dauer der Evaluation, wird dies für die Simulation aufwendiger. Da für die Messung real existierende Systeme benötigt werden, können gewisse Konfigurationen nur mit extremem Aufwand gemessen werden.

Basierend auf der Argumentation bzgl. des Aufwandes des vorhergehenden Absatzes entwickeln sich natürlich dementsprechend die Kosten für die verschiedenen Methoden.

Das letzte Kriterium „Saleability“ beinhaltet die Akzeptanz der verschiedenen Methoden und deren Ergebnissen. Diese wurde von Jain (1991) für die analytischen Modelle als „Gering“, für die Simulation als „Gemäßigt“ und als „Hoch“ für die Messung eingestuft.

2.1.2. Performance-Metriken

Die Datenerfassung schließt nach (Datta/Ioannou 1994) auch die Abbildung und Einordnung mit ein. Damit stellt die Auswahl der Metrik für die Bewertung und die Darstellung der Performance eines Computersystems eine wichtige Entscheidung dar.

2.1.2.1. Grundlagen zur Auswahl der Metrik

Nach (Hellermann 1975) nimmt die Auswahl der Metrik entscheidenden Einfluss auf die Interpretation der erhobenen Daten und damit auf die Information, welche erst durch die Interpretation der Daten entsteht (Broy 1997). Aufgrund der großen Bedeutung der Metrik in der Vermessung von Computersystemen definiert (Lilja 2000) sechs Kriterien, die eine Performance-Metrik erfüllen muss:

- Linearität:
  Ändert sich die Performance eines Systems um einen gewissen Faktor, so muss sich auch die Metrik um diesen Faktor ändern.

- Verlässlichkeit:
  Eine Metrik muss dahingehend verlässlich sein, dass ein System mit einer höheren Ausprägung einer Metrik immer performanter als ein System mit einer niedrigeren Ausprägung sein muss.

- Wiederholbarkeit:
  Wird bei identischen Experimenten die gleiche Ausprägung ermittelt, gilt diese Metrik als wiederholbar. Diese Definition beinhaltet auch die deterministische Eigenschaft einer Metrik.

- Einfachheit der Messung:
  Damit sollen Fehler bei der Ermittlung verhindert und so die Verwendung vereinfacht werden.

- Konsistenz:
  Die Definition der Metrik und deren Einheiten darf sich nicht von System zu System ändern. Dies ist nötig, um die Metrik für Vergleiche nutzen zu können.

- Unabhängigkeit:
  Eine Metrik muss in ihrer Definition unabhängig von äußeren Einflüssen sein. Diese darf nicht für bestimmte Stakeholder optimiert sein.

Um zu gewährleisten, dass die in dieser Arbeit durchgeführte Performance-Modellierung und Simulation anhand einer validen und für einen Vergleich von Performance-Werten tragfähigen Maßgröße geschieht, stütz sich die Wahl der verwendeten Metrik im folgenden Kapitel 2.1.2.2 auf diese Kriterien.

2.1.2.2. Auswahl der Metrik

Bei der Auswahl der zu verwendenden Bewertungsmaße bietet die Literatur zwar eine Fülle von Möglichkeiten, welche jedoch stark von dem jeweils betrachteten Anwendungsbereich abhängen und damit nicht allgemeingültig sind (Risse 2006). An. Die hierbei am weitesten verbreiteten Metriken sind neben den statischen Systemeigenschaften des zu untersuchenden Objekts an sich Antwortzeit und Durchsatz. (Lilja 2000) charakterisiert diese wie folgt.

- Testobjekteigenschaften:
  Diese können durch die Ausstattung und die Konfiguration des Systems angegeben werden. Jedoch beinhaltet dies keinerlei Aussagen über die zu erwartende Leistungsfähigkeit unter einem gewissen Workload.

- Durchsatz:
  Der Durchsatz als alleiniges Bewertungskriterium bietet sich vornehmlich für stark transaktionsorientierte Systeme ohne Benutzerbeteiligung an. Beispiele hierfür sind nach (Anderson/Mißbach 2005) und (Hellermann 1975) Flugbuchungssysteme oder Batch-Systeme, welche, wie z.B. in einem SAP-System, über die Nachtstunden eine über den Tag angesammelte Menge von Hintergrundaufträgen abzuarbeiten haben. Jedoch weist (Hellermann 1975) darauf hin, dass der Durchsatz nicht für Systeme mit Benutzerschnittstelle geeignet ist. Wird, wie in Absatz 2.3.4 noch beschrieben wird, zusätzlich eine maximale Antwortzeit eingeführt, so kann auch die Bewertung eines Systems mit Benutzerschnittstelle mittels des Durchsatzes erfolgen.

- Antwortzeit:
  Bei Systemen mit Benutzerschnittstellen wird in der Literatur nur in seltenen Fällen nicht die Antwortzeit als Metrik für ein System verwendet. Immer wieder zeigen Untersuchungen, dass bereits ab einer kurzen Wartezeit die Konzentration des Benutzers abnimmt und die Gedanken abschweifen. Hellermann (1975) beziffert diese Zeit mit 4 Sekunden.

Damit ist bei der Auswahl der Metrik nicht nur die Beschaffenheit des zu untersuchenden Systems von Bedeutung, sondern auch dessen Verwendungszweck. Wie Abbildung 2-1 zeigt, muss sich dabei entweder zwischen der Antwortzeit und dem Durchsatz entschieden werden. Bei maximalem Durchsatz neigt die Antwortzeit ebenfalls zu maximaler Ausprägung. Ist die Antwortzeit minimal, kann ein System häufig keinen hohen Durchsatz erzielen. In den

häufigsten Fällen kommt es zu einer Definition der maximalen bzw. minimalen Ausprägung auf beiden Achsen. Dies ist z.B. beim SAP-SD-Benchmark der Fall. Wie in Absatz 2.3.5 dargestellt, ist das Bewertungskriterium die Anzahl an durchgeführten Operationen pro Sekunde bei einer Antwortzeit von unter 2 Sekunden.

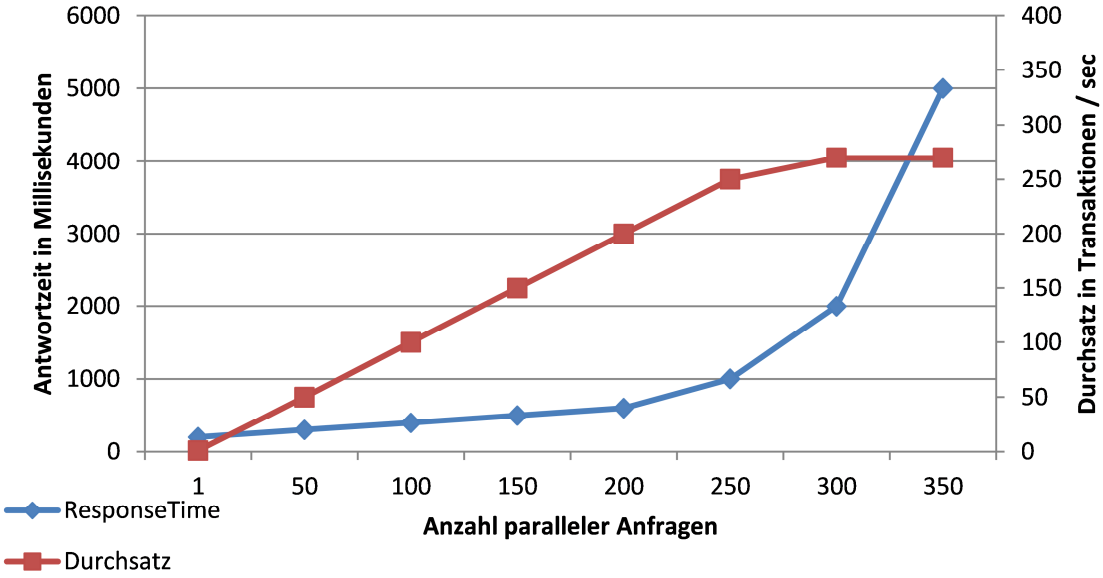


Abbildung 2-1: Durchsatz und Antwortzeit in Abhängigkeit der Anzahl paralleler Anfragen
Quelle: Eigene Darstellung

Da die vorliegende Arbeit die Performance-Evaluation von SAP-ERP-Systemen unter einer ansteigenden Anzahl von parallelen Benutzern fokussiert, wird die Antwortzeit als Metrik und damit als Vergleichskriterium sowohl für Simulations- und Messergebnisse in verschiedenen Szenarien als auch für die Szenarien selbst verwendet. Zudem erfüllt die Antwortzeit die von Lilja (2000) erhobenen Anforderungen an eine Metrik für die Performance-Analyse von Computersystemen.

2.1.2.3. Antwortzeit

Für die Antwortzeit gibt es verschiedene Definitionen, welche sich jedoch in der Tatsache decken, dass die Messung der Antwortzeit bei Beginn des Absendens der Anfrage startet. Nudd et al. (2000) definiert die Antwortzeit, wie in Abbildung 2-2 dargestellt, von Beginn der Anfrage des Clients zum Zeitpunkt T1 bis zum Zeitpunkt T3, an dem die Antwort vollständig beim Client eingetroffen ist. Erst ab diesem Zeitpunkt ist sichergestellt, dass die Antwort am Client verarbeitet werden kann. Andere Definitionen messen die Antwortzeit von Zeitpunkt T1 bis Zeitpunkt T2.

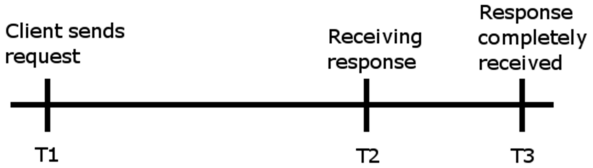


Abbildung 2-2: Antwortzeit aus Sicht des Clients
Quelle: In Anlehnung an (Menascé/Almeida 2002)

Abbildung 2-2 stellt die Antwortzeit aus der Sicht des Anwenders dar. Dies ist die Zeitspanne, die ein Anwender warten muss, bis er die Antwort auf seine Anfrage erhält und weiterarbeiten kann.

Aus der Sicht des Servers hingegen beginnt die Antwortzeit zum Zeitpunkt T3 in Abbildung 2-4, sobald die Anfrage des Clients vollständig eingetroffen und verarbeitet werden kann. Sie endet zum Zeitpunkt T4, wenn die Anfrage bearbeitet wurde und mit dem Senden der Antwort begonnen werden kann. Diese Zeitspanne entspricht der Definition der Bearbeitungszeit.

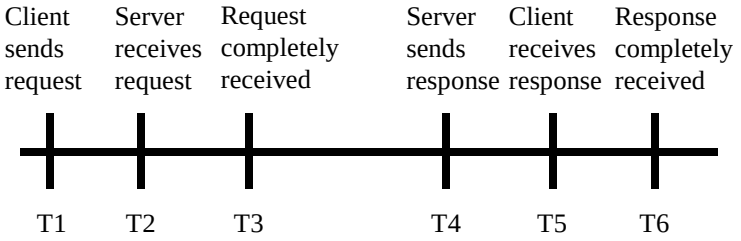


Abbildung 2-3:Antwortzeit aus Serversicht
Quelle: Eigene Darstellung

Da in dieser Arbeit die Performance des SAP-Systems und dessen interne Vorgänge im Mittelpunkt stehen und die Übertragungszeiten im Netzwerk nicht betrachtet werden, wird für den Rest dieser Arbeit die Antwortzeit als die Zeitspanne zwischen den Zeitpunkte T3 und T4 in Abbildung 2-3 verstanden.

2.2. Messtheorie

Das Messen bildet stellt einen Pfeiler moderner Wissenschaften, vor allem der Naturwissenschaften, dar. Laut Jaenecke (1982) lässt sich die Entwicklung empirischer Wissenschaft ohne eine exakte, wissenschaftlich fundierte, Messtechnik nicht vorstellen.

2.2.1. Messen

Nach Orth (1974) besteht Messen aus der Bestimmung einer Eigenschaft eines Objekts. Dies setzt die Existenz einer homomorphen (strukturerhaltenden) Abbildung eines empirischen Relativs auf ein numerisches Relativ voraus. Ein Lineal ordnet die Werte eines numerischen Relativs (eine Zahl) seinem eigenen empirischen Relativ (der Länge) zu.

2.2.2. Fehlertheorie

Wann muss ein Messwert als Fehler interpretiert werden? Um einen Messwert als Fehler zu interpretieren, muss der Messung ein Erwartungswert, z.B. in Form eines Konfidenzintervalls, zugeordnet werden. Ist ein Messwert nicht innerhalb dieses Bereichs, ist er als Fehler zu interpretieren. Allgemein formulieren Spillner/Linz (2007) einen Fehler als „Nichterfüllung einer festgelegten Anforderung, eine Abweichung zwischen dem Ist Verhalten und dem Sollverhalten“.

Je nach Art des Auftretens wird in der Literatur zwischen zufälligen und systematischen Fehlern unterschieden. Aufgrund von Fehlerraten, also einer relativen Häufigkeit des

Auftretens eines Fehlers, kann diese Kategorisierung vorgenommen werden. Die für die Einteilung in einen systematischen Fehler verwendeten Schwellwerte sind individuell für den jeweiligen Messvorgang abzustimmen. Dieser Schwellwert ist von großer Bedeutung, da im Gegensatz zu zufälligen Fehlern systematische Fehler das Ergebnis invalidieren.

2.2.3. Plausibilitätskontrollen

Bevor die erhobenen Daten für die Performance Evaluation genutzt werden können, ist es nötig, deren Plausibilität zu überprüfen. Damit wird sichergestellt, dass fehlerhafte Daten nicht bewertet werden bzw. der Testfall bei einer zu großen Anzahl fehlerhafter Daten wiederholt wird. Nach Prechelt (2001) gliedert sich die Plausibilitätskontrolle in die Prüfung von Konsistenz und Glaubwürdigkeit. Dabei besteht die Möglichkeit, dass für die Messung bereits Erwartungswerte definiert wurden, welche für die Plausibilitätskontrolle verwendet werden. Stehen diese nicht zur Verfügung, so kann dieser Erwartungswert statistisch aus einer großen Anzahl von Wiederholungen der Messung ermittelt werden.

2.2.3.1. Konsistenzprüfung

Für die Prüfung der Konsistenz werden Regeln verwendet, welche die logische Korrektheit der gemessen Werte beurteilen. Für die im Rahmen dieser Arbeit erhobenen Daten (Antwortzeiten und Zugriffszahlen) gilt grundsätzlich, dass keine negativen Werte vorkommen können. Aufgrund der detaillierten Analyse für die Parametrisierung des Performance-Modells können weitere Regeln abgeleitet werden. Für den im Rahmen dieser Arbeit verwendeten Workload werden ebenso Werte gleich Null als inkonsistent bewertet.

2.2.3.2. Glaubwürdigkeitsprüfung

Eine Verfeinerung der Auslese stellt die Glaubwürdigkeitsprüfung dar. Damit werden Bedingungen erstellt, welche die Werte, welche bereits als konsistent identifiziert wurden, bzgl. der Glaubwürdigkeit beurteilen. So können beispielsweise extrem hohe oder niedrige Werte auf Basis einer großzahligen Ausführung identifiziert und deren Auswirkung auf die Beurteilung der erhobenen Daten über Mittelwertbildungen eingeschränkt werden.

2.2.4. Zusammenfassen und Beschreibung von Daten

Um eine große Menge an Messwerten in einem Wert zusammenzufassen, wird häufig der Mittelwert verwendet. Da die erhobenen Daten jedoch nicht mit dem Mittelwert identisch, sondern gestreut sind, muss nach Becker (1993) dieser Streuung durch eine zusätzliche Beschreibung Rechnung getragen werden.

2.2.4.1. Spannweite

Die intuitivste Angabe dieser Streuung kann durch die sogenannte Spannweite erfolgen. Diese wird durch Subtraktion des kleinsten Wertes von dem größten Wert der Datenreihe gebildet. Laut Becker (1993) ist diese jedoch sehr anfällig für Extremwerte und damit nur bedingt aussagekräftig, da diese keine Information über die Verteilung der Werte zwischen den Extrempunkten enthält.

2.2.4.2. Interquartilsabstand

Eine bessere Beschreibung der Streuung mit Berücksichtigung der Verteilung der Messwerte bietet der Interquartilsabstand (Becker 1993). Um diesen Abstand zu errechnen, muss eine Häufigkeitsverteilung aller Messwert innerhalb einer Datenreihe erstellt und diese in vier gleich große Abschnitte unterteilt werden. Als Interquartilsabstand wird die Spannbreite von der Obergrenze des ersten Quartils zur Obergrenze des dritten Quartils bezeichnet und ist damit das Maß für die Streuung der mittleren 50 Prozent der Verteilung.

2.2.4.3. Standardabweichung

Eine weitere Möglichkeit für die Beschreibung von Verteilungen von Messwerten ist die Stichprobenvarianz S^2 und die daraus gewonnene Stichprobenstandardabweichung S. Da diese anhand von Stichproben errechnet wird, handelt es sich dabei nach Fahrmeier et al. (1999) lediglich um Schätzwerte für die tatsächliche Varianz und Standardabweichung der Grundgesamtheit.

$$stddev = \sqrt{stddev^2} = \sqrt{\frac{\sum_{i=1}^{n}(x_i - \bar{x}_A)^2}{n-1}}$$

In dieser Formel stellen x_i die n unabhängigen Messwerte dar. Das für die Berechnung der Standardabweichung benötigte arithmetische Mittel wird durch $\bar{x}_A$ dargestellt.

2.2.4.4. Variationskoeffizient

Im Umfeld der deskriptiven Statistik wird als Maß für die Streuung häufig der empirische Variationskoeffizient oder der quadrierte empirische Variationskoeffizient verwendet. Diese berechnen sich durch die folgenden Formeln:

$$cv = \frac{stddev}{\bar{x}_A} \text{ bzw. } cv^2 = \frac{stddev^2}{{\bar{x}_A}^2}$$

Wie aus der Formel zu sehen ist, hat der Variationskoeffizient keine Abhängigkeit zur Dimension der zu messenden Größen und kann somit allgemein sehr gut als Beschreibung für eine Streuung verwendet werden.

2.2.4.5. Ausreißer

„We shall define an outlier in a set of data to be an observation (or subset of observations) which appears to be inconsistent with the remainder of that set of data“ (Barnett/Lewis 1994). Ausreißer sind demnach Werte, welche aus einem definierten Rahmen fallen. Dieser Rahmen muss aufgabenbezogen definiert und kann nicht allgemeingültig festgelegt werden. Ausreißer sind nicht zwingend falsch. Nach Spillner/Linz (2007) entstehen sie zwar oft durch fehlerhafte Bearbeitung, ebenso sind jedoch auch unerwartete externe Einflüsse häufige Ursache. Unabhängig von der Ursache können sie nachgewiesen negatives Einflusspotential auf die Analyse von Datensätzen haben (Schendera 2007). Um für die in der vorliegenden Arbeit erhobenen Daten als Kriterium für die Identifikation von Ausreißern einzuführen, eignet sich nach Jehle (2010) die Bestimmung eines Konfidenzintervalls. Werte, die

außerhalb dieses Konfidenzintervalls liegen, können damit als Ausreißer identifiziert und von der Evaluation ausgeschlossen werden.

2.2.4.6. Konfidenzintervall

Unter einem Konfidenzintervall versteht man die Menge von Daten eines Datensatzes zwischen zwei Werten x_1 und x_2. Die Wahrscheinlichkeit 1-α, mit der sich der Mittelwert innerhalb dieses Bereichs befindet, wird als Konfidenzniveau bezeichnet (Bankhofer/Vogel 2008). Nach Lilja/Yi (2006) wird α als Signifikanzlevel bezeichnet. Einhergehend mit der Wahrscheinlichkeit 1-α befinden sich auch im Verhältnis genau so viele Werte des Datensatzes innerhalb dieses Intervalls (Kemmesies 2009).

Um die beiden Grenzen des Konfidenzintervalls x_1 und x_2 zu bestimmen, müssen die Werte nach Lilja/Yi (2006) mit folgender Formel transformiert werden:

$$z_i = \frac{\bar{x} - x}{\frac{stddev}{\sqrt{n}}}$$

Die somit erzeugten z-Werte folgen der sogenannten t-Verteilung, welche die Eigenschaft besitzt, dass sie sich mit steigendem n der Standard-Normalverteilung nähert (Bortz/Döring 1995). Damit können die Grenzen x_1 und x_2 nach Lilja/Yi (2006) mit der folgenden Formel unter Zuhilfenahme der Tabelle für t-Verteilungen berechnet werden:

$$x_{1,2} = \bar{x} \pm t_{\frac{1-\alpha}{2};n-1} * \frac{stddev}{\sqrt{n}}$$

Für die vorgestellte Ermittlung der Eckpunkte des Konfidenzintervalls muss sichergestellt werden, dass die zufällige Abweichung Δ_z der Normalverteilung folgt. Für die vorliegende Arbeit gilt diese Anforderung als erfüllt, da nach (Lilja 2000) dies für eine Messreihe mit mehr als 30 unabhängigen Messwerten angenommen werden kann.

2.2.4.7. Oszillierende Messwerte

Trotz Berücksichtigung des im vorhergehenden Kapitel besprochenen Einschwingeffekts zeigen die Messwerte trotz großzahliger Wiederholungen scheinbar zufällige Schwankungen. Nach der Elimination von Ausreißern zeigen sich noch Schwankungen von über 10 Prozent des errechneten Mittelwerts, welche auch durch eine signifikante Erhöhung der Anzahl an Wiederholungen nicht eliminiert werden konnten. In der Literatur werden als Ursache für diese Schwankungen nicht deterministisch auftretende Systemereignisse wie Fehler bei Seitenzugriffen, Fehler bei Hauptspeicherzugriffen oder Auswirkungen von Scheduling-Aktionen auf Betriebssystemebene genannt, welche zusätzliche Ressourcen benötigen (Parupudi/Winograd 1972; Lilja 2000).

2.2.4.8. Vergleich von Alternativen

Der in der Literatur „Vergleich von Alternativen“ (Lilja 2000) genannte Vorgang dient dazu, um verschiedene Messreihen und häufig damit verbundene Experimente oder Systemkonfigurationen miteinander zu vergleichen. In der vorliegenden Arbeit wird diese

Methode verwendet, um zu zeigen, dass sich die in 6.3.2 dargestellten Systemkonfigurationen tatsächlich unterscheiden. Dabei werden die verschiedenen „Alternativen“, in der vorliegenden Arbeit Szenarien genannt, durch die Anwendung statistischer Methoden miteinander verglichen. In der Literatur (Lilja 2000) werden für diesen Vergleich häufig Konfidenzintervalle, arithmetischer Mittelwert, Median und Modus verwendet. Aufgrund der Oszillation der Messwerte (2.2.4.7) und der fehlenden Klassen von Messwerten scheiden die Verwendung von Konfidenzintervallen und Modus aus.

Aus den verbleibenden Methoden wird in der vorliegenden Arbeit der Median verwendet, da sich der Median in den verschiedenen Messreihen, wie auch in (Ferschl 1970) dargestellt, als robuster gegenüber einzelnen stärker ausgeprägten Werten verhält. Dies basiert darauf, dass der Median im Gegensatz zum arithmetischen Mittel nicht die vollständige Zahlenreihe betrachtet, sondern nur das nach einer Sortierung mittlere Element (bei gerader Anzahl den Durchschnitt der beiden mittleren). Aus diesem Grund wirken sich einzelne Ausreißer weniger stark auf das Ergebnis aus, als dies beim arithmetischen Mittel der Fall ist.

2.3. Benchmark/Benchmarking

„Unter dem Begriff Benchmark (BM) versteht man eine Menge von Programmen, die als geschlossenes Auftragspaket von einem EDV-System bearbeitet werden sollen“ (Schreiber/Thomas/Wolf 1974). Die für die Bearbeitung dieses Auftragspaketes benötigte Zeit kann demnach als Leistungsmaß für das EDV-System dienen. (Prechelt 2001) detailliert diese Definition. Danach ist ein Benchmark ein „definierter Anwendungsfall für ein Programm oder eine Methode, der eine Vorschrift einschließt, wie das Ergebnis der Anwendung quantifiziert werden kann“. Mit dieser Definition kommt die Vorschrift der Quantifizierung der Ergebnisse hinzu, sodass die Verwendung eines Benchmarks darauf abzielt, eine Vergleichsmöglichkeit von verschiedenen Systemen bereitzustellen. Dieses Kriterium stellt auch Hoetzel et al. (1998) heraus, in dem er die hauptsächliche Verwendung von Benchmarks darin sieht, die eigene Entwicklung zu messen und mit Konkurrenten zu vergleichen. Darüber hinaus verstehen Heinrich/Lehner (2005) Benchmarks als Möglichkeit, Systeme zu analysieren und damit zu verstehen.

2.3.1. Anforderungen an einen Benchmark

Aus den im vorherigen Abschnitt beschriebenen Anwendungsfällen ergeben sich Anforderungen an einen Benchmark. Ein zentraler Punkt für die Verwendung eines Benchmarks ist die Vergleichbarkeit verschiedener Objekte. Damit dies im Umfeld von Computersystemen gegeben ist, müssen Benchmarks nach (Versick 2010; Heinrich/Lehner 2005) folgende Anforderungen erfüllen:

- Portabilität:
  Der Benchmark muss auf allen Plattformen, die miteinander verglichen werden, lauffähig sein.

- Repräsentativität:
  Unabhängig von der Anzahl an Applikationen müssen Benchmarks repräsentative Ergebnisse liefern.

- Wiederholbarkeit:
  Bei identischen Bedingungen müssen Benchmarks gleichbleibende Ergebnisse liefern.

- Verifizierbarkeit:
  Die Ergebnisse der Benchmark-Läufe müssen nachvollziehbar und über-prüfbar sein.

- Skalierbarkeit:
  Der Benchmark soll auch bei größerer Last signifikante Werte erzeugen.

Für die spezielle Anwendung im Umfeld von SAP-ERP-Systemen können diese Anforderungen in Anlehnung an (Boegelsack 2010) ergänzt werden:

- Vergleichbarkeit:
  Um die Vergleichbarkeit zwischen verschiedenen Implementierungen von SAP-ERP-Systemen zu gewährleisten, muss ein Benchmark die internen Strukturen des SAP-Systems nutzen.

Je nach Anwendungsgebiet, sowohl bzgl. der gewünschten Aussage als auch bzgl. des zu untersuchenden Objekts, gibt es verschiedene Typen von Benchmarks, die in den folgenden Absätzen kurz dargestellt werden.

2.3.2. Instruction-Mix

Auf der untersten, maschinennahesten Schicht befinden sich die Instruction-Mix Benchmarks. Dabei wird die erzeugte Last durch die Kombination von verschiedenen Maschinenbefehlen und deren relativer Häufigkeit bestimmt. Diese elegante und am häufigsten durch den Gibson Mix (Gibson 1970) verwendete Methode, um die Leistungsfähigkeit von verschiedenen Prozessoren zu vergleichen, wird aufgrund der großen Komplexität heutiger Hardware nicht mehr verwendet, da die Ergebnisse nur vergleichbar sind, wenn es sich um Prozessoren mit gleichem „intruction set“ handelt. Ein weiterer Grund für die aktuell sehr geringe Verwendung besteht auch in der Tatsache, dass diese Art der Benchmarks völlig auf die CPU fokussiert und andere Komponenten, wie Arbeitsspeicher, I/O und darauf ausgeführte Anwendungen, außer Acht lässt.

2.3.3. Synthetische Benchmarks

„They do no ‚useful‘ work“ (Hu/Gorton 1997). Damit ist gemeint, dass diese synthetischen Programme im Gegensatz zu den im nächsten Kapitel besprochenen „Application Benchmarks“ keine realen Funktionen einer existierenden Applikation ausführen, sondern gewisse Funktionen des unter einer Applikation laufenden Systems ausführen und damit versuchen, realen Workload einer Applikation zu emulieren. Bekannte Beispiele für synthetische Benchmarks sind nach (Hu/Gorton 1997) der NFSStone (Shein/Callahan/Woodbuy 1989) Benchmark für die Messung der Performance des Network File Systems (NFS) der Firma SUN (Sandberg et al. 1985), der IOStone Benchmark (Park/Becker 1990) für die generelle Messung von I/O oder der I/O-Benchmark von Chen und Patterson (Chen/Patterson 1994), welcher im Gegensatz zu IOStone (für diese Zeit) sehr große Dateien von 100MB verwendet. Grundsätzlich haben synthetische Benchmarks den Vorteil, dass sie einfach zu erstellen und über Kontrollparameter an veränderte Gegebenheiten anzupassen sind. Damit können sie für die Emulation einer Vielzahl von Verwendungen

angepasst werden. Jedoch sind sie aufgrund der Einfachheit oft nicht umfassend genug, um die Performance eines Systems verlässlich zu bestimmen (Jain 1991), da gewisse Bottlenecks erst während der Verwendung des Systems unter realem Workload auftreten.

2.3.4. Application-Benchmarks

Auf der Ebene der Applikationen bewegen sich die sogenannten „Application Benchmarks“ (Marquard/Götz 2008). Diese zeichnen sich dadurch aus, dass sie repräsentative Funktionen einer Applikation innerhalb des Benchmarks nutzen (Hu/Gorton 1997). Damit bilden sie realen Workload am genauesten von den bisher genannten Benchmark-Typen und sind so für die Beurteilung der Performance von Systemen für die entsprechenden Applikationen am besten geeignet. Um die Benchmarks bzgl. der schon erwähnten Durchführungs- und Bewertungsrichtlinien zu präzisieren, wurde 1988 das Transaction Processing Performance Council (TCP) gegründet. Im Vergleich zu den sich seit den 70er Jahren entwickelnden „Debit-Credit“-Benchmarks (Hu/Gorton 1997), welche sich auf die „Total Cost of Ownership“ (TCO) fokussierten und durch ein Preis-Performance-Verhältnis bewertet wurden, zeichnet sich die durch das TCP eingeführte Bewertungsvorschrift dadurch aus, dass sie die Performance nach dem Durchsatz in „Transactions per Second“ bewertet. Dabei wird dieser Durchsatz nur gewertet, wenn 90 Prozent dieser Transaktionen die Zeit von 2 Sekunden nicht überschreiten. Damit wird dem entgegengesetzten Zielen der Maximierung von Antwortzeit und Durchsatz Rechnung getragen.

2.3.5. SAP-Benchmarks

Wie in Absatz 2.3.4 dargestellt, verwenden Application-Benchmarks repräsentative Funktionen einer Applikation und sind demzufolge spezifisch für diese. Da sich die vorliegende Arbeit mit der Performance-Analyse von SAP-ERP-Systemen, genauer dem ABAP-Stack, auseinandersetzt, wird in diesem Abschnitt der von der SAP AG angebotenen und von Kunden und Partnern am meisten genutzte Benchmark, der SD-Benchmark, besprochen. Nach (SAP 2011h) bedient sich dieser Benchmark eines Szenarios aus dem „Sales and Distribution“-Umfeld, um die Performance eines SAP-Systems zu beurteilen. Die Durchführung wird durch das „SAP Benchmarking Council“ überwacht, welches aus Vertretern der SAP AG und der Hardware-Partner besteht. Die Metrik, mit welcher das „gebenchmarkte“ System beurteilt wird, ist ein „SAP Application Performance Standard“ (SAPS). Ein SAPS besteht aus „2000 fully business processed order line items per hour“ wobei diese 2000 Vorgänge ca. 2400 SAP-Transaktionen beinhalten, welche sich wiederum aus 6000 Dialog-Schritten und 2000 Verbuchungsvorgängen zusammensetzen. Eine Hardware-Plattform hat damit einen SAPS, wenn pro Stunde 2000 Kundenaufträge mit einer Antwortzeit von unter 2 Sekunden durchgeführt werden können. Da jedoch real existierende Systeme in den seltensten Fällen nur aus „SD“- Anwendungen bestehen, hat dieser Benchmark keine verlässliche Aussagekraft für die Beurteilung eines SAP-Systems in einer Kundenumgebung. Dieser Benchmark wird vielmehr von den Hardware-Partnern verwendet, um die Leistungsfähigkeit der angebotenen Hardware-Plattformen zu bewerten. Nach (Boegelsack 2010) ist dieser Benchmark umstritten, da er „keine realitätsnahen Ergebnisse liefert und lediglich zur Demonstration der maximalen Leistungsfähigkeit von stark optimierten Serversystemen dient“. Des Weiteren ist der SD-Benchmark stark auf Operationen auf CPU und Hauptspeicher optimiert, sodass er bei der Beurteilung eines real

existierenden Systems, welches auch I/O-Operationen enthält, keine verlässliche Aussage über die Performance unter einer steigenden Anzahl paralleler Anfragen zulässt.

2.3.6. Fazit

Existierende Benchmarks eigenen sich im begrenzten Maße für die Beurteilung der Leistungsfähigkeit von Hardware-Plattformen in speziellen SAP-Workloads. Jedoch bieten sie keine verlässliche Methode, um die Performance von real existierenden SAP-Systemen unter einer steigenden Anzahl von parallelen Anfragen zu analysieren.

2.4. Performance-Modellierung und Simulation

Nach Jonkers (1994) können Formalismen zur Performance-Modellierung nach „deterministic“ und „probabilistic“ eingeordnet werden. Während in deterministischen Modellen Quantitäten wie die Zeit statisch sind, beinhalten wahrscheinlichkeitsbasierte Ansätze eine gewisse Ungewissheit. Dies hat den Vorteil das Modelle die nicht-deterministische Aspekte paralleler Prozesse enthalten gelöst werden können (Jonkers 1994). Die wichtigsten Vertreter dieser wahrscheinlichkeitsbasierten Modelle sind Markov-Ketten, Petri-Netze und Warteschlangenmodelle. Obwohl Markov-Ketten und Petri-Netze, mit den verschiedenen Klassen wie „Timed Petri Nets“ (TPN), „Stochastic Petri Nets“ (SPN), und „Queueing Petri Nets“ (QPN) Vorteile wie die Unterstützung des Modellierers im Verständnis der internen Prozesse durch die graphische Notation, besitzen diese auch wichtige Nachteile. Bolch et al. (2006) führt auf, dass bei der Verwendung von Markov-Ketten für die Modellierung komplexer Systeme die Anzahl der Zustände sehr groß werden kann. Nach Bause (1993) ist es mit Petri-Netzen sehr schwierig, Scheduling-Strategien zu beschreiben, welche für die Modellierung von Parallelismus bei Sperren und Synchronisationen von Ressourcen benötigt werden. Im Bereich der Performance-Simulation ist die Warteschlangentheorie weit verbreitet. Der LQN-Formalismus stellt eine Erweiterung der häufig verwendeten „Queuing Network models“ (Rolia/Servcik 1995) dar. Diese werden nach Ufimtsev (2006) wie folgt bewertet:

Vorteile des LQN-Formalismus:

- Schichten-Architektur bildet komponentenbasierte, verteilte und geschichtete Systeme im Geschäftsbereich
- Leichte Erweiterbarkeit der LQN-Modelle um neu, als Flaschenhals identifizierte Ressourcen oder Geräte
- Vermeidung des sogenannten des „State explosion problem“ vieler anderer (SPN basierter) Formalismen durch Verwendung eines höheren Abstraktionsgrades
- Verfügbarkeit robuster Lösungs- und Simulationswerkzeuge

Nachteile:

- Keine Unterstützung dynamischer Quantitäten in der Modellierung und Parametrisierung

- Große Menge an exakt interpretierten Daten für die Modellierung benötigt

Da der LQN-Formalismus die Modellierung von hierarchischen und parallelen Aktivitäten (Rolia/Servcik 1995; Woodside 2002) unterstützt, welche in verschiedenen Servern mit gemeinsamen oder geteilten Warteschlangen repliziert sind (Omari et al. 2007), kann dieser sehr gut für die Modellierung der Systemarchitektur eines SAP-ERP-Systems verwendet werden. Da es ebenfalls möglich ist Submodelle mehrfach zu verwenden (Xu et al. 2005) ist dieser Formalismus sehr flexible im Hinblick auf Modellerweiterungen. Das in Computer-Systemen häufig zur Reduzierung der Antwortzeit verwendete Phasen Konzept beinhaltet die Ausführung von Tasks, die im Anschluss an die Beantwortung einer Benutzeranfrage durchgeführt werden müssen, jedoch nicht für die Antwort benötigt werden, in einer der Antwort nachgelagerten Phase. Damit wirkt sich diese zweite Phase nicht direkt auf die Antwortzeit aus, muss jedoch trotzdem berücksichtigt werden, da diese ebenso Systemressourcen benötigt.

Aus diesen Gründen wurde für die Erstellung des in der vorliegenden Arbeit beschriebenen Artefakts der LQN-Formalismus für die Modellierung und Simulation eines SAP-ERP-Systems verwendet. Im Folgenden werden deshalb zunächst die Grundlagen der Warteschlangentheorie dargestellt, und darauffolgend die erweiternden Konzepte des LQN-Formalismus vorgestellt.

2.4.1. Warteschlangentheorie

Die Grundlagen und die Entstehung der Warteschlangentheorie geht zurück auf Agner Krarup Erlang (1878 - 1929), der Stauerscheinungen im Telefonnetz der Stadt Kopenhagen untersuchte. Seit dieser Zeit entstanden Erweiterungen und Weiterentwicklungen für die Anwendung in den verschiedensten Bereichen, um die Leistung von System zu bewerten und zu analysieren.

2.4.1.1. Grundlagen der Warteschlangentheorie

Das Grundelement eines Warteschlangennetzes bildet dabei ein sogenannter Bediener, auch Service Center genannt, siehe Abbildung 2-4. Verschiedene Arten von Bediener können zu komplexen Warteschlangennetzen kombiniert werden, um Systeme oder Abläufe zu modellieren und hinsichtlich deren Leistung bzgl. Länge der Warteschlange, durchschnittliche Verweilzeit in der Warteschlange, usw. zu beurteilen.

Eine Warteschlange wird durch folgende Parameter charakterisiert:

- Ankunftsrate λ: Beschreibt die Anzahl von Aufträgen pro Zeiteinheit
- $A(\lambda)$: Bezeichnet die Länge der Warteschlange und wird gebildet aus der durchschnittlichen Anzahl von Aufträgen in der Warteschlange (wartend und aktiv)
- D: Bezeichnet die Bearbeitungsdauer für einen Auftrag
- $T(\lambda)$: Bezeichnet den Durchsatz an Aufträgen durch die Warteschlange pro Zeiteinheit

- R(λ): Bezeichnet die durchschnittliche Verweildauer eines Auftrags in der Warteschlange. Kann auch als Antwortzeit bezeichnet werden, da in dieser Zeitspanne sowohl Wartezeit als Ausführungszeit integriert ist.

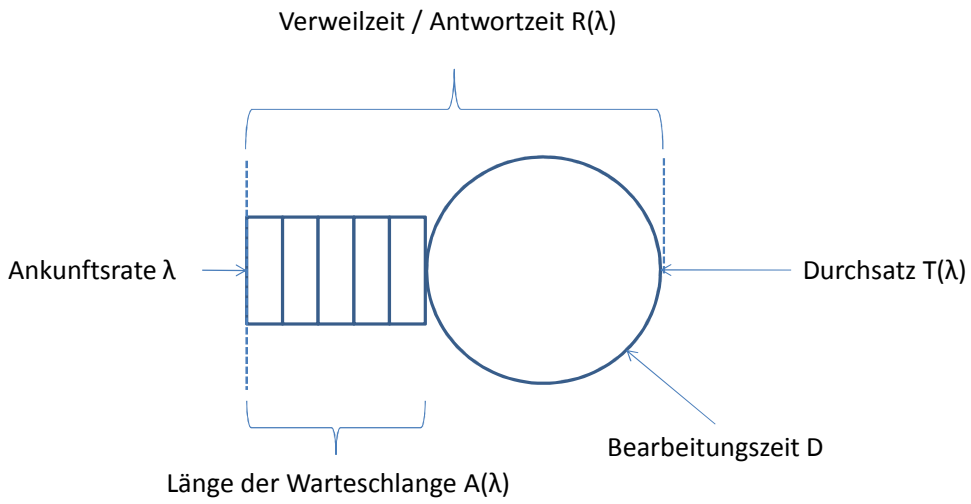


Abbildung 2-4: Grundelemente einer Warteschlange mit Parametern
Quelle: In Anlehnung an (Gross 2008)

Eine weitere wichtige Komponente der Service Center ist der Mechanismus, Scheduling-Strategie genannt, mit dem ankommende Anfragen aus der Warteschlange entnommen und einem Bearbeiter übergeben werden. Hierbei unterscheidet man zwischen:

- First-Come-First-Serve (FCFS):
  FDFS bezeichnet eine Strategie, bei welcher der Warteschlange dasjenige Element entnommen wird, welches bereits am längsten in der Warteschlange verweilt.

- Last-Come-Last-Serve (LCLS):
  LCLS stellt den entgegengesetzten Algorithmus dar. Hier wird dasjenige Element als erstes entnommen, welches die kürzeste Verweildauer in der Warteschlange aufweist.

- Service-In-Random-Order (SIRO)
  SIRO steht für die zufällige Auswahl einer Anfrage aus der Warteschlange, unabhängig von der Wartezeit in der Schlange.

- Round-Robin (RR):
  RR bezeichnet eine FCFS Strategie, bei der die Bearbeitung nach einer für alle Anfragen gleich langen Zeitspanne abgebrochen und die Anfrage wieder am Ende der Warteschlange eingereiht wird.

- Processor Sharing (PS):
  PS entspricht einem Round-Robin Verfahren mit unendlich kleinen Zeitscheiben, sodass es einer gleichzeitigen Bearbeitung aller ankommenden Anfragen gleichkommt.

Neben den vielfältigen Scheduling-Strategien wird zwischen drei unterschiedlichen Arten von Service Centern unterschieden. Der erste Typ ist das, bereits in Abbildung 2-4 dargestellte,

elementare Warteschlangenmodell mit einem Bearbeiter. Um eine größere Anzahl von parallelen Bearbeitungsvorgängen modellieren zu können, existiert die Möglichkeit, ein Service Center mit einer beliebigen Anzahl von parallelen Bearbeitern zu modellieren. Verzögerungen im Modell werden durch sogenannte Delay Center abgebildet, welche die Ausführung um eine gewisse Zeitspanne verzögern sollen. Damit die Verzögerung jedoch genau der gewünschten Spanne entspricht, hat diese Art von Service Center eine unendliche Anzahl von Bearbeitern, sodass jede Anfrage sofort bedient werden kann, und zu der gewünschten Verzögerung keine zusätzliche Wartezeit in der Queue hinzukommt. Die Notation dieser verschiedenen Typen von Warteschlangen zeigt Abbildung 2-5.

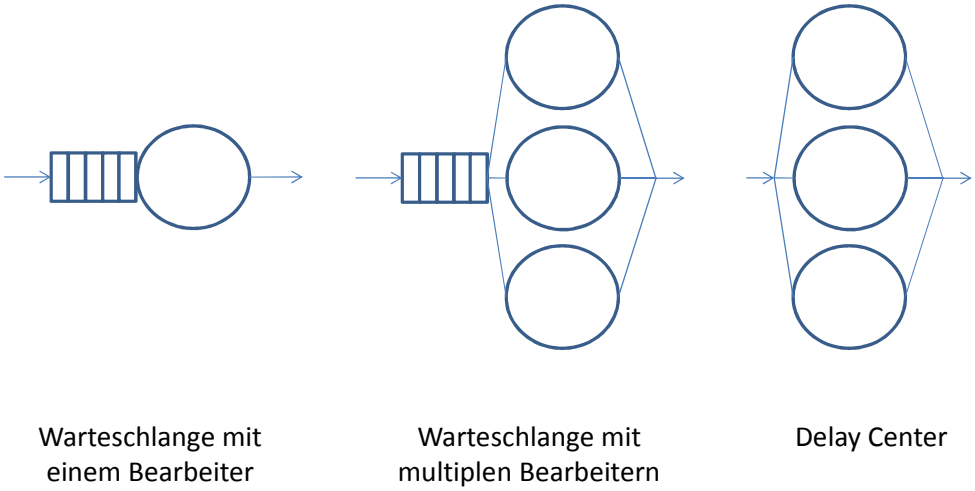


Abbildung 2-5: Verschiedene Typen von Warteschlangen
Quelle: In Anlehnung an (Gross 2008)

Für die Beschreibung elementarer Warteschlangenmodelle wird häufig die Kendall-Notation benutzt:

$$A / B / m [/ c] [/ p] [/ D]$$

- A: statistische Verteilung der Zeiten zwischen zwei aufeinanderfolgenden Ankünften
- B: statistische Verteilung der Servicezeiten
- m: Anzahl identischer Bearbeiter
- c: optionaler Parameter für die maximale Kapazität der Warteschlange
- p: optionaler Parameter für die Population, also die maximale Anzahl von Ankünften
- D: optionaler Parameter für die Scheduling-Strategie

Werden die optionalen Parameter nicht spezifiziert, so nehmen „c“ und „p“ den Wert „∞„ an. Die standardmäßig verwendete Scheduling-Strategie ist „FCFS“. Während es sich bei „m“, „c“ und „p“ um natürliche Zahlen handelt, müssen für „A“ und „B“ statistische Verteilungen angegeben werden. Dazu werden die englischen Abkürzungen verwendet:

- M: Exponentialverteilung (Markovian Distribution)

- D: Konstante (Deterministic Distribution)
- H: Hyperexponentialverteilung
- Ek: Erlangverteilung
- PH: Phasenverteiltung
- G: Beliebige Verteilung (General Distribution)

Im Rahmen dieser Arbeit wird die Exponentialverteilung für Modellierung Ankunftsrate (indirekt über die Bedenkzeit) und eine konstante Bearbeitungszeit verwendet.

2.4.1.2. Evaluation von Warteschlangen

Für die Evaluation von Warteschlangen bezüglich Durchsatz, Verweilzeit und Länge der Warteschlange existiert ein Theorem, das nach seinem Erfinder John D. C. Little (Little 1961) benannt ist. Hierbei werden die folgenden Notationen verwendet:

- λ: Ankunftsrate im System
- D: Bedienzeit
- R(λ): Verweilzeit bei Ankunftsrate λ
- U(λ): Auslastung des Service Centers bei Ankunftsrate λ
- T(λ): Durchsatz des Service Centers bei Ankunftsrate λ
- A(λ): durchschnittliche Anzahl von Anfragen in der Warteschlange bei Ankunftsrate λ
- s: Anzahl identischer Bediener eines Service Centers

Little's Law (Little 1961) zeigt die grundsätzliche Abhängigkeit von durchschnittlicher Anzahl von Anfragen und durchschnittlicher Verweilzeit in einem Service Center von der Ankunftsrate λ. Für den maximalen Durchsatz eines Warteschlangensystems gilt:

$$T_{sat} \leq \frac{s}{D}$$

Definiert man die Auslastung eines Service Centers als den Anteil der Zeit, in welcher der Bediener ausgelastet war, so ergibt sich für die Auslastung:

$$U(\lambda) = \frac{D \times \lambda}{s}$$

Die Verweilzeit in einem Service Center mit einem Bediener kann mit folgender Formel berechnet werden:

$$R(\lambda) = \frac{D}{1 - U(\lambda)} = \frac{D}{1 - (D \times \lambda)}$$

Für die Berechnung der Verweilzeit in Service Centern mit multiplen, identischen Bedienern wurde die Formel erweitert. Da die Verweilzeit für den Fall einer größeren Anzahl von Bedienern als Anfragen im System null ist, wurde bei dieser Formel eine Fallunterscheidung eingefügt:

$$R(\lambda) = \frac{D \times \left(1 + A(\lambda)\right)}{min(1 + A(\lambda), s)}$$

2.4.2. Layered Queueing Networks (LQN)

Der Einsatzbereich herkömmlicher Warteschlangennetze bestand in der Nachbildung von Hardware bzw. Computersystemen. Mit dem Konzept der Rendezvous wurden die Warteschlangennetze von Woodside et al. (2003) zu Stochastischen Rendezvous Netzwerken (SRVN) erweitert und ermöglichen damit die Modellierung von Software-System-Komponenten wie Remote Procedure Calls. Die Kombination von SRVN mit dem Konzept des „layered solving“ von Warteschlangennetzwerken (Rolia/Servcik 1995) führte zum Konzept der LQN (Woodside et al. 1995), (Franks et al. 1996b). So können mehrere herkömmliche Warteschlangennetze, zu komplexen azyklischen Graphen kombiniert, gelöst und simuliert werden.

2.4.2.1. LQN-Konzept

Die Kombination von Warteschlangennetzen für die Modellierung von Software-Systemen erlaubt es Software-Servern, im Modell „Task“ genannt, sowohl die Rolle eines Clients als auch die eines Servers einzunehmen. Ein Client befindet sich im LQN immer auf einer höheren Ebene als der Server. Demzufolge besitzen Tasks auf der obersten Ebene nur die Rolle eines Clients und Tasks auf der untersten Ebene die Rolle eines Servers. Verbindungen zwischen Tasks auf der gleichen Ebene sind dabei nicht möglich. Eine detaillierte Erläuterung zu den Komponenten des LQN Modells gibt Woodside (2002).

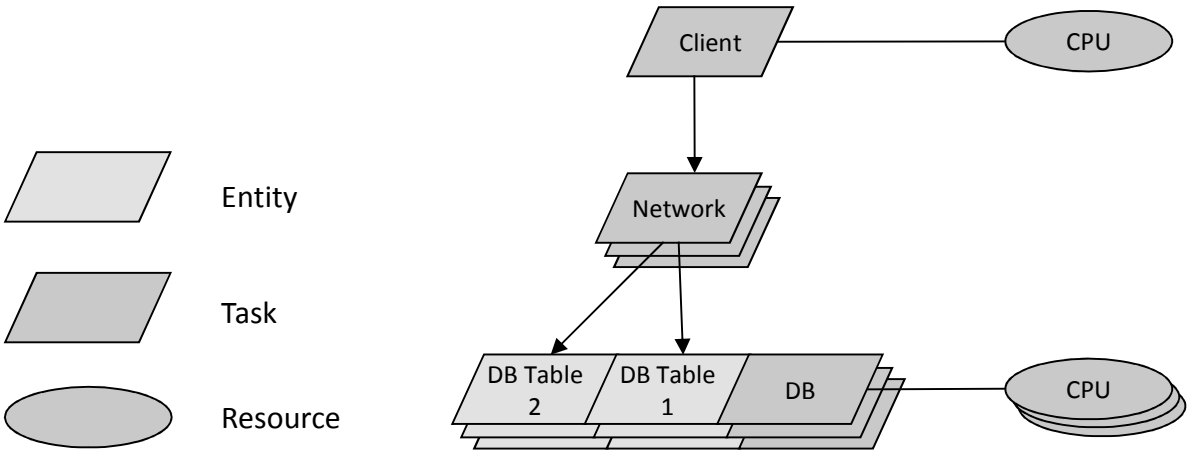


Abbildung 2-6: Beispiel eines LQN Modells
Quelle: Eigene Darstellung

Die Hauptkomponente bildet der sogenannte „Task“. Ein Task stellt entweder eine Hardware-Ressource, wie beispielsweise ein CPU, oder eine Software-Einheit dar. Ein Task ist mit einer

herkömmlichen Warteschlangenkomponente vergleichbar, da er eine eigene unendliche Warteschlange beinhaltet, um ankommende Anfragen bis zu deren Bearbeitung aufzunehmen. Im Falle von Software-Einheiten werden diese vom Bearbeiter, welcher ebenfalls Bestandteil eines Tasks ist, abgearbeitet. Sowohl ein Software- als auch ein Hardware-Server kann einzeln oder in größerer bis unendlicher Anzahl modelliert werden, analog zur Anzahl von parallelen Anfragen, welche dieser verarbeiten kann. Im Falle einer Software-Komponente kann dies ein Objekt darstellen, welches mehrmals instanziiert wurde. Auf Hardwareseite werden Multiserver oft verwendet. um mehrere CPUs oder eine CPU mit mehreren Cores zu modellieren. Abbildung 2-6 zeigt die LQN-Notation anhand eines vereinfachten Beispiels einer Netzwerkkommunikation. Hierbei sind sowohl Multi- als auch Single-Server für Hardware- und Software-Einheiten vertreten. Um die Modellierung von verschiedenen Ausführungszeiten oder Anfrageverhalten zu unterstützen, kann ein Task auch mehrere unterschiedliche Bearbeiter, im LQN „Services“ genannt, beinhalten. Abbildung 2-6 zeigt für die Notation am Beispiel der Services „DB Table 1“ und „DB Table 2“ des „DB“-Tasks. Ebenso werden die 3 verschiedenen Typen von Tasks, reine Client-Task, reine Server-Tasks und sogenannte „Network“-Tasks. Für die effiziente Modellierung von Semaphoren, wurde, wie in (Franks 2011) beschrieben, ein „Semaphore“-Task eingeführt. Im Gegensatz zu einem Standard-Task besitzt dieser neue Typ einen Zustand. Dies wird durch 2 Entrys („signal“ und „wait“) umgesetzt. Bei einem Aufruf des Wait-Entrys wird der Semaphor gesperrt. Diese Sperre wird erst wieder durch eine Anfrage an den Signal-Entry gelöst. Damit ist es möglich, einfache, sogenannte Binärsemaphore innerhalb eines LQN-Modells zu verwenden. Für Modellierung der Servicezeiten spielt die Art einer Anfrage an einen Server eine wichtige Rolle. LQN unterstützt 3 verschiedene Arten von Anfragen, in LQN „calls“ genannt“ an einen Server. Die Implikationen der jeweiligen Anfragetypen (synchron, asynchron und weitergeleitet) werden im Folgenden kurz dargestellt.

2.4.2.2. Synchrone Aufrufe

Schickt ein Client einen synchronen Aufruf an einen Server, so wartet der Client auf die Antwort des Servers. Während dieser Wartezeit ist der Client blockiert.

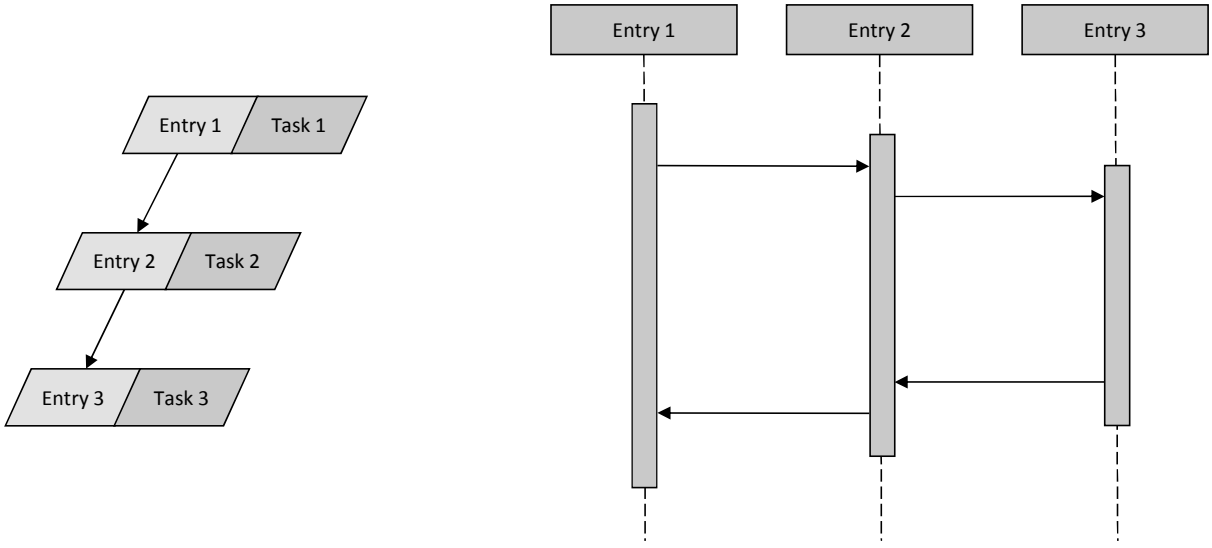


Abbildung 2-7: Synchroner Aufruf
Quelle: Eigene Darstellung

Damit bildet sich die gesamte Bearbeitungszeit aus der Summe der Bearbeitungszeiten von Client und Server. Abbildung 2-7 zeigt auf der linken Seite die Notation dieser Aufrufe. Die rechte Seite stellt die Auswirkungen auf die Ausführungszeit der aufrufenden Komponente „Entry 1“ dar. Ein Beispiel für diese Art von Aufrufen sind Datenbankanfragen aus einem Programm heraus. Hier wird die Anfrage an die Datenbank gestellt, und das Programm wartet, bis diese Anfrage beantwortet ist, bevor es die Bearbeitung weiterführt.

2.4.2.3. Asynchrone Aufrufe

Im Gegensatz zu synchronen Aufrufen wartet der Client nicht auf die Antwort des Servers, sondern setzt die Bearbeitung, wie in Abbildung 2-8 gezeigt, ohne Unterbrechung fort. Es gibt dabei keine Validierung einer erfolgreichen Bearbeitung der Anfrage durch den Server. Das mit dieser Aufrufart modellierte Verhalten ähnelt einem „fork“-Aufruf in Unix-Betriebssystemen. Ein weiteres Beispiel für derartige Aufrufe sind parallel angestoßene Java Threads.

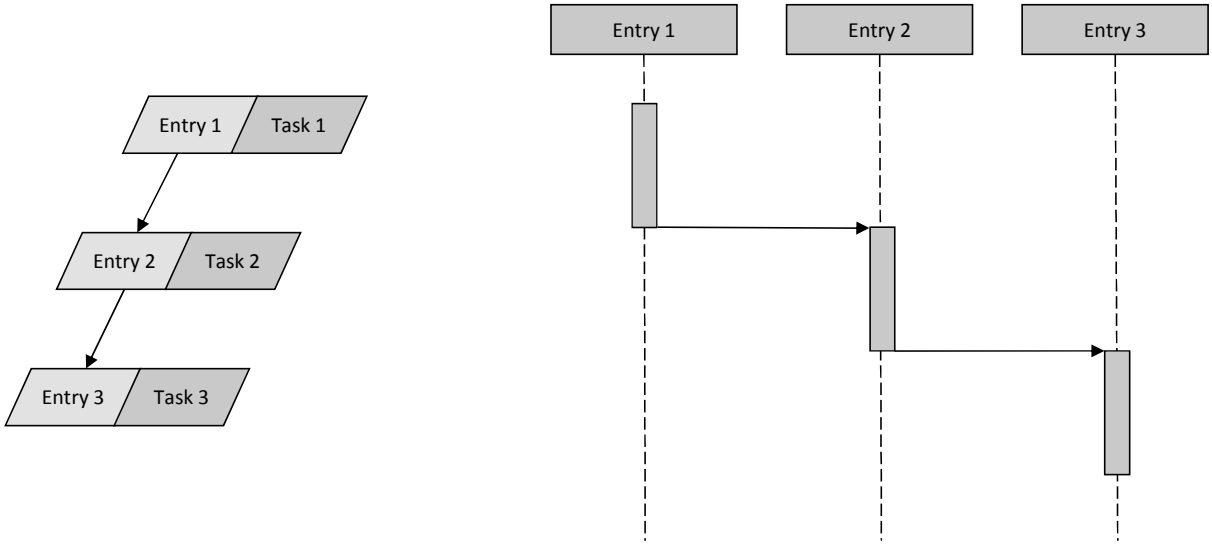


Abbildung 2-8: Asynchroner Aufruf
Quelle: Eigene Darstellung

2.4.2.4. Weitergeleiteter Aufruf

Der dritte Typ von Aufrufen, die sogenannten „forwarded calls“, wird in Abbildung 2-9 dargestellt. „Entry 2“ leitet den von „Entry 1“ empfangenen Aufruf nach Bearbeitung durch seinen Service an eine weitere Komponente „Entry 3“ weiter. „Entry 3“ gibt die Antwort dann an direkt an „Entry 1“ zurück. Die weiterleitende Komponente, im Beispiel „Entry 2“, ist damit ab dem Zeitpunkt der Weiterleitung nicht mehr belegt und steht Anfragen anderer Clients zur Verfügung. Diese Art der Weiterleitung kann z.B. für Warteschlangen verwendet werden (Risse 2006).

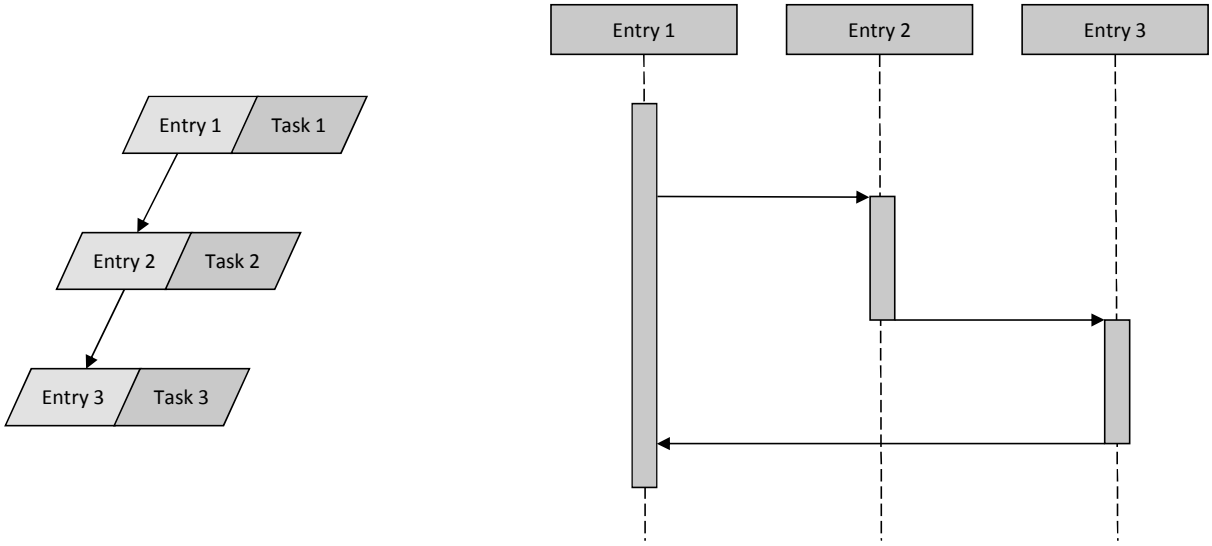


Abbildung 2-9: Weitergeleitete Aufrufe
Quelle: Eigene Darstellung

2.4.2.5. LQN Transformationen

Wie bereits erwähnt, dürfen LQN Modelle keine Zyklen enthalten. Dies widerspricht jedoch oftmals der Realität in zu modellierenden Systemen. So gibt es Software-Pattern, die eine Komponente aufrufen, welche dann ihrerseits die Ausführung durch einen Methodenaufruf der aufrufenden Komponente bestätigt. Um dies zu umgehen, hat Shousha et al. (1998) eine Regel für die Überführung von zyklischen Modellen in valide LQN Modelle eingeführt, siehe Abbildung 2-10.

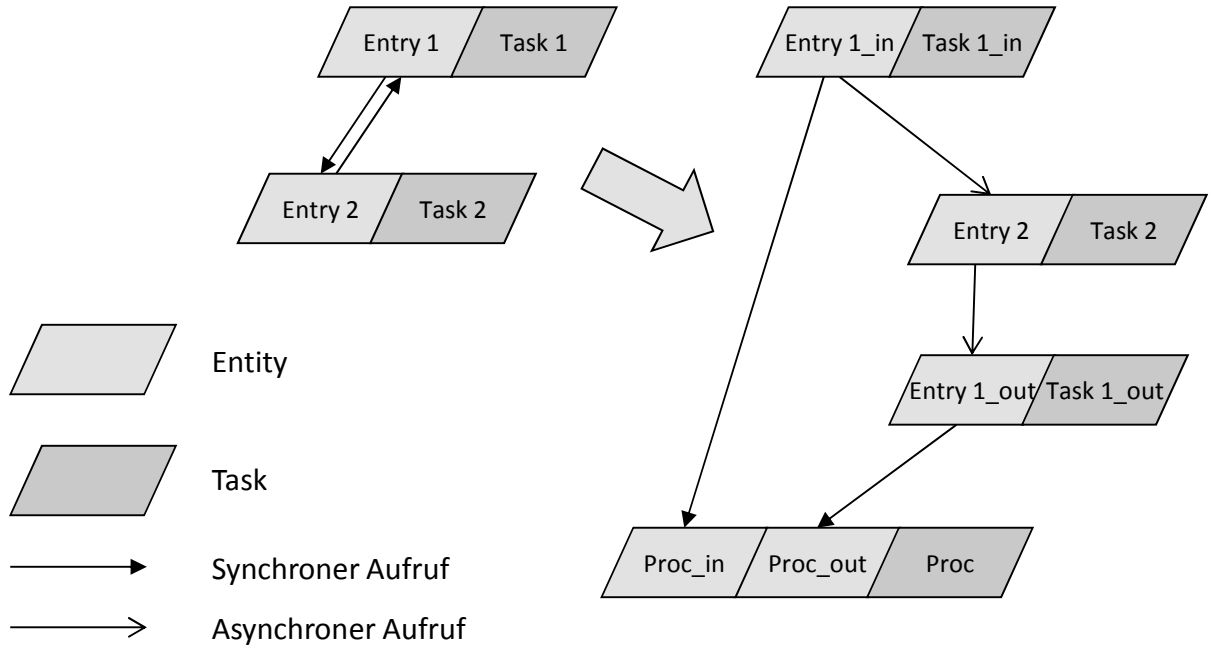


Abbildung 2-10: Transformation eines Zyklus´
Quelle: In Anlehnung an (Shousha et al. 1998)

Grundlegend für diese Transformation ist das Konzept von Semaphoren. Diese werden in LQN durch Tasks mit einfacher Multiplizität und Servicezeit gleich Null modelliert. Diese Semaphore geben die Anfrage an das geschützte Objekt mittels synchroner Aufrufe weiter.

Durch den synchronen Aufruf in Kombination mit einer Multiplizität von 1 bleibt der Semaphor für die Dauer der Bearbeitung blockiert, und es kann kein Zugriff auf die geschützte Komponente durch einen anderen Task oder Entry erfolgen.

Mittels Semaphoren kann der Zugriff auf einen Task als exklusiver Zugriff modelliert werden. Wie bereits erwähnt, ist es auch für diese Transformation nötig, die Modellierung der Semaphore mit synchronen Aufrufen durchzuführen.

Generell gibt es offene und geschlossene Warteschlangennetze. Offene Warteschlangennetze zeichnen sich durch eine oder mehrere Warteschlangen aus, die Anfragen von „außen“ mit einer Ankunftsrate λ bekommen. Der LQN-Formalismus schließt diese offenen Warteschlangennetze jedoch aus. Nach Shousha et al. (1998) existiert eine Transformation, siehe Abbildung 2-11, um offene Warteschlangennetze in geschlossene umzuwandeln.

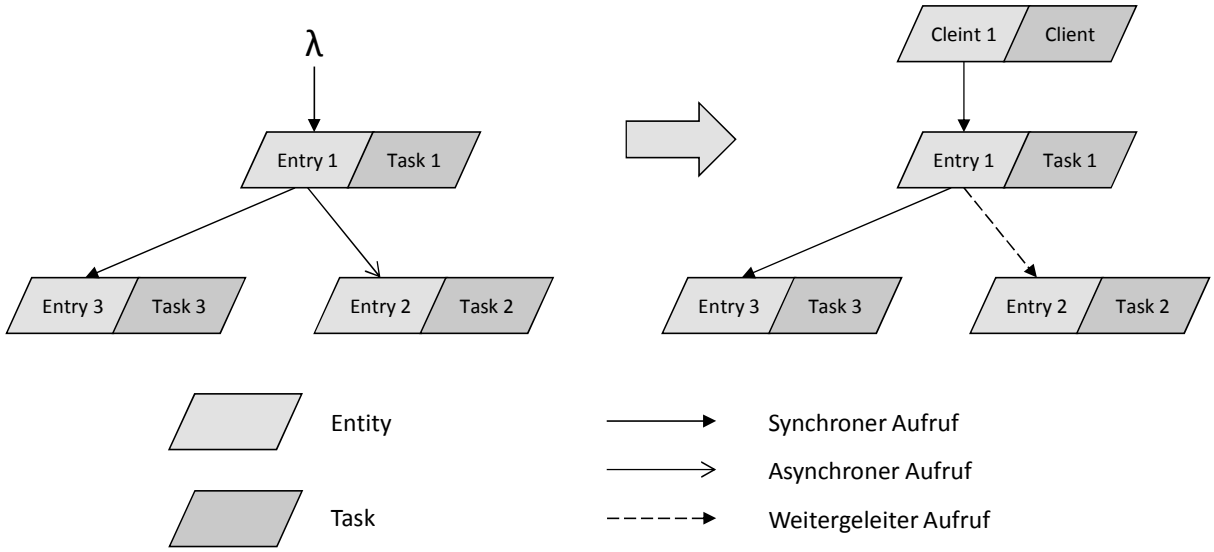


Abbildung 2-11: Transformation eines "open arrival"
Quelle: In Anlehnung an (Shousha et al. 1998)

Diese Umwandlung geschieht durch Einfügen eines Tasks, der die externe Auftragsquelle modelliert. Hierbei ist zu beachten, dass dieser Task eine sehr geringe Servicezeit haben sollte, damit die Modellierung der Performance nicht verfälscht wird. So wird eine Sättigung des Systems erreicht und gleichzeitig eine Überlastung vermieden. Dafür ist es nötig, die asynchronen Aufrufe durch weitergeleitete Aufrufe zu ersetzen, da somit der Entry „Client 1“ und damit der Task „Client“ erst wieder freie Ressourcen haben, wenn „Entry 2“ die ihm weitergeleitete Anfrage abgearbeitet hat. Da bei synchronen und weitergeleiteten Anfragen die Absender auf die Beantwortung dieser warten, müssen diese beiden Arten von Anfragen nicht transformiert werden.

2.4.2.6. Layered Queueing Network Solver (lqns)

Für die Lösung und die Simulation von LQN-Modellen existieren einige Werkzeuge, welche von der „Real-Time and Distributed Systems Group“ (Group 2011) der Carleton University, Ottawa, Kanada entwickelt wurden. LQN Modelle können sowohl mittels eines Simulators als auch mittels eines mathematischen Approximationsverfahrens evaluiert werden. Grundsätzlich unterstützen beide Werkzeuge Eingabedateien der gleichen

Beschreibungssprache, welche sowohl die Struktur des Modells als auch die Parameter für die Evaluation enthält. Diese Werkzeuge werden im Folgenden kurz erläutert.

Der LQNS löst LQN Modelle mit mathematischen Approximationsverfahren. Dabei unterstützt er verschiedene Methoden, wie „Mean Value Analysis“ (Petriu 1994) und „The Method of Layers“ (Rolia/Servcik 1995). Die Verwendung von analytischen Verfahren führt auf Kosten etwas höherer Abweichungen zu deutlich kürzeren Zeiten für die Evaluation eines Modells, als der im folgenden Absatz vorgestellte Simulator benötigt. Er unterstützt LQN Modelle mit folgender Ausprägung:

- Phases: 4
- Scheduling: FIFO, HOL, PPR
- Open Arrivals: yes
- Phase Type: stochastic, deterministic
- Think Time: yes
- Coefficient of variation: yes
- Interprocessor delay: yes
- Asyncrhonous Connections: yes
- Forwarding: yes
- Multi-Servers: yes
- Infinite-Servers: yes
- Max Entries: 1000
- Max Tasks: 1000
- Max Processors: 1000
- Max Entries per Task: 1000

2.4.2.7. Layered Queueing Network Simulator (lqsim)

Der „lqsim“ hingegen baut auf dem Parasol-Simulator (Mascarenhas/Knop/Rego 1995) auf. Durch die Verwendung von Simulationstechnik ist die Evaluierung von LQN Modellen durch das lqsim Tool genauer als die analytische Lösung mit Hilfe des LQNS. Darüber hinaus sollte er auch verwendet werden, wenn das „LQNS Wiederholungsschema nicht konvergiert“ (http://www.sce.carleton.ca/rads/lqns/lqn-documentation/). Er unterstützt LQN-Modelle mit folgenden Eigenschaften:

- Phases: 4

- Scheduling: FIFO, HOL, PPR, RAND
- Open Arrivals: yes
- Phase Type: stochastic, deterministic
- Think Time: yes
- Coefficient of variation: yes
- Interprocessor delay: yes
- Asynchronous Connections: yes
- Forwarding: yes
- Multi-Servers: yes
- Infinite-Servers: yes
- Max Entries: unlimited
- Max Tasks: 1000
- Max Processors: 1000
- Max Entries per Task: unlimited

Sowohl die Eingabedatei des LQNS als auch des lqsim-Tools enthält das vollständige Modell zusammen mit der Parametrisierung. Für ein Experiment ist es oft nötig, mehrere Konfigurationen eines Systems und damit eines Modells durchzuführen. Dies erfordert die Erstellung und Verwaltung einer separaten Ein- und Ausgabedatei für jeden Evaluationslauf. Dieser Vorgang wird durch das Tool MultiSRVN und dessen PERL-basiertem Nachfolger „Software Performance Experimenter“ (Hubbard 1997) unterstützt. In Kombination mit den Evaluationstools bietet SPEX die Möglichkeit, deren Eingabedateien für die verschiedenen Evaluationsläufe zu parametrisieren, diese durchzuführen und deren Ergebnisse zu verwalten und mittels csv-Exportfunktion in externen Programmen, wie Microsoft Excel, zu analysieren (Franks et al. 1996a).

2.4.3. Fazit

Die vorliegende Arbeit entscheidet sich für die Simulation als Methode für die Vorhersage bzw. Abschätzung der Performance von ERP-Systemen unter einer steigenden Anzahl von parallelen Benutzern. Diese Entscheidung basiert auf dem größeren Funktionsumfang des Simulators, sowie auf der relativ geringen Dauer einer Simulationsausführung des erstellten Modells. Aufgrund der Beschaffenheit des SAP-ERP-Systems mit einer in allen existierenden Systemen identischen Infrastruktur (Puffer, Enqueue, Dispatcher und Work-Prozesse) bietet dies die Möglichkeit, ein gültiges und damit kostengünstiges Vorgehen für die Erstellung und die Parametrisierung von Performance-Modellen zu entwerfen. Damit können die Kostennachteile der Messung umgangen und die Genauigkeit im Vergleich zu den

günstigeren analytischen Modellen gesteigert werden. Aufgrund der Auslegung auf die Beantwortung von Benutzeranfragen wird in der vorliegenden Arbeit die Antwortzeit als Maß für die Bewertung der Performance verwendet und damit Forschungsfrage 1 bzgl. der zu verwendeten Metrik beantwortet.

Da der LQN-Formalismus die Modellierung von hierarchischen und parallelen Aktivitäten (Rolia/Servcik 1995; Woodside 2002) unterstützt, welche in verschiedenen Servern mit gemeinsamen oder geteilten Warteschlangen repliziert sind (Omari et al. 2007), kann dieser sehr gut für die Modellierung der Systemarchitektur eines SAP-ERP-Systems, wie in Kapitel 3 verwendet werden. Da es ebenfalls möglich ist Submodelle mehrfach zu verwenden (Xu et al. 2005) ist dieser Formalismus sehr flexible im Hinblick auf Modellerweiterungen. Das in Computer-Systemen häufig zur Reduzierung der Antwortzeit verwendete Phasen Konzept beinhaltet die Ausführung von Tasks, die im Anschluss an die Beantwortung einer Benutzeranfrage durchgeführt werden müssen, jedoch nicht für die Antwort benötigt werden, in einer der Antwort nachgelagerten Phase. Damit wirkt sich diese zweite Phase nicht direkt auf die Antwortzeit aus, muss jedoch trotzdem berücksichtigt werden, da diese ebenso Systemressourcen benötigt.

3. Aufbau eines SAP-ERP-Systems

Im Bereich von Standardsoftware für ERP Anwendungen gibt es eine Vielzahl von Anbietern. Der Marktführer in diesem Bereich ist die Firma SAP mit dem Produkt SAP-ERP (Leimbach 2008). Aufgrund der Wichtigkeit dieser Anwendungen für die betrieblichen Prozesse von Unternehmen (Krcmar 2010) ist die Performance dieser Instanz von ERP-Systemen ein relevanter Untersuchungsgegenstand.

Um die Einflussgrößen auf die Performance von SAP-ERP-Systemen zu analysieren, wird in den folgenden Unterkapiteln nun deren Architektur, die Konfigurationsmöglichkeiten, die verwendeten Speicherarten, die Bearbeitung von Anfragen, die Puffermechanismen und die Bestandteile der Antwortzeit analysiert.

3.1. Architektur des SAP-ERP-Systems

Grundsätzlich gliedert sich die Architektur eines SAP-ERP-Systems, einer klassischen Dreischichtarchitektur (Gat 1998), in folgende Teile:

- Client
- Applikationsserver
- Datenbank

Hierbei stehen auf der Client- Ebene mehrere Möglichkeiten des Zugriffs zur Verfügung. Der wohl am häufigsten genutzte Client ist hierbei die SAP-Gui, welche die Präsentation der Daten und Programme für Windows-Plattformen bereitstellt. Plattformübergreifend gibt es analog die SAP-Java-Gui. Bei diesen beiden Programmen handelt es sich um Fat- Clients. Der weitverbreitete Web Browser als Thin-Client wird ebenso unterstützt. Die zugehörige Anwendung nennt sich SAP-WebGui. Eine weitere Möglichkeit auf Daten zuzugreifen wird über die RFC Schnittstelle bereitgestellt. Damit ist es möglich, aus selbstentwickelten Anwendungen über sogenannte remotefähige Funktionsbausteine auf die Daten im SAP System zuzugreifen. SAP liefert hierbei den SAP-JCO für die Anbindung von SAP-Systemen an JAVA basierte Software, um mittels eines „remote function calls“ (RFC) ABAP Programme aufzurufen. Remotefähige Funktionsbausteine können im SAP-WebAS-Java auch als WebService exportiert und damit für den Zugriff über das SOAP Protokoll freigegeben werden.

Auf der zweiten Schicht werden von SAP zwei unterschiedliche und voneinander unabhängige Applikationsserver bereitgestellt:

- SAP-WebAS-ABAP (ABAP-Stack)
- SAP-WebAS-Java (Java-Stack)

Die Lösungen der SAP basieren entweder auf einem der beiden Applikationsserver, oder auf beiden (Dual Stack). ABAP steht hierbei für die proprietäre, plattformübergreifende

Programmiersprache der SAP. Diese zweite Schicht stellt die Verarbeitungslogik und greift auf Daten zurück, welche in der Datenbankschicht abgelegt sind.

SAP unterstützt mehrere Hersteller von relationalen Datenbanksystemen, indem die Applikationsserver Datenbankzugriffe auf einer Abstraktionsschicht in Form der OpenSQL-Schnittstelle kapseln. Da diese Schnittstelle von der SAP entwickelt wurde und dauernd gepflegt wird, muss ein Datenbanksystem für die Verwendung innerhalb eines SAP-Systems von SAP zertifiziert sein.

Durch die längere Historie des ABAP Stacks für die von SAP bereitgestellten Anwendungen existieren für den ABAP- Stack ca. 83500 Programme, wohingegen der Anteil der in Java entwickelten Programme bei ca. 700 liegt (Boegelsack 2010). Durch die breitere Verwendung und damit größerer Wichtigkeit fokussiert sich diese Arbeit auf SAP-Systeme, die auf dem SAP-WebAS-ABAP basieren. Daher ist im Rest dieser Arbeit der Begriff SAP-System als Bezeichnung für SAP-Produkte, die auf dem SAP-WebAS-ABAP basieren, zu verstehen.

Der SAP-WebAS besteht aus betriebssystemabhängigen Komponenten, dem sogenannten SAP-Kernel, welche eine Laufzeitumgebung für die Kompilate der ABAP Programme bereitstellen (Schneider 2008). Daraus resultiert die Plattformunabhängigkeit der ABAP Programme, da diese in einem, für das Betriebssystem kompilierten, Kernel ausgeführt werden. Der SAP-Kernel besteht hierbei aus den folgenden Komponenten (kurz: DVEBMGS):

- **D**isp+work:
  Diese Prozesse, auch Dialog-Work-Prozesse genannt, sind für die Bearbeitung der Benutzeranfrage zuständig. Ein globaler Dispatcher-Prozess verteilt dabei die Anfragen auf die „disp+work“-Prozesse. Diese Prozesse sind für die Bearbeitung von Benutzeranfragen mit tendenziell geringerer Laufzeit zuständig. In produktiv genutzten Systemen gibt es keine feste Zuordnung von Benutzern zu Dialog-Work-Prozessen. Sobald eine Anfrage abgearbeitet ist, schickt der Work-Prozess die Anfrage an den Client zurück und wartet auf die Zuteilung einer weiteren Anfrage durch den Dispatcher.

- **V**erbucher:
  Da der SAP-Kernel ein eigenes Transaktionskonzept, über dem der Datenbank implementiert, werden Daten erst nach Abschluss einer SAP-Transaktion auf Geschäftsprozessebene durch den Verbucher in die Datenbank geschrieben.

- **E**nqueue:
  Um konkurrierenden Zugriff auf die Datenbank, genauer auf Bereiche einzelner Tabellen zu regeln, implementiert die Programmiersprache ABAP das Konstrukt von Enqueues. Dabei handelt es sich um logische Sperren, welche die Konsistenz der Daten gewährleisten. Der Enqueue- Prozess verwaltet diese Sperren in einer internen Sperrtabelle.

- **B**atch: Im Gegensatz zu den Dialog-Work-Prozessen verarbeiten Batch-Prozesse Programmaufrufe, welche eine längere Laufzeit aufweisen.

- **M**essage:
  Für die Kommunikation innerhalb der Server (Instanzen) eines SAP-Systems wird der Message-Server- Prozess, kurz Message-Server, verwendet. Darüber hinaus ist er auch zuständig für die Verteilung der Informationen bzgl. der Auslastung der einzelnen Instanzen, welche die Lastverteilung in einem SAP-System bilden.

- **G**ateway:
  Der Gateway-Prozess implementiert die Schnittstelle für Services, welche TCP/IP basieren, und wird damit für die Verbindung zu externen Programmen, auch über die RFC Schnittstelle, verwendet.

- **S**pool:
  Dieser Prozess steuert und verwaltet die Druckaufträge innerhalb eines SAP-Systems.

Das SAP-System besteht insgesamt aus 3 verschiedenen Typen von Instanzen. Gleichzusetzen mit der Datenbankschicht ergibt sich die Datenbankinstanz, welche in jedem SAP-System existieren muss. Ähnlich verhält es sich mit der Zentralinstanz. Diese besteht aus den eben genannten Prozessen und bildet den Kern eines SAP-Systems. Um die Skalierbarkeit für eine große Anzahl von Benutzern zu gewährleisten, bietet SAP die Möglichkeit der Installation von Dialoginstanzen. Abbildung 3-1 zeigt den Aufbau von Zentral- und Dialog-Instanzen. Hierbei besteht der Hauptunterschied darin, dass der Message-Server nur in der Zentralinstanz vorkommt.

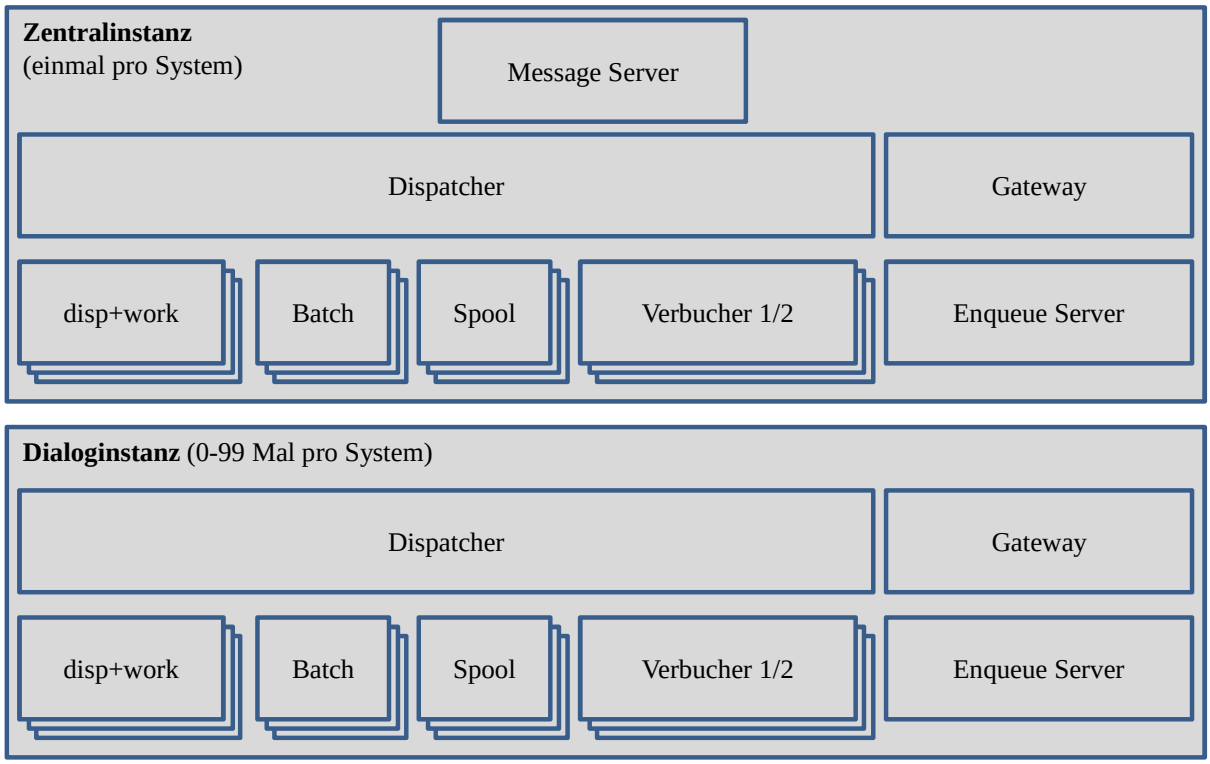


Abbildung 3-1: Aufbau der SAP-Zentral- und -Dialog-Instanz
Quelle: In Anlehnung an (SAP 2011k)

Auch bei der Implementierung bildet der Message Server eine Ausnahme. Zusammen mit dem Gateway-Prozess werden diese beiden Prozesse in einer eigenen Binary ausgeliefert. Die

restlichen Prozesse dagegen sind alle innerhalb einer einzigen Binary (disp+work) implementiert. Beim Start einer SAP-Instanz wird zuerst der Dispatcher gestartet, welcher dann, anhand der technischen Konfiguration des Systems, die restlichen SAP-Prozesse startet.

3.2. Konfiguration des SAP-WebAS-ABAP

Diese technische Konfiguration (im Gegensatz zum inhaltlichen Customizing) eines SAP-ERP-Systems kann im System verändert werden und ist in der Datenbank als sogenanntes Profil gespeichert. Da die Informationen bereits beim Start des SAP-Systems benötigt werden, muss das Profil in Form von Profildateien auf Dateisystemebene verfügbar sein. Damit ist es nötig, im System erstellte Modifikationen des Profils durch die sogenannte Aktivierung mit der Version des Profils im Dateisystem zu synchronisieren. Es enthält eine Vielzahl von Konfigurationsmöglichkeiten. Im Kontext dieser Arbeit sind dabei vor allem folgende Einstellungen wichtig:

- Anzahl der jeweiligen Systemprozesse (DVEBMGS)
- Puffergröße in KB
- Puffergröße in Anzahl Entrys
- Größe Shared Memory Segmente
- Schachtelung von Shared Memory Segmenten

Mittels dieser Einstellungsmöglichkeiten wird ein SAP-ERP-System entsprechend den in dieser Arbeit für die Evaluation des Simulationsmodells verwendeten Szenarien konfiguriert.

3.3. Verwendung verschiedener Speicherarten im SAP-WebAS-ABAP

Neben der bereits erläuterten Funktion der einzelnen Prozesse sind auch die Größe und Anordnung von Shared Memory Segmenten von großer Bedeutung für den Betrieb und damit die Performance von SAP-Systemen. Die Prozesse benutzen das Shared Memory in Kombination mit Semaphoren, siehe Abbildung 3-2, für die Zusammenarbeit innerhalb des Systems.

Wie bereits erwähnt, gibt es keine feste Zuordnung von Benutzer zu Work-Prozess. Damit muss ein Work-Prozess auf den Kontext jedes am System angemeldeten Benutzers zugreifen können. Zu diesem Kontext gehören Variablen, interne Tabellen und Bildschirmlisten (Schneider 2008). Übernimmt ein Work-Prozess eine Benutzeranfrage, so wird zuerst der Benutzerkontext aus dem Roll-Puffer im Shared Memory in das lokale Memory des Work-Prozesses, genauer die Roll Area, geladen. Sollte diese nicht ausreichen, wird weiterer Speicher aus dem „extended memory“ allokiert und in Form von Verweisen im lokalen Speicher des Work-Prozesses für diesen verfügbar gemacht. Ist auch dieser nicht ausreichend, so wird im lokalen Speicher des Work-Prozesses „private memory“ allokiert. Ab diesem Zeitpunkt steht der Work-Prozess nicht mehr für Anfragen anderer Benutzer zur Verfügung und befindet sich im sogenannten „priviliged mode“. Der von diesem Prozess allokierte Speicher kann nur durch einen Neustart des Work-Prozesses wieder freigegeben werden. Ist auch die Größe des „private memory“ nicht ausreichend, so wird der Prozess terminiert,

neugestartet und eine Abbruchmeldung an den Benutzer zurückgegeben. Kann die Benutzeranfrage jedoch ohne Probleme verarbeitet werden, so wird der Benutzerkontext durch den Work-Prozess wieder in den Roll-Puffer zurückkopiert. Die Größe von „roll area“, „roll buffer“ und „extended memory“ ist ebenfalls im bereits erwähnten Profil einer Instanz konfigurierbar.

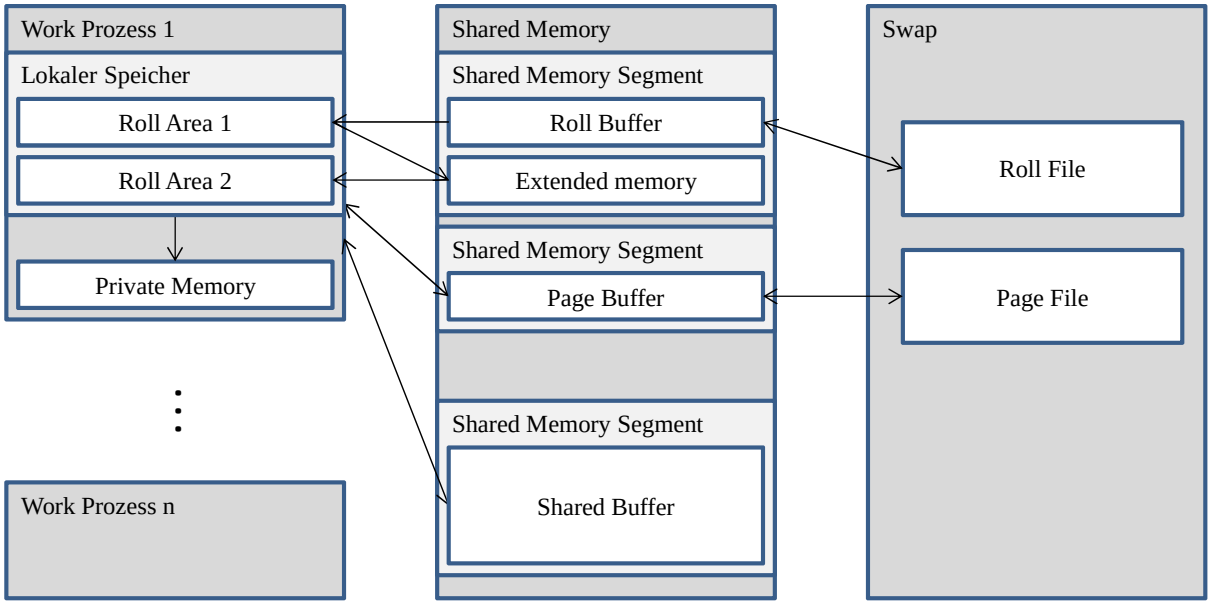


Abbildung 3-2: Speicherarten und deren Verwendung im SAP-WebAS-ABAP
Quelle: In Anlehnung an (SAP 2011a)

Generell benutzten alle Prozesse den vom Betriebssystem bereitgestellten virtuellen Speicher. Dieser besteht zum einen aus Random-Access-Memory (RAM), welcher sehr schnelle Zugriffszeiten leistet, jedoch aber sehr teuer und daher im System eine knappe Ressource ist. Zum anderen wird ein Teil des virtuellen Speichers in der Auslagerungsdatei (Swap) abgelegt, welche durch deutlich langsamere Zugriffzeiten gekennzeichnet ist, jedoch im Verhältnis zum RAM als günstig bezeichnet werden kann. Dieser virtuelle Speicher wird vom Betriebssystem lokal und global, d.h. gemeinsam nutzbar, verwendet. Dieser gemeinsam nutzbare Speicher wird auch „shared memory“ genannt, da er von mehreren Prozessen gemeinsam genutzt werden kann, wohingegen lokaler Speicher nur genau einem Prozess zur Verfügung steht. Der SAP-WebAS-ABAP nutzt diese beiden Arten von Speicher wie folgt (SAP 2011a):

- Lokaler Roll-Speicher:
  Dieser wird, wie bereits besprochen, für die initialen Daten eines Benutzerkontextes verwendet. Er wird beim Start des Work-Prozesses fest allokiert und kann nur von diesem Prozess verwendet werden. Die Umsetzung erfolgt direkt im Swap-Speicher des Betriebssystems.

- Globaler Roll-Speicher:
  Aus Gründen der verfügbaren Größe wird auch dieser Speicher im Swap-Bereich des virtuellen Speichers abgelegt. Dieser Roll-Puffer wird für die temporäre Speicherung der Benutzerkontexte verwendet, wenn diese nicht durch einen Work-Prozess verwendet werden.

- Privater Speicher:
  Dieser Speicher wird nur, wie bereits besprochen, verwendet, wenn sich der Work-Prozess im „privileged“-Modus befindet, und der Prozess für die Freigabe dieses Speichers gestoppt werden muss.

- Erweiterter Speicher:
  Dieser Speicher kann von den Work-Prozessen im SAP-System verwendet werden, wenn der lokale Speicher nicht mehr ausreicht. Eine weitere wichtige Verwendung des Shared Memory zeigt sich im Bereich der verschiedenen Puffer, welche die Abläufe im SAP-System beschleunigen.

Unterkapitel 3.5 zeigt eine Übersicht über die verfügbaren Puffer sowie deren Einschätzung für die Wichtigkeit im Rahmen dieser Arbeit.

3.4. Anfragebearbeitung im SAP-WebAS-ABAP

Transaktionen stellen in einem SAP-System einen Oberbegriff zu den Programmen mit einer n:m Beziehung dar. Programme können in mehreren Transaktionen verwendet werden, genauso wie eine Transaktion für gewöhnlich mehrere verschiedene Programme benutzt.

3.4.1. Bearbeitung von Transaktionen

Beim Aufruf einer Transaktion wird das festgelegte Programm gestartet. Abbildung 3-3 zeigt, dass dabei der Aufruf von der SAP-Gui aus den Dispatcher erreicht. Dieser speichert die aktuelle Auslastung der Work-Prozesse in einer Tabelle ab, welche sich in seinem eigenen Prozess-Speicher befindet. Mit Hilfe der Informationen über die Verfügbarkeit der Prozesse weist der Dispatcher die Anfrage einem freien Work-Prozess zu.

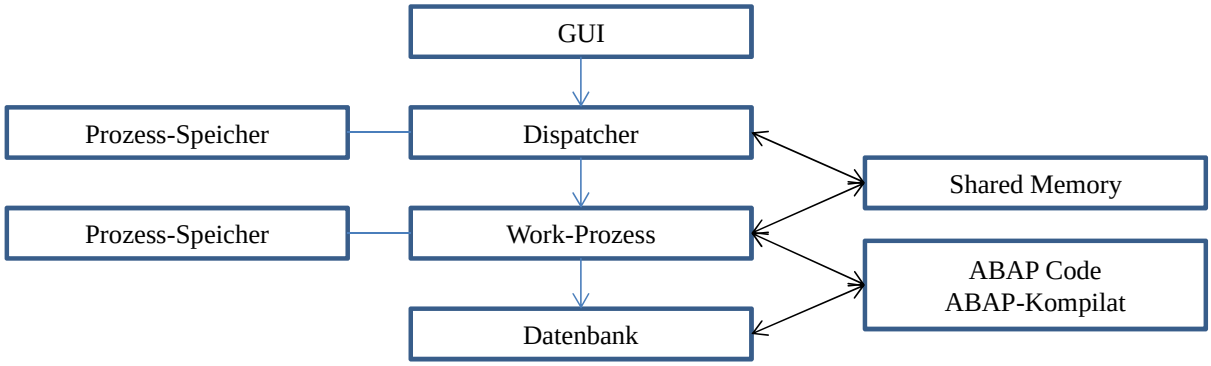


Abbildung 3-3: Ausführung eines ABAP Programms
Quelle: In Anlehnung an (Boegelsack 2010)

Dieser Work-Prozess liest nun den Benutzerkontext aus dem Shared Memory in seinen lokalen Prozessspeicher und lädt auch erforderliche Programm. Sollte es zum ersten Mal gestartet oder der Programmcode seit dem letzten Aufruf modifiziert worden sein, so wird es zuerst automatisch kompiliert und dann vom ABAP Interpreter des SAP-Kernels ausgeführt. Muss das Programm noch kompiliert werden, so wird der Quellcode aus der Datenbank geladen. War es bereits kompiliert und wurde das Programm seit dem Neustart des WebAS ABAP schon ausgeführt, so befindet es sich noch im Shared Memory, respektive in einem Puffer, und kann direkt geladen werden. Andernfalls wird es aus der Datenbank angefragt.

Wird das Programm geladen, so erhält der Dispatcher diese Information, um sie in der von ihm geführten FIFO Queue, welche sich im Shared Memory befindet, zu verwalten.

3.4.2. Wechselseitiger Zugriff von Programmen auf Tabellen

Wenn Daten in einem Datenbankobjekt innerhalb eines Programmes verändert werden sollen, so muss sichergestellt sein, dass dieser Datensatz nicht parallel durch einen anderen Work-Prozess gelesen wird. Ist dies nicht sichergestellt, so kann dies zu Daten-Inkonsistenz innerhalb des Systems führen. Um dies zu verhindern, gibt es im WebAS ABAP das Sperrkonzept. Dabei muss innerhalb eines Programms vor dem Zugriff auf ein Datenbankobjekt eine Sperre gesetzt werden. Damit diese Sperre global gültig ist, werden diese Sperren durch den sogenannten Enqueue-Server innerhalb einer Sperrtabelle, welche sich im Hauptspeicher befindet, verwaltet. Sperren werden von Work-Prozessen gesetzt, können an Update-Prozesse übergeben werden (3.4.2.1), und sind so lange gültig, bis der zu jeder Sperre gehörige „DEQUEUE“-Baustein oder ein allgemeiner „DEQUEUE_ALL“-Baustein aufgerufen wird. Wird während dieser Sperrzeit durch einen anderen Work-Prozess versucht, eine Sperre zu beantragen, so wird dies durch den Enqueue-Server verweigert, und der Work-Prozess wartet, bis die Sperre gelöscht wird. Dies kann starke, negative Auswirkungen auf die Performance eines Systems haben, wenn zentrale Objekte mit zu groben Sperren belegt werden und damit nicht für andere Programme zugreifbar sind.

3.4.2.1. Sperrtabelle

Wird eine Sperre von einem Work-Prozess beantragt, so überprüft der Enqueue-Server, ob das zu sperrende Datenbankobjekt bereits gesperrt ist, indem er die Sperrtabelle nach dem Datenbankobjekt durchsucht. Die Struktur der Einträge der Sperrtabelle ist in Abbildung 3-4 dargestellt.

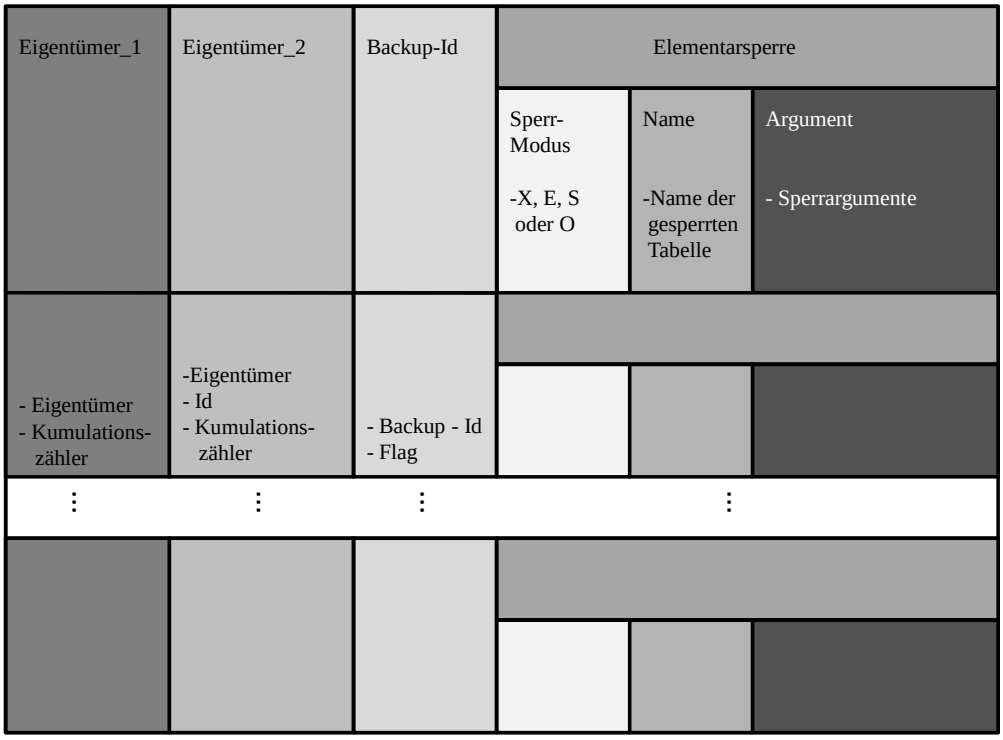


Abbildung 3-4: Struktur der Sperrtabelle
Quelle: In Anlehnung an (SAP 2011j)

Beim Start einer Transaktion oder eines Programms werden zwei Benutzer angelegt, welche Sperren anfordern dürfen. Wird nun eine Sperre gesetzt, so werden die Felder Eigentümer_1 und Eigentümer_2 mit den IDs dieser beiden Benutzer sowie einem Zähler, wie oft diese Sperre bereits durch diese Benutzer gesetzt wurde, belegt. Dabei handelt es sich nicht um einen globalen Zähler. Dieser Zähler existiert, da Sperren während einer Programmausführung kumuliert werden können. Genauer wird dies in 3.4.2.3 dargestellt. Um Daten in die Datenbank zu schreiben, gibt es im SAP-WebAS-ABAP eigene Verbucher-Prozesse, welche das Schreiben in die Datenbank, wie in Abbildung 3-5 dargestellt, asynchron vornehmen.

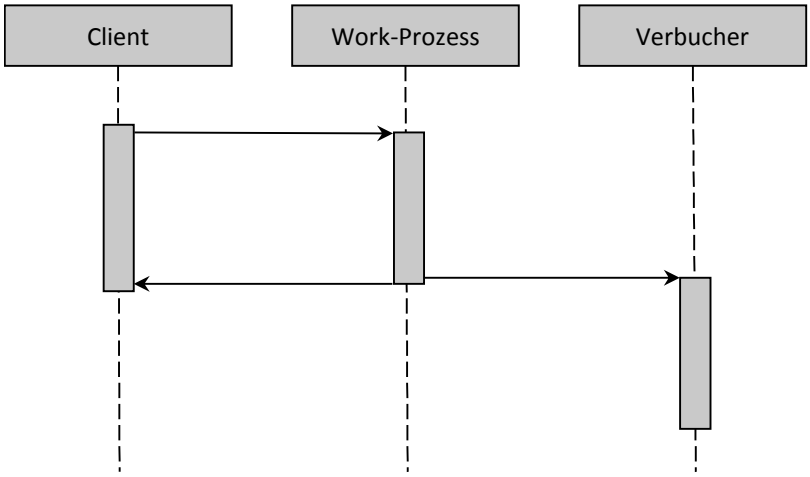


Abbildung 3-5: Aufruf eines Verbucher-Prozesses
Quelle: Eigene Darstellung

Dies geschieht u.a., damit die Antwortzeit an den Client minimiert wird. Für die Sperrverwaltung bedeutet dies jedoch, dass Sperren von einem Work-Prozess an den Verbucher übergeben werden müssen und diese Sperre auch im Falle eines zwischenzeitlichen Neustarts des Verbucher-Prozesses erhalten bleiben muss. Deshalb gibt es die Möglichkeit, beim Anfordern einer Sperre ein Backup-Flag zu setzen, welches dem Enqueue-Server signalisiert, diese Sperre persistent in eine Datei auf Betriebssystemebene zu schreiben. Dieses Backup-Flag wird zusammen mit einer ID in der dritten Spalte des Sperreintrags vermerkt. Um die Beeinflussung anderer Work-Prozesse durch Sperren zu minimieren, stellt der WebAS ABAP verschiedene Typen von Sperren bereit, welche in der vierten Spalte des Sperreintrags registriert sind. Die gesperrte Datenbanktabelle und das betreffende Feld des Primärschlüssels sind in den letzten beiden Spalten eingetragen. Dies ermöglicht eine feingranulare Sperrung einzelner Felder von Datensätzen, was wiederum die Auswirkung auf die Ausführung anderer Programme minimiert.

3.4.2.2. Typen von Sperren – Sperrmodi

Um die bereits angesprochenen wechselseitigen Sperren zu minimieren, gibt es verschiedene Sperrmodi, um die Ausdehnung der Sperre auf das gewünschte Objekt zu beschränken. Folgende Sperren sind im SAP-WebAS-ABAP vorgesehen (SAP 2011g):

- Lesesperre: S (Shared).
- Schreibsperre: E (Exclusive)

- Erweiterte Schreibsperre: X (eXclusive noncumulative)
- Optimistische Schreibsperre: O (Optimistic)

Lesesperren können von verschiedenen Benutzern nacheinander gesetzt werden und schließen sich nicht gegenseitig aus. Besteht jedoch eine Lesesperre auf einem Objekt, so werden Anforderungen von Schreibsperren oder erweiterten Schreibsperren abgewiesen, bis die Lesesperren entfernt worden sind. Damit sind auch Lesesperren mit großer Vorsicht zu verwenden, da sie Schreibvorgänge verzögern oder verhindern können.

Ist eine Schreibsperre durch einen Benutzer innerhalb eines Programms gesetzt, so kann dieses Objekt von keinem anderen Programm des gleichen Benutzers oder durch Programme anderer Benutzer gesetzt werden. Es besteht jedoch die Möglichkeit der Kumulation (siehe 3.4.2.3).

Im Gegensatz zu einfachen Schreibsperren können erweiterte Schreibsperren nicht kumuliert werden. Hat also ein Programm eine erweiterte Schreibsperre auf ein Objekt gesetzt und versucht nun erneut dieses Objekt mittels einer erweiterten Schreibsperre zu schützen, so wird diese Anfrage abgewiesen, und das Programm muss warten, bis die erste erweiterte Schreibsperre wieder entfernt worden ist.

Optimistische Sperren (SAP 2011d) entstanden aus den Problemen mit den vorher bestehenden pessimistische Sperren. Anwendungsfall ist die Darstellung von Daten im Änderungsmodus. Gibt es eine hohe Wahrscheinlichkeit für eine tatsächliche Änderung der Daten, so wäre die pessimistische Sperre geeigneter. Jedoch zeigte sich in der Praxis, dass Daten zwar häufig bereits im Änderungsmodus angezeigt werden, die Wahrscheinlichkeit für eine Änderung jedoch sehr gering ist. Aus diesem Grund wurde die optimistische Sperre eingeführt, welche sich wie eine Lesesperre verhält, jedoch bei Bedarf in eine Schreibsperre umgewandelt werden kann. Dabei schließen sich mehrere optimistische Sperren erst dann gegenseitig aus, wenn eine dieser Sperren in eine Lesesperre umgewandelt wird. Zu diesem Zeitpunkt erlöschen alle anderen optimistischen Sperren. Diese Art von Sperre kann nicht gesetzt werden, wenn bereits eine Schreibsperre besteht.

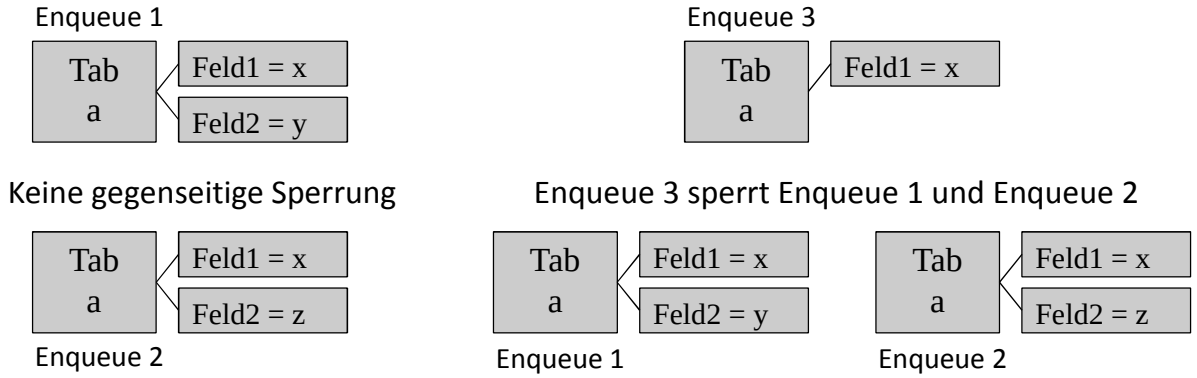


Abbildung 3-6: Wechselwirkungen von Enqueue-Vorgängen
Quelle: Eigene Darstellung

Für einen performanten Ablauf von ABAP-Programmen ist eine Analyse der verwendeten Sperren daher äußerst wichtig. Dabei ist es nicht ausreichend, die Enqueue-Objekte nur mit

dem Namen zu erfassen, da diese nicht einmalig für eine Datenbanktabelle erstellt werden, sondern durch den ABAP-Entwickler, welcher das Programm implementiert. Bei einer fundierten Performance-Analyse muss daher analysiert werden, welche Datenbanktabelle mit welchem Eigentümer und auf welche Art gesperrt wird, da verschiedene Sperrobjekte sich auf das gleiche Datenbankobjekt beziehen können. Diese Wechselwirkungen werden in Abbildung 3-6 dargestellt. Hier beziehen sich die Enqueue-Objekte „Enqueue 1“, „Enqueue 2“ und „Enqueue 3“ auf jeweils die gleiche Tabelle mit den gleichen Feldern „Feld1“ und „Feld2“. Aufgrund der unterschiedlichen Belegungen der beiden Felder sperren sich die Enqueues 1 und 2 nicht. Jedoch kann ein Enqueue-Objekt, wie in Abbildung 3-6 „Enqueue 3“, welches eine grob granularere Sperrung vornimmt, die beiden Enqueues 1 und 2 sperren.

3.4.2.3. Kumulation von Sperren

Wie bereits erwähnt, können Sperren kumuliert werden. Diese Kumulation ist abhängig vom Besitzer der Sperre. Wie in 3.4.2.1 dargestellt, hat eine Sperre zwei Eigentümer, einen Dialog-Eigentümer und einen Verbucher-Eigentümer. Beim Setzen einer Sperre ist es daher nötig, mittels des „_SCOPE“-Parameters anzuzeigen, für welchen Eigentümer diese Sperre gelten soll.

- _SCOPE = 1: Sperre bezieht sich nur auf den Dialog-Eigentümer
- _SCOPE = 2: Sperre gehört dem Verbuchungs-Eigentümer
- _SCOPE = 3: Sperre gehört beiden Eigentümern und wird erst freigegeben, wenn dies der letzte der beiden Eigentümer getan hat.

Die Anhäufung von Sperren über die verschiedenen Ausprägungen des „_SCOPE“-Parameters ist als Beispiel in Abbildung 3-7 dargestellt.

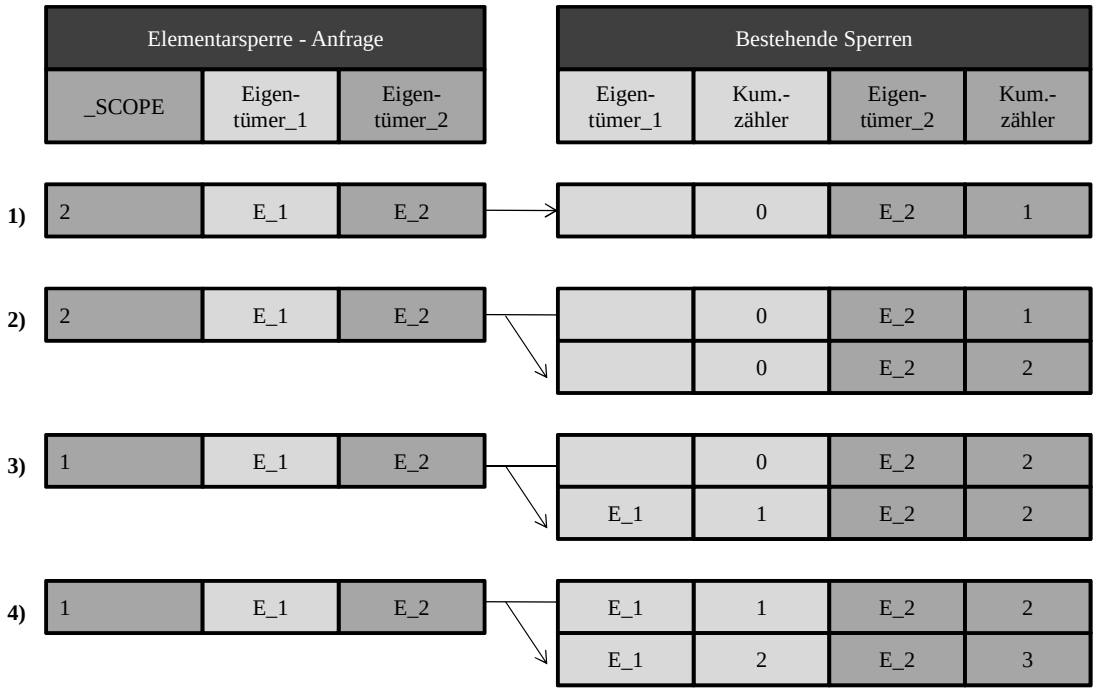


Abbildung 3-7: Kumulation von Sperren
Quelle: In Anlehnung an (SAP 2011c)

Hierbei handelt es sich um 4 Sperranfragen auf das gleiche Objekt der gleichen Eigentümer aus dem gleichen ABAP-Programm heraus. Dabei werden je nach Ausprägung des „_SCOPE“-Parameters die bereits erwähnten Kumulationszähler der beiden Eigentümer inkrementiert. Die Sperre wird entfernt, wenn der letzte Kumulationszähler auf null gestellt wird.

3.4.3. Scheduling von Programmen auf Work-Prozesse

Die Queue des Dispatcher-Prozesses (siehe 3.4.1) enthält Informationen über alle Programme, die sich gerade in Ausführung befinden, und jene, die noch zu prozessieren sind und aufgrund von einer zu geringen Anzahl von Work-Prozessen noch nicht verarbeitet werden konnten (Royalty 2009). In diesem Fall wird diese FIFO Queue als Grundlage für das „kooperative Multitasking“ (Boegelsack 2010) verwendet.

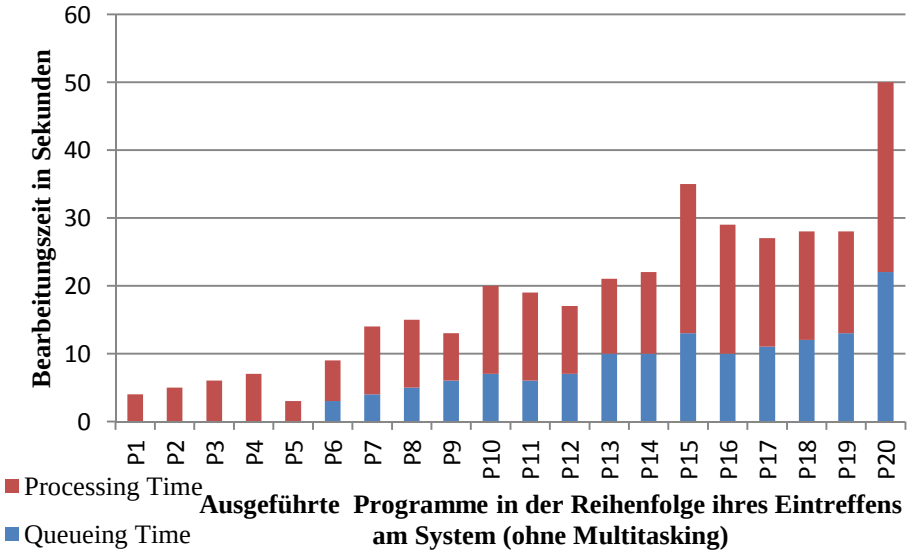


Abbildung 3-8: Verteilung der Wartezeiten ohne kooperatives Multitasking
Quelle: Eigene Darstellung

Im Gegensatz zum „Präemptiven Multitasking“, bei dem die steuernde Komponente ein Programm nach einem festgelegten Algorithmus unterbricht (Kombination aus Zeitscheibenverfahren und Round Robin bei modernen Betriebssystemen), ist es hier dem Programm selbst überlassen, ob es während der Ausführung angehalten werden kann oder nicht. Wird dies nicht unterstützt, so kann der Dispatcher das Programm nicht zu Gunsten eines wartenden Programmes neu schedulen. Wird dies aber unterstützt, so kann er das Programm stoppen und in die FIFO-Queue einreihen. Anhand eines beispielhaften Systems mit 5 konfigurierten Dialog-Work-Prozessen werden nun die Auswirkungen von fehlendem Multitasking, Abbildung 3-8, und „kooperativem Multitasking“, Abbildung 3-9, erläutert. Hierbei bleiben die Processing-Zeiten, welche für die reine Ausführung der Programme benötigt werden, unverändert. Der Unterschied zeigt sich in der Ausprägung der Wartezeiten in der Queue des Dispatchers.

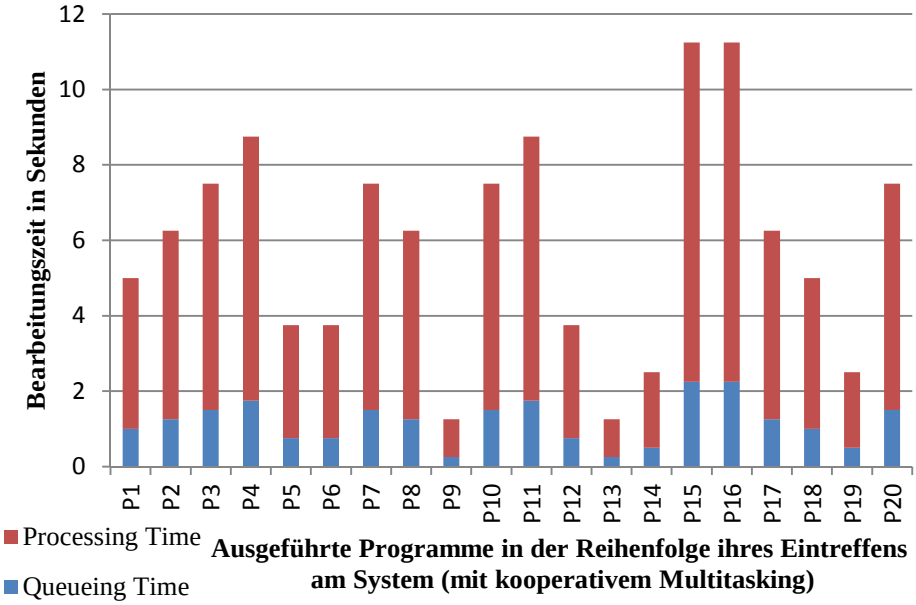


Abbildung 3-9: Verteilung der Wartezeiten mit kooperativem Multitasking
Quelle: Eigene Darstellung

Hier weisen die ersten 5 Anfragen keine Queueing-Zeit auf, da 5 Work-Prozesse zur Verfügung stehen. Bieten diese 5 Anfragen nicht die Möglichkeit zur Unterbrechung durch den Dispatcher, so verweilen die nächsten Anfragen in der Queue, bis ein freier Work-Prozess zur Verfügung steht. Der Dispatcher kann kein Multitasking anwenden. Damit wachsen die Queueing-Zeiten der Anfragen entsprechend der Ausführungszeit ihrer direkten Vorgänger an. Dies führt zu ungleich verteilten, hohen Antwortzeiten, welche sich aus den Queueing-Zeiten und der Ausführungszeit zusammensetzen. Anders ist es bei Programmen, welche die Unterbrechung durch den Dispatcher zulassen. Hier ist die Summe aller Queueing-Zeiten zwar identisch mit den Zeiten ohne kooperatives Multitasking, jedoch anteilig zur Ausführungszeit der Programme auf diese verteilt. Dies führt zu einer annähernd gleichmäßigen Erhöhung der Antwortzeiten und zu einem stabileren Antwortzeitverhalten des Systems, da die Wahrscheinlichkeit von Timeouts vermindert wird.

3.5. Pufferung im SAP-WebAS-ABAP

Der SAP-WebAS-ABAP verwendet eine Vielzahl von Puffern, die helfen sollen, Zugriffe auf die Datenbank zu reduzieren und diese durch, um Größenordnungen schnellere, Leseoperationen aus dem Hauptspeicher zu ersetzen. Damit wird die Belastung sowohl des Datenbankservers als auch des Netzwerks reduziert und die Performance des Gesamtsystems beträchtlich erhöht.

3.5.1. Aufbau der Puffer

Die Puffer sind dabei, wie bereits erläutert, im Shared Memory des SAP-WebAS-ABAP angelegt, sodass den Work-Prozessen ein schneller Zugriff möglich ist. Ein Puffer besteht aus folgenden Komponenten (SAP 2011e):

- Modus Tabelle:
  Diese Tabelle ist allen Work-Prozessen zugänglich und enthält die Informationen, in

welchem Pool die entsprechenden Speicherbereich abgelegt werden. Die Pools werden mit einer zweistelligen ganzen Zahl als Schlüssel identifiziert. Hierbei stellen die Pools 10, 20, und 40 reine Speicherpools dar, in welche Speicherbereiche, identifiziert durch andere Schlüssel, abgelegt werden können. Die Größen dieser Speicherpools werden durch das Instanzprofil in Kilobyte definiert. Mit der Konfigurationszeile „ipc/shm_psize_33=-10“ wird der Puffer für die Speicherung von einzelnen Datenbanksätzen in den Speicherpool 10 gelegt. Tabelle 3-1 zeigt die vollständige Zuordnung von Schlüsselwerten zu den verschiedenen „shared memories“, welche im SAP-WebAS-ABAP verwendet werden.

- SAP Global Management Table:
  Bei dieser Tabelle handelt es sich um die Zuordnung von SAP-Speicherbereichen zu physikalischen Adressen. Diese Tabelle wird beim Systemstart durch den Dispatcher-Prozess erzeugt, und ist, bei aktiviertem Semaphorenschutz, nur durch das SAP-Shared-Memory-Management aufrufbar. Dabei handelt es sich um einen Agenten, welcher in jedem Work-Prozess enthalten ist.

- Address-Table:
  Diese Tabelle ist in jedem Work-Prozess enthalten und speichert die Zuordnung von virtuellen Adressen des Work-Prozess-Adressraums zu physischen Adressen der gemeinsamen Speicherbereiche.

- Shared Memory Objekte:
  Dies ist die abstrakte Bezeichnung für Objekte im Share Memory, wie z.B. die Puffer.

- Header:
  Dieser enthält Informationen über das jeweilige Segment des gemeinsam genutzten Speichers.

- ID:
  Eine ID wird vergeben, wenn ein Teil des gemeinsam genutzten Speichers durch das „Shared Memory Management“ eines Work-Prozesses angefragt wird.

- Storage Class:
  Damit wird die Speicherklasse (Permanent, Shared, Paging, Short) angegeben.

- Subdivision:
  Dadurch wird ein Speicherbereich markiert. Diese Markierung wird verwendet, um diesen Bereich wieder freigeben zu können.

- Size include header:
  Diese Information beinhaltet die Größe des gesamten Puffers mitsamt der Header.

- Alignment:
  Das Alignment dient dem Abgleich der Speicherbereich bzgl. der Hardwarebeschränkungen (Page Size).

Die Vielzahl von verwendeten Puffern im SAP-WebAS-ABAP (Tabelle 3-1) sind grundsätzlich in 7 Gruppen aufgeteilt.

Die Repository Puffer speichern Strukturinformationen zu den einzelnen Datenbanktabellen. Dazu gehören die Tabellennamen, die Namen der einzelnen Felder und die Datenstrukturen für deren initiale Werte. Aufgrund der enthaltenen Informationen werden sie auch als Nametab-Puffer (kurz NTAB) oder ABAP Dictionary-Puffer bezeichnet. Diese Informationen sind in der Datenbank auf mehreren Tabellen verteilt. Dementsprechend existieren 4 Puffer in dieser Gruppe. Der Table-Definitions-Puffer (kurz TTAB) enthält die Beschreibung einer jeden Tabelle, auf die seit dem Systemstart zugegriffen wurde. Analog dazu werden Strukturinformationen aller verwendeten Felder einer Datenbanktabelle im Field-Description-Puffer (kurz FTAB) hinterlegt. Die Beschreibungen werden durch den „database access agent“, welcher in jedem Work-Prozess enthalten ist, bei Aufruf einer bisher noch nicht verwendeten Tabelle aus den Tabellen DDNT (für den TTAB-Puffer) und DDNTF (für den FTAB-Puffer) geholt, im Puffer abgelegt und dann verwendet.

Im Gegensatz zu TTAB- und FTAB-Puffer entspricht der Inhalt des Short-Nametab-Puffers kurz SNTAB) nicht direkt dem Inhalt einer Datenbanktabelle. Da es sich dabei um eine kurze Zusammenfassung der beiden zuvor genannten Puffer (TTAB und FTAB) handelt, wird der Inhalt aus deren Einträgen bzw. aus deren zugrunde liegenden Tabellen erzeugt.

Speichersegment	Schlüssel
System administration	01
Disp. administration tables	02
Disp. communication areas	03
statistic area	04
ABAP program buffer	06
Update task administration	07
Paging buffer	08
Roll buffer	09
Factory calender buffer	11
TemSe Char-Code convert Buf.	12
Alert Area	13
Presentation buffer	14
Semaphore activity monitoring	16
Roll administration	17
Paging adminitration	18
Table-buffer	19
Taskhandler runtime admin.	30
Dispatcher request queue	31
Table buffer, part.buffering	33
Enqueue table	34

Speichersegment	Schlüssel
DB statistics buffer	41
DB TTAB buffer	42
DB FTAB buffer	43
DB IREC buffer	44
DB short nametab buffer	45
DB sync table	46
DB CUA buffer	47
Number range buffer	48
Spool admin	49
Extended memory admin.	51
Message Server buffer	52
Export/Import buffer	54
Spool local printer+joblist	55
Profilparameter in shared mem	57
Enqueue ID for reset	58
Memory pipes	62
ICMAN shared memory	63
Online Text Repository Buf.	64
Export/Import Shared Memory	65

Tabelle 3-1: Speichersegmente im SAP-WebAS-ABAP
Quelle: Ausgabe SAP Programm (SAP 2009)

Der Initial-Record-Layouts-Puffer (kurz IREC) wird für die Zuweisung initialer Feldwerte verwendet. Nach einem „REFRESH“-Aufruf oder beim Einfügen einer neuen Zeile in eine Tabelle werden die entsprechenden Felder mit den Informationen aus dem IREC Puffer

initialisiert, bevor ihnen ein Wert zugewiesen wird. Damit wird sichergestellt, dass jedes Feld einer Tabelle einen Wert enthält, auch wenn die „Insert“-Anweisung nicht für jedes einzelne Feld einen Wert beinhaltet.

Da der SAP-WebAS-ABAP sowohl den Quellcode der ABAP-Programme als auch deren Kompilate (Loads) in der Datenbank vorhält, wird auch der Zugriff auf diese durch die „Program Execution Area“ (PXA), auch kurz ABAP-Puffer genannt, unterstützt. Ist dessen Größe zu gering, so werden ältere Einträge nach dem Last-Recently-Used-Verfahren (LRU) entfernt. Diesem Puffer liegen die Tabellen D010L (für die ABAP Loads), D010T (für die Texte) und D010Y (für Symbole) zugrunde. Um schnellen Zugriff auf die Inhalte zu gewährleisten, wird eine Hash-Struktur verwendet.

Für die beschleunigte Anzeige der Bildschirmmasken im SAPGui werden die SAPGui-Puffer „presentation buffer“ (Screen-buffer) und „menu buffer“ (CUA-buffer) verwendet. Der Screen-Puffer speichert die von den Programmen erzeugten Bildschirme, während der CUA-Puffer für den schnellen Zugriff auf SAPGui-Objekte, wie Schaltflächen oder Menüs, verantwortlich ist. Dabei wird der Zugriff auf die Tabellen D345T (CUA-Texte) und D342L (CUA-Loads) gepuffert. Auch bei diesen Puffern werden Objekte nach dem LRU-Prinzip verdrängt.

Um während der Ausführung die Ablage und das Einlesen von Daten zu beschleunigen, werden die Roll- und Paging-Bereiche des SAP-WebAS-ABAP, welche sich auf Dateisystemebene in Dateien oder im Swap des Betriebssystems befinden, durch die Roll- und Paging-Puffer sowie durch den Erweiterungsspeicher unterstützt. Dabei werden die Benutzerkontexte nach Beendigung eines Auftrags durch einen Work-Prozess vom Erweiterungsspeicher zurück in den Roll-Puffer gelegt. Während der Paging-Puffer aufgrund veränderter Programmierung in aktuelleren Anwendungen nicht mehr verwendet wird und damit grundsätzlich an Bedeutung verliert (SAP-NetWeaver-WebApplication-Server), wird der Erweiterungsspeicher stark für die Erstellung interner Tabellen, welche nur zur Laufzeit eines ABAP-Programms gültig sind, verwendet.

Im Gegensatz zu den bisher besprochenen Puffern hat der SAP-Calendar-Buffer eine Verzeichnisstruktur. Darin sind die definierten Werks- bzw. Feiertagskalender abgelegt. Diese Informationen werden häufig aus den verschiedenen Anwendungen heraus verwendet, um die Ausführbarkeit bestimmter Operationen zu bestimmen. Dieser Puffer unterstützt keine Verdrängungsmechanismen, da er bei zu wenig verfügbarer Speichermenge nur die erforderlichen Daten der Tabellen TFACS und THOCS speichert (SAP 2011f).

Zusätzlich zu der großen Anzahl von Puffern werden Datenbankzugriffe beschleunigt, indem der SAP-Cursor-Cache Informationen zu geparsten SQL-Aufrufen an die Datenbank zwischenspeichert. Die Implementierung dieses Puffers ist abhängig von der Datenbank und stellt sich daher für die verschiedenen, von SAP unterstützten Datenbankprodukte unterschiedlich dar. Grundsätzlich besteht er aus dem „Statement ID Cache“ und dem „Statement Cache“.

Ein weiterer Puffer für die Programmausführung stellt der „Online Text Repository“-Puffer (OTR) dar. Dieser beinhaltet Textstücke, welche durch Business-Server-Pages (BSP), Exception-Builder und http-Services verwendet werden.

Der Export/Import-Puffer speichert Daten, die für mehrere Work-Prozesse zugreifbar sein sollen. Durch die ABAP-Kommandos „export to shared buffer“, „import from shared buffer“ und „delete from shared buffer“ werden diese Daten abgelegt und verwaltet. Die Pufferverwaltung verdrängt bei zu gering konfigurierter Größe Objekte nach dem LRU-Prinzip.

Eine Vergrößerung des für Work-Prozesse verfügbaren Speicherplatzes kann durch ein Verschieben von Daten in das Export/Import Shared Memory (kurz ESM) erreicht werden. Mittels der ABAP-Befehle „export to shared memory“, „import from shared memory“ und „delete from shared memory“ kann auf das ESM zugegriffen werden. Da in diesem Puffer keine Daten aus der Datenbank zwischengespeichert werden, sondern er durch ABAP-Programme selbst erzeugte Daten enthält, werden Inhalte bei vollständiger Verwendung des verfügbaren Speichers nicht ausgelagert. Damit können keine weiteren Objekte in den Puffer eingelagert und weitere Programme nicht ausgeführt werden. Durch die sehr enge Verbindung zu der Ausführung von ABAP-Programmen ist dieser Puffer äußert wichtig für die Performance des gesamten Systems.

3.5.2. Tabellen-Pufferung

Noch größeren Einfluss auf die Performance des gesamten Systems haben die Tabellenpuffer, welche häufig angefragte Daten aus der Datenbank für einen schnelleren Zugriff zwischenspeichern. Dabei wird zwischen der Pufferung einzelner Tabellenzeilen im „Single Record Buffer“ (kurz TABLP) und der Ablage einer ganzen Tabelle oder generischer Bereiche einer Tabelle im „Generic Key Table Buffer“ (kurz TABL) unterschieden.

3.5.2.1. Einzelsatzpufferung

Abbildung 3-10 zeigt beispielhaft eine Tabelle, welche mit Einzelsatzpufferung konfiguriert wurde. Die weißen Zeilen sollen gepufferte Einträge darstellen.

Key 1	Key 2	Key 3	Spalte 4	Spalte 5
10000	A	FCED24A		
10000	G	BC46722		
10000	G	6723CA8		
10000	K	7812DAB		
10000	S	237FD34		
10010	I	89CAF82		
10010	I	247BD21		
10010	I	DAAC774		
10010	K	CA34BD2		
10010	Z	42DCBA3		
10010	Z	723649C		
10020	D	BFA7390		
10020	L	8725CAF		
10020	N	F896CDA		
10020	s	FABC723		
10020	Z	3234DFF		
10050	H	FFF7634		
10050	H	234BD89		
10050	S	FFBACCD		
10050	U	8232343		
10050	W	24724DC		

Abbildung 3-10: Einzelsatzpufferung einer beispielhaften Tabelle
Quelle: In Anlehnung an (Schneider 2008)

Wird eine Anfrage an die Datenbank gestellt, so werden die betroffenen Zeilen einer Tabelle, für welche die Einzelsatzpufferung vorgesehen ist, in diesem Puffer abgelegt, bevor sie an das Programm übergeben werden. Die Kapazität dieses Puffers wird mit den Parametern „rtbb/max_tables“ (Maximale Anzahl von Tabellen im Puffer) und „rtbb/buffer_length“ (Größe des Puffers in KB) über 2 Dimensionen definiert. Der erste der beiden Parameter definiert also die maximale Anzahl von Tabellen, von denen einzelne Sätze im Puffer abgelegt werden können. Der zweite Parameter bestimmt, abhängig von der Größe der einzelnen Sätze, wie viel Speicher für die Datenbankzeilen zur Verfügung steht.

3.5.2.2. Vollständige Pufferung

Neben der Pufferung einzelner Sätze bietet der SAP-WebAS-ABAP auch die Möglichkeit, ganze Tabellen für den Zugriff zu puffern, siehe Abbildung 3-11. Hierbei ist keine genauere Spezifikation der Primärschlüsselattribute nötig, da ja die gesamte Tabelle ausgewählt ist. Jedoch ist zu beachten, dass trotz vollständiger Pufferung nur Zugriffe mit „where“- Klauseln unterstützt werden, welche sich auf die Primärschlüsselattribute der Datenbanktabelle beziehen. Obwohl diese Art der Pufferung meist nur bei kleinen Tabellen Verwendung findet, würde die Pufferung einer Tabelle, auf welche nur über Sekundärschlüssel zugegriffen wird, unnötig Ressourcen in den Tabellenpuffern verbrauchen und zu keiner Verbesserung der Bearbeitungsgeschwindigkeit führen. Da es sich bei der vollständigen Pufferung um einen Spezialfall der Pufferung generischer Inhalte handelt, werden diese Tabellen im „Generic buffer“ (TABL) abgelegt.

Key 1	Key 2	Key 3	Spalte 4	Spalte 5
10000	A	FCED24A		
10000	G	BC46722		
10000	G	6723CA8		
10000	K	7812DAB		
10000	S	237FD34		

Abbildung 3-11: Vollständige Pufferung einer beispielhaften Tabelle
Quelle: In Anlehnung an (Schneider 2008)

3.5.2.3. Pufferung generischer Tabellenbereiche

Im bereits angesprochenen „Generic buffer“ (TABL) werden zusätzlich zu ganzen Tabellen auch generische Regionen einer Tabelle abgelegt. Abbildung 3-12 zeigt den Vergleich der generischen Pufferung anhand von einem Feld des Primärschlüssels auf der linken Seite und die Pufferung anhand von zwei Feldern des Primärschlüssels auf der rechten Seite. Dabei kann nicht frei gewählt werden, nach welchem Feld des Primärschlüssels die Auswahl der zu puffernden Zeilen geschehen soll. Im DDIC, welches die Beschreibungen aller Objekte im SAP-WebAS-ABAP enthält, sind die Felder der Primärschlüssel jeder Tabelle sortiert und damit geordnet. Entsprechend dieser Ordnung können die für die Pufferung zu verwendenden Felder ausgewählt werden. Ist das zweite Feld des Primärschlüssels für die Pufferung entscheidend, so muss zusätzlich zu diesem auch das erste Feld verwendet werden. Dies führt, wie Abbildung 3-12 zeigt, meist zu einer größeren Anzahl von generischen Bereichen.

Key 1	Key 2	Key 3	Spalte 4	Spalte 5
10000	A	FCED24A		
10000	G	BC46722		
10000	G	6723CA8		
10000	K	7812DAB		
10000	S	237FD34		
10010	I	89CAF82		
10010	I	247BD21		
10010	I	DAAC774		
10010	K	CA34BD2		
10010	Z	42DCBA3		
10010	Z	723649C		
10020	D	BFA7390		
10020	L	8725CAF		
10020	N	F896CDA		
10020	s	FABC723		
10020	Z	3234DFF		
10050	H	FFF7634		
10050	H	234BD89		
10050	S	FFBACCD		
10050	U	8232343		
10050	W	24724DC		

Key 1	Key 2	Key 3	Spalte 4	Spalte 5
10000	A	FCED24A		
10000	G	BC46722		
10000	G	6723CA8		
10000	K	7812DAB		
10000	S	237FD34		
10010	I	89CAF82		
10010	I	247BD21		
10010	I	DAAC774		
10010	K	CA34BD2		
10010	Z	42DCBA3		
10010	Z	723649C		
10020	D	BFA7390		
10020	L	8725CAF		
10020	N	F896CDA		
10020	s	FABC723		
10020	Z	3234DFF		
10050	H	FFF7634		
10050	H	234BD89		
10050	S	FFBACCD		
10050	U	8232343		
10050	W	24724DC		

Abbildung 3-12: Pufferung von Tabellen anhand generischer Regionen mit einem und zwei verwendeten Schlüsselfeldern
Quelle: In Anlehnung an (Schneider 2008)

Grundsätzlich kann damit feiner spezifiziert werden, welche Datenbankzeilen wirklich gepuffert werden sollen, und die Pufferung nicht benötigter Zeilen vermieden und damit Ressourcen im Puffer eingespart werden. Da auch dieser Puffer über die Anzahl der möglichen Einträge und die maximale Größe des zur Verfügung stehenden Speichers definiert wird, kann eine feinere Spezifikation der zu puffernden Tabellenbereiche zwar die im Puffer benötigte Speichermenge reduzieren, jedoch gleichzeitig auch den Bedarf an möglichen Einträgen erhöhen.

Die Verdrängung gepufferter Inhalte wird in diesem Puffer asynchron durchgeführt. Ist die Pufferqualität eines generischen Bereiches zu gering, so wird dieser Bereich aus dem Puffer verdrängt. Eine mögliche Ursache hierfür kann die fehlende Spezifikation der angefragten Datenbankzeilen in der „where“- Klausel der SQL-Anfrage sein. Verdrängungen können auch durch eine Unterschreitung einer voreingestellten Schwelle bzgl. der vorzuhaltenden freien Ressourcen im Puffer sein. Dann werden die zu verdrängenden generischen Bereiche durch eine Gewichtung bzgl. der Zeitpunkte, zu denen auf sie zugegriffen wurde, bewertet. Die Gewichtung verringert sich, je mehr Zeit seit dieser Anfrage verstrichen ist.

3.5.2.4. Konfiguration der Pufferung

Abbildung 3-13 zeigt anhand der Tabelle „TCURR“, wie die Pufferung für eine Tabelle konfiguriert wird. Hier wird für das, die Tabelle repräsentierende Objekt, im DDIC unter „Pufferungsart“ festgelegt, ob Einzelsatzpufferung, Pufferung eines generischen Bereichs oder vollständige Pufferung verwendet werden soll. Im Falle der generischen Puffer wird hier festgelegt, wie viele Felder des Primärschlüssels, welcher im DDIC mit einer festgelegten Reihe spezifiziert ist, für die Identifikation der generischen Bereiche verwendet werden.

Technisch ist die vollständige Pufferung einer Tabelle also nur ein Spezialfall der generischen Pufferung. Dies zeigt sich auch in der Tatsache, dass der TABL Puffer für beide Pufferungsarten verwendet wird. Die Konfiguration der Pufferungsart ist ein äußerst komplexer Vorgang, der tiefgreifende Kenntnis über die Programmierung der ABAP-Programme verlangt, welche auf diese Tabelle zugreifen. Bei nicht optimaler Abstimmung von Zugriffsarten und Pufferung können Verdrängungen (3.5.2.3) resultieren, welche negative Auswirkungen auf die Performance des gesamten Systems haben. Diese Probleme können nur durch Rückgängigmachen der Konfiguration oder durch aufwendige Programmierung behoben werden. Aus diesem Grund wird die Pufferung bei Kundenentwicklungen durch den Programmierer bzw. den Software-Architekten festgelegt. SAP-Hinweisen zufolge wird Administratoren dringend davon abgeraten, Änderungen an der Pufferungsart SAP-eigener Tabellen ohne Beratung durch den SAP-Support durchzuführen.

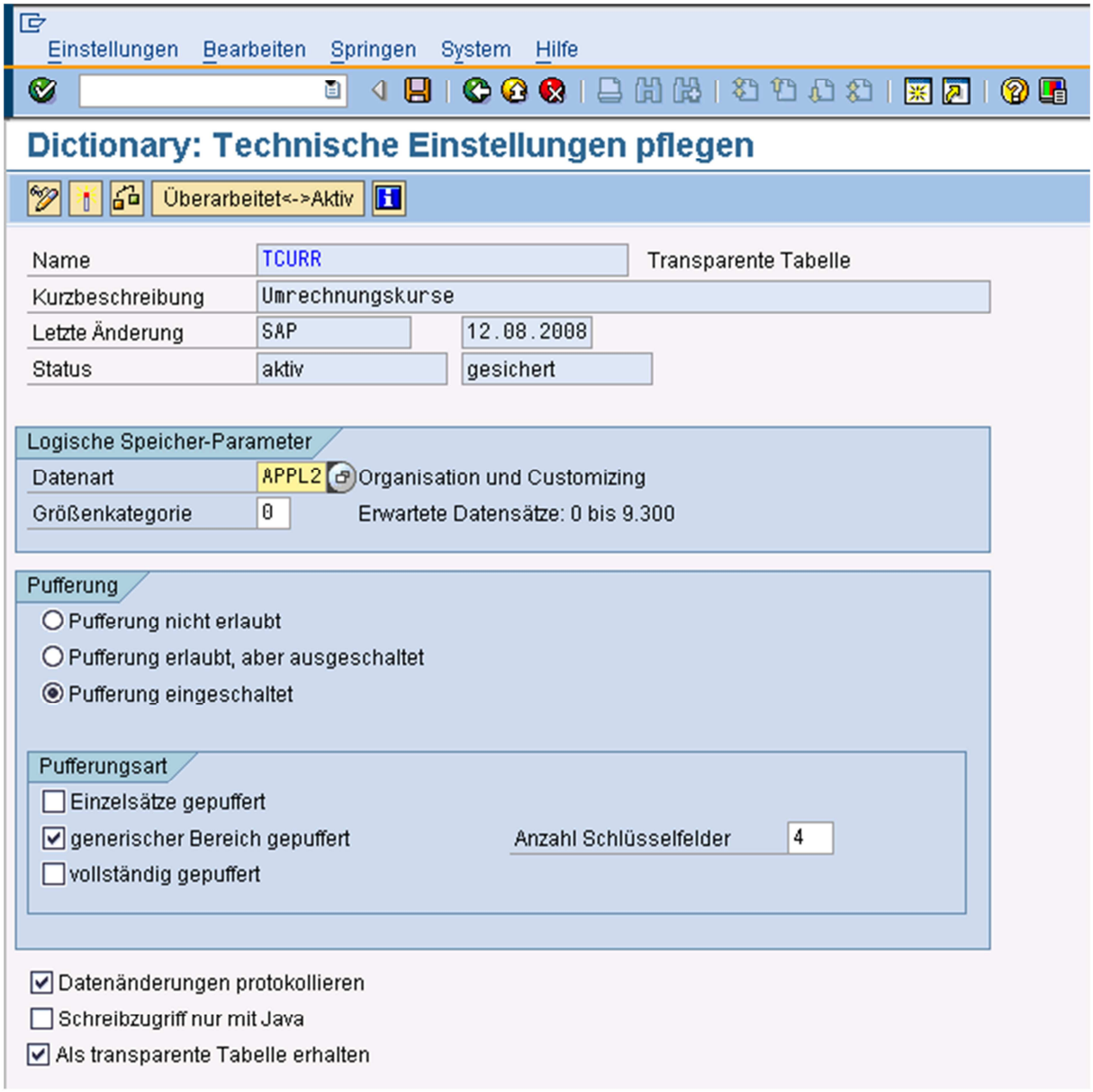
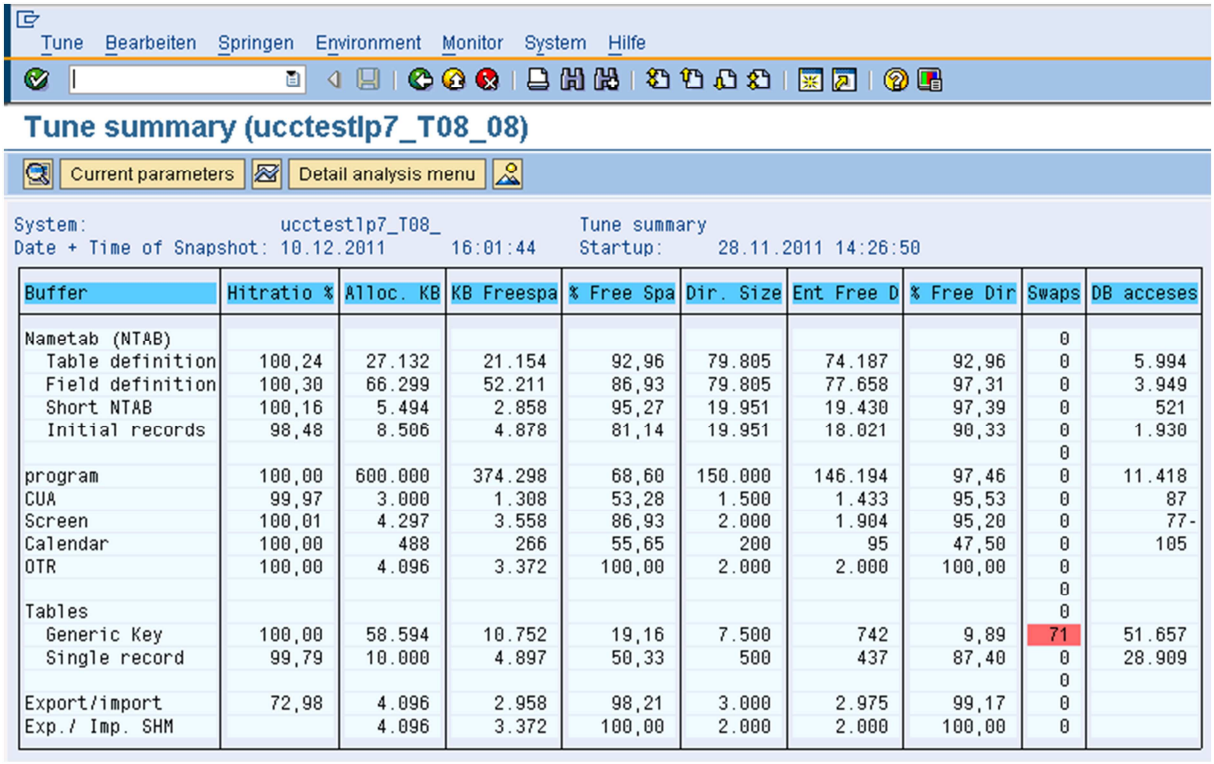


Abbildung 3-13: Konfiguration der Pufferung einer Tabelle
Quelle: Screenshot Transaktion SE11(SAP 2008b)

3.5.2.5. Pufferzustände

Um die Zustände der Puffer im SAP-WebAS-ABAP zu analysieren und damit deren Qualität zu beurteilen, zeichnet der SAP-Kernel Puffer- und Datenbankzugriffe auf und stellt entsprechende Statistiken über die Transaktion ST02, siehe Abbildung 3-14, bereit.



Buffer	Hitratio %	Alloc. KB	KB Freespa	% Free Spa	Dir. Size	Ent Free D	% Free Dir	Swaps	DB acceses
Nametab (NTAB)								0	
Table definition	100,24	27.132	21.154	92,96	79.805	74.187	92,96	0	5.994
Field definition	100,30	66.299	52.211	86,93	79.805	77.658	97,31	0	3.949
Short NTAB	100,16	5.494	2.858	95,27	19.951	19.430	97,39	0	521
Initial records	98,48	8.506	4.878	81,14	19.951	18.021	90,33	0	1.930
								0	
program	100,00	600.000	374.298	68,60	150.000	146.194	97,46	0	11.418
CUA	99,97	3.000	1.308	53,28	1.500	1.433	95,53	0	87
Screen	100,01	4.297	3.558	86,93	2.000	1.904	95,20	0	77-
Calendar	100,00	488	266	55,65	200	95	47,50	0	105
OTR	100,00	4.096	3.372	100,00	2.000	2.000	100,00	0	
								0	
Tables								0	
Generic Key	100,00	58.594	10.752	19,16	7.500	742	9,89	71	51.657
Single record	99,79	10.000	4.897	50,33	500	437	87,40	0	28.909
								0	
Export/import	72,98	4.096	2.958	98,21	3.000	2.975	99,17	0	
Exp./ Imp. SHM		4.096	3.372	100,00	2.000	2.000	100,00	0	

Abbildung 3-14: Anzeige der Pufferzustände
Quelle: Screenshot Transaktion ST02(SAP 2011i)

Dabei werden die meistverwendeten Puffer (linke Spalte) zusammen mit den wichtigsten Eigenschaften (Spalten 2 bis 9) dargestellt. In Spalte 2 zeigt diese Transaktion die Trefferquote als Prozentsatz aller Anfragen auf die entsprechenden Datenbanktabellen, welche aus dem zugehörigen Puffer beantwortet werden konnten. Je nach Puffer und Einschwingzustand des Systems sollte die Trefferquote über 99 Prozent betragen. Die konfigurierte Größe des Puffers in Kilobyte wird in Spalte 3 („Alloc. KB“) angezeigt. Hierbei ist zu beachten, dass nicht die volle allokierte Speichermenge für gepufferte Inhalte zur Verfügung steht, sondern ein kleiner Teil für das Management des Puffers verwendet wird. Spalte 4 zeigt die noch freie Menge an Speicher ebenfalls in Kilobyte. Den prozentualen Anteil an freiem Speicher im jeweiligen Puffer zeigt Spalte 5. Analog dazu zeigen die Spalten 6 bis 8 („Dir. Size“, „Ent Free D“ und „% Free Dir“) die maximale Anzahl von Einträgen, die Anzahl freier Einträge sowie den prozentualen Anteil freier Einträge im Puffer an. Je nach Puffer ist der Begriff „Eintrag“ unterschiedlich definiert. Für den Puffer „Tables Generic Key“ sind Einträge oder Entrys als generische Bereiche von Datenbanktabellen definiert. Damit kann der „Generic Key Buffer“ mit einer Konfiguration wie in Abbildung 3-14 maximal 7500 verschiedene generische Datenbankbereiche (dazu zählen auch vollständig gepufferte Tabellen) beinhalten. Die Anzahl der aus dem jeweiligen Puffer verdrängten Objekte wird in Spalte 9 dargestellt. Mit „Swaps“ werden in dieser Statistik nur verdrängte Objekte gezählt. Invalidierungen werden nur in der detaillierten Übersicht (3.5.2.6) dargestellt. Die letzte Spalte beinhaltet die Anzahl an Anfragen, welche nicht durch den

Puffer beantwortet werden konnten und somit direkt von der Datenbank geladen wurden. Anhand dieser Zahlen zeigt sich die Häufigkeit der Verwendung der verschiedenen Puffer. So haben die Tabellenpuffer bei einer Trefferquote von über 99% trotzdem noch ca. 29.000 bzw. ca. 52.000 Zugriffe nicht beantworten können. Diese Übersicht bietet einen schnellen Überblick über die Zustände und möglichen Probleme der SAP-Puffer. Für eine genaue Analyse der Puffer bietet diese Transaktion noch weitere Detailierungsstufen, welche im folgenden Absatz 3.5.2.6 beschrieben werden.

3.5.2.6. Analyse der Pufferinhalte

Die detaillierten Zustände der gepufferten Objekte zeigt die Übersicht „Performance analysis: Table call statistics“. Dies ist beispielhaft für den „generic key buffer“ in Abbildung 3-15 dargestellt. Hier werden alle Tabellen mit generischer oder vollständiger Pufferung zusammen mit den Zugriffstatistiken aufgelistet. Hierbei wird die Anzahl von Zugriffen durch ABAP-Programme den tatsächlichen Datenbankzugriffen gegenübergestellt. Dabei ist zu beachten, dass in den Spalten mit der Überschrift „DB activity“ auch die Zugriffe und betroffenen Datenbankzeilen integriert sind, die zum Füllen der Puffer benötigt wurden.

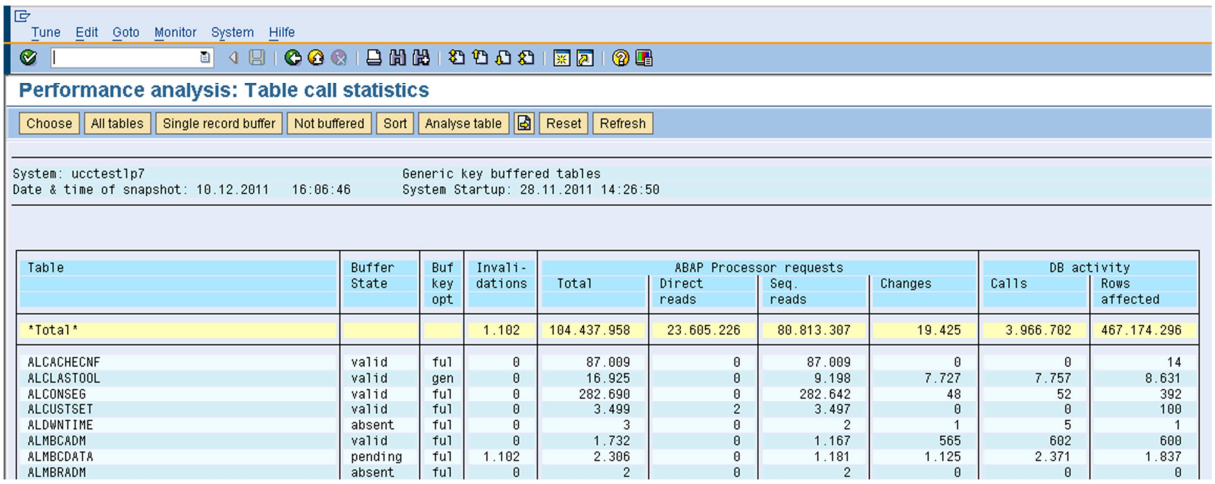


Table	Buffer State	Buf key opt	Invali-dations	ABAP Processor requests Total	Direct reads	Seq. reads	Changes	DB activity Calls	Rows affected
Total			1.102	104.437.958	23.605.226	80.813.307	19.425	3.966.702	467.174.296
ALCACHECNF	valid	ful	0	87.009	0	87.009	0	0	14
ALCLASTOOL	valid	gen	0	16.925	0	9.198	7.727	7.757	8.631
ALCONSEG	valid	ful	0	282.690	0	282.642	48	52	392
ALCUSTSET	valid	ful	0	3.499	2	3.497	0	0	100
ALDWNTIME	absent	ful	0	3	0	2	1	5	1
ALMBCADM	valid	ful	0	1.732	0	1.167	565	602	600
ALMBCDATA	pending	ful	1.102	2.306	0	1.181	1.125	2.371	1.837
ALMBRADM	absent	ful	0	2	0	2	0	0	0

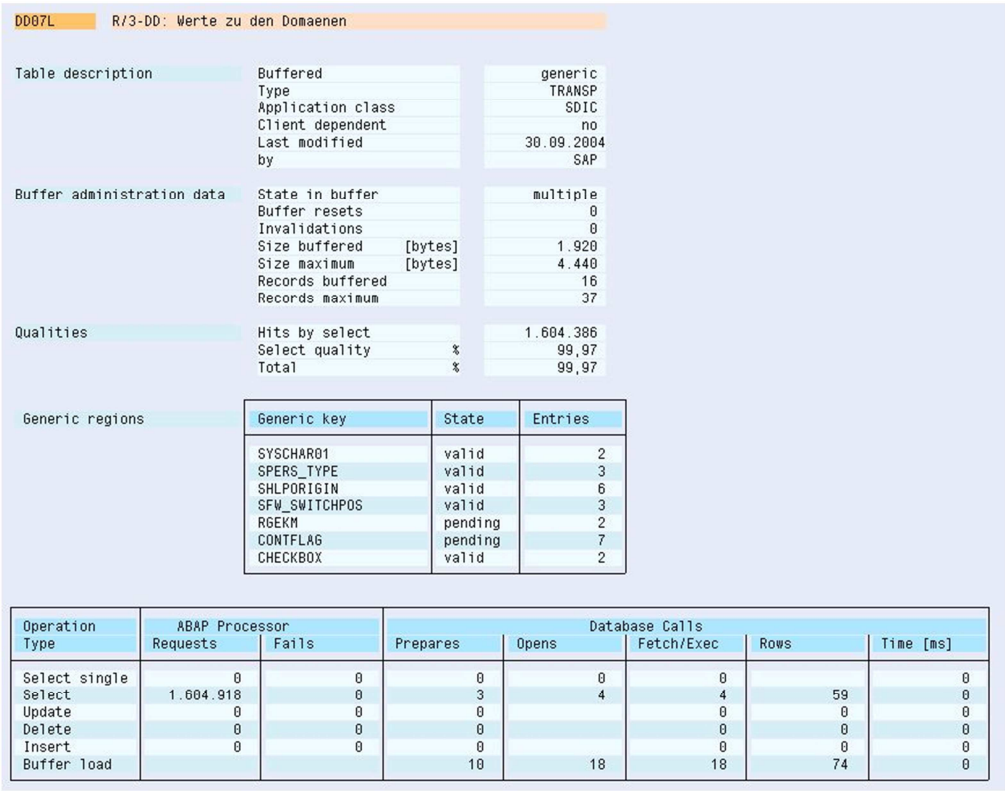
Abbildung 3-15: Zugriffsstatistiken auf generische Datenbankbereiche im Puffer
Quelle: Screenshot Transaktion ST02 (SAP 2011i)

Den Zustand der generischen Bereiche pro Datenbanktabelle zeigt die Spalte „Buffer State“. Tabelle 3-2 zeigt die verschiedenen Zustände gepufferter Objekte mit einer kurzen Beschreibung.

Status	Bedeutung
valid	Tabelle (bzw. Teile von ihr) befindet sich gültig im Puffer, d.h., der nächste Zugriff kann aus dem Puffer erfolgen
invalid	Tabelle wurde invalidiert. Die Tabelle kann noch nicht wieder in den Puffer geladen werden, da eine Operation, die die Tabelle geändert hat, noch nicht mit einem "commit" abgeschlossen wurde
pending	Tabelle wurde invalidiert. Sie kann beim nächsten Zugriff noch nicht geladen werden, da die Karenz läuft
loadable	Tabelle wurde invalidiert. Die Karenz ist abgelaufen und die Tabelle wird beim nächsten Zugriff geladen
loading	Tabelle wird zurzeit geladen
absent, displaced	Tabelle befindet sich nicht im Puffer (z.B. weil sie nie geladen wurde oder weil sie verdrängt wurde)
multiple	Kann bei generischen Puffern auftreten: Einige generische Bereich sind gültig, andere wurden aufgrund von Änderungen invalidiert
error	Fehler beim Laden der Tabelle. Diese Tabelle kann nicht gepuffert werden.

Tabelle 3-2: Zustände gepufferter Objekte
Quelle: (Schneider 2008)

Eine genaue Beschreibung des Zustandes der gepufferten Objekte wird durch einen Klick auf eine Tabelle erzeugt. Abbildung 3-16 zeigt dies beispielhaft für die vollständig gepufferte Tabelle „DD07L“, welche sich im Zustand „multiple“ befindet.



DD07L R/3-DD: Werte zu den Domaenen

Table description	Buffered		generic
	Type		TRANSP
	Application class		SDIC
	Client dependent		no
	Last modified		30.09.2004
	by		SAP
Buffer administration data	State in buffer		multiple
	Buffer resets		0
	Invalidations		0
	Size buffered	[bytes]	1.920
	Size maximum	[bytes]	4.440
	Records buffered		16
	Records maximum		37
Qualities	Hits by select		1.604.386
	Select quality	%	99,97
	Total	%	99,97

Generic regions

Generic key	State	Entries
SYSCHAR01	valid	2
SPERS_TYPE	valid	3
SHLPORIGIN	valid	6
SFW_SWITCHPOS	valid	3
RGEKM	pending	2
CONTFLAG	pending	7
CHECKBOX	valid	2

Operation Type	ABAP Processor Requests	ABAP Processor Fails	Database Calls Prepares	Database Calls Opens	Database Calls Fetch/Exec	Database Calls Rows	Database Calls Time [ms]
Select single	0	0	0	0	0		0
Select	1.604.918	0	3	4	4	59	0
Update	0	0	0		0	0	0
Delete	0	0	0		0	0	0
Insert	0	0	0		0	0	0
Buffer load			10	18	18	74	0

Abbildung 3-16: Detailanalyse der Tabellenzugriffe
Quelle: Screenshot ST02 (SAP 2011i)

Hierbei wird die Tabelle kurz anhand der Pufferungsart, dem Typ, der zugehörigen Anwendungsklasse, der Mandantenabhängigkeit, dem letzten Modifikationsdatum und dem letzten Benutzer, welcher das Objekt modifiziert hat, beschrieben. Für die detaillierte Bestimmung von benötigten Ressourcen für dieses Objekt beinhaltet der Absatz „Buffer administration data“ Informationen zu der aktuell durch die Sätze benötigten Speichergröße sowie die Anzahl aktuell gepufferter Datenbankzeilen. Zusätzlich wird auch noch für beide Angaben eine sogenannte High-Water-Mark mit angezeigt.

Es werden zusätzlich noch Informationen über den Zustand, die Anzahl von Einträgen und dem „Generic key“ gegeben. Die Spaltenüberschrift „Entries“ meint hier die Anzahl der Datensätze in diesem generischen Bereich der Tabelle, während derselbe Ausdruck bei der Bestimmung der Puffergrößen die Anzahl an generischen Bereichen im Puffer bezeichnet. Der generische Schlüssel zeigt, dass diese Tabelle „DD07L“ anhand eines Feldes des Primärschlüssels gepuffert wird. Die unterschiedlichen Inhalte des ersten Primärschlüsselfeldes aller gepufferten Datensätze dieser Tabelle sind in der Tabellenspalte „Generic key“ abgelegt und dienen als Identifikatoren der entsprechenden generischen Bereiche.

Im untersten Bereich von Abbildung 3-16 werden noch detailliert für alle SQL Anfragetypen die vom ABAP-Prozessor ausgehenden Anfragen den tatsächlich durch die Datenbank durchgeführten Operationen gegenübergestellt. Besonders wichtig für die Performance Analyse ist die Information, welche Last durch das Füllen der Puffer verursacht wurde.

Damit können für jede Datenbanktabelle der benötigte Speicherplatz im Puffer pro Tabellenzeile, die Anzahl von generischen Bereichen und damit die Anzahl benötigter Einträge bestimmt werden. Eine manuelle Bestimmung dieser Daten ist jedoch bei ca. 900 Tabellen alleine im „generic key buffer“ eine sehr umfangreiche und langwierige Arbeit, sodass diese Informationen nur für die Analyse von Problemfällen und nicht für die alltäglichen Konfigurationsarbeiten verwendet werden.

3.5.2.7. Arten gepufferter Daten

Dabei ist es jedoch sehr wichtig zu beurteilen, welche Art von Daten gepuffert werden sollten und welche besser ungepuffert direkt aus der Datenbank abgefragt werden. Ein wichtiges Entscheidungskriterium ist hierbei die Häufigkeit der Änderungen. Je nach Pufferungsart wird bei der Änderung einer gepufferten Tabelle ein Teil der Puffer invalidiert und muss dann von neuem aus der Datenbank aktualisiert werden. Treten Änderungen häufig auf, so werden dafür unnötig Ressourcen verwendet.

In SAP-ERP-Systemen wird nach Schneider (2008) grundsätzlich zwischen 3 verschiedenen Arten von Daten unterschieden:

- Bewegungsdaten
- Stammdaten
- Customizing-Daten.

Unter Bewegungsdaten versteht man Lieferungen, Materialbewegungen, usw., welche innerhalb der Geschäftsprozesse, welche im SAP-System implementiert sind, angelegt bzw. verändert werden. Die zugrunde liegenden Tabellen werden daher häufig verändert und wachsen kontinuierlich an. Aus diesem Grund werden Bewegungsdaten nicht gepuffert. Ebenso verhält es sich bei den Stammdaten, wie Materialien, Kunden oder Lieferanten. Die zugrunde liegenden Tabellen wachsen zwar langsamer an, sind jedoch durch die Frequenz der Einfüge- bzw. Update-Operationen auch nicht für die Pufferung geeignet. Hinzu kommt, dass Puffer nur Datenbankzugriffe nach den Primärschlüsselfeldern unterstützen und Stammdaten oft aufgrund anderer Kriterien ausgewählt werden, sodass eine Pufferung viele Anfragen nicht unterstützen würde.

3.5.2.8. Zusammenfassung

Um eine optimale Performance des Gesamtsystems zu erreichen, ist es zusätzlich nötig, genügend Kapazitäten in diesem Puffer bereitzustellen, sodass Verdrängungen (Swaps) nach dem LRU-Prinzip vermieden werden. Der Verdrängungsvorgang bedeutet keine direkte Verminderung der Performance. Diese Beeinträchtigung tritt erst dann zu Tage, wenn der verdrängte Satz wieder benötigt wird, dann nicht mehr im Puffer vorhanden ist und deshalb wieder direkt von der Datenbank geholt werden muss. Daher ist eine optimale Konfiguration der Puffer eine Grundvoraussetzung für ein leistungsfähiges SAP-ERP-System. Aufgrund der bereits dargestellten Konfiguration der Puffer ist dies oft problematisch, da ein Puffer die maximale Anzahl von gepufferten Tabellen erreicht haben kann, obwohl noch Platz zur Verfügung steht. Ebenso stellt die Berechnung des benötigten Speicherplatzes im Puffer eine schwierige Aufgabe dar, da sich die Größen der Zeilen, je nach Tabelle, unterscheiden und dem Administrator nur sehr eingeschränkte Informationen über die Auswirkungen der einzelnen, im System durchgeführten, Prozesse auf die Puffer zur Verfügung stehen. Damit ist eine ausreichende Pufferkonfiguration oft ein iterativer Prozess von Anpassungen.

3.6. Antwortzeitkomponenten im SAP-WebAS-ABAP

Analog zur Abarbeitung von Anfragen durch den SAP-WebAS-ABAP (3.4) besteht die Antwortzeit aus verschiedenen Komponenten (SAP 2007). Für die Evaluation und Analyse der Performance eines SAP-ERP-Systems stellen diese Antwortzeitkomponenten eine wichtige Informationsquelle dar. Die Antwortzeit wird hierbei rein aus Systemsicht (vgl. 2.1.2.3) betrachtet. Die Zeitmessung startet, sobald eine Anfrage am System eintrifft und endet, sobald die Ausgabe an den Client beginnt.

Abbildung 3-17 zeigt die Komponenten der Ausführungszeit eines ABAP Programms. Sobald eine Anfrage am System ankommt, beginnt die Messung der Antwortzeit zum Zeitpunkt t_0. Ist kein freier Work-Prozess vorhanden, so wird die Anfrage in der Warteschlange des Dispatcher-Prozesses abgelegt, bis sie zum Zeitpunkt t_2 an einen freien Work-Prozess übergeben werden kann. Diese erste Komponente der Antwortzeit wird als Queuetime bezeichnet. Danach wird der Benutzerkontext in den Work-Prozess geladen. Diese Zeitspanne wird als RollIn-Time (ROLLINTI) bezeichnet. Grundsätzlich bleibt das Programm im Work-Prozess, bis es zum Zeitpunkt t_5 beendet ist. Die Zeit im Work-Prozess wird als „Time in Workprocess“ bezeichnet und bildet zusammen mit der Queuetime und der RollIn-Time die Antwortzeit des Systems. Ist die Bearbeitung abgeschlossen, wird der Benutzerkontext wieder aus dem Work-Prozess „herausgerollt“. Diese als RollOut-Time bezeichnete Zeitspanne wird

nicht mehr auf die Antwortzeit aufgerechnet, da die Anfrage zu diesem Zeitpunkt bereits bearbeitet ist.

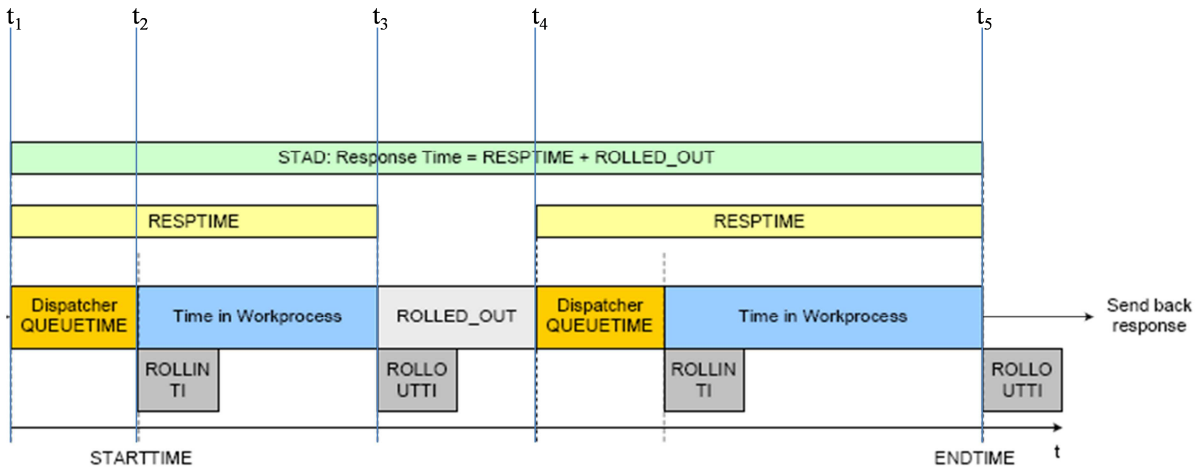


Abbildung 3-17: Komponenten der Antwortzeit im SAP-WebAS-ABAP
Quelle: In Anlehnung an (SAP 2007)

Abbildung 3-17 zeigt noch einen Spezialfall für die Bearbeitung einer Anfrage auf. Werden während der Programmausführung Daten von einem externen System abgefragt und dauert diese Abfrage länger als 500ms, so wird der Work-Prozess zum Zeitpunkt t_3 wieder für andere Anfragen bereitgestellt, indem der Benutzerkontext aus dem Work-Prozess herausgerollt wird. Sobald die Anfrage vom externen System zum Zeitpunkt t_4 eintrifft, wird das Programm wieder eingeplant. Die Zeitspanne ($t_4 – t_3$), in der auf die Antwort des externen Programms gewartet wurde und das Programm nicht in einem Work-Prozess aktiv war, wird als „ROLLED_OUT- Time“ bezeichnet und zählt auch zu der Antwortzeit des Programms.

Die SAP-Transaktion STAD liefert eine feingranularere Aufteilung der Antwortzeitkomponenten. Wie bereits besprochen, werden die Zeitspannen „Queuetime“ und „ROLLED_OUT“ nicht zur „Time in Workprocess“ addiert. Jedoch zählen sie zur Antwortzeit. Abbildung 3-18 zeigt diesen Zusammenhang durch die Differenz von „Total time in workprocs“ zu „Response time“. Diese 282 Millisekunden entsprechen der Summe von „Queuetime“ und „ROLLED_OUT“, in Abbildung 3-18 „Wait for work process“ und „Roll time Wait“ genannt.

Grundsätzlich bereitet die Transaktion einige Informationen aus den Statistiksätzen für die Analyse auf. Dies geschieht nach den folgenden Formeln (SAP 2007), welche auf der linken Seite die Terminologie der Transaktion STAD nutzen, wohingegen die Terminologie der rechten Seite aus Abbildung 3-17 stammt:

- Response Time = RESPTIME + ROLLED_OUT
- Time in WP = RESPTIME - QUEUETIME
- Wait Time = QUEUETIME
- Roll-In Time = ROLLINTI

- Roll-Out Time = ROLLOUTTI
- RollWait Time = ROLLED_OUT

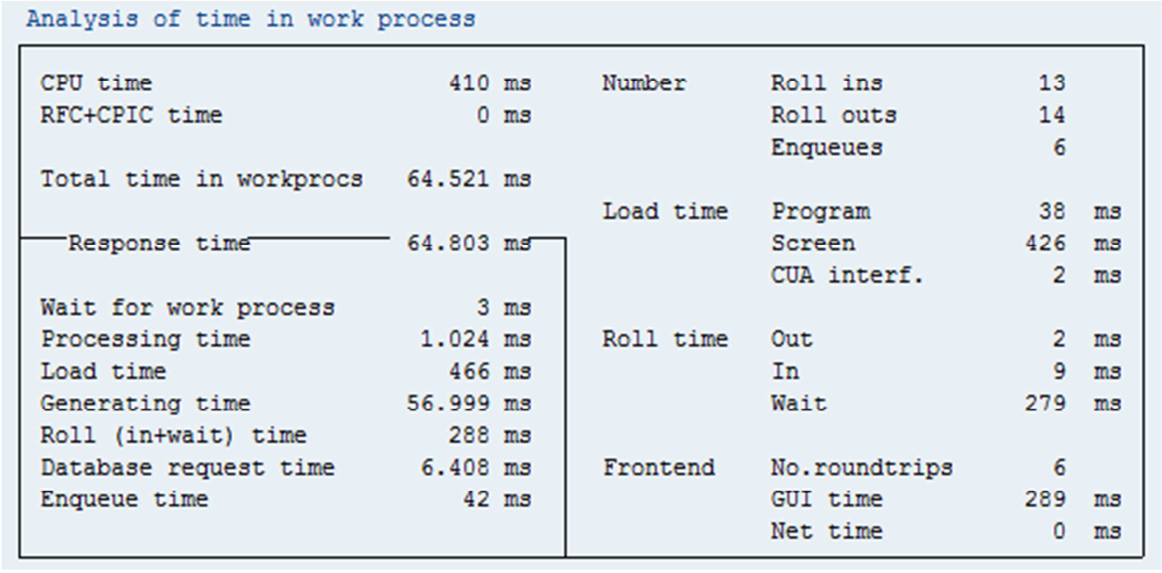


Abbildung 3-18: Darstellung der Antwortzeit in der Transaktion STAD
Quelle: Screenshot Transaktion STAD (SAP 2011i)

Unter „Processing time“ wird die Zeitmenge verstanden, welche rein für das Prozessieren von ABAP Ausdrücken verwendet wird. Die Zeitspanne, welche für das Laden des Programms aus dem Puffer oder der Datenbank benötigt wird, wird in dieser Transaktion als „Load time“ bezeichnet. Liegen Programme noch nicht in kompilierter Version vor, so werden diese automatisch vor der Ausführung kompiliert. Die dafür benötigte Zeit wird als „Generating time“ der Antwortzeit hinzugefügt. Die bereits besprochenen Zeiten „RollIn-Time“ und „ROLLED_OUT“ sind hier sowohl separat als „Roll time In“ und „Roll time Wait“ als auch in Summe als „Roll (in+wait) time“ aufgeführt. Eine weitere wichtige Komponente der Antwortzeit stellt die als „Database request time“ aufgeführte Dauer für Anfragen an die Datenbank dar. In dieser Zeit werden nur Anfragen berücksichtigt, die tatsächlich von der Datenbank beantwortet worden sind. Die Dauer von Anfragen an gepufferte Objekte ist laut der internen Dokumentation der Transaktion STAD (SAP 2011i) nicht berücksichtigt, da diese um mehrere Größenordnungen kleiner. Die letzte noch verbleibende Komponente stellt die „Enqueue Time“ dar, welche benötigt wird, um auf die Freigabe auf einen Zugriff auf geschützte Objekte zu warten. Dieser Mechanismus im SAP-WebAS-ABAP wird ausführlich in 3.4.2 besprochen.

Die in diesem Absatz besprochenen Komponenten der Antwortzeit sind nicht komplementär, sondern überlappen sich teilweise, sodass deren Summe nicht die gesamte Antwort wiedergibt. So ist z.B. die „Processing time“ Teil von anderen Zeitspannen wie „Database request time“ und „Enqueue time“.

3.7. Zusammenfassung

In diesem Kapitel wurden die Architektur des Systems und der darauf basierende Ablauf der Anfragebearbeitung analysiert. Daraus ergeben sich verschiedene Einflussfaktoren auf die

Performance eines SAP-ERP-Systems. Diese müssen auf Basis deren Einflusses bzgl. der Abbildung im Performance-Modell des Systems bewertet werden. In der vorliegenden Arbeit wird sich dafür entschieden, das in (Rolia et al. 2009b) gewählte Abstraktionslevel, um die Modellierung von Tabellenpuffer- und Sperrverwaltung zu erweitern. Die Tabellenpuffer werden im Gegensatz zu den restlichen Puffern stark durch den Workload beeinflusst, da Objekte invalidiert und verdrängt werden können. Als Beispiel werden Objekte im Programmpuffer bei Ausführung eines zusätzlichen Workloads nur einmal bei der ersten Ausführung erneuert. Im Gegensatz dazu, können die Inhalte des Tabellenpuffers bei jeder Ausführung eines Workload-Schrittes invalidiert bzw. verdrängt werden. Ebenso sind die Auswirkungen des Sperrmechanismus stark von der Ausführung des Workloads abhängig, da Sperrobjekte erst durch die gegenseitige Wechselwirkung Einfluss auf die System-Performance nehmen. Deshalb wird für deren Analyse ein Werkzeug, wie Simulation oder Lasttests benötigt, welche den Ablauf der Programme und die Abfolgen der Sperrvorgänge berücksichtigten. Damit wird Forschungsfrage 1 bzgl. der zu modellierenden Komponenten beantwortet.

4. Messwerkzeuge

Um den Lastzustand des zu untersuchenden Objekts während der Testausführung überwachen zu können, werden Monitore, auch Messwerkzeuge genannt, eingesetzt. (Jain 1991) unterscheidet dabei zwischen Software- und Hardware-Monitoren und charakterisiert diese bzgl. deren Einflussnahme auf das Testobjekt in intrusiv bzw. nicht intrusiv. Die in dieser Arbeit verwendeten Monitore beziehen sich auf Tools des Betriebssystems, siehe 4.1, und des SAP-Systems, siehe 4.2, und fallen deshalb in die Kategorie Software-Monitore. Der Einfluss auf das zu untersuchende Objekt durch das jeweilige Messinstrument wird bei dessen Darstellung aufgegriffen und diskutiert.

4.1. Betriebssystem

Grundsätzlich wird bei der Charakterisierung von Systemlast zwischen Memory-, CPU- und I/O-Last unterschieden (Lilja 2000). Diesem Ansatz folgend werden in den nächsten Absätzen die verwendeten Monitore für die einzelnen Bereiche und deren Anwendung vorgestellt.

4.1.1. Memory (RAM)

Für die Messung und die Darstellung der Verwendung von Arbeitsspeicher während der Ausführung des Testobjekts bieten UNIX-basierte Betriebssysteme grundsätzlich Tools wie „mpstat“ oder „iostat“ an. Auf der in dieser Arbeit verwendeten Power7-Hardware mit dem Betriebssystem AIX 6.1 steht noch zusätzlich das „topas“-Werkzeug (IBM 2011c) zur Verfügung. Dieses Tool bietet die Möglichkeit, aus einer virtuellen Maschine einer sogenannten logischen Partition, kurz LPAR, heraus sowohl die Memory Verwendung der LPAR als auch die der zugrunde liegenden physikalischen Hosts zu messen.

4.1.1.1. Aufzeichnung

Das Betriebssystem AIX stellt für die Aufzeichnung der Ausgabe des „topas“-Kommandos ein weiteres Tool, „topasrec“ (IBM 2011f), zur Verfügung. Folgende Parameter wurden in der vorliegenden Arbeit für die Aufzeichnung verwendet:

- -o <Name Ausgabedatei>: Spezifiziert den Namen der Ausgabedatei
- -s <Intervall>: Dauer des Zeitintervalls zwischen zwei Aufzeichnungen in Sekunden, sollte ein Vielfaches von 60 sein
- -c <Anzahl>: Anzahl der Aufzeichnungen
- -C: Aktivierung der Aufzeichnung der Memory- und CPU-Verwendung über die Grenzen der logischen Partitionen hinweg

Mittels dieser Parameter kann die Ausführungsdauer der Aufzeichnungen gesteuert werden. Damit ergibt sich die Laufzeit wie folgt:

Laufzeit = <Intervall> * <Anzahl>

4.1.1.2. Aufbereitung

Das Programm zeichnet die Daten im Binärformat aus. Um diese weiter verarbeiten zu können, existiert ein weiteres Tool, „topasout“ (IBM 2011e). Damit werden Werte der Aufzeichnung im Binärformat in einer kommaseparierten Textdatei (csv-Format) gespeichert.

4.1.1.3. Darstellung

Für die graphische Darstellung wird mit der Excel-Datei „Topas_cec_analyser_v1.0.xls“ (IBM 2011d) eine Möglichkeit angeboten, die Werte aus der csv-Datei zu lesen und in Diagrammen innerhalb einer Excel-Datei aufzubereiten.

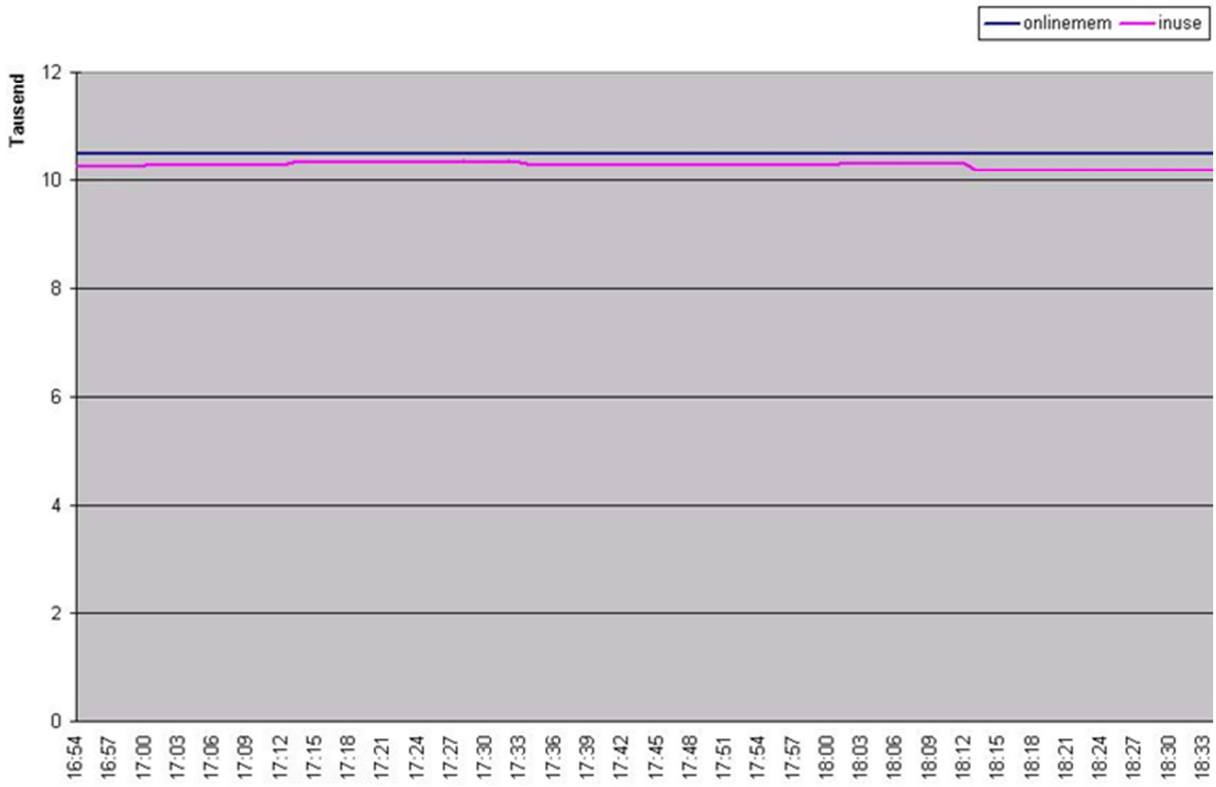


Abbildung 4-1: Exemplarische Darstellung des Memory Verbrauchs einer LPAR gemessen mit "topas"
Quelle: Screenshot eines Diagramms erzeugt mit „Topas_cec_analyser_v1.0.xls“ (IBM 2011d)

Abbildung 4-1 zeigt den Speicherverbrauch einer logischen Partition unterteilt in „onlinemem“ und „inuse“. Auf der y-Achse ist die Speichermenge in GB aufgetragen, während auf der x-Achse die Zeit vermerkt ist. Damit kann während einer Testausführung der aktuelle Verbrauch „inuse“ und die maximal verfügbare Speichermenge „onlinemem“ gemessen und die Entwicklung in Abhängigkeit der Zeit dargestellt werden.

4.1.1.4. Einfluss auf die Testausführung

Das Tool „topasrec“ ist als intrusiv einzustufen, da es Betriebssystemressourcen benötigt. Jedoch wird dieses Tool auch für die Überwachung der (virtualisierten) Systeme im regulären Betrieb verwendet, sodass der Einfluss auf die System-Performance der normalen Last zuzurechnen ist, und nicht zusätzlich durch die Überwachung des Testobjekts verursacht wird. Aus diesem Grund wurde das Tool auch innerhalb der Testausführungen in dieser Arbeit verwendet.

4.1.2. CPU

Die Toolgruppe „topas“ bietet auch die Möglichkeit, die Auslastung der CPU aufzuzeichnen und darzustellen. Jedoch konnte in dieser Darstellung nicht unterschieden werden, ob und vor allem welcher Anteil der CPU-Auslastung durch System- oder Benutzerprozessen verursacht wurde. Da diese Information jedoch wichtig für die Betrachtung und Analyse der Last ist, wurde als Monitor für die Last das Betriebssystem-Tool „mpstat“ verwendet.

4.1.2.1. Ausgabe „mpstat“

Abbildung 4-2 zeigt eine exemplarische Ausgabe des Tools „mpstat“ (IBM 2011b) und damit den Zustand (Snapshot) des Systems zum Zeitpunkt der Tool-Ausführung.

```
System configuration: lcpu=12 ent=0.3 mode=Uncapped

cpu  min  maj  mpc  int   cs  ics  rq  mig lpa sysc us sy wa id   pc  %ec  lcs
  0 5899    1    0  246  236  107   0    1 100 1486 76 20  0  4 0.10 32.1  174
  1    0    0    0   15   56   28   0    1 100  115  6  9  0 85 0.02  7.5   41
  2    0    0    0    9    0    0   0    1 100    0  0  2  0 98 0.02  6.6   10
  3    0    0    0   10    0    0   0    1 100    0  0  2  0 98 0.02  6.6   11
  4    0    0    0    0    0    0   0    1   0    0  0 64  0 36 0.00  0.0    0
  5    0    0    0    0    0    0   0    1 100    0  0 41  0 59 0.00  0.0    0
  6    0    0    0    0    0    0   0    1 100    0  0 53  0 47 0.00  0.0    0
  7    0    0    0    0    0    0   0    1 100    0  0 53  0 47 0.00  0.0    0
  8    0    0    0    0    0    0   0    1   0    0  0  7  0 93 0.00  0.1    0
  9    0    0    0    0    0    0   0    1 100    0  0  4  0 96 0.00  0.1    0
 10    0    0    0    0    0    0   0    1 100    0  0  6  0 94 0.00  0.1    0
 11    0    0    0   11    0    0   0    1 100    0  0 58  0 42 0.00  0.1   11
  U    -    -    -    -    -    -   -    -   -    -  -  -  0 73 0.22 73.4    -
ALL 5899    1    0  291  292  135   0   12 100 1601 12  4  0 84 0.16 53.1  247
--------------------------------------------------------------------------------
```

Abbildung 4-2: Exemplarische Ausgabe des Tools "mpstat" (IBM 2011b)
Quelle: Screenshot

Wie in Abbildung 4-2 dargestellt, werden in der Ausgabe die folgenden Eigenschaften für jede CPU (gekennzeichnet durch eine eindeutige CPU-ID) und aufsummiert für alle (CPU-ID „ALL“) ausgegeben. Die folgende Liste beschreibt nur die Werte, welche für die vorliegende Arbeit verwendet wurden, für eine detailliertere Beschreibung des „mpstat“-Tools sei auf die Dokumentation (IBM 2011b) verwiesen:

- us: Anteil der CPU-Zeit, welche für Benutzeranfragen verwendet wurde
- sy: Anteil der CPU-Zeit, welche für Kernel-Tasks benötigt wurde
- wa: Anteil der CPU-Zeit, in welcher die CPU wartete, während I/O Anfragen ausstanden
- id: Anteil der CPU-Zeit, in welcher die CPU ohne ausstehende I/O-Anfragen wartete

4.1.2.2. Aufzeichnung

Das Tool „mpstat“ bietet neben der Anpassung der in der Ausgabe enthaltenen Informationen auch Parameter für die Steuerung der automatischen Ausgabe über die Angabe eines Intervalls und der Anzahl von Snapshots. Mit folgendem Befehl kann damit eine

Aufzeichnung der CPU-Auslastung über den Zeitraum <Intervall> * <Count> in Sekunden in eine Datei <output> gestartet werden (IBM 2011b):

mpstat <Intervall> <Count> > <output>

4.1.2.3. Darstellung

Die im vorhergehenden Abschnitt dargestellte Aufzeichnung beinhaltet eine Aneinanderreihung von Blöcken, wie in Abbildung 4-2 dargestellt. Um die Inhalte auszuwerten und grafisch aufzubereiten, wurde eine Funktionalität in die Analysekomponente integriert, welche die Ausgabe parst und die Werte einem Zeitpunkt zuordnet. Für die grafische Darstellung wurde analog zu „topasout“ eine Funktion implementiert, welche die Werte als kommaseparierte Datei speichert, sodass eine Diagramm-Darstellung mittels Microsoft Excel ermöglicht wird.

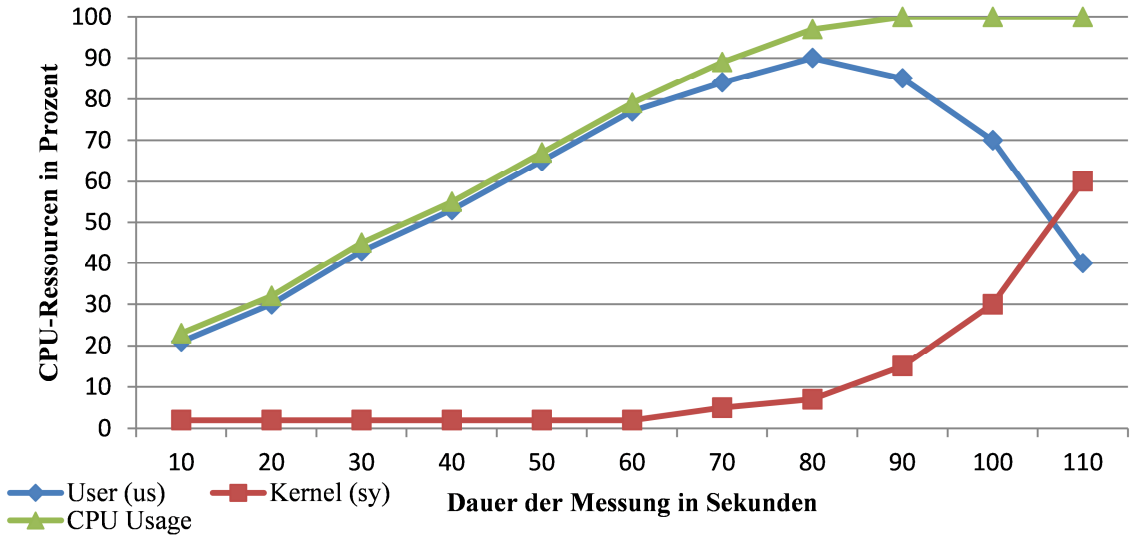


Abbildung 4-3: Grafische Aufbereitung der "mpstat"- Daten als csv-Datei in Microsoft Excel
Quelle: Eigene Darstellung

Abbildung 4-3 zeigt exemplarisch den prozentualen CPU-Verbrauch, aufgeteilt auf Gesamt-CPU, Benutzer-CPU und Kernel-CPU in Abhängigkeit der Zeit. Zusätzlich wurde eine Funktionalität implementiert, welche die Werte für größere, durch den Anwender zu bestimmende, Zeitintervalle aufsummiert und das arithmetische Mittel der Komponenten „us“, „sy“ und „ALL“ berechnet.

4.1.2.4. Einfluss auf die Testausführung

Wie bereits in 4.1.2.4 diskutiert, ist auch das „mpstat“-Tool als intrusiv zu bewerten, da es für die Aufzeichnung der Performance-Werte Ressourcen des Betriebssystems benötigt. Analog zum „topas“-Tool wird „mpstat“ ebenso für das Monitoring im Regelbetrieb verwendet und stellt somit keine zusätzliche Systembelastung durch die Aufzeichnung während des Testverlaufs dar.

4.1.3. I/O

Für die Messung und Aufzeichnung der Last auf das I/O-System wird das Tool „iostat“ (IBM 2011a) verwendet. Dies findet sich auf allen gängigen UNIX-basierten Betriebssystemen. Um die I/O-Last, welche durch den in dieser Arbeit verwendeten Workload erzeugt wird, zu messen, müssen die Lese- und Schreibvorgänge auf das I/O-System gemessen werden. Im Versuchsaufbau wurde das Storage-System über Fibre-Channel-Adapter als Festplatten eingebunden. Damit können die Lese- und Schreibvorgänge auf die, auf Dateisystemebene abgelegten, Datenbankdateien über das „iostat“-Tool gemessen und aufgezeichnet werden.

4.1.3.1. Ausgabe „iostat“

Abbildung 4-4 zeigt exemplarisch die Ausgabe des „iostat“-Befehls:

Disks:	% tm_act	Kbps	tps	Kb_read	Kb_wrtn
hdisk1	5.9	801.8	33.0	2476618170	7899041873
hdisk2	0.1	3.6	0.9	238988	46505255
hdisk3	2.8	711.9	25.1	1308699450	7903194285
hdisk0	0.1	3.8	0.9	2612151	46497191

Abbildung 4-4: Exemplarische Ausgabe des "iostat"-Befehls
Quelle: Screenshot Ausgabe „iostat“-Kommando (IBM 2011a)

Für die Beschreibung der Last werden nur die folgenden Beschreibungen verwendet:

- % tmp_act: Anteil der Zeit, in welcher das I/O-System aktiv war
- Kbps: Übertragungsrate (Lesen und Schreiben) in Kilobyte pro Sekunde
- tps: Anzahl der I/O-Vorgänge pro Sekunde

Die beiden letzten Spalten „Kb_read“ und „Kb_wrtn“ zeigen die Gesamtanzahl von geschriebenen und gelesenen Kilobytes auf die „disks“ an und können so nicht als Indikatoren für den Lastzustand des I/O-Systems zu einem bestimmten Zeitpunkt verwendet werden.

4.1.3.2. Aufzeichnung mit „iostat“

Analog zum „mpstat“-Tool bietet auch der „iostat“-Befehl die Möglichkeit eine Serie von Snapshots mittels deren Anzahl und der dazwischen liegenden Zeitspanne in Sekunden aufzuzeichnen. Mit dem folgenden Befehl wurde die Last auf das I/O-System in der vorliegenden Arbeit aufgezeichnet und in einer Datei abgelegt:

Iostat <Intervall> <Anzahl> > <Ausgabe>

4.1.3.3. Darstellung I/O-Last

Um die Entwicklung der Last grafisch aufzubereiten, wurde für die Ausgabe des „iostat“-Befehls ebenso eine Funktionalität in die Analysekomponente der vorliegenden Arbeit integriert. Der Inhalt der Ausgabedatei besteht aus einer Aneinanderreihung der Ausgaben, wie in Abbildung 4-4 dargestellt. Diese Ausgabe wird geparst und die Werte der

identifizierten Spalten werden einem Zeitstempel zugeordnet abgelegt. Diese so aufbereiteten Daten können nun wieder im csv-Format als Datei exportiert und damit in Microsoft Excel importiert und als Diagramm dargestellt werden. Zusätzlich zu der Darstellung der I/O-Operationen einer Testausführung, wie beispielhaft in Abbildung 4-5 gezeigt, können mehrere Testausführungen kombiniert werden. Dabei werden die Werte einer Testausführung mittels der Bildung eines Median aggregiert. Die so zusammengefassten I/O-Operationen jeder Testausführung können ebenfalls als csv-Datei exportiert und nach einem Microsoft-Excel-Import als Diagramm dargestellt werden.

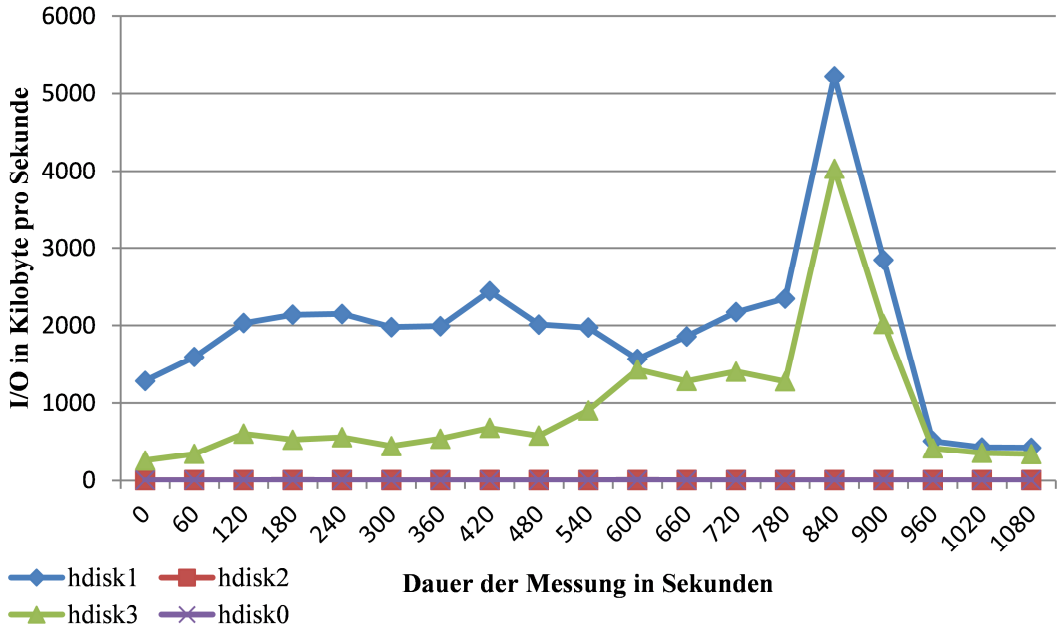


Abbildung 4-5: Grafische Darstellung der I/O-Operationen als Diagramm in Microsoft Excel in Kilobyte pro Sekunden
Quelle: Eigene Darstellung

4.1.3.4. Auswirkungen auf die Testausführung

Analog zu den Tools „topas“ und „mpstat“ wird auch „iostat“ für das Monitoring im Regelbetrieb verwendet, sodass dessen Einsatz für die Vermessung des Testfalls trotz intrusiver Charakteristik des Tools an sich, das Testergebnis nicht verfälscht.

4.2. SAP-System

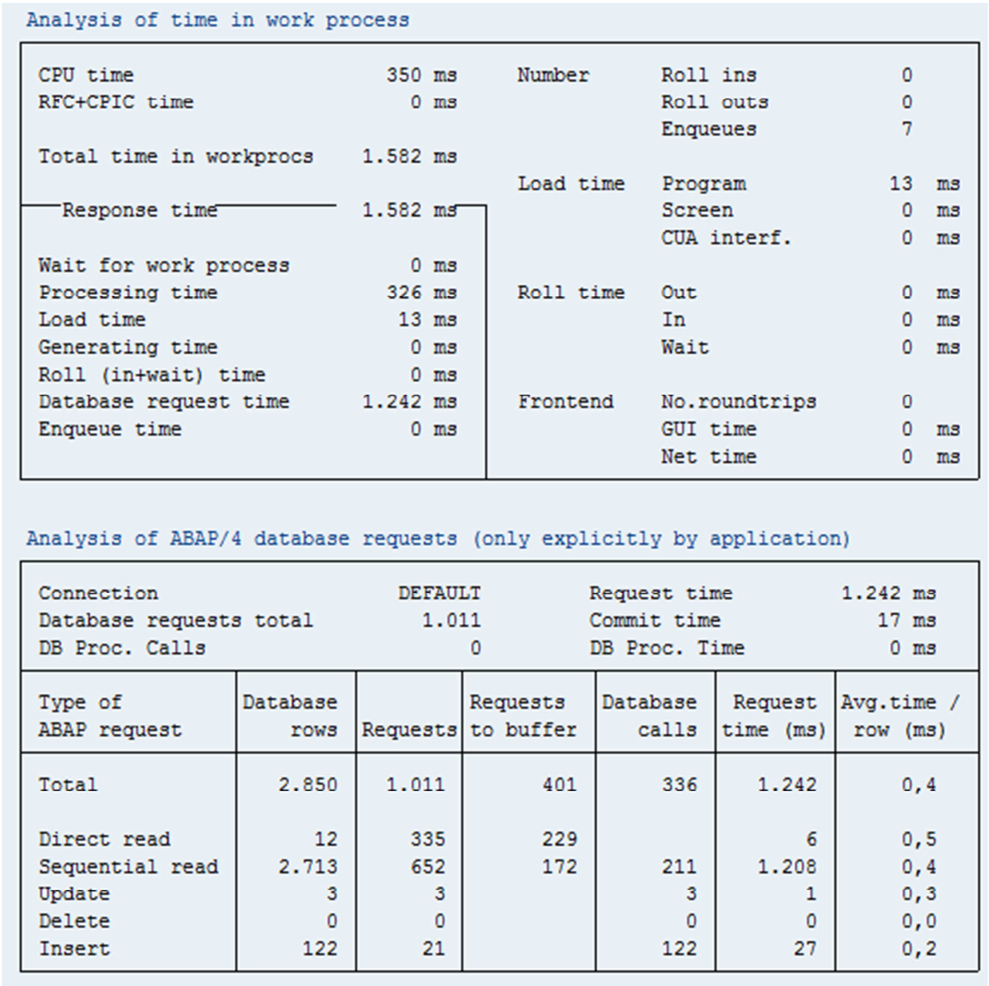
Der SAP-WebAS-ABAP bietet eine große Anzahl an Messinstrumenten für die Analyse und das Monitoring der Performance. Die Oberfläche dieser Tools ist als Transaktion im Sinne der SAP implementiert und kann nur über die SAPGui aufgerufen werden. Da dies jedoch die Auswertung groß angelegter automatisierter Testläufe stark beeinträchtigt, wurden im Rahmen dieser Arbeit zusätzlich zu den SAP-Transaktionen noch remotefähige Funktionsbausteine identifiziert und über die WebService-Schnittstelle in den Prototypen integriert.

4.2.1. Transaktion STAD

Wie bereits kurz in 3.6 aufgezeigt, bietet die Transaktion STAD eine detaillierte Aufschlüsselung der Komponenten der Antwort- bzw. Bearbeitungszeit. Für die Performance-Analyse sind jedoch, zusätzlich zu den gemessenen Zeiten, weitere Informationen über die Anzahl und Ausprägung von Systemereignissen nötig. Die „Performance Records“, welche von der Transaktion STAD ausgelesen werden, liefern deshalb zusätzlich eine Vielzahl an Informationen, darunter die Anzahl von Roll-ins, Roll-outs und Enqueue-Vorgängen sowie die Aufschlüsselung aller Datenbank- und Pufferzugriffe innerhalb eines Programmablaufs. Parallel dazu wird, wie in Abbildung 4-6 beispielhaft dargestellt, auch die verbrauchte CPU-Zeit durch den Kernel gesammelt und in dieser Transaktion angezeigt, welche über alle Komponenten der Antwortzeit verteilt ist.

4.2.1.1. Antwortzeit- bzw. Bearbeitungszeitkomponenten

Die Komponenten der Antwortzeit in der Transaktion STAD wurden bereits in Absatz 3.6 erläutert. Darüber hinaus bietet die Transaktion Informationen über die Anzahl von Roll-In, Roll-out und Enqueue-Vorgängen, wobei vor allem die letzteren, wie in 3.4.2 beschrieben, einen großen Einfluss auf die Performance des SAP-Systems haben können.



Analysis of time in work process

CPU time	350 ms	Number	Roll ins	0	
RFC+CPIC time	0 ms		Roll outs	0	
			Enqueues	7	
Total time in workprocs	1.582 ms				
		Load time	Program	13	ms
Response time	1.582 ms		Screen	0	ms
			CUA interf.	0	ms
Wait for work process	0 ms				
Processing time	326 ms	Roll time	Out	0	ms
Load time	13 ms		In	0	ms
Generating time	0 ms		Wait	0	ms
Roll (in+wait) time	0 ms				
Database request time	1.242 ms	Frontend	No.roundtrips	0	
Enqueue time	0 ms		GUI time	0	ms
			Net time	0	ms

Analysis of ABAP/4 database requests (only explicitly by application)

Connection	DEFAULT	Request time	1.242 ms
Database requests total	1.011	Commit time	17 ms
DB Proc. Calls	0	DB Proc. Time	0 ms

Type of ABAP request	Database rows	Requests	Requests to buffer	Database calls	Request time (ms)	Avg.time / row (ms)
Total	2.850	1.011	401	336	1.242	0,4
Direct read	12	335	229		6	0,5
Sequential read	2.713	652	172	211	1.208	0,4
Update	3	3		3	1	0,3
Delete	0	0		0	0	0,0
Insert	122	21		122	27	0,2

Abbildung 4-6: Transaktion STAD – Bearbeitungszeit und Datenbankzugriffe
Quelle: Screenshot Transaktion STAD (SAP 2011i)

Des Weiteren werden einige der bereits besprochenen Antwortzeitkomponenten weiter aufgeschlüsselt. Für die „Load time“ werden die Zeitspannen für das Laden von Kompilaten, Screens und CUA-Interfaces unterschieden. Auch die Anzahl an Roll-In-Vorgängen ist eine wichtige Information, da diese während einer Programmausführung nur auftreten, wenn externe Programme aufgerufen werden, und diese nicht innerhalb von 500 Millisekunden antworten.

4.2.1.2. Datenbank und Pufferzugriffe

Weitere ausführliche Informationen werden auch über die Anfragen an die Datenbank bzw. die Puffer bereitgestellt. Diese Anfragen werden in 5 Typen unterschieden:

- Direct read: Direkter lesender Zugriff auf einen einzelnen Datensatz (ABAP Kommando „select single“)
- Sequential Read: Lesender Zugriff auf eine Menge von Datensätzen
- Update: Modifizierender Zugriff auf einen bereits bestehenden Datensatz
- Delete: Löschen eines bestehenden Datensatzes
- Insert: Einfügen eines neuen Datensatzes

Für jeden dieser Typen, sowie für alle in Summe, werden die folgenden Informationen bereitgestellt:

- Database rows: Anzahl der benutzen Datensätze
- Requests: SQL-Anfragen an die Datenbank
- Requests to buffer: Anzahl von SQL-Anfragen, welche durch die Puffer beantwortet werden konnten
- Database calls: Anzahl an tatsächlichen Operationen (Select, Insert, Update und Delete)
- Request time (ms): Dauer der Anfragen
- Avg.time / row (ms): durchschnittliche Anfragedauer pro benutztem Datensatz

Die „Request time“ setzt sich hierbei ausschließlich aus Anfragen auf die Datenbank zusammen. Die Dauer der Pufferzugriffe wird hierbei ignoriert, da diese um mehrere Größenordnungen geringer und damit nicht von Bedeutung, ist. Darüber hinaus wird für die Datenbank noch die Anzahl und die Dauer von „DB Proc Calls“, sogenannten „Stored Procedures“ sowie die „Commit time“ angezeigt, welche für die Speicherung der Daten in der Datenbank benötigt wurde.

4.2.1.3. Automatisierter Zugriff auf STAD-Performance-Records

Um die Informationen der statistischen „Performance records“ für eine große Anzahl von Programmausführungen automatisiert auslesen und analysieren zu können, ist es notwendig,

eine alternative Schnittstelle zur SAPGui zu identifizieren. Über die beschriebene WebService-Schnittstelle können remotefähige und als WebService exportierte Funktionsbausteine aufgerufen werden.

Der Funktionsbausteins „SWNC_STATREC_READ“ wurde für das Auslesen der STAD-Records identifiziert. Für die Spezifikation der auszulesenden Performance Records können folgende Übergabeparameter angegeben werden:

- ASTAT_FILE: Name der Anwendungsstatistikdatei. Default: Lokale Datei
- FLAG_READ_ASTAT: Flag, um das Lesen der Anwendungsstatistiken zu aktivieren
- FLAG_READ_STAT: Flag, um das Lesen der Performancestatistiken zu aktivieren
- NO_BUFFER_FLUSH: Flag, um das Leeren des Statistikpuffers zu verhindern
- NO_OF_RECORDS: Anzahl der zu lesenden Sätze. Default: alle (-1)
- READ_CLIENT: Mandant, für welchen die Performance Daten ausgegeben werden sollen. Default: alle
- READ_DIALOGSTEPID: Dialogstep-ID, für welche die Performance Daten ausgegeben werden sollen
- READ_END_DATE: Enddatum der auszulesenden Performance Records
- READ_END_TIME: Endzeitpunkt der auszulesenden Performance Records
- READ_START_DATE: Startdatum der auszulesenden Performance Records
- READ_START_TIME: Startzeitpunkt der auszulesenden Performance Records
- READ_TCODE: Filterung der Performance Records nach Transaktionsname
- READ_TIMEZONE: Bestimmung der Zeitzone für die Beurteilung der Start- und Endzeitpunkte.
- READ_TRANSIDS: Filterung der Performance Records nach TransaktionsGuiID
- READ_USERNAME: Filterung nach Benutzernamen
- READ_WORKPROCESS: Filterung nach Nummer des Work-Prozesses
- STATISTIC_FILE: Auswahl einer bestimmten Datei mit gespeicherten STAD-Records

Mit dieser Parametrisierung ist es möglich, Performance-Daten aus den Kernel auszulesen und diese in großer Zahl automatisiert zu verarbeiten. Hierbei ist zu beachten, dass ein Datensatz aus 80 Einzelwerten besteht. Deshalb ist es nötig, die auszuwertenden Sätze genau zu spezifizieren, um unnötige Belastungen für Netzwerk und SAP-System zu vermeiden. Des Weiteren sind diese Felder nicht dokumentiert, bzw. es konnte keine Dokumentation

gefunden werden. Um nun die Felder der vom Baustein übermittelten Datensätze den Einträgen in der STAD-Transaktion zuzuordnen, wurden die Feldnamen und deren Inhalte anhand einer signifikanten Menge von Performance-Records mit den Inhalten der STAD-Transaktion abgeglichen und damit einander zugeordnet:

Name in der STAD	Feldname im Funktionsbaustein
CPU time	CPUTI
RFC+CPIC time	RFCTI
Total time in workprocess	= RESPTI - QUEUETI - ROLLED_OUT
Response time	RESPTI
Wait for work process	QUEUETI
Processing time	PROCTI
Load time	= REPLOADTI + CUALOADTI
Generating time	GENERATETI
Roll (in+wait) time	RFCTI
Database request time	DBREQTIME
Enqueue time	LOCKTI
Number Roll ins	ROLLINCNT
Number Roll outs	ROLLOUTCNT
Enqueues	LOCKCNT
Load time Program	REPLOADTI
Load time CUA Interf.	CUALOADTI
Roll time Out	ROLLOUTTI
Roll time In	ROLLINTI
Roll time Wait	ROLLED_OUT
Commit time	COMMITTI
DB Proc. Time	DBPREQTIME
Request time	DBREQTIME
DB Proc. Calls	DBPCOUNT()

Tabelle 4-1: Zuordnung von SWNC_STATREC_READ- zu STAD-Feldern
Quelle: Eigene Tabelle

Wie in Tabelle 4-1 dargestellt, werden einige Angaben der Transaktion STAD nicht direkt vom Kernel erzeugt, sondern ergeben sich aus der Kombination mit anderen Werten. In diesen Fällen wurde die für die Berechnung verwendete Formel angegeben. Tabelle 4-2 zeigt die Zuordnung der Felder des Bausteins in Analogie zum unteren Bereich von Abbildung 4-6. Dabei stellen die linke Spalte und die Überschrift von Tabelle 4-2 die Matrix aus Abbildung 4-6 dar, die restlichen Felder der Tabelle beinhalten die Namen der Felder des Bausteins mit den entsprechenden Werten.

Type of ABAP Request	Database rows	Requests	Requests to buffer	Database calls	Request time (ms)
Direct Read	READDIRREC	READDIRCNT	READDIRBUF		READDIRTI
Sequential Read	READSEQREC	READSEQCNT	READSEQBUF	PHYREADCNT	READSEQTI
Update	UPDREC	UPDCNT		PHYUPDCNT	UPDTI
Delete	DELREC	DELCNT		PHYDELCNT	DELTI
Insert	INSREC	INSCNT		PHYINSCNT	INSTI

Tabelle 4-2: Zuordnung von SWNC_STATREC_READ- zu STAD-Feldern für Datenbankzugriffe
Quelle: Eigene Tabelle

Darüber hinaus bietet der Baustein noch weitere Angaben über das zugehörige Programm. Tabelle 4-3 zeigt die für die vorliegende Arbeit benötigten Daten:

Feldname	Beschreibung
REPORT	Name
ACCOUNT	Benutzer
MANDT	Mandant
STARTDATE	Startdatum
STARTTIME	Startzeitpunkt
ENDDATE	Enddatum
ENDTIME	Endzeitpunkt

Tabelle 4-3: Angaben im Baustein SWNC_STATREC_READ zum ausgeführten Programm
Quelle: Eigene Tabelle

4.2.1.4. Beurteilung des Messinstruments hinsichtlich des Einflusses auf die Performance

Nach (Jain 1991) sind Messwerkzeuge, sogenannte Monitore, bezüglich ihrer Beeinflussung des Testobjekts intrusiv bzw. nicht intrusiv. Intrusive Monitore verursachen durch die eigene Ressourcenverwendung zusätzliche Last auf das Testobjekt. Im Falle der SAP-Performance-Records ist die Ermittlung der Performance-Werte und deren Ablage in einer Datei auf Betriebssystemebene als intrusiv zu bewerten, da der SAP-Kernel diese Werte ermittelt und dafür zusätzliche CPU- und Hauptspeicher benötigt. Durch das Schreiben dieser Werte in eine Datei werden weitere I/O-Ressourcen benötigt. Im Gegensatz zu anderen Messwerkzeugen des SAP-WebAS-ABAP kann das Mitschreiben der Performance Records nicht abgestellt werden und ist in jedem SAP-System auf Basis des SAP-WebAS-ABAP aktiv. Aus diesem Grund wird es im Rahmen dieser Arbeit als nicht intrusiv eingestuft, da es nicht ausschließlich für diese Messungen aktiviert wurde. Im Gegensatz dazu beeinflusst das Auslesen der Performance-Werte die Systemressourcen, sodass dieser Vorgang nicht während des Testlaufs durchgeführt werden darf.

4.2.2. Verwendung Speicherressourcen durch die Puffer

4.2.2.1. Messinstrument

Um die Informationen aus der Transaktion „ST02“ in Testserien zu verwenden, wurde der remotefähige Funktionsbaustein „SAPTUNE_GET_SUMMARY_STATISTICS“ identifiziert

und verwendet. Als WebService implementiert liefert dieser detaillierte Informationen über den aktuellen Zustand der Puffer im SAP-WebAS-ABAP. Dies ist exemplarisch in Tabelle 4-4 dargestellt.

Buffer	Used [KB]	Hitratio [%]	Alloc [KB]	Freesp [KB]	Freespace [%]	Dir Size	FreeDir [%]	Swaps	DB Accs
TTAB	486	95,3646	6797	5214	91,47369	19990	91,47073	0	59536
FTAB	984	97,4883	31564	29018	96,720215	19990	98,554276	5246	20450
SNTAB	16	97,8395	3625	2984	99,46667	4997	98,81929	0	3795
IRBD	196	44,2553	6625	5804	96,73333	4997	94,576744	672	17484
TABL	2835	99,2928	29297	24888	89,773834	5000	99,04	45	116574
TABLP	101	86,4173	10000	9711	98,97064	500	97	5	130183
PXA	278248	99,0965	300000	72	0,025868416	75000	91,305336	1259	117483
CUA	78	99,5748	3000	2309	96,7323	1500	99,73334	0	116
PRES	15	99,6205	4297	4078	99,63352	2000	99,55	0	177
CALE	211	100	488	267	55,857742	200	49	0	1020
EIBUF	9	91,0683	4096	3272	99,72569	2000	99,95	0	0
OTR	0	100	4096	3281	100	2000	100	0	0
ESM	0	99,5268	4096	3281	100	2000	100	0	0

Tabelle 4-4: Ausgabe des Funktionsbausteins SAPTUNE_GET_SUMMARY_STATISTICS
Quelle: (Gradl et al. 2011a)

Im SAP-System wird die Größe der Puffer in Kilobyte und Anzahl von Einträgen definiert. Ein Puffer kann also voll sein, obwohl noch Platz übrig ist, indem die maximale Anzahl an Einträgen überschritten wird. Beim Starten des SAP-WebAS-ABAP werden die Puffer mit initialisiert und belegen die volle konfigurierte Größe im Hauptspeicher.

Der Web-Service liefert die aktuell von allen Puffern benutzte Menge an Hauptspeicher (Used[KB]), die Trefferquote, die insgesamt allokierte Menge an Speicher (Alloc[KB]), den prozentualen Anteil an freiem Speicher (Freespace[%]), die Anzahl der maximal verfügbaren Einträge (DirSize), den prozentualen Anteil an freien Einträgen (FreeDir[%]), die Anzahl ersetzter bzw. neu geladener Einträge (Swaps) und die Anzahl der Datenbankzugriffe, die zum Füllen der Puffer benötigt wurden (Schneider 2008).

Die Trefferquote ist dabei durch die folgende Formel definiert:

$$x = \frac{logical\ accesses - physical\ accesses}{logical\ accesses} * 100$$

Ein Teil des Speichers in den jeweiligen Puffern wird dabei für die eigene Verwaltung verwendet. Dies ist der Grund folgender Differenz:

$$Used\ [KB] + Freesp\ [KB] \neq Alloc\ [KB]$$

Obwohl die Daten nicht über die SAPGui angezeigt wurden, wurde für die Ausführung des Funktionsbausteins die Infrastruktur des SAP-WebAS-ABAP verwendet. Damit wurde ein Teil der vom Baustein zurückgelieferten Belegung der Puffer durch diesen selbst verursacht. Um diese Beeinflussung durch das Messinstrument auszugleichen, wurde der folgende Ansatz entwickelt und evaluiert.

4.2.2.2. Messung

Die Verwendung von Pufferressourcen hängt stark vom jeweiligen Workload, welcher vom SAP-System abzuarbeiten ist, ab. Um alle Auswirkungen zu integrieren, muss die Analyse für den jeweiligen Workload durchgeführt werden. Da das verwendete Messinstrument, wie bereits angesprochen, die Infrastruktur des SAP-WebAS-ABAP benutzt, sind diese Messungen als intrusiv (Jain 1991) zu bezeichnen und daher beeinflusst. Um signifikante Messergebnisse zu erzeugen, muss das Maß dieser Beeinflussung identifiziert und in die Analyse integriert werden.

Abbildung 4-7: Ablauf der Messung
Quelle: Eigene Darstellung

Dazu müssen die Puffer zuerst geleert werden. Dies geschieht durch Invalidierung sämtlicher Inhalte. Während des in Abbildung 4-7 dargestellten Vorgangs muss sichergestellt sein, dass auf dem ganzen System kein anderer Workload, von Benutzern oder Hintergrundprozessen erzeugt, vom System abgearbeitet wird. Um die Puffer zu leeren, wurde der SAP-Funktionscode „$SYNC“ identifiziert und verwendet, da „dieser alle Puffer des Applikationsserver zurücksetzt“ (SAP 2009). Um die Beeinflussung durch das Messinstrument zu identifizieren, wurden das Messinstrument und damit der Funktionsbaustein SAPTUNE_GET_SUMMARY_STATISTICS mehrmals in Folge ausgeführt. Dabei wurde erkannt, dass der für die Verwaltung des Puffers verwendete Speicher nicht nur den verfügbaren Speicher reduzierte, sondern dass dies nicht sofort geschieht, sondern nach einer gewissen Anzahl von Ausführungen des Messinstruments und damit einer gewissen Anzahl von Zugriffen auf gepufferte Objekte. Dieses Verhalten, die sogenannte Karenz (Schneider 2008), wurde in der Pufferverwaltung implementiert, um im Falle falscher Konfiguration bzw. fehlerhafter Programmierung große Mengen von Pufferinvalidierungen zu verhindern. Somit bedarf es einer im SAP-System konfigurierbaren Anzahl von Anfragen auf gepufferte Objekte, bis diese Anfragen dann direkt aus dem Puffer und nicht mehr von der Datenbank direkt beantwortet werden. Diese festgestellten Eigenschaften sind in Abbildung 4-8 dargestellt:

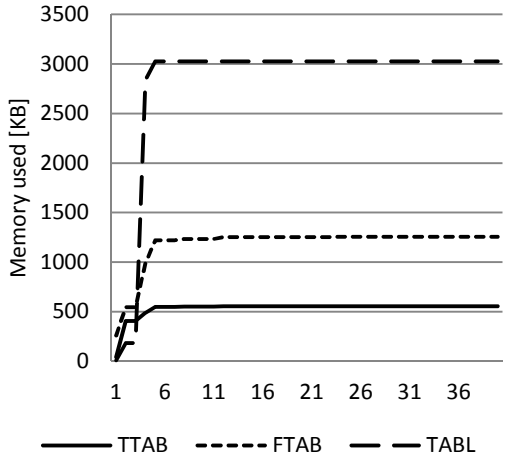


Abbildung 4-8: Pufferverwendung durch das Messinstrument
Quelle: (Gradl et al. 2011a)

Abhängig von der Zugriffshäufigkeit des Messinstruments auf die einzelnen Puffer benötigt es eine unterschiedliche Anzahl von Ausführungen (dargestellt auf der x-Achse in Abbildung 4-8), bis die Puffer verwendet und damit die Verwaltungsinformationen angelegt werden. Bei der Messung der verwendeten Pufferressourcen durch einzelne Schritte des Workloads muss dies berücksichtigt werden, da andernfalls die Messwerte eine signifikante Standardabweichung, wie in Tabelle 4-5 dargestellt, aufweisen.

Buffer	Durchschnitt	Standardabweichung
TTAB	3622,15	20,61240046
FTAB	26161,1	2306,299152
SNTAB	272,75	2,953588362
IRBD	1898,2	32,60222789
TABL	27628,95	3,605186323
TABLP	991,5	27,0972906
PXA	277185,55	444,9519755
PRES	116,35	3,013565819

Tabelle 4-5: Verwendung von Pufferressourcen durch das Messinstrument
Quelle: (Gradl et al. 2011a)

Betrachtet man nun nur die Werte des Messinstruments, nachdem die Puffer vollständig initialisiert waren, zeigten die Messwerte eine konstante Ausprägung ohne Abweichung. Damit können diese Werte als Grundniveau der Pufferbelastung bei der Vermessung des Bedarfs an Pufferressourcen von Workload-Schritten dienen.

4.2.2.3. Vermessung eines Workload-Schrittes

Die Vermessung der einzelnen Workload-Schritte folgt nun dem gleichen Prinzip wie die Vermessung des Messinstruments selbst. Wie Abbildung 4-9 zeigt, werden zuerst die Einflüsse vorheriger Workloads durch den OK-Code „/$SYNC“ (SAP 2011b) entfernt.

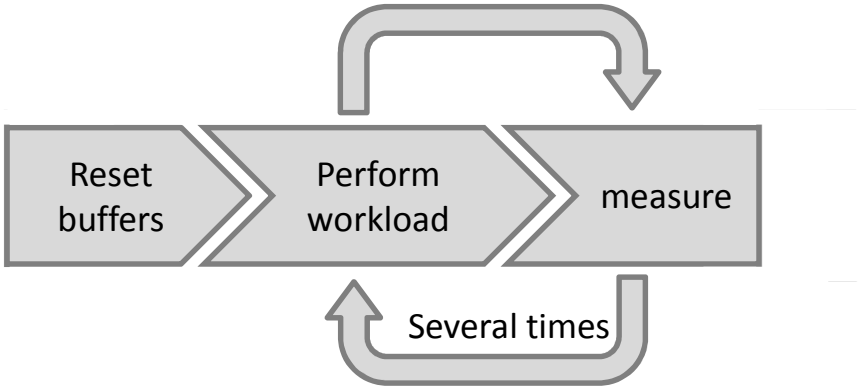


Abbildung 4-9: Vorgehen für die Messung der Puffernutzung eines Workload-Schrittes
Quelle: (Gradl et al. 2011a)

Um die bereits beschriebenen Einflüsse der Pufferverwaltung in SAP-WebAS-ABAP auszugleichen und sicherzustellen, dass die Puffer auch vollständig initialisiert und benutzt wurden, muss der Workload mehrmals ausgeführt werden. Da Workload Daten im System erzeugen kann, welche dann wiederum in den Puffer geladen werden, kann im Zuge der Messung die Menge benötigter Pufferressourcen ansteigen. Grundsätzlich ist dieser Einfluss als eher gering einzuschätzen, da Tabellen nur dann für die Pufferung empfohlen sind, wenn der Anteil an modifizierenden Datenzugriffen < 1 Prozent ist (SAP 2011l). Bleibt die gemessene Menge an benötigten Pufferressourcen konstant oder steigt linear an, so kann die Messung beendet werden, da die erwähnten Einflüsse der SAP-Pufferverwaltung eliminiert sind.

4.2.2.4. Auswertung der Messergebnisse

Im Messzyklus wurden das Messinstrument und der Workload-Schritt mehrere Male nacheinander ausgeführt. Dies hat zur Folge, dass die Puffer nun sowohl die Daten des Messinstruments als auch die des Workload-Schritts beinhalten.

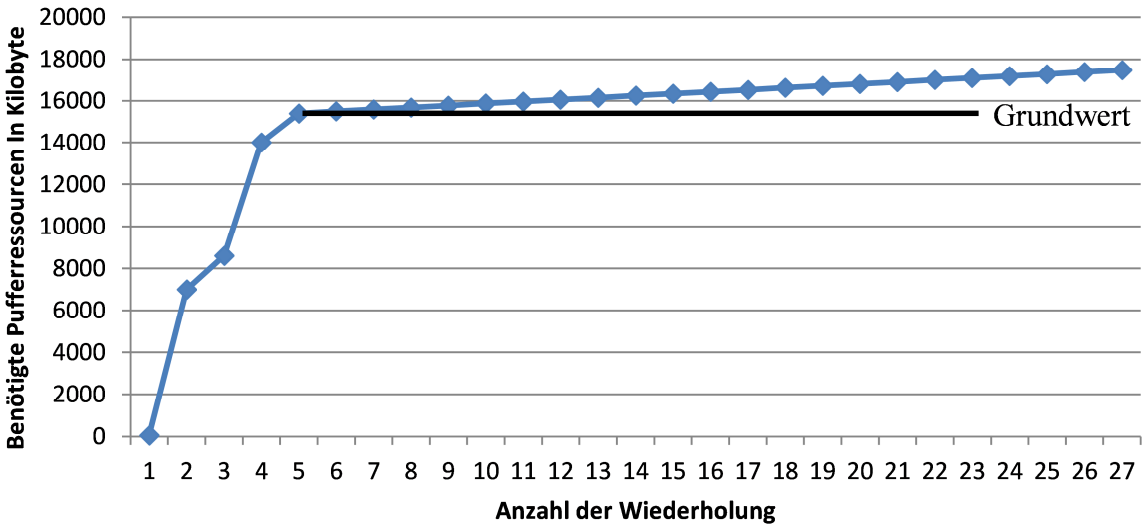


Abbildung 4-10: Identifikation benötigter Pufferressourcen anhand eines beispielhaften Workload-Schrittes
Quelle: (Gradl et al. 2011a)

Um signifikante Ergebnisse zu erhalten, muss nun die zuvor identifizierte, vom Messinstrument benötigte Menge an Pufferressourcen von den gemessenen Ergebnissen subtrahiert werden.

Abbildung 4-10 zeigt den für einen exemplarischen Workload-Schritt ermittelten Bedarf an Ressourcen im Puffer. Dieser Workload-Schritt erzeugt Daten, welche bei der nächsten Ausführung wieder in den Puffer geladen werden. Damit steigt die Menge benötigter Ressourcen ab der 6. Wiederholung linear an. Der Anfang dieses linearen Anstiegs kann nun für die Bestimmung des Ressourcenbedarfs als Grundwert verwendet werden. Diesem linearen Anstieg liegt eine Expansionsrate zugrunde, welche ebenfalls aus den Messwerten ermittelt und für die Bestimmung des Ressourcenbedarfs verwendet werden kann. Dieser ergibt sich zusammen mit der Anzahl geplanter Ausführungen des Workloads mit folgender Formel:

$$buffer\ demand = base\ value + (number\ of\ planned\ executions * expansion\ rate)$$

Diese Methode hängt nicht von einem speziellen Workload ab, da alle Programme im SAP-WebAS-ABAP über die gleichen Schnittstellen auf die Puffer zugreifen und diese nicht vom Programm selbst ausgelöst werden. Das SAP-System stellt damit eine Abstraktionsschicht zur Verfügung, die entscheidet, ob vom Programm angefragte Daten direkt von der Datenbank oder aus den Puffern gelesen werden.

4.2.2.5. Evaluation anhand einer Fallstudie

Der für die Evaluation verwendete Workload basiert auf einer Fallstudie (Weidner 2006) des SAP-University-Alliances-Programms (SAP 2012b) und beinhaltet verschiedene Schritte aus dem SAP-Modul „Produktionsplanung“. Insgesamt besteht der Geschäftsprozess in dieser Fallstudie aus 17 Schritten, welche Kernanwendungen dieses Moduls beinhalten und häufig in realen Systemen verwendet werden.

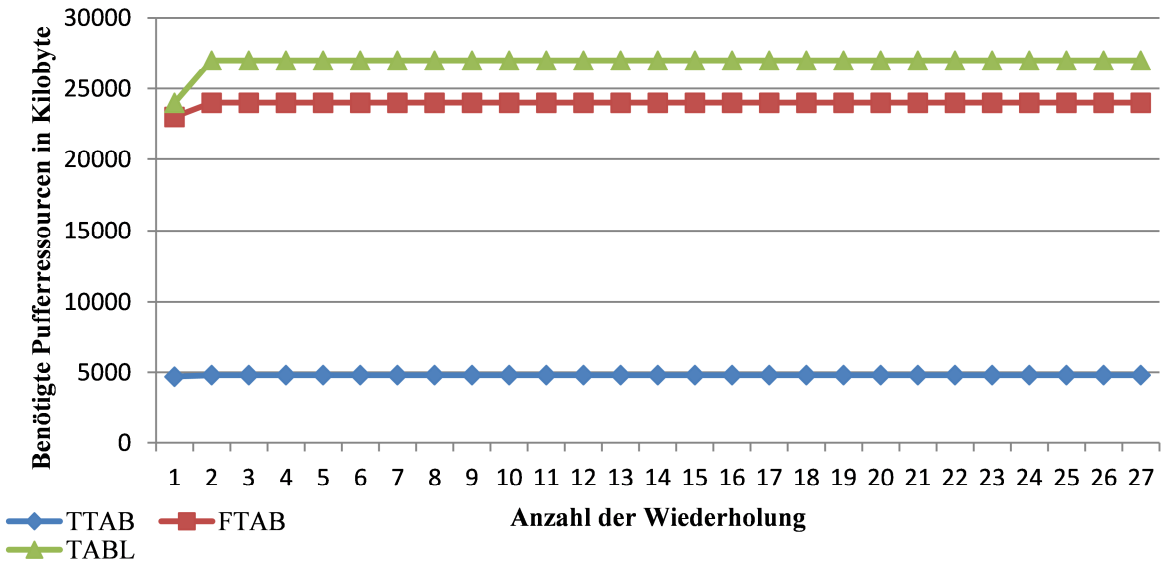


Abbildung 4-11: Durch das Messinstrument benutzte Pufferressourcen
Quelle: (Gradl et al. 2011a)

Folgt man dem in den vorherigen Absätzen entwickelten Vorgehen, so startet die Messung mit der Ausführung des OK-Code „/$SYNC“, um die Puffer im System zu invalidieren. Danach wird die Menge der durch das Messinstrument benötigten Pufferressourcen ermittelt, siehe Abbildung 4-11.

In dieser Teststellung wurde eine konstante Benutzung von Pufferressourcen gemessen. Nach dem Zurücksetzen der Puffer wurde der bereits beschriebene Workload gefolgt vom Aufruf des Messinstruments SAPTUNE_GET_SUMMARY_STATISTICS wiederholt, insgesamt 36-mal, ausgeführt. Wie Abbildung 4-11 zeigt, wurde die konstante Ressourcenbenutzung bereits nach 5-maliger Ausführung erreicht. Dies wurde durch den großen Umfang des Workloads verursacht, der bereits bei einmaliger Ausführung eine Vielzahl an Anfragen an die Puffer richtet. Eine genauere, technische Beschreibung dieses Workloads findet sich in Kapitel 6.1.

Die gemessenen Ressourcenbedarfe in den Puffern, wie in Tabelle 4-5 dargestellt, zeigten eine signifikante Abweichung vom arithmetischen Mittel, da die identifizierten Effekte der Pufferverwaltung, sowie die vom Messinstrument benötigten Ressourcen nicht berücksichtigt wurden.

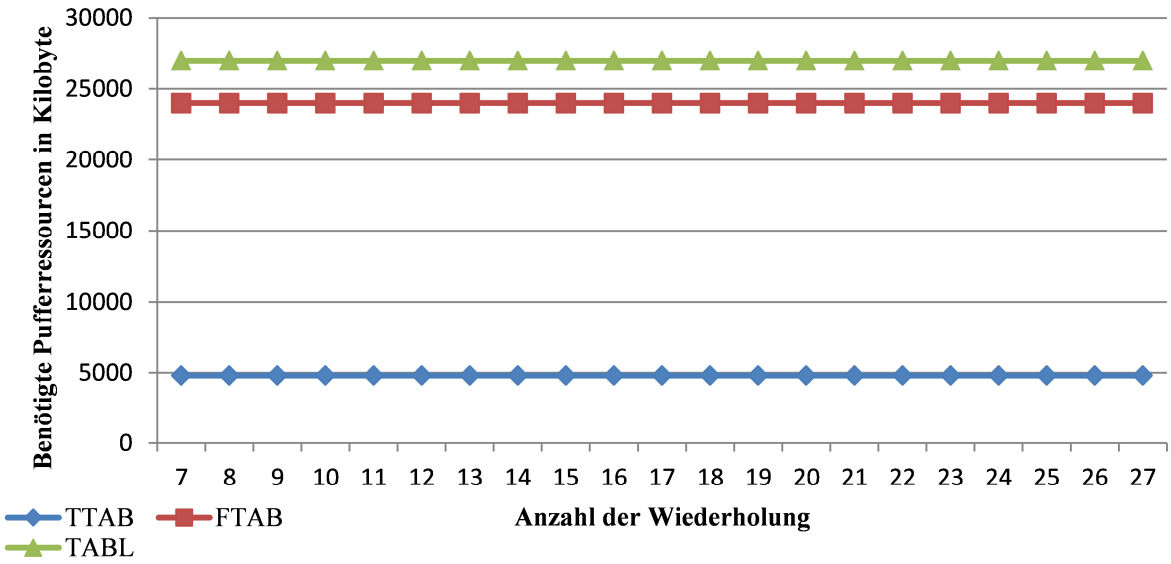


Abbildung 4-12: Vom Geschäftsprozess benötigte Ressourcen in den Puffern TTAB, FTAB und TABL
Quelle: (Gradl et al. 2011a)

Durch die Verwendung der vorgestellten Methode zeigte sich für den gesamten Geschäftsprozess ein konstanter Bedarf an benötigten Pufferressourcen, ohne Abweichung vom arithmetischen Mittel. Durch die Verwendung des errechneten und in Abbildung 4-11 dargestellten Bedarfs an Pufferressourcen durch das Messinstrument selbst, konnte der Ressourcenbedarf des Geschäftsprozesses, wie in Abbildung 4-12 gezeigt, ermittelt werden. Diese Abbildung zeigt, dass durch die vorgestellte Methode signifikante Ergebnisse erzielt werden können.

4.2.3. Analyse der Tabellenpufferinhalte

Die bereits in Kapitel 3.5.2 vorgestellten Funktionalitäten der Transaktion „ST02“ bzgl. der Analyse von Pufferinhalten sind für die Analyse der Performance eines SAP-ERP-Systems, basierend auf dem SAP-WebAS-ABAP ebenfalls von zentraler Bedeutung.

4.2.3.1. Der Baustein „SAPWL_TABSTAT_SINCE_STARTUP“

Um diese Informationen für die Auswertung einer großen Anzahl von Testläufen automatisiert bereit zu stellen, wurde analog zu den STAD Performance Records ein remotefähiger Funktionsbaustein identifiziert und mittels der WebService-Schnittstelle in den Prototypen der vorliegen Arbeit integriert. Der Baustein „SAPWL_TABSTAT_SINCE_STARTUP“ benötigt für die Ausführung keine Parameterübergabe, da er die zum Zeitpunkt der Ausführung aktuellen Daten der Puffer ausliest, und mittels Tabellen der folgenden SAP-Strukturen zurückgibt:

Tabellenname	Zugrundliegende Struktur
TABLE_CALLS	DBSTATDEC
TABLE_CALLS_64	DBSTATDEC65
GENERIC_REGIONS	RSTBXOBUF

Tabelle 4-6: Rückgabeparameter des Funktionsbausteins "SAPWL_TABSTAT_SINCE_STARTUP"
Quelle: Analyse Funktionsbaustein Transaktion SE37 (SAP 2008c)

Die Strukturen „DBSTATDEC“ und „DBSTATDEC 64“ unterscheiden sich lediglich in der Kodierung der enthaltenden Integer-Werte. Beide Tabellen enthalten Informationen analog zu dem bereits beschriebenen Bildschirm der Transaktion „ST02“ aus Abbildung 3-16 und geben Auskunft über die Größe aller generischen Bereiche dieser Tabelle im Puffer sowie die Anzahl der darin enthaltenen Datensätze.

Detaillierte Informationen über die generischen Bereiche enthält die Tabelle „GENERIC_REGIONS“. Diese enthält u.a. die Anzahl der Datensätze pro generischem Bereich und deren Größe.

4.2.3.2. Verwendungsmöglichkeiten

Mit diesen Informationen ist es möglich, nach Durchführung eines Workload-Schrittes zu bestimmen, ob und wie (Anzahl der Einträge und Größe) sich der Inhalt der Tabellenpuffer verändert hat, indem die Ausgabe dieses Bausteins nach dem Workload-Schritt mit der Ausgabe vor dem Workload-Schritt verglichen wird. So kann festgestellt werden, welche generischen Bereiche sich vergrößern, und da sich die Datensätze pro generischem Bereich hinsichtlich der benötigten Speichermenge sehr stark ähneln, um wie viel sich der Speicherplatzbedarf im Puffer pro Workload-Schritt erhöht.

4.2.3.3. Einfluss des Messinstruments auf die Performance des Testobjekts

Der Funktionsbaustein „SAPWL_TABSTAT_SINCE_STARTUP“ liefert die aktuelle Zugriffsstatistik auf die Tabellenpuffer des SAP-WebAS-ABAP. Analog zu den STAD-Performance-Records werden diese Statistiken permanent durch den SAP-Kernel mitgeschrieben und sind daher ebenso als nicht intrusiv (Jain 1991) zu charakterisieren. Im

Gegensatz dazu ist die Last, welche durch den Aufruf des Funktionsbausteins erzeugt wird, als intrusiv einzustufen, sodass der Baustein nicht während eines Testlaufs einsetzt werden darf, da er sonst das Messergebnis verfälschen würde.

4.2.4. SAP-Performance-Trace

Der SAP-WebAS-ABAP bietet für die detaillierte Analyse des Ablaufs von ABAP Programmen die Möglichkeit die einzelnen Aktionen in verschiedenen Bereichen mittels des sogenannten „Performance Trace“ aufzuzeichnen. Damit ist es möglich, für die Bereiche Puffer, Enqueue, SQL und RFC die Vorgänge zu loggen und zu analysieren sowie einzelne SQL-Anweisungen im Detail zu untersuchen.

Den zentralen Einstiegspunkt bietet die Transaktion „ST05“. Abbildung 4-13 zeigt auf der linken Seite die zur Verfügung stehenden Traces. Auf der rechten Seite finden sich die Bedienelemente zum Starten und Stoppen eines Traces. Es besteht auch die Möglichkeit, den Trace beim Start bzgl. eines Benutzers, einer Transaktion, eines Programms, zu verwendender bzw. auszuschließender Tabellen zu konfigurieren, um die Menge der aufzuzeichnenden Daten sowie die durch den Trace erzeugte Systemlast zu reduzieren.

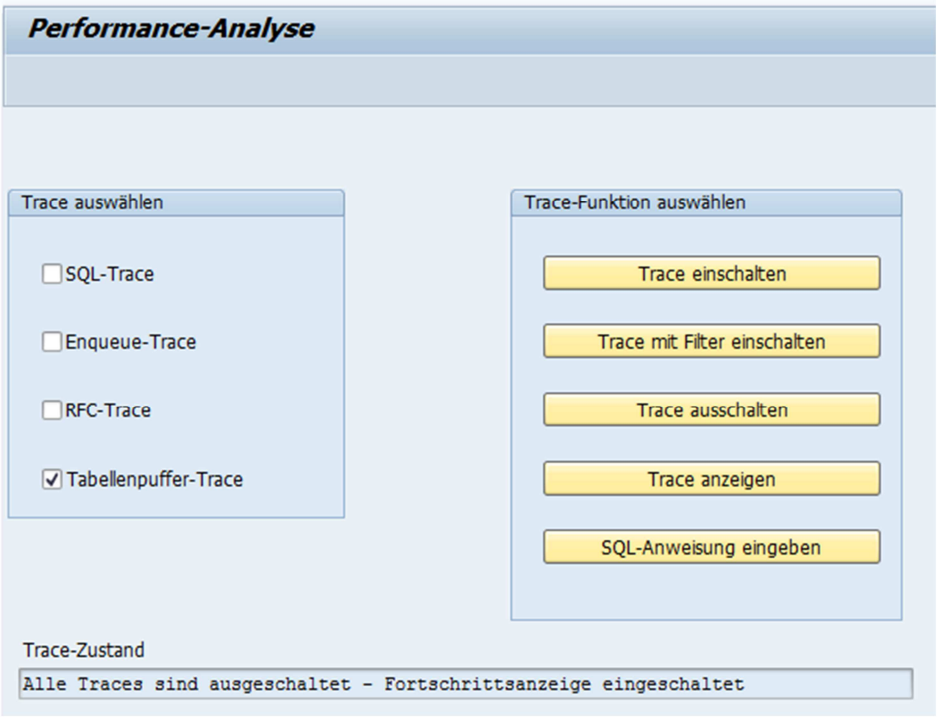


Abbildung 4-13: Administrationsoberfläche der Performance-Traces in Transaktion "ST05"
Quelle: Screenshot „ST05“ (SAP 2008a)

4.2.4.1. Aufzeichnung starten, stoppen und speichern

Um ein ABAP-Programm mittels des Performance Trace zu analysieren, muss dieser vor Beginn gestartet werden. Aufgrund der hohen Systemlast, welche der Trace erzeugt, ist es ratsam, die Aufzeichnung auf den/die gewünschten Benutzer einzuschränken. Nach der Ausführung kann der Trace geschlossen werden und ist damit in einer Datei auf Betriebssystemebene verfügbar. Soll der Trace längere Zeit zur Verfügung stehen, so kann dieser in eine separate Datei auf Betriebssystemebene gespeichert werden.

4.2.4.2. Analyse Funktionalitäten

Wurde ein Trace aufgezeichnet, kann dieser mit dem Bedienelement „Trace anzeigen“ dargestellt werden. Für die graphische Aufbereitung ist es ebenfalls möglich, die Menge an Datensätzen nach Kriterien (SQL-Trace, Enqueue-Trace, RFC-Trace, Tabellenpuffer-Trace, Starttag, Startzeit, Endetag, Endezeit, Benutzer, Objekte, Dauer, Operationen, DB-Verbindung, TransGUID, Anzahl Trace-Sätze) zu filtern. Damit ist es möglich, die zu analysierenden Vorgänge aus der großen Menge an Einträgen im Performance Trace zu identifizieren. Darüber hinaus kann die Menge an Detailinformationen zu den einzelnen Schritten über die Wahl von „Trace-Liste“ bzw. „Ausführliche Trace-Liste“ gesteuert werden.

Für die externe Analyse können die aufgezeichneten Daten in die folgenden Formate exportiert werden:

- Unkonvertiert (Textbasierte Ausgabe)
- Tabellenkalkulation (Excel-Format)
- Rich Text Format
- HTML Format

Die verschiedenen Performance-Trace (Puffer, Enqueue, SQL und RFC) können in dieser Transaktion auch gleichzeitig angezeigt werden, sodass damit auch die Möglichkeit von Wechselwirkungen, wie z.B. zwischen Pufferzugriffen und SQL-Abfragen, analysiert werden kann. Um diese Einträge den verschiedenen Performance-Trace zuordnen zu können, sind diese, wie in Abbildung 4-14 exemplarisch dargestellt, farblich hinterlegt.

Transaktion SE80	Workprozess-Nr. 0	Prozess-Typ DIA	Mandant 000	Benutzer MASTER-ADM	TransGUID 4F094C3A548B0680E1008000839F09B6	Datum 09.01.2012

hh:mm:ss.ms	Dauer	Programm	Objektname	Oper	Curs	Array	Sätze	RC	Con	Anweisung
17:45:21.331	12	CL_WB_S…	D345T	RETRIEV			0	0		C 82 MENUSYST D 33.766
17:45:21.332	2	SAPLWB_…	D342L	RETRIEV			0	0		C 120 MENUSYST MEN
17:45:21.332	9	SAPLWB_…	D345T	RETRIEV			0	0		C 82 MENUSYST D 33.766
17:45:21.332	5	CL_WB_S…	AAB_ID_ACT	OPEN				0		R 6 000
17:45:21.332	1	CL_WB_S…	AAB_ID_ACT	FETCH			0	64		
17:45:21.332	1	CL_WB_S…	AAB_ID_ACT	CLOSE			0	0		
17:45:21.333	682	SAPLSEU1	TRDIR	OPEN	98			0	R/3	SELECT WHERE "NAME" = 'SAPLWB_INITIAL_TOOL' FETCH FIRST 1 ROWS
17:45:21.333	32	SAPLSEU1	TRDIR	FETCH	98	1	1	0	R/3	
17:45:21.333	11	SAPLSEU1	TRDIR	CLOSE	98		0	0	R/3	
17:45:21.333	446	SAPLSMPI	EUDB	OPEN	76			0	R/3	SELECT WHERE "RELID" = 'CU' AND "NAME" = 'SAPLWB_INITIAL_TOOL'
17:45:21.334	15	SAPLSMPI	EUDB	FETCH	76	1	1	0	R/3	
17:45:21.334	5	SAPLSMPI	EUDB	CLOSE	76		0	0	R/3	
17:45:21.334	363	SAPLSMPI	EUDB	OPEN	110			0	R/3	SELECT WHERE "RELID" = 'CU' AND "NAME" = 'SAPLWB_INITIAL_TOOL'
17:45:21.334	13	SAPLSMPI	EUDB	FETCH	110	1	1	0	R/3	
17:45:21.335	7	SAPLSMPI	EUDB	CLOSE	110		0	0	R/3	
17:45:21.335	601	SAPLSMPI	RSMPTEXTS	OPEN	45			0	R/3	SELECT WHERE "PROGNAME" = 'SAPLWB_INITIAL_TOOL' AND "SPRSL" = '
17:45:21.336	462	SAPLSMPI	RSMPTEXTS	FETCH	45		86	0	R/3	
17:45:21.336	2	SAPLSMPI	RSMPTEXTS	FETCH	45	1	0	100	R/3	
17:45:21.336	6	SAPLSMPI	RSMPTEXTS	CLOSE	45		0	0	R/3	
17:45:21.337	11	CL_WB_R…	WBREGISTRY	READ SI			0	64		R 52 REPINITIAL MASTER-ADM

Abbildung 4-14: Farbliche Hervorhebungen im Performance Trace
Quelle: Screenshot „ST05“ (SAP 2008a)

Die Anzahl der Spalten sowie deren Benennung ist bei allen Arten des Performance-Trace identisch:

- hh:mm:ss.ms: Startzeitpunkt der Aktion
- Dauer: Ausführungszeit der Aktion, gemessen in Mikrosekunden

- Programm: Name des Programms, von welchem die Aktion angestoßen wurde
- Objektname: je nach Performance Trace, Datenbankobjekt, RFC Verbindung oder Enqueue-Objekt
- Oper: Name der Operation
- Curs: Identifikationsnummer des Zeigers auf eine Ergebnismenge
- Array: Anzahl der durch die Anweisung geholten Arrays
- Sätze: Anzahl der durch die Anweisung geholten Tabellenzeilen
- RC: Rückgabewert der Anweisung
- Con: Bezeichnung der Verbindung
- Anweisung: vollständige Darstellung der Anweisung. Unterscheidet sich stark nach Art des Performance-Trace.

Auf die unterschiedliche Interpretation der Spalten „Operation“ und „Anweisung“ sowie deren Analyse wird in den Abschnitten zu den verschiedenen Performance-Trace (4.2.5 bis 4.2.7) eingegangen.

4.2.4.3. Einfluss des Performance-Trace auf die Messergebnisse

Wie bereits im vorhergehenden Absatz erwähnt, erzeugt das Erstellen von Performance-Trace eine hohe Last auf das System. Aus diesem Grund muss nicht nur der Einsatz in produktiven SAP-Systemen genau abgewogen werden. Auch für die Aufzeichnung von Testläufen ist der Performance Trace nicht geeignet, da er die Messergebnisse verfälschen würde. Er eignet sich jedoch sehr gut, um einzelne logische Arbeitsschritte (LUW) hinsichtlich deren Ressourcennutzung eingehend zu analysieren.

4.2.5. Analyse von ABAP-Programmen hinsichtlich der Pufferzugriffe

Der ABAP-Puffer-Trace zeichnet alle Operationen auf den SAP-Puffer auf. In den folgenden Abschnitten, werden die Informationen des Traces dargestellt und deren logische Gruppierung zu Sequenzen eingeführt.

4.2.5.1. Operationen für Pufferzugriffe

Abbildung 4-15 zeigt exemplarisch Pufferzugriffe auf 5 verschiedenen Datenbanktabellen. Die Tabellennamen sind in Spalte „Objektname“ verzeichnet.

hh:mm:ss.ms	Dauer	Programm	Objektname	Oper	Curs	Array	Sätze	RC	Con	Anweisung
17:45:21.316	16	SAPLSMT…	TSTCT	READ SI			1	0		P 42 DSE80
17:45:21.317	5	SAPLSUSE	TSTC	READ SI			1	0		P 40 SE80
17:45:21.317	1	SAPLSUSE	TSTCP	READ SI			1	0		P 40 SE80
17:45:21.317	1	SAPLSUSE	TSTC	READ SI			1	0		P 40 SEU_INT
17:45:21.317	8	SAPLSUSE	USRBF2	OPEN				0		R 50 000MASTER-ADM S_TCODE
17:45:21.317	3	SAPLSUSE	USRBF2	FETCH			1	0		
17:45:21.317	1	SAPLSUSE	USRBF2	FETCH			1	0		
17:45:21.317	1	SAPLSUSE	USRBF2	FETCH			1	0		
17:45:21.317	1	SAPLSUSE	USRBF2	FETCH			0	64		
17:45:21.317	1	SAPLSUSE	USRBF2	CLOSE			0	0		
17:45:21.317	6	SAPLSUSE	UST12	OPEN				0		R 50 000S_TCODE &_SAP_ALL
17:45:21.317	0	SAPLSUSE	UST12	FETCH			0	64		
17:45:21.317	0	SAPLSUSE	UST12	CLOSE			0	0		
17:45:21.317	1	SAPLSUSE	UST12	OPEN				0		R 50 000S_TCODE S_TCD_ALL
17:45:21.317	1	SAPLSUSE	UST12	FETCH			0	64		
17:45:21.317	1	SAPLSUSE	UST12	CLOSE			0	0		

Abbildung 4-15: Puffer-Trace - Anzeige "ST05"
Quelle: Screenshot ST05 (SAP 2008a)

Abhängig von dem Puffer, auf welchen zugegriffen wird, sind die Operationen in der Spalte „Oper“ unterschiedlich ausgeprägt. Für die Zugriffe auf Puffer werden die folgenden Anweisungen verwendet:

- READ SI
- OPEN
- LD OPEN
- LD END
- FETCH
- RETRIEVE
- LOAD
- CLOSE

Die Operationen auf die Tabellenpuffer treten in Sequenzen unterschiedlicher Länge auf. Die Operation „READ SI“ bildet dabei eine Sequenz mit Länge 1. Dies bedeutet, dass genau ein Datensatz aus dem Puffer gelesen wird. Im Gegensatz zu der Operation „FETCH“ sind keine vorbereitenden Operationen nötig. Bevor ein Datensatz mittels der Operation „FETCH“ aus dem Puffer gelesen werden kann, muss die Anfrage an den Puffer gestellt und die Ergebnismenge mittels „OPEN“ geöffnet werden. Nun kann eine Folge von „FETCH“-Operationen Daten aus dem Puffer lesen. Ist dies erfolgreich, so wird es durch den Eintrag „0“ in der Spalte „RC“ vermerkt. Tritt ein Fehler auf oder sind keine weiteren Datensätze in der Ergebnismenge vorhanden, so wird dies mit einem Returncode > 0 verzeichnet. Sind die Lesevorgänge an den Puffer abgeschlossen, so wird die Ergebnismenge mit „CLOSE“ geschlossen. Sind jedoch die angefragten Datensätze noch nicht im Tabellenpuffer vorhanden, so wird dieser aus der Datenbank gefüllt. Dies wird mit der Operation „LD OPEN“ durchgeführt und mit „LD END“ beendet. Im Gegensatz zu den Tabellenpuffern treten die Operationen auf die CUA- oder Export-/Import-Puffer nicht in Sequenzen, sondern einzeln,

ohne gegenseitige Abhängigkeiten auf. Um einen Datensatz aus dem Puffer zu lesen, wird die Operation „RETRIEVE“ verwendet. Mit „LOAD“ werden Daten in die Puffer geladen.

4.2.5.2. Analyse der Anweisungen für Pufferzugriffe

Grundsätzlich besteht der Eintrag in der Spalte „Anweisung“ für den Puffer-Trace aus 4 Komponenten, welche durch Leerzeichen getrennt sind. Die Identifikation dieser 4 Bereiche wird jedoch durch die Tatsache erschwert, dass die Inhalte der einzelnen Felder ebenfalls Leerzeichen enthalten dürfen.

Im ersten Bereich des Anweisungsfeldes ist die Art des Puffers codiert, auf welchen zugegriffen wird. Dabei stehen die folgenden Abkürzungen für die jeweiligen Puffer:

- P: Einzelsatzpuffer
- R: Vollständiger oder generischer Puffer
- C: CUA-Puffer
- S: Export-/Import Puffer

Die Länge des Primärschlüssels der zugehörigen Tabelle, wird im zweiten Bereich der Anweisung hinterlegt. Dieser ist nicht zwangsweise identisch mit der Länge des generischen Schlüssels, welcher erst über eine Detailansicht (Abbildung 4-17) der Anweisung dargestellt wird.

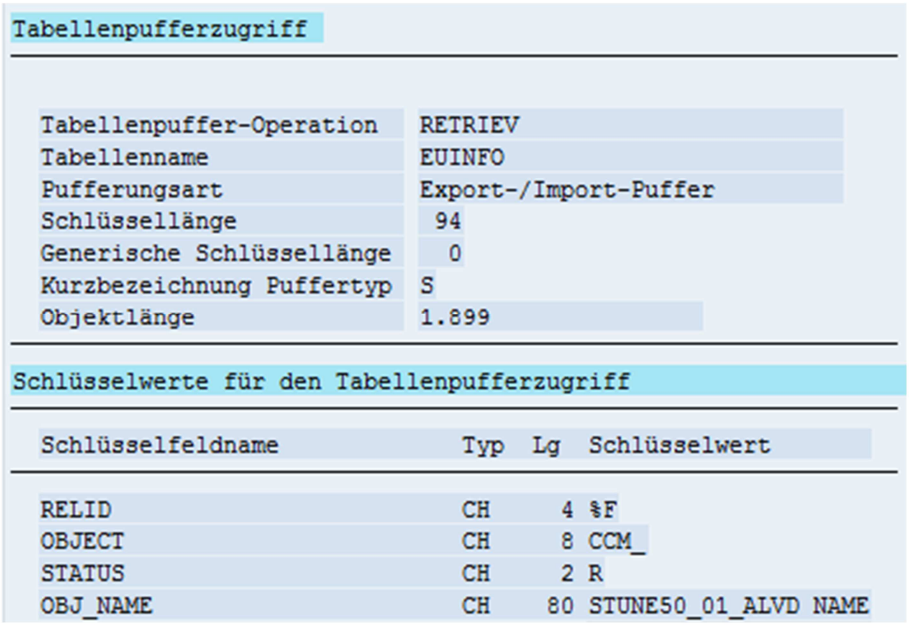
Im dritten Bereich werden die für den Pufferzugriff verwendeten Werte der Felder des Primärschlüssels angezeigt. Die Länge der gesamten Anweisung auf den Puffer wird im Bereich 4 angezeigt.

Um nun die durch die Anweisung betroffenen Datensätze zu identifizieren muss der String aus Bereich 3 analysiert werden.

```
S 94 %FCCM_RSTUNE50_01_ALVD NAME
```

Abbildung 4-16: Beispiel für eine Anweisung aus dem SAP-Puffer-Trace
Quelle: Screenshot „ST05“ (SAP 2008a)

Die in Abbildung 4-16 beispielhaft dargestellte Spezifikation eines Objekts, welches aus einem der Puffer gelesen werden soll, besteht aus einer Aneinanderreihung von Werten für die ausgewählten Felder des Primärschlüssels. Eine Möglichkeit zur Analyse dieser Spezifikation bietet die Transaktion „ST05“ mittels einer Detailansicht, welche in Abbildung 4-17 für die Spezifikation aus Abbildung 4-16 dargestellt ist. Diese zeigt, neben den Informationen über die Operation, die Tabelle und deren Art der Pufferung auch die Spezifikation des Datensatzes, aufgeteilt auf die verwendeten Felder des Primärschlüssels.



Tabellenpufferzugriff

Tabellenpuffer-Operation	RETRIEV
Tabellenname	EUINFO
Pufferungsart	Export-/Import-Puffer
Schlüssellänge	94
Generische Schlüssellänge	0
Kurzbezeichnung Puffertyp	S
Objektlänge	1.899

Schlüsselwerte für den Tabellenpufferzugriff

Schlüsselfeldname	Typ	Lg	Schlüsselwert
RELID	CH	4	%F
OBJECT	CH	8	CCM_
STATUS	CH	2	R
OBJ_NAME	CH	80	STUNE50_01_ALVD NAME

Abbildung 4-17: Detailanalyse zur Anweisung einer Pufferaktion
Quelle: Screenshot „ST05“ (SAP 2008a)

Diese Detailansicht zeigt, dass die Werte, entsprechend der Reihenfolge ihrer Spalten im Primärschlüssel, einfach aneinandergereiht werden. In Abbildung 4-17 werden also alle Datensätze im Zielpuffer angesprochen, welche zur Tabelle in der Spalte „Objektname“ gehören und die unter „Schlüsselwerte für Tabellenpufferzugriff“ verzeichneten Kriterien erfüllen.

Wie bereits erwähnt, kann dies auch zu Leerzeichen in dieser Spezifikation führen, da die Werte, falls sie kürzer als die definierte Länge des Feldes sind, mit Leerzeichen aufgefüllt werden, bis die Länge des Wertes und die des Feldes gleich sind. Für die Identifikation der Zugriffskriterien sind also folgende Informationen, welche im SAP-DDIC abgelegt sind, nötig:

- Anzahl Primärschlüsselfelder
- Bezeichnung Primärschlüsselfelder
- Länge jedes einzelnen Primärschlüsselfeldes
- Reihenfolge der Primärschlüsselfelder

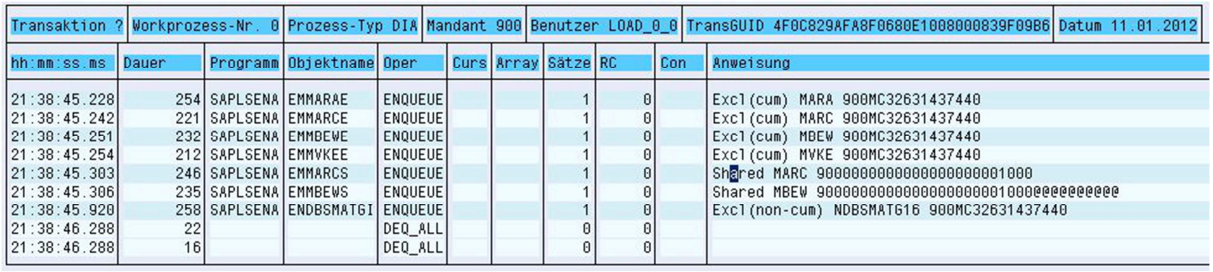
Mit den Informationen aus dem Puffer-Trace und den Objektbeschreibungen aus dem SAP-DDIC ist es möglich, für jedes ABAP-Programm festzustellen, welche Datensätze es aus welchem Puffer benötigt.

4.2.6. Analyse von ABAP-Programmen hinsichtlich der Enqueue-Vorgänge

Wie bereits besprochen, wird der wechselseitige Zugriff auf Objekte über Enqueue genannte Sperren implementiert. Für die Analyse dieser Sperrvorgänge bietet der Performance Trace die Möglichkeit diese mitzuloggen.

4.2.6.1. Beschreibung eines Sperrvorgangs

Die Beschreibung der Enqueue-Vorgänge, siehe Abbildung 4-18, unterscheidet sich in einigen Spalten von den bisher eingeführten Traces. Die Zeit, welche gewartet werden muss, wenn ein Enqueue-Vorgang nicht sofort möglich ist, da das zu sperrende Objekt bereits durch ein anderes Programm gesperrt war, wird durch die Performance-Traces nicht direkt erfasst. Die Spalte „Dauer“ zeigt ebenfalls nicht, wie lange das Objekt gesperrt wurde, sondern die Dauer für das Setzen der Sperre. In der Spalte Objektname ist der Name des Sperrobjekts verzeichnet, welches vom ABAP-Programm aufgerufen wurde, um die Sperre zu setzen.



Transaktion ? | Workprozess-Nr. 0 | Prozess-Typ DIA | Mandant 900 | Benutzer LOAD_0_0 | TransGUID 4F0C829AFA8F0680E1008000839F09B6 | Datum 11.01.2012

hh:mm:ss.ms	Dauer	Programm	Objektname	Oper	Curs	Array	Sätze	RC	Con	Anweisung
21:38:45.228	254	SAPLSENA	EMMARAE	ENQUEUE			1	0		Excl(cum) MARA 900MC32631437440
21:38:45.242	221	SAPLSENA	EMMARCE	ENQUEUE			1	0		Excl(cum) MARC 900MC32631437440
21:38:45.251	232	SAPLSENA	EMMBEWE	ENQUEUE			1	0		Excl(cum) MBEW 900MC32631437440
21:38:45.254	212	SAPLSENA	EMMVKEE	ENQUEUE			1	0		Excl(cum) MVKE 900MC32631437440
21:38:45.303	246	SAPLSENA	EMMARCS	ENQUEUE			1	0		Shared MARC 900000000000000000001000
21:38:45.306	235	SAPLSENA	EMMBEWS	ENQUEUE			1	0		Shared MBEW 900000000000000000001000@@@@@@@@@@
21:38:45.920	258	SAPLSENA	ENDBSMATGI	ENQUEUE			1	0		Excl(non-cum) NDBSMATGI6 900MC32631437440
21:38:46.288	22			DEQ_ALL			0	0		
21:38:46.288	16			DEQ_ALL			0	0		

Abbildung 4-18: Beispielausgaben für den Enqueue Trace
Quelle: Screenshot „ST05“ (SAP 2008a)

Abbildung 4-18 zeigt die wichtigsten und am häufigsten vorkommenden Operationen im Enqueue-Trace. Eine vollständige Liste von Operationen, welche durch ein ABAP-Programm angefordert werden, mit einer jeweils kurzen Beschreibung, zeigt Tabelle 4-7.

Operation	Beschreibung
ENQUEUE	Sperre setzen, oder Kumulationszähler erhöhen
DEQUEUE	Sperre lösen, oder Kumulationszähler reduzieren
DEQUEUE_ALL	Alle Sperren eines Owner aufheben
REMOVE	Löschen beliebiger Sperren (durch Administrator Transaktion SM12)
GETUSER	Ermitteln der eigenen Owner-ID
ENQUEUE_PERM	Sichern der an einen Verbuchungsauftrag vererbten Sperren
QUERY	Lesen von Sperreinträgen aus der Sperrtabelle
GET_STAMP	Eindeutige, aufsteigende Nummer anfordern
STATISTCS	Enqueue-Server-Statistik lesen

Tabelle 4-7: Sperroperationen innerhalb von ABAP-Programmen

4.2.6.2. Analyse des gesperrten Objekts

Wie bereits bei der Analyse von gepufferten Objekten ist die Spezifikation des gesperrten Objekts im Feld der Spalte „Anweisung“ abgelegt. Der Inhalt dieses Feldes besteht aus Bereichen, welche wieder durch Leerzeichen voneinander getrennt. Der erste Bereich beinhaltet den Sperrmodus. Im Gegensatz zur Dokumentation der Sperrverwaltung (SAP 2011g), welche vier Sperrmodi erklärt und die in 3.4.2.2 beschrieben sind, listet die technische Hilfe in der Transaktion „ST05“ nur drei Sperrmodi auf. In den Analysen verschiedener Workload-Schritte in dieser Arbeit wurden analog zur technischen Dokumentation nur die 3 folgenden Arten von Sperren identifiziert und analysiert:

- Shared: Lesesperre (S)

- Excl(cum): Schreibsperre (E)
- Excl(non-cum): Erweiterte Schreibsperre (X)

Diese Auflistung der Sperrarten bringt die Bezeichnung aus den Sätzen des Performance-Traces mit den Bezeichnungen in Verbindung, welche in der Dokumentation und in 3.4.2.2 verwendet wurden.

Der zweite Bereich im Anweisungsfeld zeigt die Tabelle, welche durch den Enqueue-Vorgang ganz oder teilweise gesperrt wird. Analog zu der genauen Spezifikation der Datensätze, welche aus dem Puffer gelesen werden sollen (4.2.5.2), sind die zu sperrenden Reihen im Enqueue Trace im dritten Bereich des Anweisungsfeldes beschrieben. Bei der Detailanalyse der Sperranweisung wird jedoch die Spezifikation der zu sperrenden Tabellenbereiche nicht nach den Feldern des Primary Key aufgespalten, sondern eine Liste angezeigt, welche alle in dem jeweiligen Sperrbaustein enthaltenen Sperranfragen darstellt.

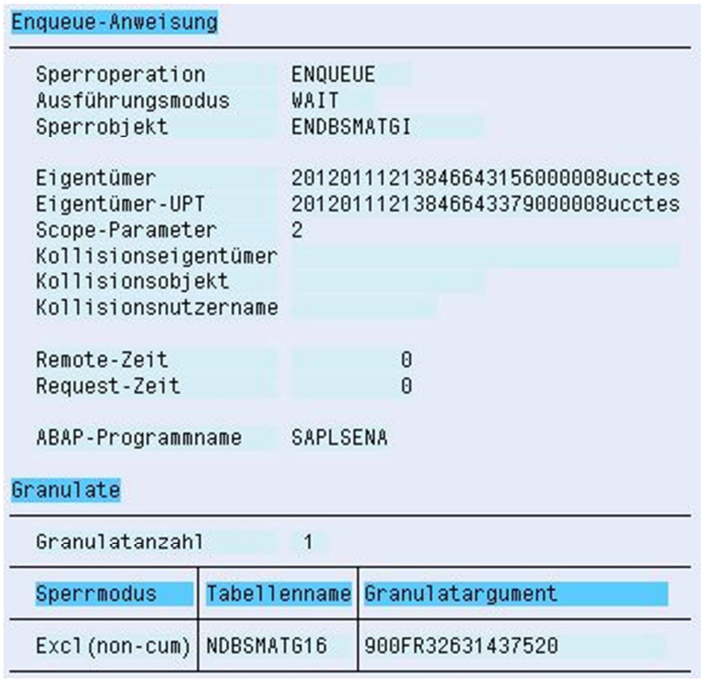


Abbildung 4-19: Enqueue-Anweisung – Detailansicht
Quelle: Screenshot „ST05“ (SAP 2008a)

Die Detailansicht aus Abbildung 4-19 zeigt neben der bereits bekannten Informationen über Sperroperation, Namen des Sperrobjekt, ID des Eigentümers, Name des zugehörigen Programms und dem „_SCOPE“-Parameter im Bereich „Granulate“ diese bereits angesprochene Liste mit allen Sperranfragen eines Enqueue-Bausteins. Damit können für jeden Enqueue-Baustein folgende Informationen einer jeden beinhalteten Sperre ermittelt werden:

- Sperrmodus
- Tabellenname
- Granularitätsargument

Das Granularitätsargument ist analog zur Spezifikation aus 4.2.5.2 aufgebaut und dementsprechend auszuwerten.

4.2.7. Analyse von ABAP-Programmen hinsichtlich der SQL-Zugriffe

Da ein Großteil der Last auf einem SAP-System durch das unterliegende Datenbank-Management-System verarbeitet wird, bietet der Performance-Trace eine Möglichkeit, sämtliche SQL-Zugriffe auf die Datenbank zu protokolieren und zu analysieren.

4.2.7.1. Beschreibung von SQL Zugriffen

Der SQL-Trace bietet zunächst eine Übersicht über alle Operationen, welche durch die Programme eines Benutzers im ausgewählten Zeitraum ausgeführt werden. Beispielhaft wird dies durch Abbildung 4-20 und Abbildung 4-21 veranschaulicht.

hh:mm:ss.ms	Dauer	Programm	Objektname	Oper	Curs	Array	Sätze	RC	Con
21:38:45.215	2.070	SAPLHTT...	USR02	OPEN	43			0	R/3
21:38:45.217	45	SAPLHTT...	USR02	FETCH	43	1	1	0	R/3
21:38:45.217	14	SAPLHTT...	USR02	CLOSE	43		0	0	R/3
21:38:45.218	644	SAPLHTT...	USR02	EXECSTM	20	1	1	0	R/3
21:38:45.219	475	SAPLHTT...	USR02	EXECSTM	20	1	1	0	R/3
21:38:45.219	1.660	SAPLHTT...		EXECSTA			0	0	R/3

Abbildung 4-20: Beispielhafter Auszug aus SQL Trace - linke Seite
Quelle: Screenshot „ST05“ (SAP 2008a)

Nach der millisekundengenauen Angabe des Startzeitpunkts der SQL-Operation in Spalte 1, zeigt Spalte 2 die Dauer der betreffenden SQL-Operation in Mikrosekunden sowie das aufrufende Programm in Spalte 3. Die Zieltabelle, auf welche die Operation, aufgelistet in Spalte 5, abgesetzt wird, ist in Spalte 4 verzeichnet. Die Identifikationsnummer des Zeigers auf die Ergebnismenge sowie die Anzahl der beinhalteten Arrays, die Anzahl der bearbeiteten Datenbankzeilen, den Rückgabewert als auch die Bezeichnung der Verbindung zeigen die Spalten 6 bis 10.

```
Anweisung
SELECT WHERE "MANDT" = '900' AND "BNAME" = 'LOAD_0_0'  WITH UR ── OPTLEVEL( 5 ) ── QUERY_DEGREE( 1 ) ── LOCATION( SAPMSYST , 2448 ) ── SYSTEM(

UPDATE SET "BCODE" = 0xC37A3E989D9BB4D8 , "GLTGV" = 00000000 , "GLTGB" = 00000000 , "USTYP" = 'B' , "CLASS" = ' ' , "LOCNT" = 0 , "UFLAG" = 0 ,
UPDATE SET "BCODE" = 0xC37A3E989D9BB4D8 , "GLTGV" = 00000000 , "GLTGB" = 00000000 , "USTYP" = 'B' , "CLASS" = ' ' , "LOCNT" = 0 , "UFLAG" = 0 ,
COMMIT WORK
```

Abbildung 4-21: Beispielhafter Auszug aus SQL Trace - rechte Seite
Quelle: Screenshot „ST05“ (SAP 2008a)

Das Anweisungsfeld ist aufgrund der Breite separat in Abbildung 4-21 dargestellt. Dieses Feld beinhaltet die Anweisung in SQL-Syntax. Die Darstellung der Anweisung ist aufgrund der beschränkten Anzahl von verfügbaren Zeichen häufig nicht vollständig.

4.2.7.2. Detailanzeige einer SQL-Operation

Die Informationen für diese vollständige Darstellung werden bei der Erstellung des Performance-Trace bereits mit aufgezeichnet, sodass sie für die Analyse zur Verfügung stehen.

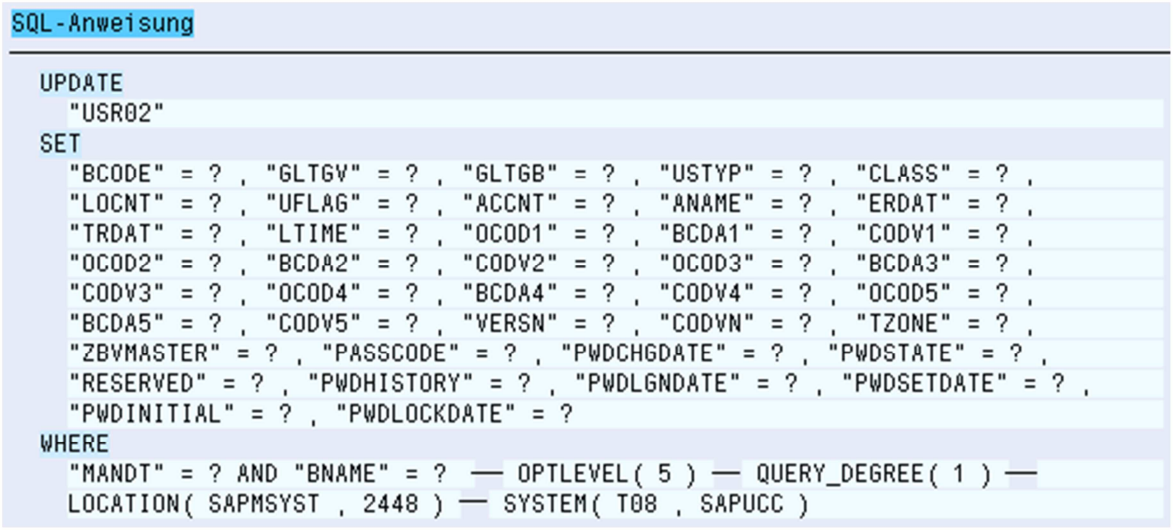


Abbildung 4-22: Detailansicht SQL-Trace am Beispiel einer Update-Anweisung
Quelle: Screenshot „ST05“ (SAP 2008a)

Abbildung 4-22 und Abbildung 4-23 zeigen dies exemplarisch für einen Update-Vorgang auf die Tabelle „USR02“. Die Darstellung von SQL-Statements enthält dabei keine Belegung der Felder, sondern zeigt nur die Namen der Tabellenfelder sowie deren Verwendung im Statement. Die zugehörigen Werte werden in einem eigenen Bereich unterhalb des Statements aufgelistet. Wie Abbildung 4-23 zeigt, werden die Werte über Variablennamen (A0 bis A<n>) den Feldern aus Abbildung 4-22 zugeordnet. Die Nummer im Variablennamen ergibt sich dabei aus der Position im Statement.

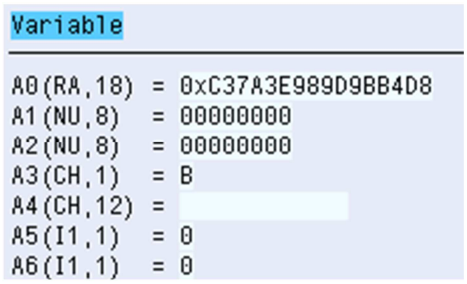


Abbildung 4-23: Variablenbelegung in der Detailansicht des SQL-Trace
Quelle: Screenshot „ST05“ (SAP 2008a)

Zusätzlich zum Wert sind in Klammern auch noch der Typ und die Länge des Tabellenfeldes angegeben, siehe Tabelle 4-8.

Abkürzung	Typ
RA	Raw-Data
NU	Number
I<Länge>	Integer der Länge <Länge>
CH	Character

Tabelle 4-8: Datentypen im SQL-Trace
Quelle: Eigene Darstellung

Daraus ist ersichtlich, welche Felder der Tabelle mit neuen Werten belegt werden. Damit kann bereits in der Analysephase festgestellt werden, ob sich eine Operation auf

Tabelleninhalte bezieht, welche in den Puffern enthalten sind, indem die Tabelle und deren Felder mit den in den Tabellenpuffern vorhandenen Einträgen verglichen werden.

4.2.7.3. SQL Operationen

Um nun die Auswirkungen der einzelnen im SQL-Trace verzeichneten Einträge analysieren zu können, ist es nötig, die vom SAP-System verwendeten Operationen zu identifizieren. Da in der Dokumentation keine vollständige Liste aller SQL-Operationen gefunden werden konnte, zeigt die folgende Übersicht die SQL-Operationen, welche innerhalb des in 6.1 beschriebenen Geschäftsprozesses verwendet werden:

- PREPARE
- OPEN
- FETCH
- CLOSE
- EXECSTM
- EXECSTA

Mittels „PREPARE“ wird eine Anweisung für die häufige Verwendung ohne Parametrisierung an die Datenbank übergeben, um dort vorübersetzt und vorgehalten zu werden, sodass bei späteren Verwendungen die Werte direkt übergeben werden können. Mittels der Operation „OPEN“ wird eine Anfrage an die Datenbank geschickt und der Zeiger auf die zugehörige Ergebnismenge bereitgestellt. Mittels dieses Zeigers und der „FETCH“-Operation können nun die Inhalte der Ergebnismenge von der Datenbank gelesen werden. Die Operation „CLOSE“ beendet Lesevorgänge und schließt den Zeiger auf die Ergebnismenge. Um Werte in der Datenbank zu modifizieren, wird die Operation „EXECSTM“ verwendet. Um Daten in der Datenbank persistent zu speichern, müssen Datenbanktransaktionen mittels einer „commit“-Anweisung abgeschlossen werden. Dies wird durch die Operation „EXECSTA“ durchgeführt. Mittels dieser Operationen werden die 4 Gruppen von SQL-Anfragen „Select“, „Insert“, „Update“ und „Delete“ an die Datenbank übergeben.

4.2.7.4. Kapselung von SQL-Operationen zu Sequenzen

Die im vorhergehenden Abschnitt angesprochenen Operationen werden für die Durchführung der verschiedenen SQL-Anfragen verwendet. Dies führt bereits implizit zu einer Kapselung der verfügbaren Operationen für die verschiedenen SQL-Anfragen. Ein lesender Zugriff besteht aus einer „OPEN“-Operation, gefolgt von einer oder mehreren „FETCH“-Operationen. Mit der Operation „CLOSE“ wird der lesende Zugriff beendet. Wie bereits angesprochen kann einer „OPEN“-Operation auch noch eine „PREPARE“-Operation vorangehen. Schreibende Zugriffe auf die Datenbank mittels „Insert“, „Update“ und „Delete“-Querys werden durch die Operation „EXECSTM“ ausgeführt und können in beliebiger Reihenfolge aufeinander folgen, bevor die Datenbank-Transaktion mittels einer „commit“-Anweisung innerhalb einer „EXECSTA“-Operation geschlossen wird. „Update“- und „Insert“-Querys können optional noch einer „PREPARE“-Operation vorausgehen. Damit können einzelne Operationen des SQL-Trace zu Sequenzen gruppiert werden.

4.2.8. Identifikation von SQL-Anweisung zum Füllen der Puffer

Für die Parametrisierung des Modells ist es nötig, SQL-Aktionen, welche nur zum einmaligen Füllen der Puffer aufgerufen werden, zu identifizieren, da diese nicht bei jeder Ausführung der zu modellierenden Workload-Schritte ausgeführt werden, sondern nur nach einer Verdrängung oder Invalidierung der entsprechenden Puffereinträge. Diese Identifikation geschieht durch Analyse der Felder „Programm“, „Objektname“ und „Anweisung“, um festzustellen, ob sich aufeinanderfolgende Einträge von „Puffer“- und „SQL“-Traces auf die gleiche Tabelle beziehen und innerhalb des gleichen Programms ausgeführt werden. Dies ist beispielhaft in Abbildung 4-24 dargestellt. Ist dies der Fall, so muss die Spezifikation der Pufferzugriffe mit der SQL-Anweisung verglichen werden. Wird hier festgestellt, dass die gleichen Sätze der Tabelle sowohl durch die Puffer- als auch durch die SQL-Anweisungen angefragt werden, so dienen die SQL-Anweisungen dieser Sequenz dem Füllen der Puffer. Beispielhaft ist dies in Abbildung 4-24 und Abbildung 4-25 dargestellt, welche Einträge des Puffer-Traces (hellblau hinterlegte Zeilen) und des SQL-Traces (gelb hinterlegte Zeilen) zeigen.

hh:mm:ss.ms	Dauer	Programm	Objektname	Oper
21:38:55.203	5	SAPLCPE_CHECK_AUTHORIZATION	UST12	OPEN
21:38:55.203	6	SAPLCPE_CHECK_AUTHORIZATION	UST12	LD OPEN
21:38:55.203	678	SAPLCPE_CHECK_AUTHORIZATION	UST12	PREPARE
21:38:55.204	503	SAPLCPE_CHECK_AUTHORIZATION	UST12	OPEN
21:38:55.205	28	SAPLCPE_CHECK_AUTHORIZATION	UST12	FETCH
21:38:55.205	2	SAPLCPE_CHECK_AUTHORIZATION	UST12	FETCH
21:38:55.205	9	SAPLCPE_CHECK_AUTHORIZATION	UST12	CLOSE
21:38:55.205	18	SAPLCPE_CHECK_AUTHORIZATION	UST12	LD END

Abbildung 4-24: Füllen der Puffer – Zusammenhang Operation, Objektname und Programm
Quelle: Screenshot „ST05“ (SAP 2011i)

In diesen Einträgen wird das Füllen der Tabellenpuffer mit Datensätzen der Tabelle „UST12“ innerhalb des Programms „SAPLCPE_CHECK_AUTHORIZATION“ beschrieben. Hierbei wird in der ersten Zeile versucht, die Datensätze der Tabelle „UST12“, welche im Feld „MANDT“ „900“, im Feld „OBJCT“, „CPE_SETTIN“ und im Feld „AUTH“ den Wert „&_SAP_ALL“ besitzen, aus dem Puffer zu lesen. Dies zeigt die Spezifikation in Abbildung 4-25. Da diese nicht vorhanden ist, wird in der zweiten Zeile das Laden der betreffenden Sätze aus der Datenbank mit der Operation „LD OPEN“ angestoßen.

Oper	Anweisung
OPEN	R 50 900CPE_SETTIN&_SAP_ALL
LD OPEN	R 50 900CPE_SETTIN&_SAP_ALL
PREPARE	SELECT WHERE "MANDT" = ? AND "OBJCT" = ? AND "AUTH" = ? ORDER BY "MANDT" , "OBJCT
OPEN	SELECT WHERE "MANDT" = '900' AND "OBJCT" = 'CPE_SETTIN' AND "AUTH" = '&_SAP_ALL'
FETCH	
FETCH	
CLOSE	
LD END	R 0

Abbildung 4-25: Füllen der Puffer - Zusammenhang Operation und Anweisung
Quelle: Screenshot „ST05“ (SAP 2011i)

Das Holen der Datensätze wird durch die Sequenz von SQL-Operationen der Zeilen 3 bis 7 in Abbildung 4-25 ausgeführt. Hier zeigt sich, dass das SQL-Kommando genau die Datensätze aus der Datenbank liest, welche zuvor in Zeile 1 aus dem generischen Tabellenpuffer angefragt wurden.

4.2.9. Automatische Analyse der Performance-Trace

Die SAP-Performance-Traces bieten eine Fülle von Informationen. Dies verursacht, wie bereits erwähnt, zusätzliche Last auf dem SAP-System. Deshalb und aufgrund der Menge von Informationen ist es nötig, die Erstellung der gewünschten Traces nur für den Testfall zu aktivieren und Daten aus dem System automatisch auszulesen und zu analysieren.

4.2.9.1. Starten und Stoppen

Um innerhalb einer Testausführung die Aufzeichnung unnötiger Daten zu vermeiden, ist es nötig, die einzelnen SAP-Performance-Traces unmittelbar vor und nach der Ausführung des Tests zu starten bzw. zu stoppen. Dies geschieht am effektivsten, indem der Trace direkt aus der Testausführung heraus gestartet und gestoppt wird. Um dies zu erreichen, wurden zwei remotefähige BAPIs für das Starten und Stoppen der Traces identifiziert.

Der standardmäßig bereitgestellte Baustein „PERFORMANCEA_TRACE_ON_ALL“ bietet die Möglichkeit, die verschiedenen Traces in einem Zielmandanten für bestimmte Benutzer zu aktivieren. Es werden folgende Parameter angeboten:

- BUF_TRACE_ON
- ENQ_TRACE_ON
- RFC_TRACE_ON
- SQL_TRACE_ON
- CLIENT
- TRACE_USER

Um die Trace-Aufzeichnung für eine Menge von Benutzern zu starten, ist es möglich, innerhalb des Parameters „TRACE_USER“ Wildcards zu verwenden.

Die Beendigung der Aufzeichnung kann auf dem gleichen Weg über den Baustein „PERFORMANCEA_TRACE_OFF“ durchgeführt werden. Für die Parametrisierung stehen folgende Möglichkeiten zur Verfügung:

- BUF_TRACE_OFF
- ENQ_TRACE_OFF
- RFC_TRACE_OFF
- SQL_TRACE_OFF

Hierbei ist es nicht möglich, einen oder mehrere Traces für einen bestimmten Benutzer zu beenden und den gleichen Trace für einen anderen ausgewählten Benutzer aktiv zu lassen.

4.2.9.2. Export der Performance Trace-Daten

Um die trotz genauer Filterung große Menge an aufgezeichneten Daten analysieren zu können, ist es nötig, diese Daten auszulesen. Im Rahmen dieser Arbeit konnte kein Baustein in entsprechenden Umfang identifiziert werden, welcher es ermöglicht, vollständig auf die aufgezeichneten Daten zuzugreifen.

Um so viele Daten wie möglich im Analyse-Tool zur Verfügung zu stellen, wurde deshalb eine Alternative unter Verwendung der Export-Funktionalität im SAPGui entwickelt. Dafür ist es nötig, den Start- und Stopzeitpunkt sowie die weiteren Parameter der Trace-Aufzeichnungen aus der Testausführung zu ermitteln. Mit diesen Informationen kann der entsprechende Trace mittels der Transaktion „ST05“ im SAPGui dargestellt und exportiert werden. Zu diesem Zweck werden 4 verschiedene Formate angeboten:

- unkonvertiert
- Tabellenkalkulation
- Rich Text Format
- HTML Format

Der Export zeigt jedoch die Einschränkung, dass dieser nur die grafisch dargestellten Informationen beinhaltet. Die Detailinformationen zu den jeweiligen Inhalten der Anweisungsfelder werden nicht mit exportiert. Zusätzlich geht beim Export die Zuordnung von Eintrag zu Trace-Typ verloren. In der grafischen Darstellung werden die Einträge entsprechend ihrer Zugehörigkeit zu Puffer-, Enqueue-, RFC- oder SQL-Trace farblich hinterlegt. Um diese Information zu erhalten, ist es nötig, den Export für jeden Trace-Typ einzeln durchzuführen. Damit entstehen beim Export pro Trace-Durchlauf bis zu vier verschiedene Dateien, welche die Aktionen der jeweiligen Traces enthalten. Um die für die im Rahmen dieser Arbeit durchgeführte Performance-Analyse benötigten Daten bereit zu stellen, müssen die im Export enthaltenen Daten im Nachgang aufbereitet werden.

4.2.9.3. Aufbereitung der Daten

Die durch den Export erzeugten Dateien enthalten für jeden Trace-Satz die Informationen zu Startzeitpunkt, Dauer, Programm, Objektname, Operation, Cursor, Array, Sätze, Returncode, Connection und Anweisung. Anhand des in jedem Eintrag vermerkten Timestamps können die Einträge bei der Analyse wieder zueinander in Beziehung gesetzt werden.

Um den im vorhergehenden Absatz angesprochenen Informationsverlust bzgl. der Detailanalyse der Anweisung zu kompensieren, wurde für die verschiedenen Typen von Performance-Trace ein Vorgehen entwickelt, um die für die vorliegende Arbeit benötigten Informationen aus den exportierten Daten zu erhalten.

Bei der Analyse der Pufferzugriffe gehen durch den Export die Informationen über die Aufschlüsselung der Spezifikation bzgl. der verwendeten Felder des Primärschlüssels

(4.2.5.2) verloren. Wie bereits erläutert, besteht diese Spezifikation aus einer Aneinanderreihung der Inhalte der verwendeten Felder des Primärschlüssels.

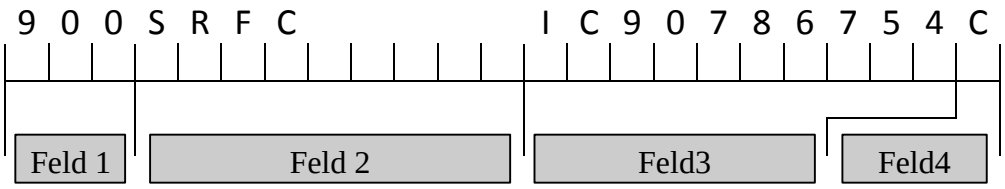


Abbildung 4-26: Spezifikation von Sätzen einer Datenbanktabelle
Quelle: Eigene Darstellung

Für das in Abbildung 4-26 schematisch dargestellte Format konnte der Baustein „RSTR_EXPLAIN_TBUF_STRING“ durch Analyse des Quellcodes der Transaction „ST05“ identifiziert werden, welcher diese Aufschlüsselung von Spezifikationen auf Basis von Informationen aus dem SAP-DDIC ermöglicht. Für die Verwendung dieses Bausteins innerhalb der Analysekomponente musste dieser zuerst kopiert und die Kopie für den externen Aufruf über die RFC bzw. WebService-Schnittstelle angepasst werden. Für die Ausführung benötigt dieser Baustein die folgenden Parameter:

- TABNAME
- SEARCH_LEN
- SEARCH_STRING

Diese Informationen sind, wie in 4.2.5.2 dargestellt, in der Spalte „Anweisung“ des Puffer-Traces vorhanden. Der Baustein liefert eine Tabelle mit den folgenden Spalten:

- NAME
- TYPE
- LENGTH
- VALUETEXT

Die Zeilen beschreiben damit die Felder des Primärschlüssels sowie die für die Spezifikation verwendeten Werte. Die Reihenfolge der Primärschlüsselattribute wird durch die Reihenfolge des Vorkommens in dieser Tabelle von oben nach unten bestimmt.

Damit ist es möglich, die in der Detailanalyse des Puffer-Traces dargestellten Informationen nach dem Export aus den Inhalten des Anweisungsfeldes zu extrahieren, und wie in Abbildung 4-27 dargestellt, die Spezifikation von Inhalten des Tabellenpuffers den verwendeten Feldern des Primärschlüssels zuzuordnen.

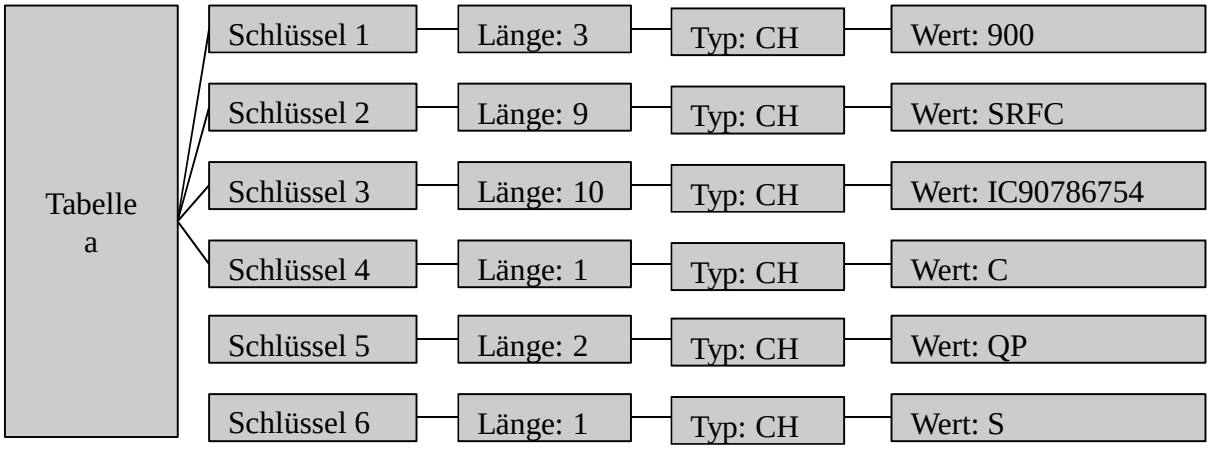


Abbildung 4-27: Aufsplitten der Spezifikation von Tabellenzeilen
Quelle: Eigene Darstellung

Da bei der Spezifikation der durch die Enqueue-Objekte zu sperrenden Datensätze (4.2.6.2) das gleiche Format verwendet wird, kann dieser WebService ebenfalls für die Aufbereitung der Informationen nach dem Export des Enqueue-Traces verwendet werden. Theoretisch ist es möglich, in einem Sperrobjekt mehrere Sperren für verschiedene Tabellen zu setzen. Die Informationen über weitere Sperren gehen mit diesem Export verloren. Für die vorliegende Arbeit wurden alle im Referenzprozess verwendeten Sperrobjekte überprüft, sodass ein Informationsverlust durch den Export der Enqueue-Traces ausgeschlossen werden konnte.

Da die Sätze des SQL-Traces die jeweiligen SQL-Anweisungen in der Spalte „Anweisung“ enthalten können diese geparst werden. So werden die betroffenen Tabellensätze der „Select“-, „Insert“-, „Update“- und „Delete“-Statements identifiziert.

4.3. Zusammenfassung

In diesem Kapitel wurden Messwerkzeuge vorgestellt mit welchen sowohl die Performance des SAP-ERP-Systems und dessen Komponenten als auch die Leistungsdaten des unterliegenden Betriebssystems aufgezeichnet werden können. Damit können die im vorhergehenden Kapitel aufgezeigten, und für die Modellierung verwendeten Komponenten analysiert werden und für Modellierung, Parametrisierung und Evaluation verwendet werden. Dieses Kapitel beantwortet die Forschungsfrage 1 bzgl. der Messwerkzeuge und –vorgänge für Analyse, Modellierung, Parametrisierung und Evaluation.

5. Modellerstellung und Parametrisierung

5.1. Einflusskriterien der Systemkomponenten auf die Gesamtantwortzeit des Systems

Die Anzahl von möglichen Einflüssen auf die Antwortzeit eines SAP-ERP-Systems ist grundsätzlich nicht beschränkt. Aus diesem Grund müssen bei der Erstellung eines Modells gewisse Grundannahmen getroffen werden, damit die zu untersuchenden Einflussgrößen dem zeitlichen und inhaltlichen Umfang dieser Arbeit entsprechen. Jedoch wird die Relevanz des in Abschnitt 5.2 erstellten Modells bewahrt, indem dieses die wichtigsten Einflussgrößen auf die Antwortzeit aus der Literatur und der Dokumentation berücksichtigt.

5.1.1. Grundannahmen

Für das in dieser Arbeit erstellte Performance-Modell der Antwortzeit eines SAP-ERP-Systems wurden einige Grundannahmen vorausgesetzt. Die große Menge an Puffern wurde auf die Betrachtung der Tabellenpuffer reduziert, da diese die häufigsten Modifikationen und Zugriffe aufweisen. Zudem können die benötigte Größe dieser Puffer ex ante durch eine Messung vorherbestimmt werden, sodass deren Konfiguration keiner Simulation bedarf. Dies führt zu der Grundannahme, dass das Modell nur für die Simulation der Tabellenpuffer verwendet wird, und die restlichen Puffer ausreichend dimensioniert sind.

5.2. Modellierung der Systemkomponenten

Wie bereits angesprochen wurden die in diesem Modell berücksichtigten Komponenten und deren Einflussmöglichkeiten auf die Antwortzeit durch eine Analyse der verfügbaren relevanten Literatur und Dokumentation ausgewählt.

5.2.1. Workload

In diesem Abschnitt wird zuerst theoretisch eine Beschreibungsmöglichkeit für verschiedene Arten von Workload anhand verschiedener Kriterien dargestellt. Die Eigenschaften des in der vorliegenden Arbeit untersuchten SAP-ERP-Systems werden dann im Hinblick auf die theoretischen Kriterien untersucht.

5.2.1.1. Workload-Dimensionen

Um die Performance eines Systems mittels eines Warteschlangenmodells nachzustellen, werden nach (Menascé/Dowdy/Almeida 2004) grundsätzlich zwei Arten von Informationen benötigt:

- Servicezeit:
  Die durchschnittliche Zeit, die ein Servicecenter benötigt, um eine bestimmte Klasse von Anfragen zu bearbeiten.

- Lastintensität:
  Durch die Lastintensität wird die Last, welche von einem System bearbeitet werden soll, gemessen.

Durch diese beiden Angaben kann ein Modell parametrisiert werden.

5.2.1.2. Workload-Typen

Mit Ausnahme der lastabhängigen Warteschlangensysteme, bei denen die Servicezeit von der Lastintensität abhängt, klassifizieren (Menascé/Dowdy/Almeida 2004; Lazowska et al. 1984) die folgenden Workload-Typen mit statischer Servicezeit anhand verschiedener Ausprägungen der Lastintensität:

- Interaktiver Workload:
  Bei einem interaktiven Workload handelt es sich um eine feste Anzahl von Benutzern, die Anfragen an das System stellen. Nach erhaltener Antwort erfolgt die nächste Anfrage nach einer system- und benutzerabhängigen Wartezeit, der sogenannten „Think time“. Damit ist die Lastintensität über die Anzahl der Benutzer und deren Thinktime vollständig definiert.

- Transaktionsbasierter Workload:
  Bei diesem Typ von Workload wird die Last nicht, wie bei interaktivem Workload, innerhalb des Warteschlangensystems erzeugt, sondern die Aufträge kommen von außen in das System. Die Intensität wird bei diesem Workload-Typ über eine Ankunftsrate definiert.

- Batch-Workload:
  Ein Workload vom Typ Batch ist charakterisiert durch eine fixe Anzahl an Anfragen, welche vom System abzuarbeiten sind. Die Intensität ist durch diese Anzahl definiert.

5.2.1.3. Gruppierung von Aufträgen

Da Warteschlangen-Systeme sowohl eine als auch mehrere Klassen von Aufträgen unterstützen, ist diese Klassifizierung nach (Menascé/Dowdy/Almeida 2004) in den folgenden Situationen sinnvoll:

- Heterogene Service-Zeiten:
  Wenn die Service-Zeiten der verschiedenen beteiligten Komponenten einer Auftragsverarbeitung signifikant unterschiedliche, durchschnittliche Servicezeiten besitzen, können diese in Gruppen zusammengefasst werden.

- Unterschiedliche Typen von Workload:
  Gruppierungen sind sinnvoll, wenn gleichzeitig verschiedene Typen, wie transaktionsbasiert und interaktiv, verwendet werden.

- Unterschiedliche Service Level:
  Da verschiedene Service Level oft eine unterschiedliche Analyse verlangen, sollten die unterschiedlichen Anfragen gruppiert werden.

Bei der Modellierung von Anfragen und Systemkomponenten ist auf die Gruppierung zu achten, damit neben einer exakten Parametrisierung auch eine detaillierte Auswertung der Simulationsergebnisse möglich ist.

5.2.1.4. Grenzen des Modells

Für die Modellierung der Performance eines Systems müssen die in diesem Abschnitt definierten Informationen über den Workload verfügbar sein. Nach (Menascé/Almeida 2000) können die Informationen über die Lastintensität über die durchschnittlich erwartete Anzahl von Benutzern bzw. deren Verhalten abgeschätzt werden. Grundsätzlich bleibt jedoch bestehen, dass ein Modell nicht generell die Performance eines Systems abbilden kann. Das Modell beinhaltet den Workload implizit. Die statische Konfiguration der Warteschlangen-Systeme kann unabhängig vom Workload sein, das vollständige Modell nicht.

5.2.1.5. Workload von SAP-ERP-Systemen

Grundsätzlich werden SAP-ERP-Systeme sowohl für die Ver- und Bearbeitung von interaktiven Benutzeranfragen als auch für Hintergrundverarbeitung verwendet. Da dies jedoch häufig gegensätzliche Ziele darstellt, bietet das SAP-System die Möglichkeit, verschiedene Betriebsmodi, häufig auch Tag- und Nachbetrieb genannt, für Dialog- und Hintergrundbearbeitung zu definieren. Diese Modi können zeitlich gesteuert werden, sodass das System damit, ohne Neustart, mit zwei verschiedenen Konfigurationen betrieben werden kann.

Da das Ziel der vorliegenden Arbeit die Performance-Vorhersage unter einer steigenden Anzahl von Benutzer ist, und die Verarbeitung von Hintergrundjobs häufig konzentriert über die Nachtstunden (Zeit mit wenig Dialog-Last) mit Hilfe einer unterschiedlichen Konfiguration durchgeführt wird, wird die Hintergrundverarbeitung nicht weiter berücksichtigt und einer späteren Bearbeitung vorbehalten. In dieser Arbeit wird hingegen auf Vorhersage der Performance eines für die klassischen Dialog-Anfragen konfiguriertes System fokussiert und das Modell damit auf einen interaktiven Workload abgestimmt. Dies impliziert die Definition der Lastintensität durch Anzahl der Benutzer und deren Thinktime.

5.2.1.6. Service-Zeiten eines SAP-ERP-Systems

Wie in 5.2.1.2 bereits vorgestellt, gibt es verschiedene Typen von Servicezeiten. Um festzustellen, ob die Servicezeit von der Lastintensität abhängt oder davon unabhängig ist, wurden mehrere Testreihen eines interaktiven Workloads unter einer steigenden Anzahl von Benutzern durchgeführt. Dabei wurden die Komponenten der Antwortzeit eines SAP-ERP-Systems, wie in 3.6 ausführlich dargestellt, vermessen.

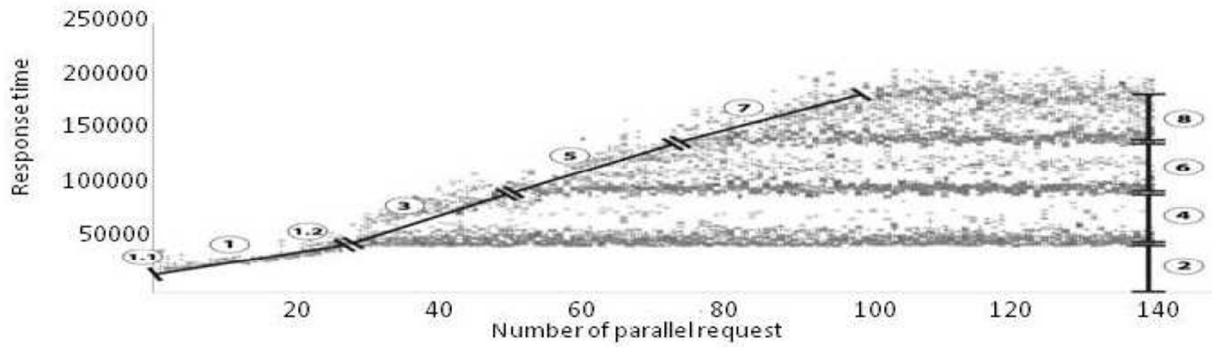


Abbildung 5-1: Antwortzeiten in Abhängigkeit des Anzahl von parallelen Anfragen
Quelle: (Gradl et al. 2011b)

Es sei an dieser Stelle noch darauf hingewiesen, dass diese Messungen nach einer Einschwingphase von drei Wiederholungen des Testfalls mit maximaler Parallelität durchgeführt wurden.

Abbildung 5-1 zeigt die Antwortzeiten einer Testausführung mit einer steigenden Benutzeranzahl von einem bis 140 parallelen Benutzern. Dabei ist zu erkennen, dass sich das Verhalten in 3 Teile spalten lässt.

Teil eins ist charakterisiert durch eine langsam ansteigende Antwortzeit, die, wie später noch dargestellt wird, dem langsam ansteigenden Zeitanteil in der Warteschlange des Enqueue-Prozesses geschuldet ist. Bis zum Punkt 1.2 können die ankommenden Anfragen parallel durch das System bearbeitet werden.

Im zweiten Abschnitt treffen mehr Anfragen am SAP-System ein als gleichzeitig abgearbeitet werden können. Damit steigt der Zeitanteil der Antwortzeit, welcher in der Warteschlange des Dispatcher-Prozesses verbracht wird. Die ersten 30 Anfragen, die am System eintreffen, können sofort bearbeitet werden und verbringen keine Zeit in der Dispatcher Queue. Dies zeigt sich in der Antwortzeit, welche in Abbildung 5-1 durch Ziffer 2 dargestellt ist. Sobald eine Anfrage abgearbeitet ist, wird durch den Dispatcher die nächste Anfrage gescheduled. Diese hat jedoch schon eine gewisse Zeit in der Dispatcher-Warteschlange verbracht und hat daher eine Queueing-Zeit, die durchschnittlich in etwas der Antwortzeit ohne Warteschlangenanteil entspricht. Damit erhöht sich die Gesamtantwortzeit entsprechend dem Queueinganteil, wie durch Ziffern 4, 6 und 8 dargestellt ist.

Die Anfragen bleiben so lange in der Queue des Dispatchers, bis die Anfrage bearbeitet wird, weil ein freier Work-Prozess zur Verfügung steht oder die konfigurierte maximale Antwortzeit überschritten ist. Dies stellt die Grenze zu Teil drei dar. In diesem Abschnitt bleibt die maximale Antwortzeit konstant, da Anfragen mit einer höheren Antwortzeit abgebrochen werden.

Um jedoch die Performance eine SAP-ERP-Systems durch die Modellierung der Antwortzeit nachzustellen, ist es nötig, die einzelnen Komponenten dieser Antwortzeit zu analysieren und sie entsprechend ihrer Abhängigkeit von der Lastintensität zu vermessen. Deshalb wurden analog zu der Messung der Gesamtantwortzeit in Abbildung 5-1 die Komponenten der Antwortzeit mittels des in 4.2.1 vorgestellten Messwerkzeugs vermessen.

Abbildung 5-2 zeigt die Anzahl von CPU-Mikrosekunden, die pro Anfrage benötigt werden. Auf der x-Achse ist die Reihenfolge der Anfragen abgetragen. Im Diagramm entsprechen damit die Punkte am Anfang der x-Achse Testläufen mit weniger parallelen Benutzern, wohingegen die Werte bei ca. 5000 schon der CPU-Zeit bei maximal parallelen Anfragen entsprechen. Damit zeigt sich dass die CPU-Zeit zwar Schwankungen unterworfen ist, jedoch diese sich nicht mit der Anzahl der parallel ausgeführten Anfragen ändert sodass in der vorliegenden Arbeit davon ausgegangen wird, dass die Servicezeiten der CPU lastunabhängig sind und deshalb für die Modellierung lastunabhängige Warteschlangensysteme verwendet werden können.

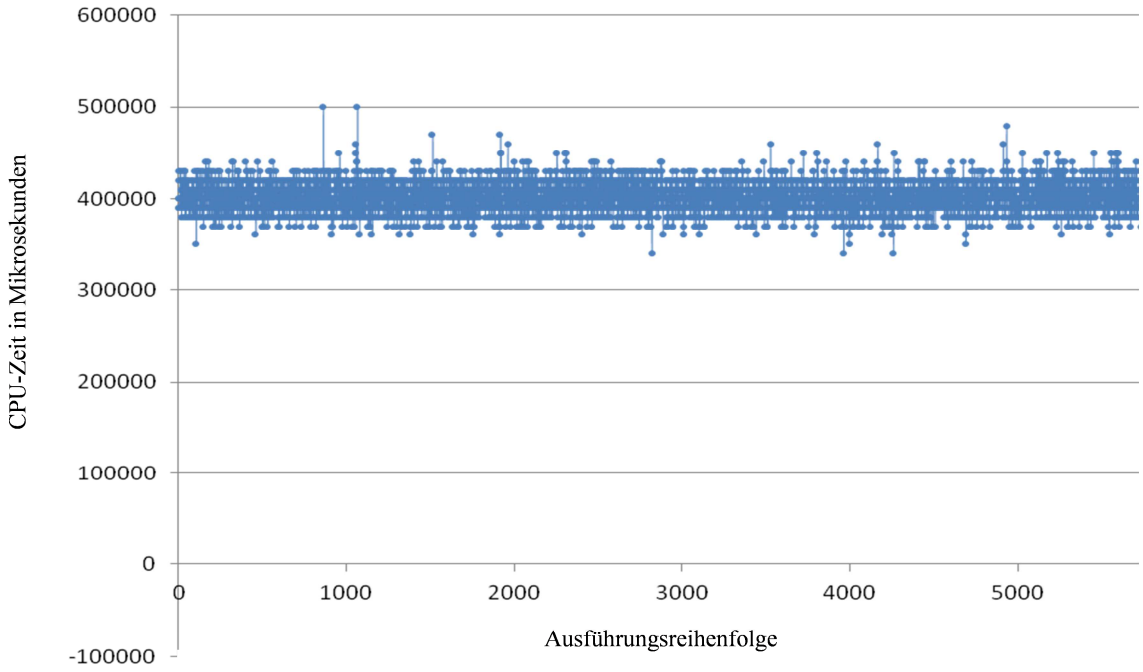


Abbildung 5-2: Durch die Anfragen verwendete CPU Zeit in Abhängigkeit des Ausführungszeitpunkts
Quelle: Eigene Darstellung

Ähnlich zur CPU-Zeit pro bearbeiteter Anfrage verhalten sich die Datenbankzeiten. Abbildung 5-3 zeigt den Datenbankanteil der Antwortzeit. Auf der x-Achse ist die Reihenfolge der Anfragen aufgetragen.

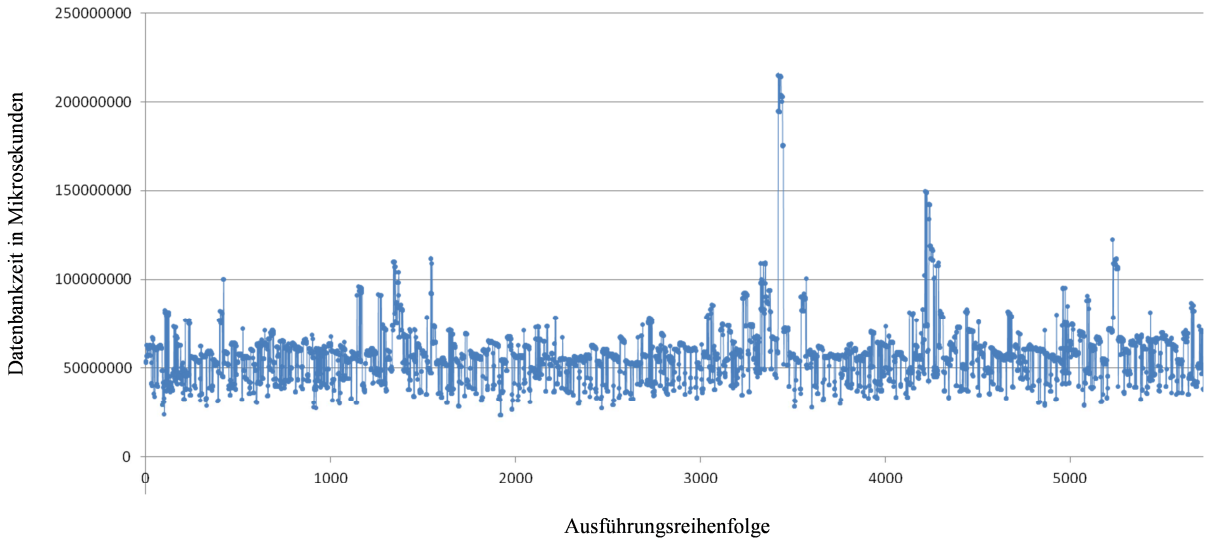


Abbildung 5-3: Datenbankanteil der Antwortzeiten in Abhängigkeit des Ausführungszeitpunkts
Quelle: Eigene Darstellung

Die y-Achse zeigt die Datenbankzeit in Mikrosekunden. Dies verdeutlicht, dass auch die Datenbankzeit große Schwankungen aufweist. Es lässt sich jedoch kein Muster erkennen, das auf eine Abhängigkeit der Datenbankzeit von der Lastintensität hindeutet. Wäre dies der Fall, so würde die Datenbankzeit mit steigendem Wert auf der x-Achse ebenfalls ansteigen, da im Bereich ab der 5000. Anfrage bereits die maximale Lastintensität erreicht ist. Daraus wird

geschlossen, dass die Ausprägung der Datenbankzeit nicht von der Intensität der Last abhängt und damit für den Testlauf ebenfalls die Datenbank als lastunabhängiges Warteschlangensystem modelliert werden kann.

Ein anderes Verhalten zeigen die Enqueue-Vorgänge. Dabei ist zu beachten, dass ein Enqueue-Vorgang aus 4 Schritten besteht:

- Bearbeitung der Anfrage durch den Enqueue-Server
- Setzen der Sperre
- Dauer der Sperre
- Lösen der Sperre

Im ersten Schritt wird das Setzen einer Sperre beim Enqueue-Server beantragt. Dies ist abhängig von der Auslastung des Enqueue-Servers selbst. Da der Enqueue-Server keine Möglichkeit zur parallelen Abarbeitung von Anträgen enthält, werden diese sequentiell bearbeitet. Damit entstehen Wartezeiten, bis der Antrag durch den Enqueue-Server abgearbeitet werden kann, welche von der Lastintensität abhängen.

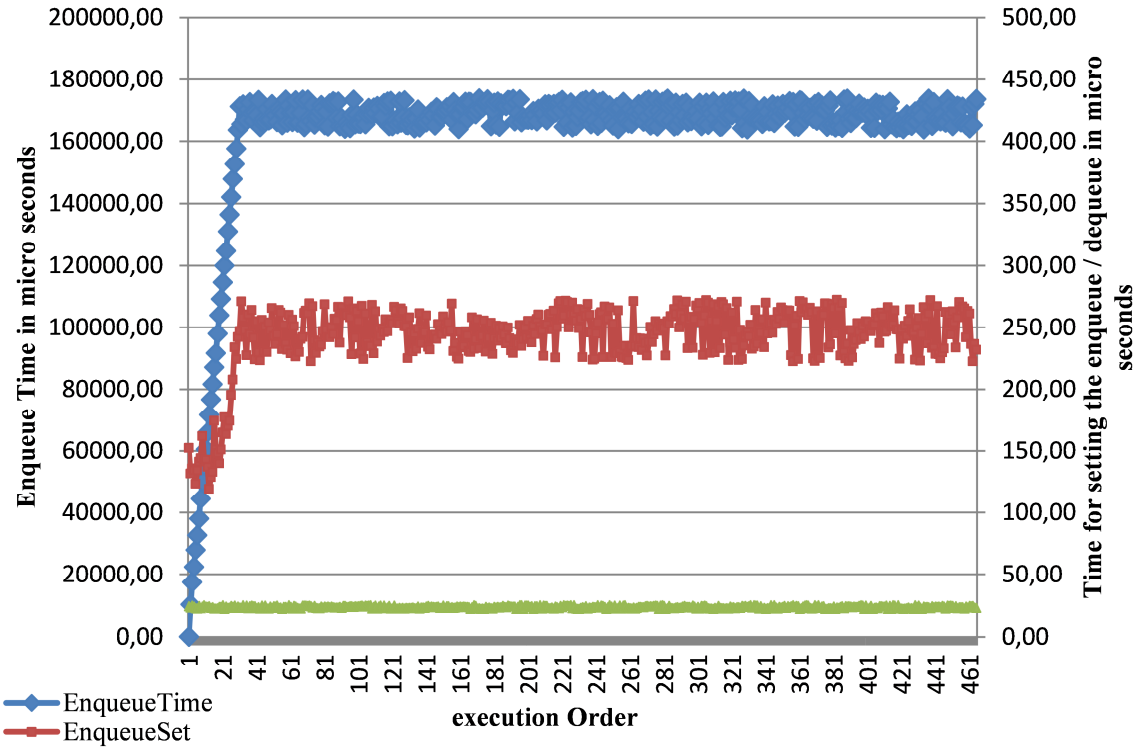


Abbildung 5-4: Enqueue-Zeiten in Abhängigkeit des Ausführungszeitpunkts
Quelle: Eigene Darstellung

Das Setzen der Sperre im zweiten Schritt ist abhängig von den bereits existierenden Sperren, da ein Objekt nur gesperrt werden kann, wenn dieses nicht mit bereits gesetzten Sperren kollidiert. Ist dies der Fall, so muss gewartet werden, bis diese Kollision nicht mehr besteht, also die betroffene(n) Sperre(n) gelöst ist/sind. Auch in diesem Schritt können Wartezeiten entstehen, die auf die Lastintensität zurückzuführen sind.

Die wirkliche Dauer der Sperre in Schritt drei hängt von der Dauer der Aktion ab, für welche die Sperre gesetzt wurde. Die Abhängigkeit der Servicezeit dieses Abschnitts von der Lastintensität kann nicht generell beurteilt werden, da im Enqueue-Server für diesen Schritt keine Ressourcen benötigt werden.

Das Lösen der Sperre in Schritt vier benötigt zu vernachlässigende Ressourcen. Den Messungen, welche in Abbildung 5-4 dargestellt sind, zeigt die Servicezeit für das Lösen von Sperren („Dequeue“) keine Queueing-Effekte am Enqueue-Server. Es scheint, als hätten „Dequeue“-Anfragen immer oberste Priorität und würden vom Enqueue-Server sofort bei deren Eintreffen bearbeitet. Dies erscheint insofern als nötig, da andernfalls Deadlock-Situationen auftreten könnten, wenn eine „Dequeue“-Anfrage auf Objekt X in der Warteschlange des Enqueue-Servers hinter einer „Enqueue“-Anfrage auf das gleiche Objekt wäre.

Grundsätzlich wird bei Performance-Analysen nur die „Enqueue“-Zeit (Schritt eins und zwei) in Betracht gezogen. Für die Modellierung des Enqueue-Vorgangs ist es jedoch nötig, alle vier Schritte und die daraus resultierenden Wechselwirkungen zu analysieren und im Modell zu berücksichtigen. Abbildung 5-4 zeigt die Zeiten für das Setzen („Enqueue“) und das Lösen („Dequeue“) von Sperren sowie die kumulierte Zeit des Enqueue-Vorgangs („EnqueueSet“ und „Enqueue“) pro gestellter Anfrage an das SAP-ERP-System in Abhängigkeit zum Zeitpunkt der Anfrage. Die kumulierte Zeit wurde mittels STAD-Records (4.2.1) ermittelt. Sie enthält die Gesamtdauer der beiden ersten Schritte eines Enqueue-Vorgangs. Die Zeiten für das Setzen und Lösen von Sperren können nur mittels der SAP-Performance-Traces (4.2.6) identifiziert werden. In diesen Zeiten ist jedoch die Wartezeit auf den Enqueue-Server und die Dauer des „EnqueueSet“-Vorgangs aufaddiert. Es konnte keine Möglichkeit identifiziert werden, um diese Summe in ihre beiden Bestandteile aufzuteilen, und so die in dieser Arbeit konzeptionierte Modellierung der Enqueue-Vorgänge mit Messdaten zu untermauern.

Die in Abbildung 5-4 dargestellten Zeiten für das Setzen von Sperren („EnqueueSet“) zeigt bis zur Auslastung aller Work-Prozesse im System eine steigende Tendenz. Aufgrund der Struktur des Enqueue-Servers, welcher als separater Prozess auf Betriebssystemebene mit einem eigenen Hauptspeicherbereich für die Sperrtabelle implementiert ist, kann die Entwicklung der „EnqueueSet“-Zeiten durch das Warten auf den Enqueue-Server interpretiert werden. Diese zeigt sich dann, sobald die maximale Anzahl von Work-Prozessen erreicht ist, als unabhängig von der Lastintensität.

Der deutlich steilere Anstieg der kumulierten „Enqueue“-Zeit in Abbildung 5-4 ist zum einen natürlich durch den Anstieg der „EnqueueSet“-Zeit verursacht. Jedoch kann dies allein aufgrund der Größenordnung der Zeiteinheiten nicht die alleinige Ursache sein. Da diese Zeiten jedoch aus der Summe der ersten beiden Schritte besteht, folgt daraus, dass die Differenz der Zeiten „Enqueue“ und „EnqueueSet“ für das Setzen der Sperre benötigt werden. Diese Differenz kann aus einer lastabhängigen Servicezeit des Enqueue-Servers und/oder aus der Wartezeit, bis eine Sperre gesetzt werden kann, bestehen. In dieser Arbeit wird aus Gründen der Komplexitätsreduktion und der Beschreibung der „Enqueue“-Zeit in der Transaktion STAD (SAP 2008a) die Servicezeit des Enqueue-Servers als unabhängig von der Last angenommen, und so in das zu erstellende Konzept des Performance-Modells eines SAP-ERP-Systems integriert.

Aus den in diesem Abschnitt beschriebenen Komponentenzeiten der Antwortzeit zeigt sich, dass diese eine Streuung aufweisen, jedoch nicht als lastabhängig charakterisiert werden müssen. Dies kann als Erklärung für die Streuung der Antwort in den Abschnitten 2, 4, 6 und 8 in Abbildung 5-1 dienen, jedoch nicht für die Existenz dieser Abschnitte.

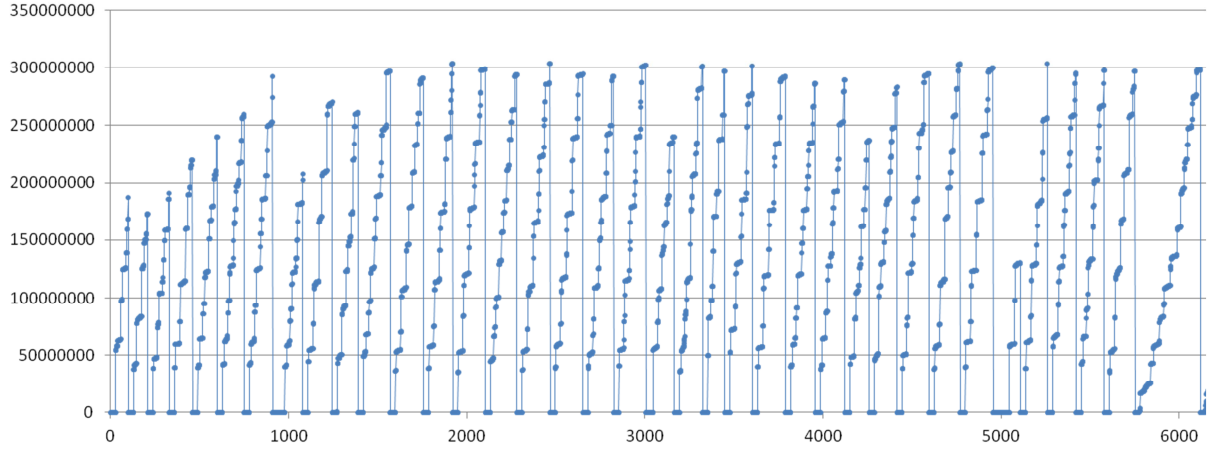


Abbildung 5-5: Dauer der Wartezeit in der Dispatcher-Queue in Abhängigkeit des Ausführungszeitpunkts
Quelle: Eigene Darstellung

Die Ursache für die Bildung dieser Abschnitte ist in Abbildung 5-5 dargestellt, in dem die Wartezeit pro Anfrage in der Warteschlange des Dispatchers in Abhängigkeit des Ausführungszeitpunktes veranschaulicht ist. Wie dem Diagramm zu entnehmen ist, gibt es in jedem „Zacken“ eine Menge von Anfragen, deren Wartezeit gleich null ist, eine Menge von Anfragen mit einer Wartezeit von ca. 50 Sekunden usw. Dies zeigt deutlich den bereits beschriebenen Queueing-Mechanismus im Dispatcher-Prozess und ist verantwortlich für die Bildung der Abschnitte in Abbildung 5-1.

5.2.2. Client

Wie bereits in den vorhergehenden Kapiteln diskutiert, liegt dem zu modellierenden System ein interaktiver Workload zu Grunde. Dem zufolge wird die Lastintensität durch die Anzahl der Benutzer und deren Thinktime definiert. Da ein SAP-ERP-System jedoch die Möglichkeit bietet, mehrere Geschäftsprozesse abzuwickeln und nicht alle Benutzer zwangsweise die gleichen Prozesse abarbeiten, wird die Lastintensität in der vorliegenden Arbeit durch das Tupel (Anzahl Benutzer, Thinktime, Geschäftsprozess) definiert.

5.2.2.1. Anforderungen an das Sub-Modell

Da es in einem System unterschiedliche Nutzergruppen gibt (Reilly) muss das Modell die Möglichkeit bieten, unterschiedliche Nutzergruppen, welche sich durch Anzahl der Benutzer, deren Thinktime und des bearbeiteten Geschäftsprozesses unterscheiden. Dies führt zu den folgenden Anforderungen an das Modell:

- Modellierung unterschiedlicher Benutzergruppen
- Variation der Thinktime der unterschiedlichen Gruppen

- Variation der Mächtigkeit der unterschiedlichen Gruppen
- Möglichkeit der Zuordnung von Gruppen zu unterschiedlichen Geschäftsprozessen

Die Erfüllung dieser Anforderungen ist nötig, um die verschiedenen Workloads, die von einem System in der Praxis bearbeitet werden müssen, zu modellieren.

5.2.2.2. Umsetzung

In diesem Absatz soll nun das Modell für die Nachbildung der Lastintensität des Workloads entwickelt werden. Dem LQN-Ansatz folgend steht für die Modellierung der Lastintensität sowohl die Spezifikation einer Ankunftsrate als auch eines Referenz-Tasks mit einer zu definierenden Thinktime zur Verfügung. Aufgrund der bereits erhobenen Anforderungen werden für die verschiedenen Benutzergruppen Referenz-Tasks verwendet. Dabei wird eine Benutzergruppe jeweils durch einen Task im LQN-Terminus modelliert. Innerhalb dieses Tasks besteht für jeden auszuführenden Workload-Schritt eine eigene Aktivität mit einer zu definierenden Thinktime t. Damit wird dem Umstand Rechnung getragen, dass Benutzer nach jedem Schritt eine Zeitspanne für die Vorbereitung der Daten und für deren Eingabe benötigen. Zusätzlich wird damit die Grundlage für die Umsetzung der Anforderung nach einer Zuordnung von Benutzergruppen zu einzelnen Geschäftsprozessschritten gelegt. Diese erfolgt über die Kombination der Aktivitäten des „Benutzergruppen“-Tasks zu den Elementen des Modells, welche die Durchführung der Schritte des Geschäftsprozesses abbilden. Die Anzahl der Benutzer pro Gruppe wird durch die Multiplizität dieses Tasks abgebildet. Die Umsetzung der Benutzerkomponente des Performance-Modells ist in Abbildung 5-6 dargestellt.

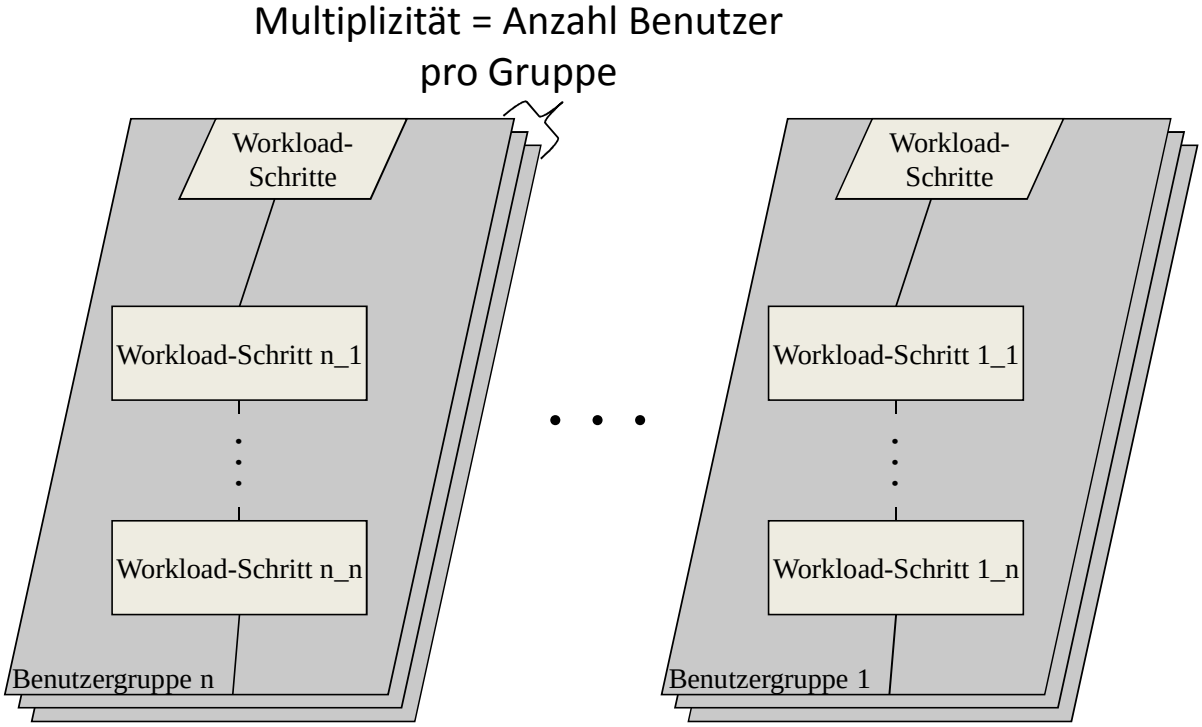


Abbildung 5-6: Modellierung verschiedener Workload-Intensitäten
Quelle: Eigene Abbildung

5.2.3. Lastschritte

Die durch das SAP-System zu prozessierende Last ergibt sich neben der Lastintensität durch die benötigten Systemressourcen. Diese bestehen aus der Nutzung der in Abschnitt 3 dargestellten Komponenten und der dafür benötigten Antwortzeit.

5.2.3.1. Anforderungen an das Sub-Modell

Um nun ein Modell für die Antwortzeit eines SAP-ERP-Systems zu erstellen, müssen Modellkomponenten konzipiert werden, welche die benötigten Ressourcen und vor allem deren Antwortzeit für jeden Schritt des Geschäftsprozesses kapseln, sodass diese dann den einzelnen Benutzeranfragen zugeordnet werden können. Ein Schritt eines Geschäftsprozesses wird aufgrund der verfügbaren Granularität der Daten als Ausführung eines ABAP-Programms modelliert. Dies hat zur Folge, dass die Ausführung eines Geschäftsprozessschrittes mit der Verwendung eines Work-Prozesses des SAP-Systems korreliert und damit nur durchgeführt werden kann, wenn ein freier Work-Prozess zur Verfügung steht. Da die Zuordnung von Anfragen zu freien Work-Prozessen durch den Dispatcher-Prozess durchgeführt wird, muss dieses Submodell auch dessen Eigenschaften umsetzen. Damit ergeben sich für das Sub-Modell die folgenden Anforderungen:

- Kapselung der Antwortzeiten der an einem Geschäftsprozessschritt beteiligten Systemkomponenten
- Modellierung und Variation der Anzahl an verfügbaren Work-Prozessen
- Modellierung der Warteschlange des Dispatchers
- Modellierung von Abbrüchen nach Überschreiten einer konfigurierten maximalen Antwortzeit

Da sich die Antwortzeit für eine Anfrage aus den Service-Zeiten der beteiligten Komponenten ergibt, kann die Anforderung nach der Modellierung einer maximalen Antwortzeit nicht mit Mitteln des LQN-Modells erfolgen. Die Identifikation von Verbindungsabbrüchen aufgrund der Überschreitung dieser maximalen Antwortzeit muss daher in der Auswertung des Ergebnisses eines Simulationslaufs, siehe Abschnitt 6.5.3.3, erfolgen.

5.2.3.2. Umsetzung

Um die Antwortzeiten der einzelnen, an einem Schritt des Geschäftsprozesses beteiligten Komponenten zu kapseln, wird in dem in dieser Arbeit entwickelten Modell ein Task verwendet, der sowohl die Anfragen der Benutzerkomponenten beantwortet, als auch Anfragen an die beteiligten Modellkomponenten stellt. Diese Anfragen werden als synchrone Verbindungen modelliert, da der Task die Antwort erst an die Benutzerkomponenten zurückliefert, wenn alle von ihm gestellten Anfragen beantwortet sind. Damit kapselt der Task alle Servicezeiten der an der Beantwortung einer Benutzeranfrage beteiligten Modellkomponenten. Die unterschiedlichen Schritte des Geschäftsprozesses werden als Entrys innerhalb dieses Tasks modelliert. Damit ist es möglich, für jeden Schritt des Geschäftsprozesses die jeweils benötigten Systemressourcen und damit dessen Antwortzeit zu modellieren. Die Kapselung aller Entrys innerhalb eines Tasks bietet darüber hinaus die Möglichkeit, die Anzahl der zur Verfügung stehenden Work-Prozesse des SAP-Systems

durch die Multiplizität des Tasks abzubilden. Da jeder Task im LQN-Modell eine eigene Warteschlange besitzt, wird mit dieser Modellierung auch die Anforderung nach der Abbildung der Queue des Dispatchers umgesetzt.

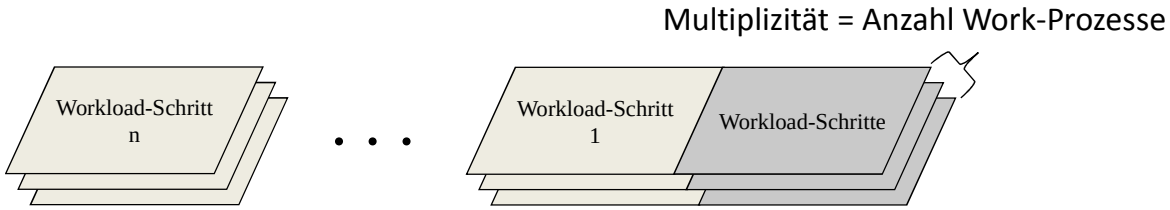


Abbildung 5-7: Modellierung der verschiedenen Workload-Schritte
Quelle: Eigene Darstellung

Abbildung 5-7 zeigt das entwickelte Sub-Modell für die Kapselung der Service-Zeiten der an einem Schritt des Geschäftsprozesses beteiligten Systemkomponenten. Um die bereits angesprochene maximale Antwortzeit zu modellieren, existiert im LQN Formalismus die Möglichkeit einer „max_service_time“. Damit ist es möglich die eine maximale Servicezeit für eine Komponente zu definieren. Dies hat jedoch nicht zur Folge, dass bei Überschreiten dieser Antwortzeit die Bearbeitung abgebrochen wird, wie es im realen System der Fall wäre. Wird diese Eigenschaft für einen Entry parametrisiert, so enthält das Ergebnis der Simulation, wie in Abschnitt 6.4.3.6 beschrieben, die Wahrscheinlichkeit dafür, dass die „max_service_time“ in diesem Entry überschritten wird.

5.2.4. Verbucher-Prozesse

Verbucher- und Verbucher2-Prozesse werden für Schreibvorgänge in die Datenbank verwendet. Mittels dieser separaten Implementierung wird eine Verkürzung der Antwortzeit des SAP-Systems erreicht. Hat ein Work-Prozess eine Anfrage bearbeitet und eine Transaktion abgeschlossen, so übergibt er die Aufgabe des Schreibens in die Datenbank und die dafür bereits erfolgten Sperren von Datenbankobjekten asynchron an einen Update-Prozess. Während dieser die Verbuchung vornimmt, sendet der Work-Prozess bereits die Antwort an den Benutzer zurück.

5.2.4.1. Anforderungen an das Sub-Modell für den Verbucher-Prozess

Um die Auswirkungen der Verbucher-Prozesse auf die Antwortzeit und die Nutzung der Systemressourcen zu modellieren, müssen folgenden Anforderungen umgesetzt werden:

- Asynchroner Aufruf durch die Work-Prozesse
- Kapselung der in einem Verbucher-Lauf benötigten Ressourcen und Antwortzeiten
- Variation der Anzahl von Verbucher-Prozessen
- Übernahme der Sperren

Die Modellierung der Übernahme von Sperren von einem Work-Prozess durch einen Verbucher-Prozess wird implizit durch die Modellierung der Enqueue-Vorgänge (5.2.5) selbst durchgeführt, sodass das folgende Kapitel nicht auf diese Anforderungen eingeht.

5.2.4.2. Umsetzung

Da Verbucher immer durch einen Work-Prozess aufgerufen werden und der LQN-Formalismus diese asynchronen Verbindungen unterstützt, erfolgt die Umsetzung durch die Verwendung einer „second phase“ (Woodside 2002). Da Verbucher2-Prozesse von einem Verbucher-Prozess aufgerufen werden und ebenfalls keinen direkten Einfluss auf die Antwortzeit besitzen, werden Verbucher- und Verbucher2-Prozesse künftig nicht mehr unterschieden und abstrakt als Verbucher-Prozesse bezeichnet und mit geringerer Priorität modelliert. Die Servicezeit der ersten Phase beträgt null, damit die Antwortzeit nicht durch die Servicezeit des Verbuchers beeinträchtigt wird. Die benötigte Servicezeit eines Verbucher-Laufs hängt von den benötigten Systemressourcen ab. Diese variieren, je nachdem für welchen Schritt des Geschäftsprozesses die Verbuchung durchgeführt werden muss. Aus diesem Grund wird die Umsetzung der Verbucher-Prozesse ähnlich zu den Lastschritten innerhalb eines Tasks vorgenommen. Über die Multiplizität dieses Tasks wird die Anzahl an verfügbaren Verbucher-Prozessen abgebildet. Analog zu den Lastschritten wird innerhalb dieses Tasks für jeden Lastschritt ein eigener Entry für einen Verbuchungsvorgang implementiert. Damit können die verschiedenen Ressourcenbedarfe unterschiedlicher Verbucher-Aufrufe modelliert werden. Abbildung 5-8 zeigt die Umsetzung des Submodells für Verbuchungsvorgänge.

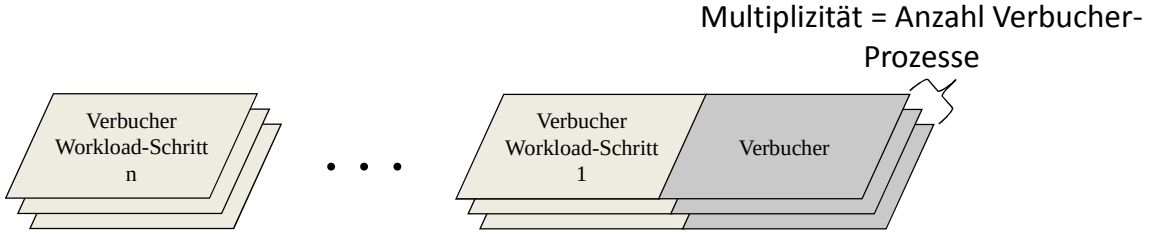


Abbildung 5-8: Modell der Verbucher-Prozesse
Quelle: Eigene Darstellung

5.2.5. Enqueue-Komponenten

Enqueue-Objekte regeln den wechselseitigen Zugriff auf geschützte Datenbankobjekte. Die Verwaltung dieser Sperrobjekte wird durch den Enqueue-Server durchgeführt, welcher Anfragen auf Sperrung eines Objekts sequentiell abarbeitet, und diese Sperren in einer Tabelle, welche im Hauptspeicher abgelegt ist, speichert. Aufgrund dieser sequentiellen Abarbeitung von Sperranfragen kann es dabei zu Wartezeiten kommen, welche sich auf die Antwortzeit der Anfrage auswirkt. Wird eine Sperranfrage durch den Enqueue-Server bearbeitet, so kann diese nur gesetzt werden, wenn diese Sperranfrage nicht mit einer existierenden Sperre kollidiert. Ist dies der Fall, so entsteht eine Wartezeit, bis die kollidierende Sperre entfernt wird. Wie im vorhergehenden Abschnitt dargestellt, können Work-Prozesse existierende Sperren an Verbucher-Prozesse übergeben.

5.2.5.1. Anforderungen an das Sub-Modell

Aus der Beschreibung der Verwendung von Enqueue-Objekten in Abschnitt 3.4 ergeben sich die folgenden Anforderungen an die Modellierung:

- Warteschlange vor dem Enqueue-Server

- Warteschlange vor dem Setzen der Sperre
- Modellierung wechselseiter Sperren durch mehrere Enqueue-Objekte
- Übergabe von Sperrobjekten durch Work-Prozesse auf Verbucher-Prozesse

5.2.5.2. Umsetzung durch das LQN Sub-Modell

Wie in Abbildung 5-9 dargestellt, wird der Enqueue-Server als Task mit der Multiplizität 1 modelliert. Damit wird gewährleistet, dass das Modell die Queueing-Effekte, durch die sequentielle Bearbeitung von Sperranfragen verursacht, berücksichtigt. Um die Bearbeitung der Sperranträge für die verschiedenen Sperrobjekte abzubilden, wird für jedes Sperrobjekt innerhalb des Enqueue-Server Tasks ein Entry modelliert. Jedes verwendete Sperrobjekt wird als selbständiger Task modelliert, welches von dem korrespondierenden Entry innerhalb des Enqueue-Server Tasks mittels einer synchronen Anfrage aufgerufen wird. Damit wird für jedes Sperrobjekt eine Warteschlange erzeugt, und damit die Wartezeit auf das Setzen einer Sperre modelliert. Wie in Abbildung 5-9 dargestellt, wird für jeden modellierten Datenbankbereich ein korrespondierendes Sperr-Objekt erstellt, um konkurrierende Zugriffe auf diesen zu vermeiden.

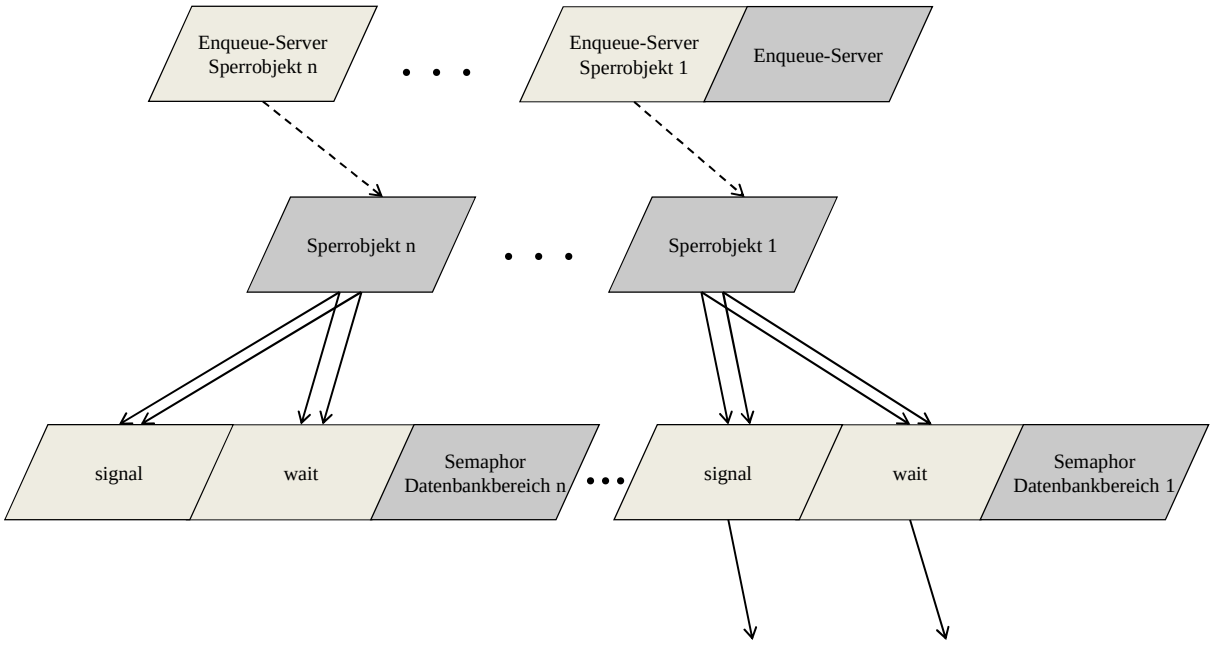


Abbildung 5-9: Modellierung Enqueue-Vorgang
Quelle: Eigene Darstellung

Um die verschiedenen Typen von Sperren (3.4.2) zu modellieren, müssen logische Komponenten eingefügt werden, da der LQN-Formalismus derzeit keine Objekte mit entsprechenden Funktionalitäten bereitstellt (Franks 2011). Die Modellierung des Sperrkonzepts wird durch Verwendung von „Activities“ und der bereits beschriebenen „Semaphore“-Task umgesetzt. Die Implementierung eines Sperrobjekts zeigt Abbildung 5-10. Dieses besteht grundsätzlich aus jeweils einem Task für Lese- und Schreibsperre mit den Multiplizitäten ∞ und 1. Dies begründet sich daraus, dass Lesesperren sich nicht gegenseitig beeinflussen, während dagegen auf einen Datenbankbereich nur eine Schreibsperre gesetzt

sein kann. Ist eine lesende SQL-Anfrage durch ein Sperrobjekt zu kapseln, so wird zuerst überprüft, ob auf dem betreffenden Datenbankbereich bereits eine Schreibsperre aktiv ist. Dies geschieht durch Aufruf des Entrys „Sperre gesetzt“. Da der Task für die Schreibsperre eine Multiplizität von 1 hat, kann bei aktiver Schreibsperre diese Anfrage nicht sofort beantwortet werden, und der lesende Zugriff muss warten, bis der Schreibzugriff beendet ist. Sobald dies geschehen ist, wird die Anfrage durch den „Sperre gesetzt“-Entry beantwortet, und die Anfrage an die Datenbank gestellt. Parallel wurde bereits das Setzten der Sperre durch einen Aufruf des „wait“-Entrys im Semaphor des Datenbankbereichs angefordert. Ist dieser nun durch frei, so wird die Sperre gesetzt, bis die Datenbankanfrage durchgeführt und die Antwort an den Client geschickt ist. Die Umsetzung von kumulativen Sperren wurde bei dieser Modellierung nicht gesondert berücksichtigt, da diese Sperren in der Analysephase identifiziert und zu einer einzelnen Sperre aggregiert werden können.

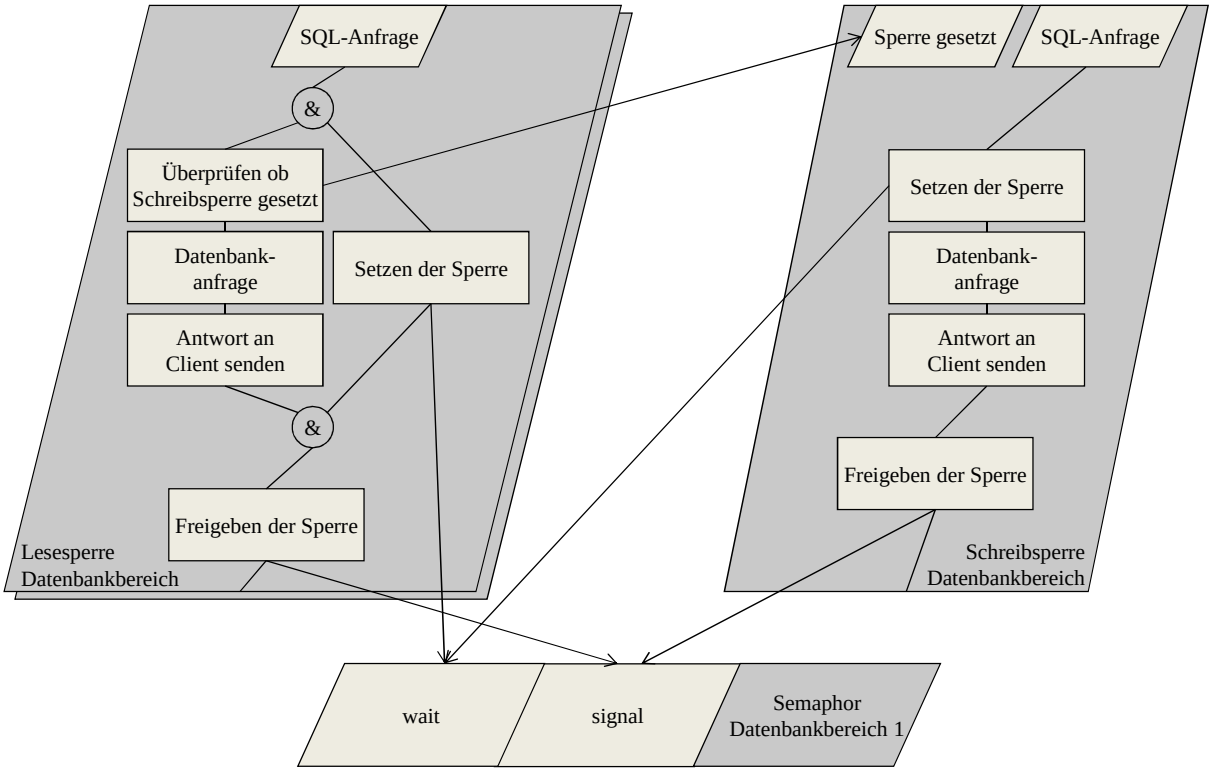


Abbildung 5-10: Modellierung von Lese- und Schreibsperren durch Aktivitäten und Semaphore
Quelle: Eigene Darstellung

5.2.6. Pufferverwendung

Die in dieser Arbeit betrachteten Tabellenpuffer gliedern sich, wie in Abschnitt 3.5.2 beschrieben, in den „Single Record Buffer“ und den „Generic Key Buffer“. Beide Puffer werden für die Zwischenspeicherung von Datenbankinhalten im Shared Memory eines SAP-Applikationsservers verwendet. Die in den Puffern zur Verfügung stehenden Kapazitäten werden jeweils definiert durch die zur Verfügung stehende Größe in Kilobyte und der maximal möglichen Anzahl von Einträgen. Ist der zur Verfügung stehende Puffer zu klein konfiguriert, werden die am wenigsten häufig verwendeten Inhalte aus dem Puffer verdrängt und die benötigten Inhalte nachgeladen. Dies führt zu Performance-Einschränkungen, da zu diesem Zeitpunkt Daten von der Datenbank geladen werden müssen. Weitere Performance-Einschränkungen im Zusammenhang mit Puffern ergeben sich, wenn gepufferte Daten durch

SAP-Programme verändert werden. Diese werden dann im Puffer invalidiert und müssen bei der nächsten Verwendung wieder aus der Datenbank geholt werden. Um diese Einschränkungen zu vermeiden, müssen die Puffer ausreichend dimensioniert sein.

5.2.6.1. Anforderungen an die Modellierung

Aus der Beschreibung der Puffermechanismen des vorherigen Abschnitts und deren möglicher Auswirkungen auf die System-Performance ergeben sich die folgenden Anforderungen an die Modellierung:

- Abbildung von Verdrängungen anhand definierter Puffergrößen
- Abbildung von Invalidierungen von Pufferinhalten

Damit ist es möglich die konfigurierte Größe der Puffer bzgl. der Auswirkungen auf die Performance mit Hilfe der Simulation zu evaluieren.

5.2.6.2. Umsetzung des Modells

Die Puffer des SAP-WebAS-ABAP werden als Task modelliert. Dieser Task enthält einen Entry für jeden gepufferten Datenbankbereich. Da das Modell die Performance eines Systems und nicht zwangsweise den genauen internen Ablauf abbilden soll, werden nicht direkt die Verdrängungen modelliert, sondern deren Auswirkung auf die System-Performance. Diese besteht aus zusätzlichen Datenbankzugriffen, welche natürlich mehr Zeit in Anspruch nehmen als Lesevorgänge aus dem Hauptspeicher. Demzufolge werden Verdrängungen durch Weiterleitungen von Pufferanfragen an die Datenbank mittels synchroner Aufrufe modelliert.

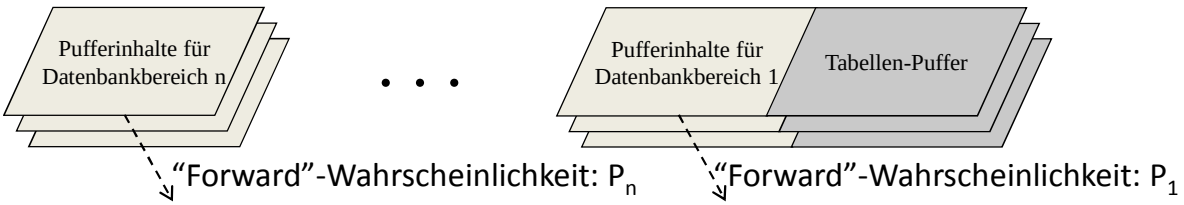


Abbildung 5-11: Modellierung der Pufferaktivitäten
Quelle: Eigene Darstellung

Um den Verdrängungsmechanismus zu berücksichtigen, werden diese Weiterleitungen, wie in Abbildung 5-11 dargestellt, in Abhängigkeit einer Wahrscheinlichkeit durchgeführt. Diese „forward probability" ergibt sich aus der gewichteten, relativen Häufigkeit der jeweiligen Pufferanfragen, da diejenigen Elemente mit den geringsten Zugriffsraten zuerst verdrängt werden. Invalidierungen wirken in der gleichen Art und Weise auf die System-Performance. Deshalb muss für jedes Objekt im Puffer die Wahrscheinlichkeit von Invalidierungen in „forward probability" integriert werden.

5.2.7. Datenbank

Der Fokus der vorliegenden Arbeit liegt auf der Modellierung der internen Abläufe des SAP-WebAS-ABAP. Zudem werden mehrere Datenbanksysteme durch den SAP-WebAS-ABAP unterstützt, welche nicht alle im Rahmen dieser Arbeit analysiert und modelliert werden können. Aus diesem Grund wird die Datenbankkomponente als Black-Box modelliert.

Trotzdem ist es nötig, Datenbankzugriffe in das Modell zu integrieren, da diese eng mit den verschiedenen modellierten Eigenschaften des SAP-WebAS-ABAP zusammenarbeiten.

5.2.7.1. Anforderungen an die Modellierung

Um das Modell für die Performance-Vorhersage verwenden zu können, werden konkrete Performance-Werte oder Abschätzungen der entsprechenden Servicezeiten der Datenbank benötigt. Um diese integrieren zu können und damit die Modellierung der bereits dargestellten Systemkomponenten zu unterstützten, müssen folgende Anforderungen in der Modellierung der Datenbankkomponente berücksichtigt werden:

- feingranulare Abbildung aller verwendeten Datenbankbereiche und der entsprechenden Zugriffe auf diese
- feingranulare Möglichkeit der Parametrisierung einzelner Datenbankzugriffe

Da die Datenbank lediglich als Black-Box-Komponente modelliert wird, ist es nötig Strukturen abzubilden, die eine genaue Parametrisierung und ebenso eine genaue Analyse der Simulationsergebnisse erlauben.

5.2.7.2. Umsetzung

Um die entsprechenden Anforderungen in der Modellierung der Datenbankkomponente zu berücksichtigen, wird jeder durch den abgebildeten Workload verwendete Datenbankbereich als Entry innerhalb eines Tasks modelliert. Damit kann eine feingranulare Parametrisierung anhand von Servicezeiten und über die Datenbank hinweg eine Gesamtkapazität von Anfragen mit Hilfe der Multiplizität erfolgen. Abbildung 5-12 zeigt die Umsetzung mittels eines LQN-Modells:

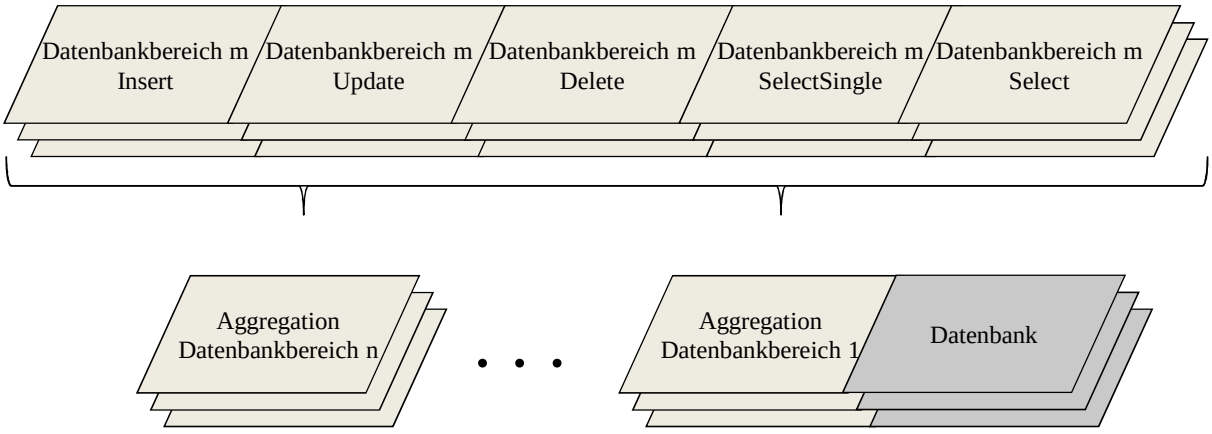


Abbildung 5-12: Modellierung Datenbank
Quelle: Eigene Darstellung

5.2.8. Vollständiges Modell

Um das vollständige Modell zu erhalten, müssen die einzelnen Submodelle zusammengefügt werden. Dies ist in Abbildung 5-13 dargestellt. Um die Vielzahl von Verbindungen zu minimieren, wurden, beginnend bei den Clients, für jede Komponente die ausgehenden Verbindungen abgebildet. Die jeweils farbig hinterlegte Verbindung zeigt den Pfad einer

exemplarischen Benutzeranfrage durch das Modell. Zur Vereinfachung der Komplexität wurde bei der Darstellung des gesamten Modells auf die Darstellung der CPU-Ressourcen verzichtet. Diese werden jedoch, wie in den Detailabbildungen der Submodelle dargestellt, verwendet.

5.2.8.1. Anpassung an zu untersuchende Anwendungen

Wie bereits erwähnt, bietet diese Modellierung die Möglichkeit, Nutzergruppen zu modellieren, welche sich durch Thinktime, Anzahl von Benutzern und verwendeten Schritten des Geschäftsprozesses unterscheiden. Die jeweilige Ausprägung dieser Gruppen muss das tatsächliche Verwendungsmuster wiederspiegeln. Ebenso müssen die Schritte aller verwendeten Geschäftsprozesse in Abhängigkeit zu deren Verwendung von Verbucher-Prozessen in das Modell integriert werden. Die Anzahl und Art der verwendeten Enqueues, Datenbankbereiche und Puffer müssen analog zu deren Beschreibungen in den Absatz 5.2.5 bis 5.2.7 in das Modell integriert werden. Sind die entsprechenden Schritte des Geschäftsprozesses im Modell abgebildet, so kann eine Zuordnung von Benutzergruppen auf die jeweils verwendeten Schritte des Geschäftsprozesses durch das Einfügen der entsprechenden Verbindungen zwischen den Benutzergruppen und den Workload-Schritten durchgeführt werden.

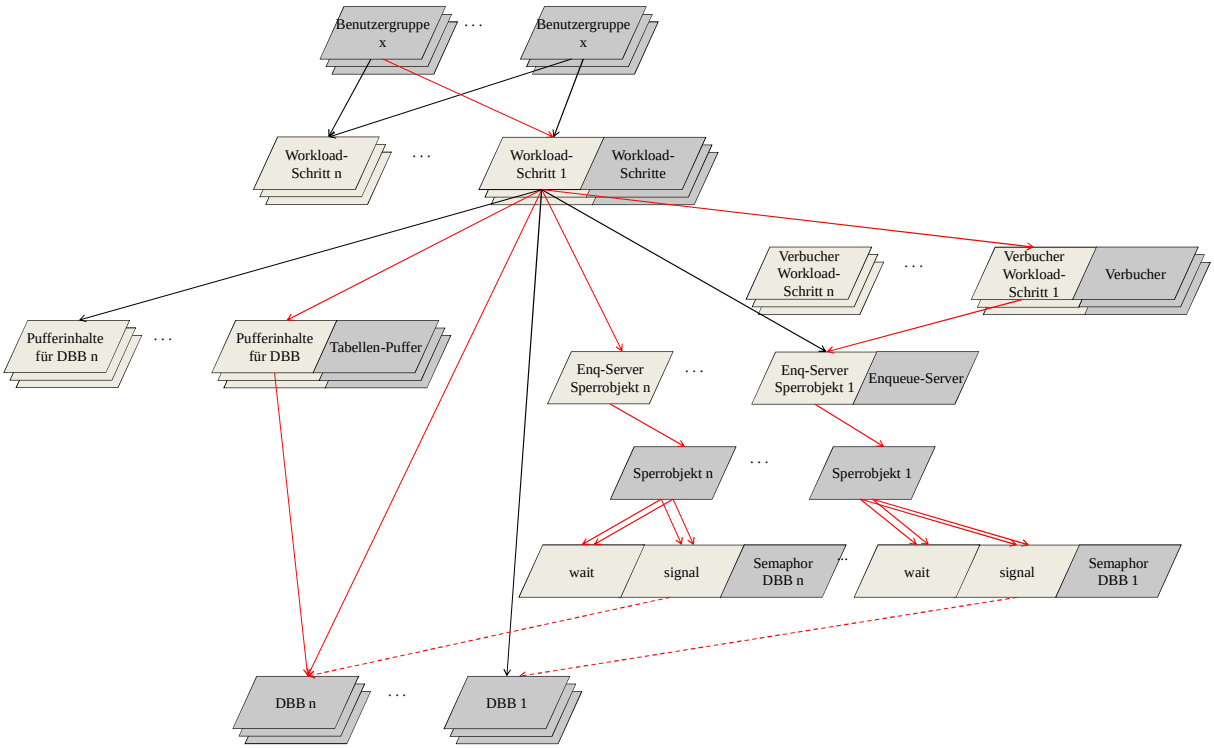


Abbildung 5-13: Schematische LQN-Modell der Antwortzeit eines SAP-ERP-Systems
Quelle: Eigene Darstellung

Die Architektur vieler Systeme baut auf dem Konzept verschiedener Schichten auf. Komponenten aus höher liegenden Schichten senden Anfragen an Komponenten auf niedrigeren Ebenen. Aus diesem Grund bietet der Ansatz „The Method Of Layers“ (Rolia/Servcik 1995) viele Vorteile gegenüber „flachen“ Warteschlangensystemen. In eben dieser Schichtenarchitektur gründet die Ähnlichkeit vieler LQN-Modelle. Dies zeigen beispielsweise die Modelle von Jenq/Kohler/Towsley (1988) und Sheikh/Woodside (1997),

die ähnlich zu dem in der vorliegenden Arbeit entwickelten Modell die Benutzer auf der obersten Ebene und die Datenbank auf der untersten Ebene darstellen.

Die Ausprägung der mittleren Ebenen variiert dabei je nach fokussierter Funktionalität des abzubildenden Systems. Die Modellierung eines Sperrkonzepts für kumulative Lese- und exklusive Schreibsperren wurde in der Literatur noch nicht behandelt.

5.2.8.2. Statische Eigenschaften des Modells

Das in Abbildung 5-13 dargestellte Modell kann individuell an zu untersuchende Geschäftsprozesse angepasst werden und damit eine große Anzahl von verschiedenen Anwendungen abbilden. Dabei ist jedoch die Grundannahme eines interaktiven Workloads fest in das Modell integriert, sodass es nicht für andere Typen von Workload verwendet werden kann.

5.3. Parametrisierung des Modells

In diesem Kapitel wird vorgestellt, welche Informationen für die Parametrisierung der jeweiligen Modellkomponenten benötigt werden, und wie sie mit Hilfe der in Kapitel 4 vorgestellten Messwerkzeuge ermittelt werden können. Dazu wird zuerst dargestellt, welche Art von Informationen als Parameter dem Modell für die Simulation übergeben werden müssen, um dann in den folgenden Kapiteln die Parametrisierung der einzelnen Submodelle zu beschreiben.

5.3.1. Parameter der einzelnen Modellkomponenten

Die Komponenten eines LQN-Modells können mit einer Vielzahl an Parametern ausgestattet werden. Im diesem Abschnitt werden diejenigen Parameter vorgestellt, die im Rahmen dieser Arbeit verwendet werden.

5.3.1.1. Task

Um die Workload-Intensität in einem LQN-Modell abzubilden, werden sogenannte „reference Tasks“ verwendet (Woodside 2002), welche keine Anfragen empfangen können, sondern die Last durch Anfragen auf die Komponenten unterer Ebenen erzeugen.

Wie bereits erwähnt, beinhalten die Task-Komponenten ein Service-Center mit einer vorgeschalteten Warteschlange. Der LQN Formalismus bzw. das in dieser Arbeit eingesetzte Simulationswerkzeug unterstützt die folgenden Scheduling-Mechanismen:

- FIFO
- PPR
- HOL

Neben dem standardmäßig verwendeten „First in first out“- werden auch noch die prioritätsbasierten Mechanismen „priority, preemptive resume“ (PPR) und „Head-of-line priority“ (HOL) angeboten. Dabei werden die Tasks nach deren Priorität aufsteigend sortiert aus der Warteschlange entnommen. Im Gegensatz zu HOL wird bei PPR-basiertem

Scheduling ein aktiver niedriger priorisierter Task zu Gunsten eines höher priorisierten Tasks beendet und später ausgeführt. Bei HOL wird ein aktiver niedrig priorisierter Task bei Ankunft eines höher priorisierten Tasks fertig prozessiert, bevor der Task mit höherer Priorität aufgerufen wird.

Ein Task kann wie bereits im Abschnitt 2.4.1.1 mehrere Service-Center besitzen. Dies wird im LQN-Formalismus durch die Spezifikation der Multiplizität umgesetzt. Die beliebige Anzahl von Service-Centern wird bei Angabe einer Multiplizität jedoch von einer Warteschlange bedient. Im Gegensatz dazu wird bei der Konfiguration von „Replicated servers“ für jeden neuen Bediener auch eine eigene Warteschlange instanziiert.

5.3.1.2. Entry

Entrys stellen unterschiedliche Services innerhalb eines Tasks dar. Sie können Anfragen stellen und bearbeiten Wenn sie als Server fungieren, unterstützen der LQN-Formalismus und das verwendete Simulationstool „lqsim“ drei Phasen der Bearbeitung. Dabei wird am Ende der ersten Phase die Antwort an den Client zurückgeschickt. Die nächsten beiden Phasen dienen der Nachbearbeitung der Anfrage. Die dafür benötigte Zeit wird nicht mehr der Antwortzeit zugerechnet. Da jedoch System-Ressourcen benötigt werden, können diese Phasen nicht vernachlässigt werden. In der Client-Funktion benötigen Entrys Ressourcen des Prozessors. Diese „service time“ wird durch die Angabe eines Durchschnitts und der Verteilung in Form des „squared variance coefficient“, welcher bereits in Abschnitt 2.2.4.4 beschrieben wurde, definiert. Dabei wird die gesamte Servicezeit in sogenannte „slices“ unterteilt (Woodside 2002). Diese Abschnitte modellieren die Zeit zwischen den Serviceaufrufen des Entrys. Sei Y die Anzahl der Anforderungen an Server-Komponenten durch einen Entry, so wird dessen Service-Zeit in (Y+1)-Zeitabschnitte unterteilt. Für die Dauer dieser Zeitabschnitte wird eine Exponentialverteilung angenommen, wobei der angegebene Variationskoeffizient berücksichtigt wird (Woodside 2002). Abhängig von diesem Koeffizienten interpretieren Simulatoren die Umsetzung unterschiedlich. Der in der vorliegenden Arbeit verwendete „lqsim“ geht abhängig vom Variationskoeffizienten (cv^2) wie folgt vor:

- $cv^2 = 0$: deterministische Service-Zeit
- $0 < cv^2 < 1$: Gammaverteilung
- $cv^2 = 1$: Exponentialverteilung

Für jeden Entry werden Verbindungen zu unterliegenden Servern definiert. Diese können die in Abschnitt 2.4.2 beschrieben folgenden Typen haben:

- Synchron
- Asynchron
- Weitergeleitet

Für jede dieser Verbindungen kann innerhalb der Modell-Parametrisierung eine Anzahl von „Rendezvous“ (Woodside 2002), also Anfragen an die jeweiligen Server-Komponenten,

angegeben werden. Diese Anzahl wird je nach Parametrisierung der „Call order“ unterschiedlich interpretiert. Die folgenden Varianten stehen für die Konfiguration der Aufrufordnung über zur Verfügung:

- deterministisch
- stochastisch

Diese Parameter müssen für jede zu benutzende Phase eines Entrys über den Parameter „ph_type_flag“ gesetzt werden. Dabei wird in einer deterministischen Phase exakt die konfigurierte Anzahl an Anfragen an unterliegende Server gestellt. Eine stochastische Phase stellt eine zufällige Anzahl an Anfragen an die unterliegenden Server, wobei die konfigurierte Anzahl von Anfragen den Durchschnitt darstellt.

5.3.1.3. Processors

Da die Analyse von Systemen mit „Shared Ressources“ ein häufiger Anwendungsfall von Warteschlangenmodellen ist, gibt es für die Prozessoren eine eigene Komponente. Diese ist ein reiner Server und bietet neben den bereits in Abschnitt 5.3.1.1 dargestellten Scheduling-Mechanismen noch die folgenden Varianten an:

- PS
- RAND

PS bedeutet, dass alle Anfragen vom Prozessor gleichzeitig bearbeitet werden. Laut (Woodside 2002) wird dies bei Simulation mit „lqsim“ (Franks et al. 2010) durch ein Round-Robin-Verfahren (siehe Abschnitt 2.4.1.1) implementiert. Eine zufällige Auswahl von Jobs aus der Queue wird durch das „Random scheduling“ (RAND) umgesetzt.

5.3.1.4. Zeiteinheiten

Da sowohl Service-Zeiten als auch Thinktimes an verschiedensten Stellen in die Parametrisierung des Modells einfließen, stellt sich die Frage nach deren Einheit. Da der Simulator in „Units of time“ rechnet, kann diese Einheit selbst festgelegt werden. Die gewählte Einheit muss jedoch in der Parametrisierung des gesamten Modells einheitlich sein.

5.3.2. Client

Wie bereits erwähnt bietet der LQN Formalismus die Möglichkeit, die Lastintensität über die Anzahl von Benutzern und deren Thinktime, also der Zeitspanne zwischen zwei Anfragen, zu modellieren. Dazu wird der Task-Typ „reference task“ verwendet.

5.3.2.1. Thinktime

Um die bereits beschriebenen Gruppen von Benutzern in die Parametrisierung des Modells zu übernehmen, müssen die durchschnittlichen Bedenkzeiten vorher ermittelt werden. Diese Zeiten hängen sowohl von Art und Umfang der benötigten Eingaben als auch von dem jeweiligen Benutzer selbst ab. Die ermittelte Zeit muss dann in die gewählte Einheit (siehe 5.3.1.4) transformiert und dem entsprechenden Entry als Thinktime übergeben werden.

5.3.2.2. Anzahl der Benutzer

Die Gesamtanzahl an Benutzern, welche im Modell abgebildet sind, ergibt sich aus der Summe der Benutzer der einzelnen Gruppen. Jede Gruppe entspricht im Modell einem Task. Die Anzahl der Gruppenmitglieder wird über die Multiplizität des als „reference tasks“ modellierten Benutzer-Tasks abgebildet.

5.3.2.3. Verbindungen

Für jeden Workload-Schritt, welcher durch einen Benutzer der jeweiligen Gruppe ausgeführt wird, muss eine Verbindung zu dem entsprechenden Element des Modells parametrisiert werden. Diese Verbindung hat die Kardinalität 1 und ist vom Typ „synchron“.

5.3.2.4. CPU Nutzung

Das Modell dient der Analyse der Antwortzeit des SAP-ERP-Systems unter einem gewissen Workload. Der CPU-Bedarf der Clients wird nicht betrachtet und daher mit einer unbegrenzten Multiplizität modelliert.

5.3.3. Enqueue-Komponenten

Bei den Enqueue-Komponenten handelt es sich um die Modellierung der wechselseitigen Zugriffkontrolle auf geschützte System-Ressourcen. Aus der Umsetzung der Modellierung (Abschnitt 5.2.5.2) ergeben sich bereits die Multiplizitäten der Enqueue-Server-Komponenten, der Sperrobjekte, der Semaphoren sowie der spezifischen Verbindung zwischen den Objekten Die Workload abhängige Parametrisierung besteht nun aus den Service-Zeiten der einzelnen Bestandteile des Enqueue-Submodells sowie aus der Anzahl der von einem Sperrobjekt gekapselten SQL-Zugriffe.

5.3.3.1. Service-Zeiten

Die Servicezeiten der verschiedenen Bestandteile des Enqueue-Submodells sind, obwohl mit einem intrusiven Verfahren (4.2.6) gemessen, so gering, dass deren Einfluss auf die Antwortzeit zu vernachlässigen ist und deshalb bei der Parametrisierung nicht berücksichtigt werden muss. Analog zu den anderen Komponenten des Enqueue-Submodells ist deren Hauptaufgabe die Modellierung von Sperren und der dadurch verursachten Wartezeiten.

5.3.3.2. Gekapselte SQL-Abfragen

Da ein Sperrobjekt eine bestimmte Anzahl von SQL-Abfragen kapselt, müssen für die Parametrisierung der Zugriffe auf Datenbankobjekte diese SQL-Abfragen aus der gesamten Menge an Anfragen extrahiert und analysiert werden. Die Identifikation der SQL-Abfragen geschieht mittels der Kriterien „Zielobjekt“ und „Timestamp“ der Sperrobjekte mit den SQL-Abfragen innerhalb des SAP-Performance-Traces. Der Datenbankbereich einer zu identifizierenden Abfrage stimmt mit dem Zielobjekt des Sperrobjekts überein. Ist dies der Fall, so muss auch noch der Zeitstempel der SQL-Abfrage innerhalb des Intervalls sein, welches durch den Zeitpunkt des Setzens der Sperre und dem Zeitpunkt des Lösens der Sperre gebildet wird. Aus allen identifizierten SQL-Abfragen pro Sperrobjekt und Sperrmodus muss nun das arithmetische Mittel gebildet werden, welches für die Parametrisierung der Anzahl der Verbindungen eines „Semaphor“-Entrys zur Datenbank verwendet wird.

5.3.4. Lastschritte und Verbucher-Aktionen

Die Lastschritte bilden die Container für die einzelnen Systemressourcen, welche während deren Ausführung verwendet werden. Bei der Parametrisierung dieser Komponente ist genau diese Kapselung zu beachten, und die Messwerte sind für die Parametrisierung dementsprechend zu bewerten. Die STAD-Records beinhalten eine Menge von Werten, welche in den folgenden Kapiteln für die Parametrisierung der Workload-Schritte bewertet und verwendet werden. Danach wird dargelegt, wie die Verbindungen von den Workload-Schritten zu den verschiedenen Komponenten und deren Anzahl aus den verfügbaren Informationen ermittelt werden können, sodass am Ende dieses Kapitels die Workload-Schritte sowie alle Aufrufe durch deren Entrys parametrisiert werden können.

5.3.4.1. CPU-Nutzung

Wie bereits angesprochen, kapseln diese Komponenten die Antwortzeiten und die Ressourcen, die die jeweiligen Schritte des Geschäftsprozesses oder der jeweilige Verbucher-Prozess benötigen. Die für die Parametrisierung der CPU-Zeit zu verwendenden Daten müssen mittels eines nicht-intrusiven Messinstruments erhoben worden sein und den einzelnen Schritten des Geschäftsprozesses bzw. des Verbucher-Prozesses zugeordnet werden können. Die Informationen der STAD-Records (4.2.1) erfüllen diese Voraussetzungen. Jedoch kapselt ein STAD-Record die CPU-Zeiten aller Vorgänge innerhalb dieses Work-Prozesses. Es erfolgt keine Aufteilung auf die einzelnen darunter liegenden Komponenten. Aus diesem Grund werden die durch Messung erhobenen CPU Zeiten nur in Komponenten definiert, welche in den STAD-Records mit gleicher Granularität verzeichnet sind. Dies ist bei den Workload-Schritten der Fall. Die für die Parametrisierung des Modells erhoben CPU-Werte müssen zu einem Mittelwert zusammengefasst werden und können somit zusammen mit dem ermittelten Quadrierten Varianzkoeffizienten für die Parametrisierung der Service-Zeit der einzelnen Workload-Schritte verwendet werden.

5.3.4.2. Multiplizität

Wie in Abschnitt 5.2.3 dargestellt, wird die Anzahl der in einem System verfügbaren Work-Prozesse über die Multiplizität des Tasks „Workload-Schritte“ abgebildet. Hierbei ist zu beachten, dass nur die Anzahl der wirklich für die Anfragen von Benutzern zur Verfügung stehenden Work-Prozesse verwendet wird. Batch-Prozesse dürfen hier nicht berücksichtigt werden.

Für die Parametrisierung der Verbucher-Prozesse ist die Anzahl der im System konfigurierten Verbucher- und Verbucher2-Prozesse zu verwenden, da im Modell zwischen diesen beiden Typen von Work-Prozessen nicht unterschieden wird (5.2.4).

5.3.4.3. Pufferaufrufe

Die in den STAD-Records vermerkte Anzahl von Pufferaufrufen beinhaltet leider keine Detailinformationen über das angefragte Objekt und die Häufigkeit der Anfragen. Diese Informationen werden durch den Performance-Trace geliefert. Dabei werden die in Abschnitt 4.2.5.1 dargestellten Sequenzen von Pufferzugriffen für die Modellierung und die Parametrisierung verwendet, um die Anzahl an Verbindungen im Modell möglichst gering zu halten. Diese Sequenzen bestehen aus einer Menge von Zugriffen auf ein gepuffertes Objekt, welche von Anweisungen zum Öffnen und Schließen der Ergebnissemenge umgeben sind.

Durch die Analyse des Puffer-Traces können diese Sequenzen, wie in Abschnitt 4.2.5 beschrieben, identifiziert werden. Damit sind die für die Modellierung der Verbindungen eines jeden Entrys im Task „Workload-Schritte“ verfügbar, da die Anzahl der Verbindungen mit der Anzahl der Sequenzen parametrisiert wird. Die Bezeichnung des gepufferten Objekts, auf welches sich die Sequenz bezieht, wird ebenfalls durch den SAP-Performance-Trace bereitgestellt.

5.3.4.4. Enqueue-Aufrufe

Analog zu den Pufferaufrufen, können die Aufrufe der Enqueue-Bausteine parametrisiert werden. Auch die dafür nötigen Informationen können aus dem Enqueue-Teil (4.2.6) des SAP-Performance-Trace entnommen werden. Die Aufteilung auf Sequenzen wird dabei anhand der verschiedenen, verwendeten Sperrobjekte vorgenommen. Auch bei den Enqueue-Vorgängen wird die Anzahl an Verbindungen mit der Anzahl der durch einen Geschäftsprozess-Schritt aufgerufenen Sequenzen modelliert. Der Name des Sperrobjekts, welcher die Zielkomponente der Verbindung darstellt, ist in der Spalte „Objektname“ verzeichnet.

5.3.4.5. Datenbankaufrufe

Für die Parametrisierung der direkten Datenbankaufrufe werden wiederum die Inhalte des SAP-Performance-Trace verwendet. Da eine logische SQL-Anfrage aus der Öffnung der Ergebnismenge, dem Auslesen daraus und deren Schließung besteht, werden die einzelnen Einträge werden wie in Abschnitt 4.2.7.4 beschrieben in Sequenzen zusammengefasst. Jedoch ist die somit identifizierte Anzahl von SQL-Abfragen nicht zwangsweise korrekt. Wie bereits erwähnt, erfasst der SAP-Performance-Trace auch die SQL-Anfragen, welche für das Füllen der Puffer verwendet wurden. Wie in Abschnitt 4.2.8 müssen diese SQL-Anfragen identifiziert werden, da sie nicht für die Parametrisierung der direkten Datenbankaufrufe durch einen Geschäftsprozess-Schritt verwendet werden dürfen. Von den verbleibenden SQL-Sequenzen müssen nun die SQL-Abfragen, welche in Abschnitt 5.3.3.2 identifiziert wurden, abgezogen werden, sodass die nun verbleibenden Anfragen für die Parametrisierung der Anzahl und des Zielobjekts einer jeden Verbindung verwendet werden können.

5.3.4.6. Aufrufe von Verbucher-Prozessen

Dieser Abschnitt gilt nur für die Parametrisierung von Verbindungen zwischen den Modellkomponenten von Geschäftsprozess-Schritt zu Verbucher-Aktion. Da Verbucher-Prozesse für asynchrone Verarbeitung verwendet werden und damit keinen direkten Einfluss auf die modellierte Antwortzeit besitzen, wurde die Umsetzung in dem Performance-Modell der vorliegenden Arbeit mittels des Konzepts der „second phase“ realisiert. Damit sind für die Parametrisierung lediglich die Anzahl und die Zielkomponenten für die Modellierung der Verbindungen nötig.

Program	T	Scr.	Wp	User	Response time (ms)
*	*			LOAD_0_0	0
SAPMHTTP /sap/bc/soap/rfc	H		7	LOAD_0_0	30.162
RSM13000	U	3000	95	LOAD_0_0	9.873
RSM13000	2	3000	99	LOAD_0_0	29

Abbildung 5-14: Übersicht der STAD-Records in der Transaktion STAD (SAP 2008a)
Quelle: Eigene Darstellung

Da Verbucher-Vorgänge durch die extra zu konfigurierenden Verbucher-Prozesse durchgeführt werden, kann die Anzahl der Aufrufe von Verbucher-Prozessen aus der Vermessung eines Workload-Schrittes anhand der Einträge, siehe Abbildung 5-14, in den STAD-Records vorgenommen werden. In der Spalte „T“ sind dabei die verschiedenen Kürzel der Prozess-Typen aufgelistet:

- H: http-Request. Wird durch einen Work-Prozess verarbeitet
- U: update-Request. Wird durch einen Verbucher-Prozess abgearbeitet
- 2: update2-Request. Verarbeitung durch einen Verbucher2-Prozess.

Mittels dieser Zuordnung kann für jeden Workload-Schritt ermittelt werden, ob und wie viele Verbucher(2)-Anfragen gestellt werden. Wie in Abschnitt 5.2.4 beschrieben, wird im Modell nicht zwischen Verbucher- und Verbucher2-Prozessen unterschieden. Damit ergibt sich die Anzahl der Verbindungen eines Workload-Schrittes zu dem entsprechenden Verbucher-Objekt durch die Summe aus Verbucher- und Verbucher2-Anfragen.

5.3.5. Pufferverwendung

Wie bereits bei der Modellierung des Puffersubmodells (5.2.6.2) dargestellt wurde, wird die Zeit für die Zugriffe auf gepufferte Elemente in den STAD-Records aufgrund der geringen Ausprägung nicht berücksichtigt. Analog dazu wird darauf verzichtet, die Auswirkungen der Pufferzugriffe auf die Antwortzeit zu modellieren. Stattdessen rückt die Abbildung der Effekte von Pufferzugriffen in den Fokus, die ein Nachladen von Inhalten aus der Datenbank zur Folge haben.

5.3.5.1. Swaps und Invalidierungen

Wie bereits erwähnt, haben die eigentlichen Verdrängungen keinen Einfluss auf die Performance, da es sich um ein Löschen von Daten im Hauptspeicher handelt. Die negativen Auswirkungen entstehen, wenn die verdrängten Objekte bei der nächsten, sie betreffenden, Anfrage wieder von der Datenbank geladen werden müssen. Wie in 5.2.6.2 dargestellt, wird dies durch eine Weiterleitungswahrscheinlichkeit von Pufferanfragen an die Datenbank umgesetzt. Um diese Wahrscheinlichkeit zu berechnen, sind folgende Informationen für den zu modellierenden Workload nötig:

- Anzahl der gepufferten Datenbankbereiche
- Anzahl der Zugriffe auf die gepufferten Datenbankbereiche

- Größe der Datenbankbereiche im Puffer
- Konfigurierte Größe der Puffer

Die Anzahl der gepufferten Bereiche sowie die jeweilige Größe und Pufferungs-Art können über den in Absatz 4.2.3 beschriebenen Baustein „SAPWL_TABSTAT_SINCE_STARTUP“ ermittelt werden.

Die Anzahl der Zugriffe auf den jeweiligen gepufferten Datenbankbereich werden über den SAP-Performance-Trace und dessen in Abschnitt 4.2.5 dargestellte Analyse identifiziert.

Die Größe des zur Verfügung stehenden Puffers kann über den in Abschnitt 4.2.2.1 dargestellten Baustein „SAPTUNE_GET_SUMMARY_STATISTICS“ in der Spalte „Alloc. [KB]“ für den jeweiligen Puffer ausgelesen werden.

Mit diesen Informationen kann nun für die beiden Puffer „single record“ und „generic key“ je eine Liste mit den zugehörigen Datenbankbereichen erstellt und nach deren Zugriffshäufigkeit absteigend sortiert werden. Zusammen mit der Information über die Größe der Puffer können nun diejenigen Datenbankbereiche identifiziert werden, die nicht mehr im Puffer Platz finden und aufgrund der geringeren Zugriffshäufigkeit eine höhere Verdrängungswahrscheinlichkeit besitzen. Damit kann nun über die Anzahlen N, N_{ib} und N_{ob} eine einfache Wahrscheinlichkeit $P_{DBA}(L)$ für die Verdrängung bzw. das Nachladen des entsprechenden Datenbankbereichs errechnet werden:

$$P_{a,b,c,d,e,f}(L) = 1 - \frac{N_{ib}}{N} \text{ bzw. } P_{g,h,i,j} = 1 - \frac{N_{ob}}{N}$$

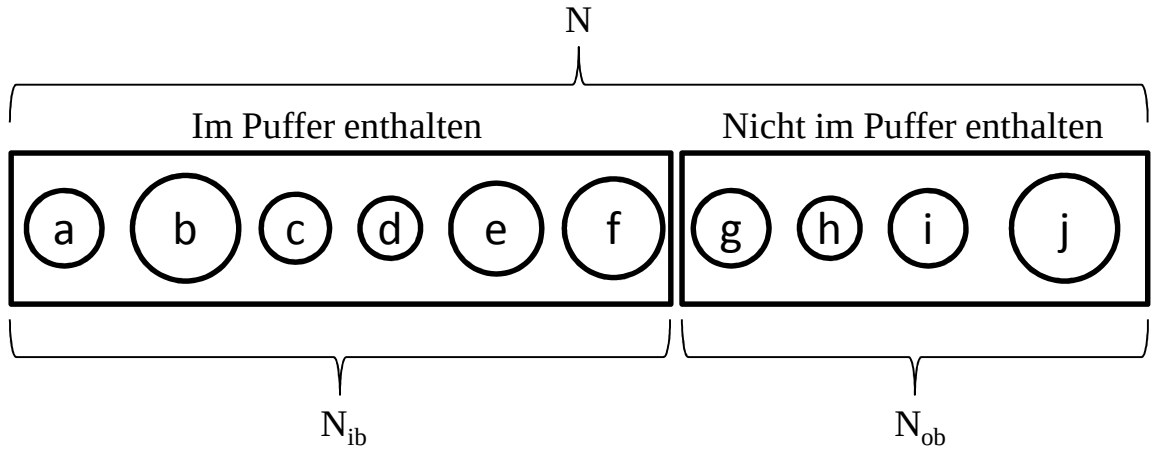


Abbildung 5-15: Beispielhafte Liste der Datenbankbereiche sortiert nach Aufrufhäufigkeit
Quelle: Eigene Darstellung

Um den in Absatz 3.5.2 beschriebenen Verdrängungsmechanismus stärker in diese Wahrscheinlichkeit einfließen zu lassen, wurde die Berechnung der Wahrscheinlichkeit für ein Nachladen der Daten in den Puffer durch die Häufigkeit der Anfragen A gewichtet.

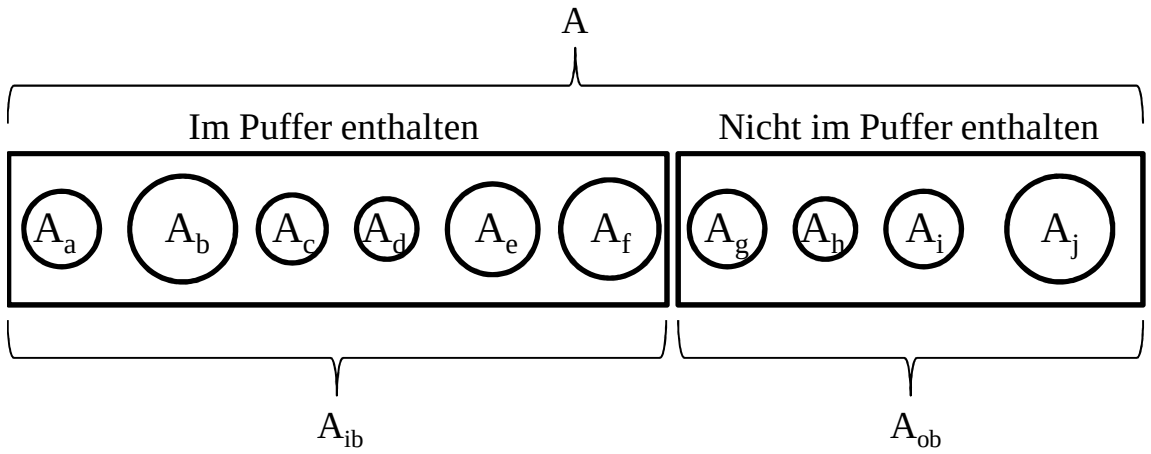


Abbildung 5-16: Beispielhafte Liste der Datenbankbereiche mit der relativen Häufigkeit der Anfragen
Quelle: Eigene Darstellung

Wie in Abbildung 5-16 dargestellt, können auch die Häufigkeiten der Anfragen an die jeweiligen Datenbankbereiche für die Berechnung der Wahrscheinlichkeit eines Nachladens des betreffenden Bereiches verwendet werden.

$$A_{ib} = \sum_{x=a}^{f} A_x \text{ bzw. } A_{ib} = \sum_{x=g}^{jf} A_x$$

Damit ergeben sich näherungsweise die folgenden Weiterleitungswahrscheinlichkeiten:

$$P(L_x) = 1 - \frac{A_{ib}}{A}; x = a, b, c, d, e, f \text{ bzw. } P(L_x) = 1 - \frac{A_{ob}}{A}; x = g, h, i, j$$

Da Invalidierungen von gepufferten Objekten ebenso negative Auswirkungen auf die Performance mit sich bringen, müssen diese ebenso in der Weiterleitungswahrscheinlich integriert werden. Durch die Analyse der Inhalte der Puffer ist deren Inhalt bekannt. Dieser Inhalt kann nun mit den SQL-Anfragen aus dem SAP-Performance-Trace abgeglichen werden, um festzustellen, ob es eine Insert- oder Update-Anfrage gibt, die ein gepuffertes Objekt betrifft. Ist dies der Fall, so erhöht dies die Wahrscheinlichkeit, dass der betreffende Datenbankbereich nicht aus dem Puffer, sondern von der Datenbank gelesen werden muss. Zusätzlich werden bei einer Invalidierung eines gepufferten Objekts, wie in 4.2.2 dargestellt, die nächsten k Anfragen auf dieses Objekt direkt aus der Datenbank gelesen. Die Anzahl k der Anfragen, kann über die Systemkonfiguration verändert werden. Um dies nun in die Weiterleitungswahrscheinlichkeit zu integrieren, wird bei der Berechnung der relativen Häufigkeit pro Datenbankbereich die Anzahl der Anfragen um das Produkt aus Invalidierungen und der Anzahl k reduziert.

5.3.6. Datenbank

Die Datenbank wird, wie bereits erwähnt, als Black Box modelliert. Für jeden Datenbankbereich existieren, falls benötigt, jeweils ein Entry für „Insert“, „Update“, „Delete“, „Select Single“ und „Select“.

5.3.6.1. Service-Zeiten

Um mit dem Modell des SAP-WebAS-ABAP genaue Vorhersagen der Antwortzeit zu gewährleisten, müssen die Servicezeiten der Datenbank implizit sämtliche möglichen

Queueing-Zeiten enthalten. Die Parametrisierung der Datenbankbereiche muss durch Messung und nachfolgender Fehleridentifikation und Ausreißerelimination anhand der bereits besprochenen Charakteristika Mittelwert und quadriertem Variationskoeffizienten geschehen.

5.3.6.2. Multiplizität

Sollte es für das verwendete Datenbanksystem, wie z.B. der MaxDB (Boegelsack et al. 2009) eine maximale Anzahl von Threads für die Bearbeitung von Benutzeranfragen geben, so kann diese Anzahl für die Parametrisierung der Multiplizität der Datenbanktasks verwendet werden. Damit kann für die Datenbank eine Kapazität von parallelen Anfragen modelliert werden, welche in vielen Datenbanksystemen so umgesetzt ist. Ist in dem abzubildenden Datenbanksystem keine obere Grenze für parallele Benutzeranfragen vorgesehen, so kann die Multiplizität des Datenbanktasks als „infinite“, also unendlich, definiert werden.

5.3.6.3. CPU-Nutzung

Ähnliches gilt für die Parametrisierung der CPU. Wenn es Messwerte bzgl. des CPU-Verbrauchs pro Anfrage gibt und dies betrachtet werden soll, so kann entweder die CPU des SAP-WebAS-ABAP bei einer Zentralinstallation verwendet, oder es können bei einer verteilten Installation eine beliebige Anzahl von CPUs des Datenbankservers modelliert werden. Zu beachten ist dabei jedoch der beträchtliche I/O-Anteil der Antwortzeit einer jeden Datenbankanfrage. Da die Servicezeit ja implizit der benötigten Zeit in der CPU entspricht, wird bei der Parametrisierung der Servicezeit über die Antwortzeit der Datenbank eine höhere CPU-Last abgebildet als dies tatsächlich der Fall ist. Um dies auszugleichen, ist eine Erhöhung der modellierten und parametrisierten CPU-Kapazität nötig, um realitätsnahe Simulationsergebnisse zu erhalten. Liegen entsprechende Messdaten über CPU-Auslastung, Anzahl der Anfragen an die Datenbank und der I/O-Wait Abschnitte vor, so kann daraus der für die Parametrisierung der CPU benötigte Aufschlag errechnet werden.

5.4. Fazit

In diesem Kapitel wurde das in der vorliegenden Arbeit erstellte Artefakt für die Modellierung und Simulation der Performance anhand des in Kapitel 2 identifizierten Leistungsmaßes beschrieben. Dies beinhaltet die Modellierung der in Kapitel 3 identifizierten System-Komponenten sowie die Konzeption logischer Komponenten für die Abbildung der internen Abläufe. Ebenso wird in diesem Kapitel die für die Simulation benötigte Parametrisierung des Modells dargestellt. Dabei wird aufgezeigt, wie die Messwerte, welche durch die in Kapitel 4 vorgestellten Messwerkzeuge ermittelt werden, für die Parametrisierung aufbereitet werden müssen. Dies beantwortet Forschungsfrage 2 nach den technischen und logischen Modellkomponenten für die Performance-Modellierung und deren Parametrisierung.

6. Evaluation der Simulationsergebnisse

Bei einem ERP-System handelt es sich um ein integriertes System, welches sämtliche, in einem Unternehmen ablaufenden, Geschäftsprozesse unterstützt. Dabei enthält es Module aus Bereichen wie:

- Beschaffung
- Produktion
- Vertrieb
- Anlagenwirtschaft
- Personalwesen
- Finanz-und Rechnungswesen

Diese Module sind über eine gemeinsame Datenbasis miteinander verbunden und ermöglichen damit die Unterstützung von modulübergreifenden Geschäftsprozessen. Um einen Geschäftsprozess mit Referenzcharakter zu implementieren, sollte dieser aus funktioneller Sicht modulübergreifend sein und häufig verwendete, grundlegende Vorgänge eines ERP-Systems abbilden.

Auf technischer Seite muss ein Referenzprozess die in dieser Arbeit ermittelten performance-kritischen Eigenschaften (Puffer- und Enqueue Nutzung sowie Hauptspeicher- und CPU-Last) beinhalten.

6.1. Workload

In diesem Kapitel wird der für die vorliegende Arbeit als Workload herangezogene Geschäftsprozess sowohl aus funktioneller als auch aus technischer Sicht beschrieben und die Implementierung dieser Fallstudie mit Hilfe von WebService-Aufrufen dargestellt.

6.1.1. Anforderungen an den Workload

Die in Abschnitt 2.3.1 bereits dargestellten Anforderungen an einen Benchmark gelten ebenso für den in diesem Kapitel verwendeten Workload, um das Performance Modell durch den Vergleich von Simulations- und Messergebnissen zu evaluieren. Da in diesem Modell die interne Struktur des SAP-WebAS-ABAP abgebildet wurde, ist das folgende Kriterium von Bögelsack (2010) besonders wichtig:

- Vergleichbarkeit:
  Um die Vergleichbarkeit zwischen verschiedenen Implementierungen von SAP-ERP-Systemen zu gewährleisten, muss ein Benchmark die internen Strukturen des SAP-Systems nutzen.

Für die Evaluation von Performance-Modellen muss diese Anforderung sogar noch spezifiziert werden, da der Workload natürlich zusätzlich zu der Variation von Lastprofilen die modellierten Konzepte prüfen, d.h. mit den modellierten Strukturen arbeiten muss:

- Nutzung der Tabellenpuffer mit auftretenden Verdrängungen
- Nutzung von Enqueue-Objekten mit wechselseitigen Sperren

6.1.2. Die PP-Fallstudie der SAP-University-Competence-Center

In einem SAP-ERP-System unterstützen zahlreiche Module dessen Anwendung in verschiedenen geschäftlichen Bereichen. Um Geschäftsprozesse durchführen zu können, muss ein SAP-System zuerst an ein Unternehmen durch einen Customizing-Prozess angepasst werden. Für Schulungs- und Demonstrationszwecke bietet die SAP ein ERP-System an, in welchem bereits eine Musterfirma, die IDES-AG, vorbereitet ist. IDES steht hierbei für „Internation Demo and Education System“.

Die SAP-University-Competence-Center (UCC) bieten für den Einsatz in Forschung und Lehre vier Fallstudien (Weidner 2006) an. Diese aufeinander aufbauenden Fallbeispiele nutzen u.a. die folgenden Module:

- Produktionsplanung- und Steuerung (PP)
- Rechnungswesen, Gemeinkostencontrolling (CO)
- Logistik Allgemein (LO)
- Projektabwicklung (PS)
- Vertriebssystem (SD)
- Materialwirtschaft (MM)
- Finanzbuchhaltung (FB)

Da diese Fallstudien häufig auf den SAP-Systemen der SAP-UCCs durch deren Kunden in der Lehre durchgeführt werden und verschiedene Module verwenden, eignet sich die PP-Fallstudie aufgrund der Abhängigkeit der restlichen SAP-UCC-Fallstudien für eine Implementierung als Referenzprozess für die Performance Evaluation eines SAP-ERP-Systems.

Wie in Abbildung 6-1 dargestellt, besteht die PP-Fallstudie aus 11 Tätigkeiten in den Modulen Materialwirtschaft, Produktionsplanung und Gemeinkostenkontrolling. Diese Tätigkeiten lassen sich in 4 Bereiche untergliedern:

1. Stammdatenpflege: Erfassung des zu erzeugenden Produkts und dessen Komponenten
2. Planung und Erstellung von Stücklisten und Arbeitsabläufen
3. Kalkulation der Produktkosten

4. Produktion des neuen Erzeugnisses

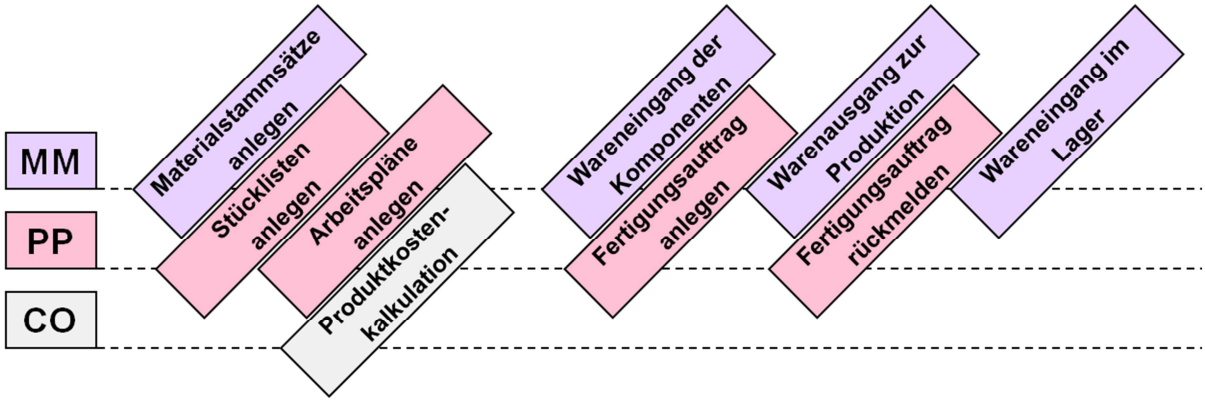


Abbildung 6-1: Schritte der Fallstudie mit den zugehörigen Modulen
Quelle: (Weidner 2006)

Damit werden innerhalb dieses Workloads Programme aus mehreren Modulen verwendet und so die Wahrscheinlichkeit erhöht, dass die vom SAP-Kernel bereitgestellte Infrastruktur auf unterschiedliche Weise genutzt wird.

6.1.2.1. Stammdatenpflege

Die im ersten Bereich zu pflegenden Stammdaten sowie deren Abhängigkeiten untereinander sind in Abbildung 6-2 dargestellt. Das als „Fertigerzeugnis“ anzulegende Motorrad „Bike“ besteht aus zwei „Halbfabrikaten“. „Halbfabrikate“ können sowohl selbst hergestellt, oder von Zulieferern bezogen werden. Diese beiden Varianten werden in dieser Fallstudie durch den „Rahmen“, welcher von außen bezogen wird, und den „Motor“, welcher selbst zusammengesetzt wird, verwendet. Der „Motor“ besteht aus den Rohmaterialen bzw. Rohstoffen „Block“ und „Welle“.

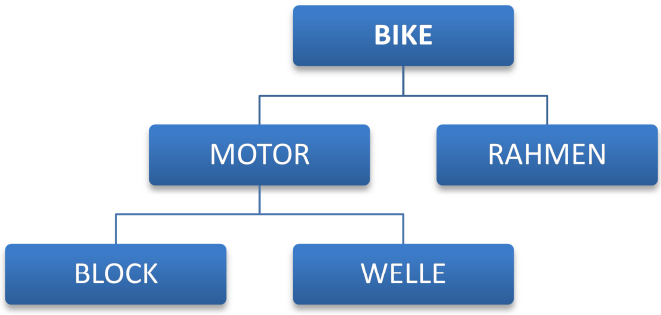


Abbildung 6-2: In der Fallstudie verwendete Materialen
Quelle: In Anlehnung an (Weidner 2006)

Damit integriert die Fallstudie in den fünf anzulegenden Materialstammsätzen die unterschiedlichen Typen von Materialien.

6.1.2.2. Stücklisten und Arbeitsabläufe

Im zweiten Bereich werden für die angelegten Materialstammdaten Stücklisten erzeugt. Da sowohl das Fertigerzeugnis „Bike“ als auch das Halbfabrikat „Motor“ aus unterschiedlichen Bestandteilen bestehen, müssen für diese beiden Elemente Stücklisten angelegt werden. „Unter einer Stückliste versteht man ein formal aufgebautes Verzeichnis für Gegenstände (Materialien, Dokumente, Kundenaufträge, etc.), das alle zugehörigen Bestandteile unter Angabe von Bezeichnung, Menge und Einheit enthält. Durch ihre Tiefe von 2 handelt es sich beim vorliegenden Beispiel um eine „mehrstufige Baukastenstückliste“ (Weidner 2006). Da für die Erstellung von „Motor“ und „Bike“, welche aus verschiedenen Elementen zusammengesetzt werden, unterschiedliche Arbeitsschritte mit unterschiedlichen Ressourcen benötigt werden, muss in diesem Bereich auch noch jeweils ein Arbeitsplan erstellt werden. „Ein Arbeitsplan beschreibt einen Fertigungsablauf zur Herstellung von Werksmaterialien bzw. zur Erbringung von Leistungen. Arbeitspläne werden dabei als Vorlage für Fertigungs- und Serienaufträge und als Grundlage für die Erzeugniskalkulation verwendet“ (Weidner 2006).

6.1.2.3. Produktkostenkalkulation

Die im zweiten Abschnitt erzeugten Stücklisten und Arbeitspläne dienen im dritten Bereich der zweistufigen Materialkostenkalkulation als Grundlage. In der ersten Stufe werden mittels einer Erzeugniskalkulation die Herstell- und Selbstkosten pro Erzeugniseinheit ermittelt. Um diese Kosten in die Materialstammsätze zu übertragen, wird in einem zweiten Schritt eine Preisfortschreibung durchgeführt. Die damit durchgeführte Preisfindung, als Grundlage für die Preispolitik, ist für die weitere Durchführung dieser Fallstudie jedoch nicht von Belang und wurde daher bei der Implementierung der Fallstudie innerhalb eines Lasttreibers nicht umgesetzt. Für eine Erweiterung des Lasttreibers um die drei bereits erwähnten UCC-Fallstudien muss dieser Schritt noch zusätzlich implementiert werden.

6.1.2.4. Produktionsprozess

Im letzten Abschnitt wird nun der Produktionsprozess durchgeführt, welcher sich in die folgenden Aktivitäten aufgliedert:

- Buchung des Wareneingangs aller benötigten Materialien (Block, Welle, Rahmen)
- Fertigungsaufträge für Materialien „BIKE“ und „MOTOR“ erzeugen
- Ermitteln von Plankosten der Fertigungsaufträge
- Freigeben der Fertigungsaufträge
- Produktion und Rückmelden der Fertigungsaufträge nach der Herstellung

Um die für den Produktionsprozess benötigten Komponenten in ausreichender Menge vorrätig zu haben, werden im ersten Schritt die Warenbewegungen manuell, ohne Bestellung, ins Lager gebucht. Um diese Vorgänge mittels eines MRP-Laufs automatisch auszulösen, müssen zusätzlich Kunden- und Lieferantenstammsätze gepflegt sein. Im zweiten Schritt werden nun die Fertigungsaufträge für „MOTOR“ und „BIKE“, in dieser Reihenfolge, erzeugt und zurückgemeldet. Diese Fertigungsaufträge beinhalten Informationen für die

Produktion, sodass ein Produkt in einer bestimmten Menge und zu einem bestimmten Termin erzeugt werden kann. In dieser Fallstudie wird die Terminierungsart „PP01“ verwendet, sodass die Startzeit der Produktion automatisch anhand des Zeitpunkts für deren Ende ermittelt wird. Diese Aufträge werden nun in Schritt 4 freigegeben. Nach der Produktion in Schritt 5 werden die Warenbewegungen automatisch durchgeführt, indem die Fertigungsaufträge zurückgemeldet werden. Damit vermindert sich im System der Bestand an Rohmaterialen, und während sich die Anzahl an gefertigten „Bike“-Materialien erhöht.

6.1.3. Umsetzung mittels remotefähiger Funktionsbausteine

Um den im vorigen Abschnitt inhaltlich dargestellten Geschäftsprozess automatisch mit einer zu wählenden Anzahl von parallelen Ausführungen implementieren zu können, wurden für die einzelnen Schritte die folgenden Funktionsbausteine identifiziert.

6.1.3.1. BAPI_MATERIAL_SAVEDATA

Mit diesem Baustein können Materialstammdaten angelegt bzw. bestehende Materialstammsätze verändert werden. Mit Hilfe der Übergabeparameter kann gesteuert werden, ob Rohmaterialien, Halbfabrikate oder Fertigerzeugnisse erstellt werden.

Dieser Baustein enthält insgesamt 23 Übergabeparameter, von welchen 19 aus Strukturen im Sinne der SAP bestehen, welche wiederum eigene Felder enthalten. Damit bietet dieser Baustein die Möglichkeit, die Erstellung von Materialen mit 971 verschiedenen, größtenteils nicht dokumentierten, Parametern zu konfigurieren.

Die mittels dieses Bausteins erzeugten Daten werden in der weiteren Ausführung des Geschäftsprozesses von anderen Prozessschritten benötigt. Die Identifikation erfolgt durch die Materialnummer, welche dem Baustein bereits beim Aufruf übergeben wird. Folgende Materialen werden erstellt:

- Rohmaterial Block
- Rohmaterial Welle
- Halbfabrikat Motor
- Halbfabrikat Rahmen
- Fertigerzeugnis Bike

Bei der automatischen Erzeugung von Materialnamen innerhalb eines Lastgenerators ist die maximale Länge desselben von großer Wichtigkeit. Werden keine eindeutigen Namen generiert, besteht die Möglichkeit, dass ein Material bereits besteht. In diesem Fall wird kein neues Material erzeugt, sondern das bestehende nur modifiziert. Dies hat einen signifikanten Einfluss auf die Dauer und den Ressourcenbedarf der Ausführung.

6.1.3.2. CSAP_MAT_BOM_CREATE

Für die Erzeugung von Stücklisten wurde der Baustein CSAP_MAT_BOM_CREATE identifiziert, da er auch die Möglichkeit bietet, zweistufige Stücklisten, wie für den Geschäftsprozess nötig, zu erstellen.

6.1.3.3. Z_BAPI_ROUTING_CREATE

Arbeitspläne können mit dem Baustein BAPI_ROUTING_CREATE erzeugt werden. Im Gegensatz zu den bisher verwendeten Bausteinen beinhaltet dieser keine eigene „Commit“-Anweisung. Ohne diese Anweisung werden jedoch die erstellten Arbeitspläne nicht persistent im System gespeichert. „The time interval between two consistent statuses, that is, the mechanism that runs during this time is called the LWU (Logical unit of Work). Each LUW is always ended with a commit, which sets all the changes made, or a rollback, which can undo all the changes.” (Keller/Kruger 2002) Innerhalb von ABAP-Programmen wird diese „Commit“-Anweisung üblicherweise direkt im Anschluss an den Aufruf des Bausteins ausgeführt. Dies ist bei einem externen Aufruf nicht direkt möglich. Die Verwendung des Bausteins „BAPI_TRANSACTION_COMMIT“, welcher genau dieses Kommando absetzt, würde aber eine sitzungsorientierte (stateful) Kommunikation erfordern, damit das „Commit“ einem vorhergehenden RFC- oder WebService-Aufruf eindeutig zugeordnet werden kann. Aus diesem Grund wurde Z_BAPI_ROUTING_CREATE als Wrapper erstellt, welcher den Baustein BAPI_ROUTING_CREATE aufruft und im ABAP-Code das „Commit“-Statement ausführt.

6.1.3.4. Z_BAPI_GOODSMVT_CREATE

Analog zu der Beschreibung in 6.1.3.3 wurde auch für diesen Baustein ein Wrapper erzeugt, um die Änderungen des originalen SAP-Bausteins „BAPI_GOODSMVT_CREATE“ persistent in die Datenbank zu schreiben. Mit Hilfe dieser BAPI werden die Warenbewegungen manuell gebucht. Damit wird sichergestellt, dass die für die Produktion benötigten Materialien auf Lager sind.

6.1.3.5. Z_BAPI_PRODORD_CREATE

Die für die Produktion benötigten Fertigungsaufträge werden mittels „BAPI_PRODORD_CREATE“ erzeugt. Für die persistente Speicherung wurde wieder ein Wrapper („Z_BAPI_PRODORD_CREATE“) erzeugt.

6.1.3.6. Z_BAPI_PRODORD_COSTING

Um die Selbst- und Herstellkosten anhand von Stücklisten und Arbeitsplänen zu berechnen, wird mittels dieses Bausteins eine Materialkalkulation durchgeführt. Für die Ausführung sind die relevanten Stücklisten und Arbeitspläne, deren Erzeugung in den Absätzen 6.1.3.2 und 6.1.3.3 beschrieben wurde zu referenzieren.

6.1.3.7. Z_BAPI_PRODORDCONF

Um die Produktionsplankosten in die Materialstammdaten zu überführen, muss nach der Produktion der Produktionsauftrag bestätigt und damit der gesamte Prozess abgeschlossen werden. Um diese durchführen zu können, benötigt diese BAPI die Informationen aus dem Fertigungsauftrag, welcher über die Auftragsnummer referenziert wird.

6.1.3.8. Ablauf der Fallstudie mittels externer Aufrufe der Funktionsbausteine

Um den Geschäftsprozess mit den Funktionsbausteine umzusetzen, die in den vorhergehenden Abschnitten beschriebenen wurden, ist es nötig, die Reihenfolge der Aufrufe anhand der wechselseitigen Abhängigkeiten zu orientieren. Abbildung 6-3 zeigt den Ablauf der Fallstudie anhand der Bausteine, welche sequentiell aufgerufen werden. Jeder Funktionsbaustein benötigt eine bestimmte Menge an Übergabeparametern, um das gewünschte Resultat, die Durchführung der PP-Fallstudie, nachzubilden. Der Großteil dieser Parameter ist statisch und damit unabhängig von der Ausführung der einzelnen Prozessinstanz. Die Parameter in der rechten Spalte von Abbildung 6-3 sind hingegen prozessinstanzspezifisch und damit dynamisch, da sie nicht bei jeder Testausführung identisch sind.

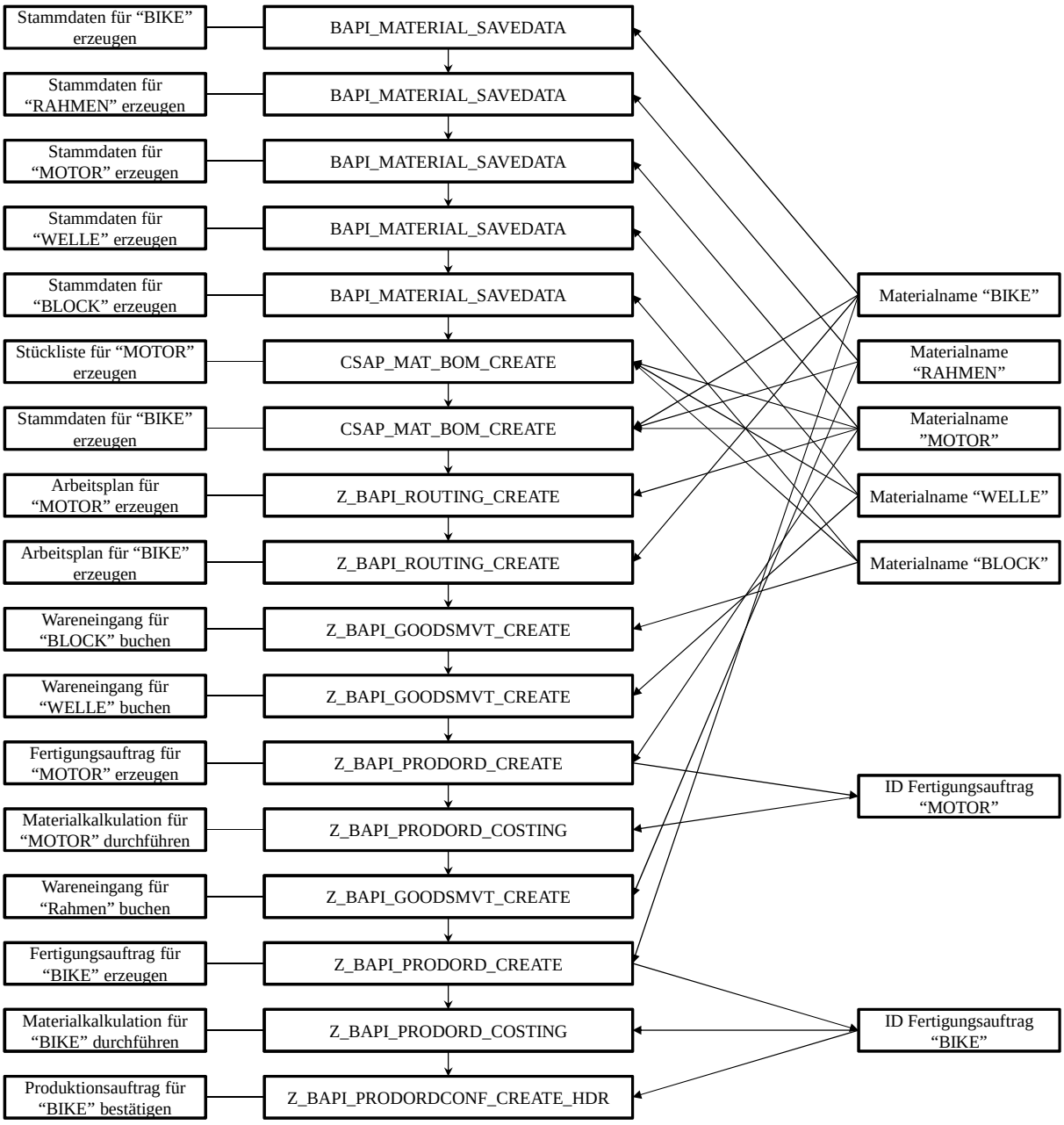


Abbildung 6-3: Ablauf der Fallstudie mit den jeweils benötigen, von der Testinstanz abhängigen, Übergabeparametern
Quelle: Eigene Darstellung

Diese Parameter können in zwei Gruppen unterteilt werden:

- Erzeugt durch den Aufrufer
- Erzeugt durch das SAP-System

Durch den Aufrufer müssen für eine Ausführung des Prozesses fünf Materialnamen erzeugt werden. Wie bereits erwähnt, ist dabei auf die Konventionen zur Erstellung zu achten, da Materialnamen das Material eindeutig beschreiben und damit während verschiedener Testläufe oder innerhalb des gleichen Testlaufs nicht mehrfach verwendet werden dürfen. Weitere dynamische Parameter, wie z.B. die Nummer des Fertigungsauftrags, werden durch das System selbst innerhalb eines BAPI-Aufrufs generiert und diesem als Rückgabewert übermittelt. Damit stehen diese dynamischen vom System erzeugten Parameter späteren BAPI-Aufrufen als Übergabeparameter zur Verfügung.

6.2. Lasterzeugung

Für die Evaluation des konzipierten Performance-Modells müssen Messwerte erhoben werden, welche einen tragfähigen Vergleich mit den Simulationsergebnissen ermöglichen. In diesem Kapitel werden zunächst die Anforderungen an einen Lastgenerator erhoben um darauf aufbauend die Implementierung darzustellen.

6.2.1. Anforderungen

Nach (Jehle 2010) muss ein Lastgenerator die folgenden Anforderungen erfüllen:

- Reproduzierbar
- Automatisch
- Rekonfigurierbar
- Modular
- Archivierbar
- Auswertbar
- Skalierbar

Aufgrund der starken Abhängigkeiten zwischen den verschiedenen Schritten des der Workload-Erzeugung zu Grunde liegenden Geschäftsprozesses wurde in der vorliegenden Arbeit die Granularität der Modularität auf die Ebene vollständiger Geschäftsprozesse abstrahiert.

6.2.2. JAVA basierter Lastgenerator

Für die Umsetzung der im vorhergehenden Kapitel diskutierten Anforderungen „reproduzierbar“, „automatisch“, „rekonfigurierbar“ und „archivierbar“ wurde im Rahmen dieser Arbeit ein Lastgenerator konzeptioniert und implementiert. Die folgenden Abschnitte

werden zunächst die abstrakte Architektur, die Schnittstellen zu den einzelnen Komponenten sowie die generelle Funktionsweise dieses Lastgenerators darstellen.

6.2.2.1. Architektur

Für die Entwicklung der abstrakten Architektur, welche in Abbildung 6-4 dargestellt ist, wurde das Modell von Chen et al. (2008) als Grundlage verwendet. Ein zentraler Punkt der Lasterzeugung ist der zusätzliche Einfluss auf das Testobjekt, welcher durch den Betrieb des Lastgenerators erzeugt wird. Dieser Einfluss muss so gering wie möglich gehalten werden, um signifikante Messergebnisse zu erhalten. Durch die Trennung von Testobjekt und Testtreiber wird dies gewährleistet. Damit ergeben sich drei Bestandteile des Lastgenerators:

- Parametrisierung
- Testtreiber
- Ergebnisse

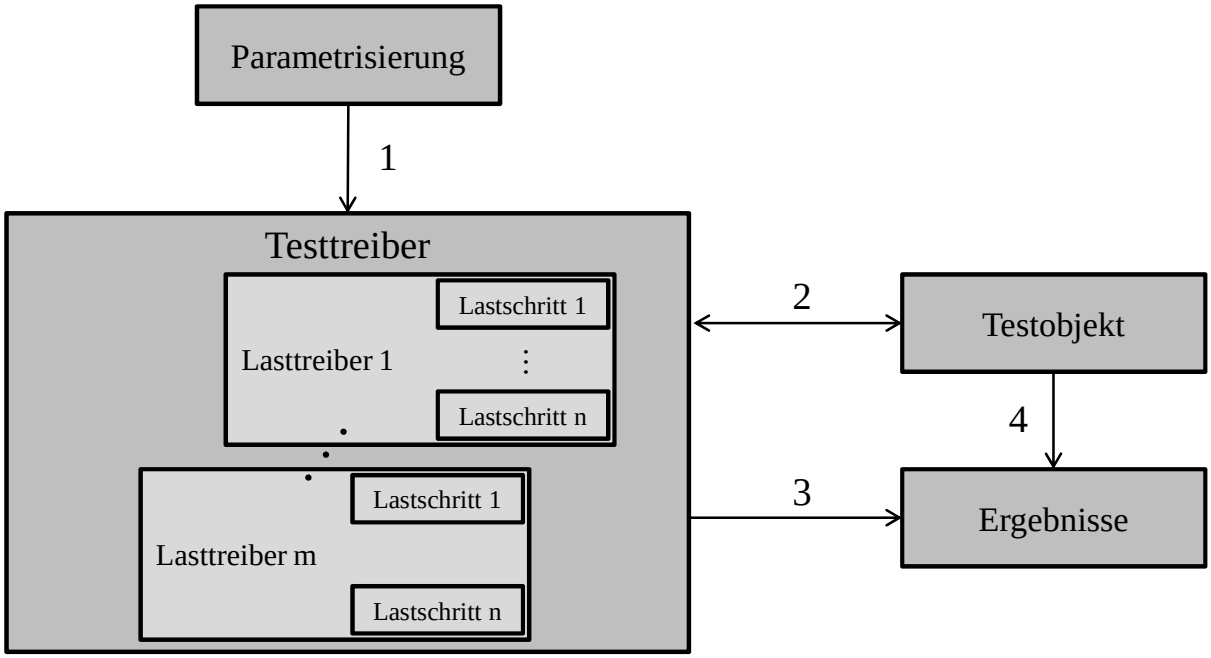


Abbildung 6-4: Abstrakte Architektur für die Performance-Messung
Quelle: In Anlehnung an (Chen et al. 2008)

Um den Testlauf zu parametrisieren, werden im ersten Schritt u.a. Informationen über die Anzahl der emulierten parallelen Benutzer, die Anzahl der Wiederholungen der einzelnen Testsequenzen sowie die Wartezeit und deren statistischer Verteilung zwischen diesen Sequenzen von außen an den Testtreiber übergeben. Dieser Architektur liegt die Abhängigkeit der Schritte des verwendeten Geschäftsprozesses von jeweils vorhergehenden zugrunde, sodass die Ausführungsreihenfolge der Lastschritte nicht verändert werden kann. Es wurde sich dabei darauf beschränkt, die Anzahl der durchzuführenden Schritte in der durch die Prozessbeschreibung festgelegten Reihenfolge auswählen zu können. Da dieser Lastgenerator nicht nur für die Ausführung von Lasttests vorgesehen ist, sondern auch die Möglichkeit der qualitativen Analyse des Testobjekts unterstützen soll, kann dies auch über die Parametrisierung festgelegt werden. Soll die Testausführung der qualitativen Analyse dienen,

so werden durch den Testtreiber noch weitere intrusive Messwerkzeuge, wie in 4.2 dargestellt, auf dem Testobjekt aktiviert.

Mit diesen Informationen kann der Testtreiber in Schritt zwei die Lasttreiber gemäß der Parametrisierung starten und somit die gewünschte Last auf dem Testobjekt erzeugen. In der Implementierung der vorliegenden Arbeit ist die Ausführung eines Lasttreibers mit der Durchführung des Geschäftsprozesses durch einen Benutzer vergleichbar. Damit muss der Testtreiber für jeden zu emulierenden Benutzer einen Lasttreiber mit den für ihn bestimmten Ausführungsparametern instanziieren und diesen gemäß den Durchführungsvorschriften, im späteren Lastmodus genannt, starten. Um die Möglichkeit zu gewährleisten, diesen Prozess an die Eigenschaften von realen Workloads anzupassen, existieren verschiedene Möglichkeiten, die Wartezeit („Think time“) zwischen zwei Lastschritten auf Basis verschiedener Verteilungen zu konfigurieren.

Ist die Ausführung des Testfalls beendet, so werden in Schritt 3 die Messergebnisse aus der Blackbox-Sicht in einer Datenbank für die spätere Analyse gespeichert. Diente die Testausführung der qualitativen Analyse, so werden in diesem Schritt die zusätzlich gestarteten Messwerkzeuge auf dem Testobjekt gestoppt, bevor die Messdaten exportiert werden.

Da in der vorliegenden Arbeit die interne Struktur des Testobjekts analysiert wird, werden in einem weiteren Schritt Performance-Werte aus dem Testobjekt ausgelesen. Je nach Ziel der Testausführung enthalten diese Werte nur die STAD-Records. Im Falle der qualitativen Analyse des Testobjekts werden in diesem Schritt noch zusätzlich Performance-Trace, Pufferstatistiken und –zustände exportiert und in der Datenbank abgelegt.

6.2.2.2. Interface des Testtreibers

Die in dieser Interface-Beschreibung in Abbildung 6-5 vorgesehenen Methoden stellen den Gesamtumfang der möglichen Konfigurationen der Testausführungen dar.

Die Methode „setConfiguration“ speichert die Konfiguration für den Zugriff auf das Testobjekt. Darin enthaltene Informationen sind:

- Beschreibung Zielsystem (System ID, Mandant, Präfix Benutzername und Passwort für Lastbenutzer)
- ServiceLocator (URL des SOAP-Handlers)
- Benutzername und Passwort für Analysebenutzer
- Benutzername und Passwort für Zeitabgleich zwischen Testobjekt und Testtreiber
- Hostname und Ports der zur Verfügung stehenden Lastserver

Mit Ausnahme der Methode „createMeasurement()“, welche die bereits angesprochene qualitative Untersuchung des Testobjekts durchführt, dienen die restlichen Methoden dazu, die Parametrisierungen des Lastlaufes einzustellen. Für die qualitative Untersuchung kann dieser Methode mittels einer Zahl x der zu analysierende Lastschritt übergeben werden. Wird

der Test gestartet, so werden zunächst die Schritte von 0 bis x-1 durchgeführt und sämtliche Puffer im System invalidiert. Danach werden die bereits dargestellten Messwerkzeuge auf dem Testsystem gestartet und der gewünschte Lastschritt x durchgeführt.

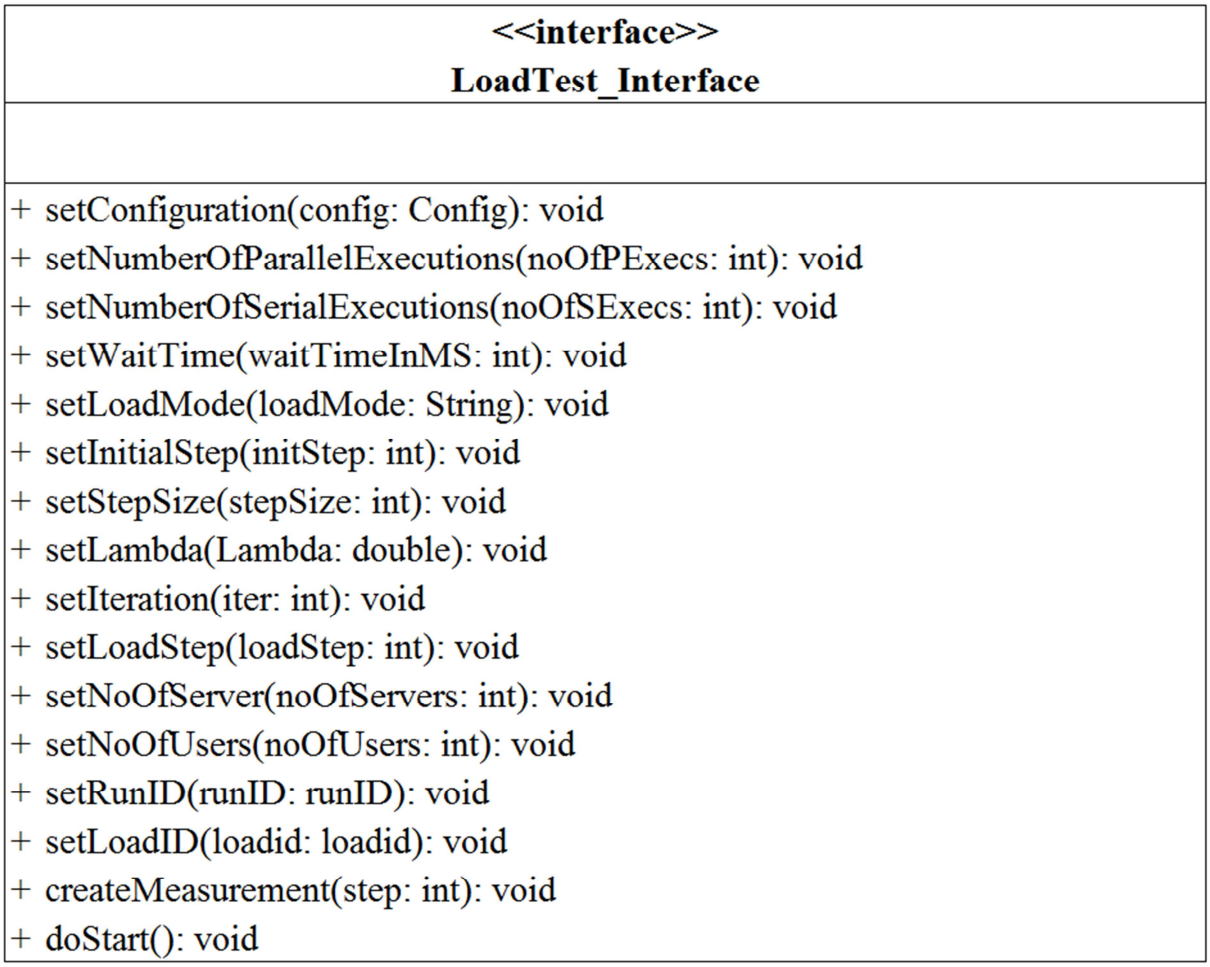


Abbildung 6-5: Interface des Lasttreibers LoadTest
Quelle: Eigene Darstellung

6.2.2.3. Interface des Lasttreibers

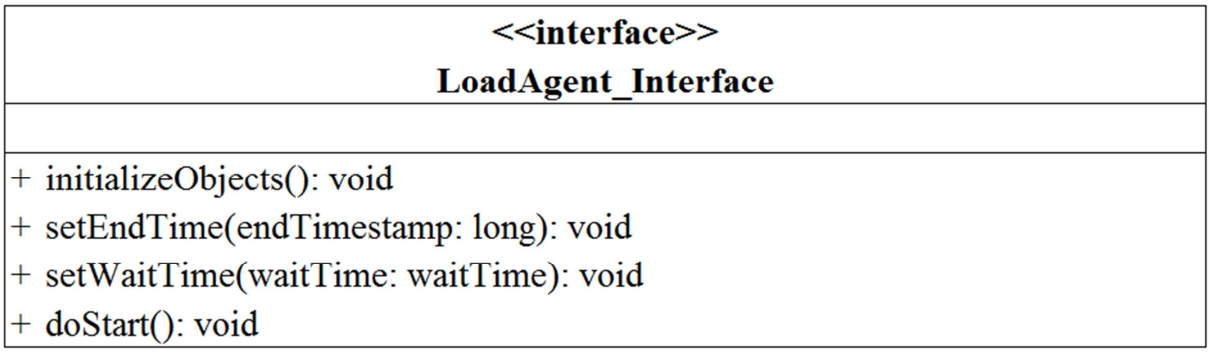


Abbildung 6-6: Interface des Lastesels LoadAgent
Quelle: Eigene Darstellung

Um nun einen Geschäftsprozess automatisch abspielen zu können, muss der Last-Agent mittels des in Abbildung 6-6 dargestellten Interface implementiert werden.

Damit die Testausführung nicht damit beginnt, dass alle „LoadAgents“ zuerst die benötigten Objekte initialisieren und damit in den ersten Sekunden keine Last erzeugt wird, muss dies vor Beginn der Testausführung mittels der Methode „initializeObjects()“ durchgeführt werden.

Um die zeitliche Ausführungsdauer der Testausführung einschränken zu können, kann dem Last-Agenten über die Methode „setEndTime()“ ein Unix-Timestamp übergeben werden. Dieser Zeitstempel übergibt den Zeitpunkt, an dem der Lastesel die Lasterzeugung beenden muss.

Mittels des Objekts „waittime“ übergibt die gleichnamige Methode die Beschreibung der „Think time“ vor jedem Lastschritt. In der, der vorliegenden Arbeit zugrunde liegenden Implementierung wurden für die „Thinktime“ zwei verschiedene Verteilungen umgesetzt:

- Exponentialverteilung mit Erwartungswert λ
- Gleichverteilung mit Konstante c

6.2.2.4. Funktionsweise des Testtreibers

Um die grundsätzliche Funktionsweise des Testtreibers zu beschreiben, wurde dessen Aufbau in Abbildung 6-7 anhand eines konzeptionellen Klassendiagramms dargestellt.

Die zentrale Steuereinheit des implementierten Testtreiber-Prototypen stellt die Klasse „LoadTestManager“ dar. Diese lädt die Konfiguration über die Instanziierung der „Configuration“-Klasse und instanziiert je nach Art der Verwendung als Client oder als Server das oder die entsprechenden Objekte der Klassen „LoadTestClient“ oder „LoadTestServer“. Um keine parallelen Instanziierungen und damit mehrfache Testläufe zuzulassen, wurde die Klasse als Singleton implementiert. Nach Krüger (2006) ist dies ein Design-Pattern für eine Klasse, von der nur ein einziges Objekt instanziiert werden kann.

In der ersten Evaluierungsphase hat sich herausgestellt, dass die benötigten Systemressourcen des Lastgenerators teilweise die verfügbaren Kapazitäten der physikalischen Rechner überstiegen, und damit die Emulation einer hohen Anzahl paralleler Benutzer nicht möglich war. Aus diesem Grund wurde eine Client-Server-Architektur eingeführt und mittels der Klassen „LoadTestClient“ und „LoadTestServer“ implementiert. Damit ist es möglich, die pure Lasterzeugung auf einer beliebigen Anzahl von Lastservern zu verteilen. Die Klassen für die folgenden Funktionalitäten wurden aus Gründen der Übersichtlichkeit in Abbildung 6-7 nicht dargestellt:

- Abgleich der Systemzeiten zwischen Clients und Servern
- Synchronisation der Server bzgl. der Initialisierung und Start der „LoadAgents“
- Synchronisation der Server bzgl. der Vorbereitungen (Erstellen der Systembenutzer) der Lastläufe

Dient die Ausführung der qualitativen Analyse des Testobjekts, so wird in dieser Klasse auch das Auslesen der Performance-Trace über die Klassen „BufferTrace“, „EnqueueTrace“,

„SQLTrace“ und „RFCTrace“ durchgeführt. Diese erben ihre gemeinsamen Daten von der Klasse „PerfTrace“ und enthalten die Einträge der verschiedenen Performance-Traces in Listen des Typs „PerfTraceItem“, „EnqItem“ und „TabAccessItem“, wobei die letzten beiden Klassen von der Klasse „PerfTraceItem“ erben.

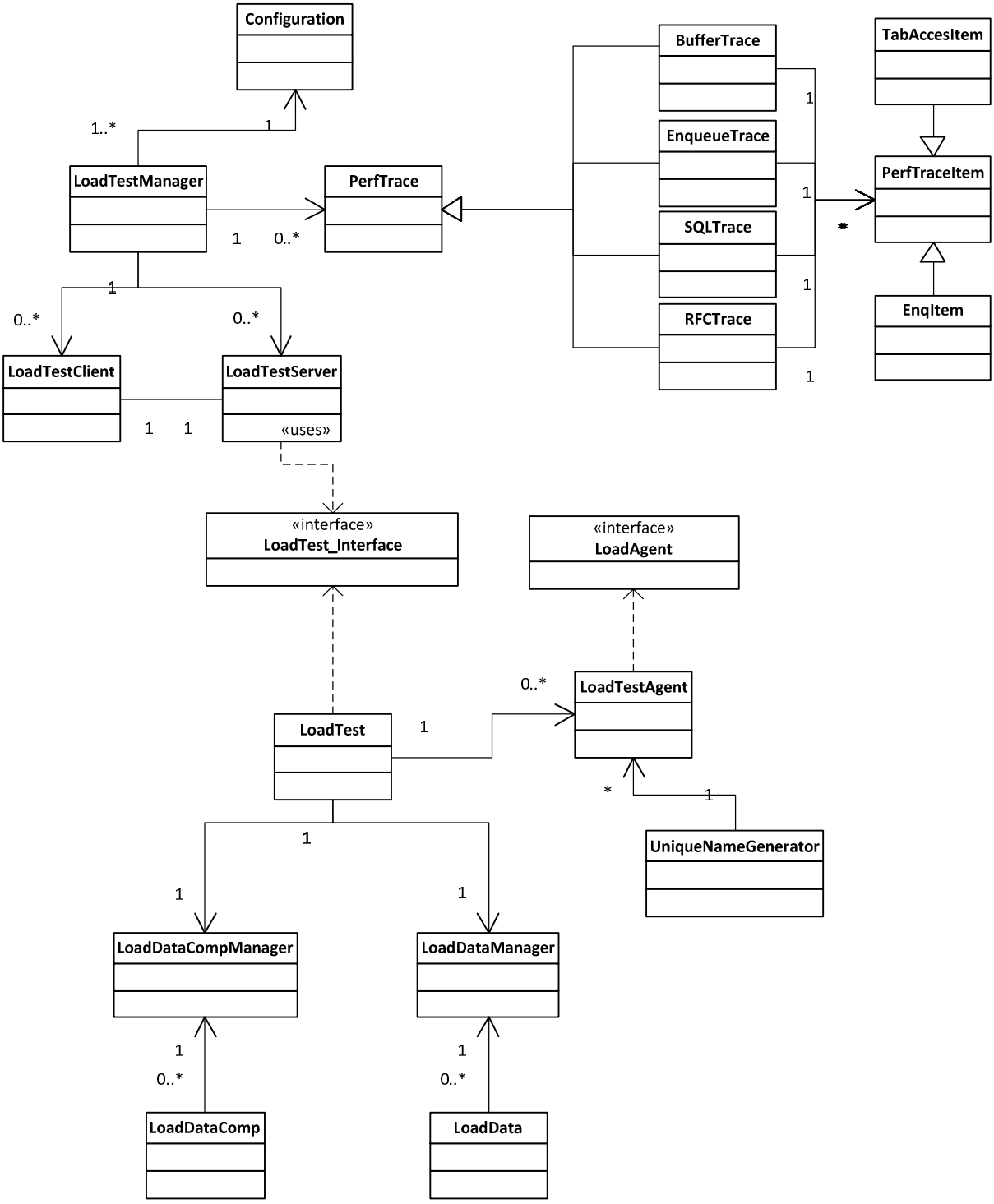


Abbildung 6-7: Konzeptionelles Klassendiagramm des Testtreibers
Quelle: Eigene Darstellung

Die Steuerung des Lastprofils ist in der Klasse „LoadTest“ implementiert. Sie übernimmt die Konfiguration von der Klasse „LoadTestServer“ und instanziiert die einzelnen Lastagenten.

Über ein „CyclicBarrier“-Objekt (Sun 2004) werden die Threads nach Beendigung des Testlaufs synchronisiert, sodass die gemessenen Antwortzeiten mit Hilfe der „LoadDataManager“-Klasse in Form von „LoadData“-Objekten in der Performance-Datenbank abgelegt werden können. Ist dieser Vorgang und damit der Testlauf beendet, können die Performance-Daten der SAP-Systemkomponenten durch eine Instanz der „LoadDataCompManager“-Klasse als „LoadDataComp“ Objekte ausgelesen und ebenfalls in der Performance-Datenbank abgelegt werden.

Die Klasse „LoadAgent“ implementiert das Interface „Runable“ der Klasse „Thread“. Damit ist es möglich, eine zu definierende Menge an parallelen Last-Threads zu erzeugen (Sun 2004) und damit parallele Benutzer zu emulieren. Um die bereits in Absatz 6.1.3 diskutierte Problematik der Erzeugung korrekter und eindeutiger Materialnamen zu lösen, werden die in den „LoadAgent“-Objekten benötigten Produktnamen mit Hilfe der Klasse „UniqueNameGenerator“ kreiert.

6.3. Messung

In diesem Kapitel werden Messungen beschrieben, die für die Performance-Analyse eines SAP-ERP-Systems und die Evaluation des Performance-Modells durch Vergleich der Simulationswerte mit gemessenen Performance Daten eine SAP-ERP-Systems im Labor als Grundlage dienen. Dazu wird zunächst die Intensität des bereits vorgestellten Workloads beschrieben, und die für die Evaluation gewählten Szenarien der Systemkonfiguration dargestellt. Im Anschluss folgt die Beschreibung der Messergebnisse.

6.3.1. Workload

Für die Erhebung der dieser Arbeit zu Grunde liegenden Performance-Daten wurde der in Absatz 6.1 dargestellte Geschäftsprozess verwendet. Dieser wurde durch den in 6.2 vorgestellten Lastgenerator erzeugt. Um eine möglichst realitätsnahe Last zu erzeugen, wurde zwischen jedem Schritt des Geschäftsprozesses eine exponentiell-verteilte Thinktime mit einem Erwartungswert von 4 Sekunden (Hellermann 1975) gewählt. Die Anzahl emulierter parallel arbeitender Benutzer wurde innerhalb eines Experiments in einzelnen Lastläufen jeweils um 25 bis zu einer Obergrenze von 125 parallelen Benutzern erhöht. Jedes Experiment wurde nach der Zurücksetzung des Systems in den Ausgangszustand 5-mal wiederholt. Um zu gewährleisten, dass der Lastgenerator über ausreichend Ressourcen verfügt, wurden während der Experimente 4 Lastserver mit jeweils 3 physikalischen Cores in einer mit 6 GB konfigurierten JVM ausgeführt.

6.3.2. Systemkonfiguration und Szenarien

Wie bereits im vorhergehenden Abschnitt ausgeführt, wurden die in diesem Kapitel vorgestellten Performance-Werte in drei Testszenarien erhoben. Diese unterscheiden sich in der Konfiguration des SAP-Systems und der darunterliegenden logischen Partition. Gleich bleiben in allen Konfigurationen die Größe des konfigurierten Arbeitsspeichers der virtuellen Maschine von 11 GB, sowie die bereits beschriebene Konfiguration des SAP-Systems mit Ausnahme der Größe der Tabellenpuffer. Dies dient der Evaluation des Modells in verschiedenen Szenarien, in welchen ein Computer-System betrieben werden kann.

Szenario1:

- Genutzte physikalische Cores: 6
- Tabellen-Puffer – „Generic Key“: 60.000 KB / 7500 Entrys
- Tabellen-Puffer – „Single Record“: 10.000 KB / 500 Entrys

Dieses Szenario stellt den Fall ausreichend konfigurierter Systemressourcen dar. Damit wird evaluiert, ob das Modell die grundlegenden Abläufe abbilden kann.

Szenario2:

- Genutzte physikalische Cores: 3
- Tabellen-Puffer – „Generic Key“: 60.000 KB / 7500 Entrys
- Tabellen-Puffer – „Single Record“: 10.000 KB / 500 Entrys

In diesem Szenario, wird überprüft, inwieweit die Modellierung und Parametrisierung der CPU-Verwendung im Modell den tatsächlichen CPU-Bedarf simulieren kann. Des Weiteren dient dieses Szenario der Evaluation des Systemverhaltens bei zu gering dimensionierter CPU-Kapazität.

Szenario3:

- Genutzte physikalische Cores: 6
- Tabellen-Puffer – „Generic Key“: 30.000 KB / 3750 Entrys
- Tabellen-Puffer – „Single Record“: 5.000 KB / 250 Entrys

In diesem Szenario stehen bei einer Halbierung der verfügbaren Tabellenpuffer ausreichend CPU-Ressourcen zur Verfügung. Mit dieser Konfiguration soll die Modellierung der Verdrängung von Objekten aus den Tabellenpuffern und die daraus resultierenden zusätzlichen Anfragen auf die Datenbank evaluiert werden.

Die Ausprägung dieser Szenarien wurde so gewählt, um die Eigenschaften der Performance eines SAP-Systems aufzuzeigen und damit die Evaluation des vorgestellten Modells bzgl. dieser Charakteristika zu ermöglichen.

6.3.2.1. Einschwingverhalten

Durch die, dieser Arbeit zu Grunde liegenden, Performance-Analyse des SAP-Systems wurde festgestellt, dass sich die Performance des Systems in Abhängigkeit von der Anzahl der durchgeführten Wiederholungen des Workloads verändert. Dieser Effekt wird in der Literatur als „Einschwingverhalten“ bezeichnet (Meyer/Guicking 1974). Die Auswirkungen dieses Effekts zeigen sich in erhöhten Antwortzeiten zu Beginn einer Testserie. Abbildung 6-8 zeigt die Auswirkungen anhand der Gesamtantwortzeit in Abhängigkeit der durchgeführten Wiederholungen des Workloads nach einen Systemneustart. Nach etwa 3 Wiederholungen

des Geschäftsprozesses zeigt dieser eine konstante Tendenz, nachdem für die erste Ausführung ca. die doppelte Zeit benötigt wurde.

Wird ein SAP-System neu gestartet, so sind zunächst die Puffer nicht mit den für die Ausführung des Workloads benötigten Daten gefüllt. Aufgrund der bereits in 4.2.2.2 beschriebenen Mechanismen sind einige Ausführungen des Workloads nötig, damit alle Puffer gefüllt sind und das System seine maximale Performance leisten kann. Da Puffer auch auf der Datenbank-, Dateisystem- und Betriebssystemebene existieren, kann die Anzahl der für den Einschwingvorgang benötigten Wiederholungen nicht aus der Analyse des SAP-Systems abgeleitet werden.

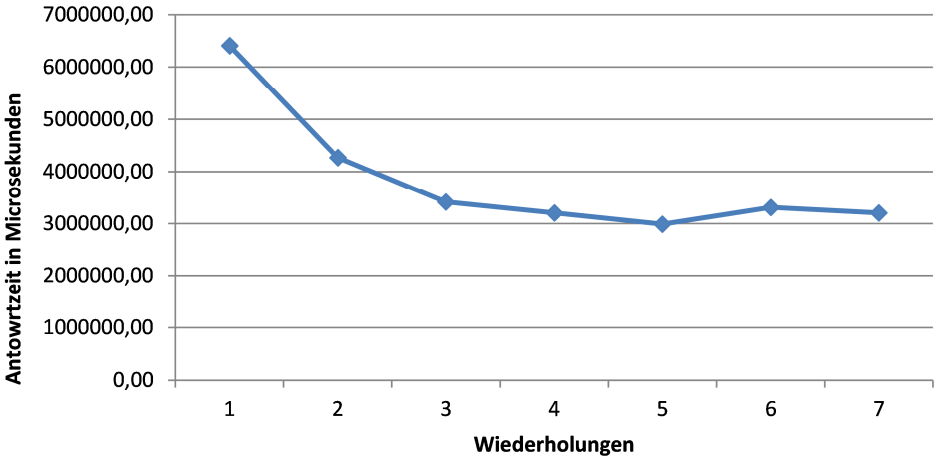


Abbildung 6-8: Einschwingverhalten des Systems anhand der Antwortzeit in Abhängigkeit der Anzahl der Wiederholungen
Quelle: Eigene Darstellung

Nach Barham et al. (2003) werden bei komplexen Systemen, wie einem ERP System, ca. 3 Wiederholungen benötigt, um die negativen Auswirkungen des Einschwingvorgangs auf die Performance zu eliminieren. Abbildung 6-8 bestätigt diese Aussage zwar, jedoch kann nicht grundsätzlich davon ausgegangen werden. Je nach Komplexität und Beschaffenheit des Systems kann dieser Wert z.T. stark variieren und muss durch Messung für jedes System speziell identifiziert werden.

6.3.3. Szenario 1

Wie bereits dargestellt, stehen im ersten Szenario für den Workload ausreichende Systemressourcen zur Verfügung. Wie sich jedoch bei der Durchführung der Experimente zeigte, ist die Datenbank in diesem Szenario nicht korrekt auf diesen Workload ausgelegt und zeigt ein stark lastabhängiges Verhalten. In den späteren Szenarien wurde diese Konfigurationsschwäche korrigiert. Da sich diese Konstellation jedoch sehr gut für die Evaluation der Kapselung von Datenbankabfragen innerhalb von Sperrobjekten eignet, wurden die Experimente nicht wiederholt, sondern das Szenario so belassen.

6.3.3.1. Antwortzeiten

Im Gegensatz zu den erwarteten Ergebnissen zeigt Abbildung 6-9 eine fast konstant ansteigende Entwicklung der Antwortzeit. In dieser Abbildung wurde der Median aus den Messungen der Antwortzeit in Abhängigkeit von der Anzahl paralleler Benutzer aufgetragen.

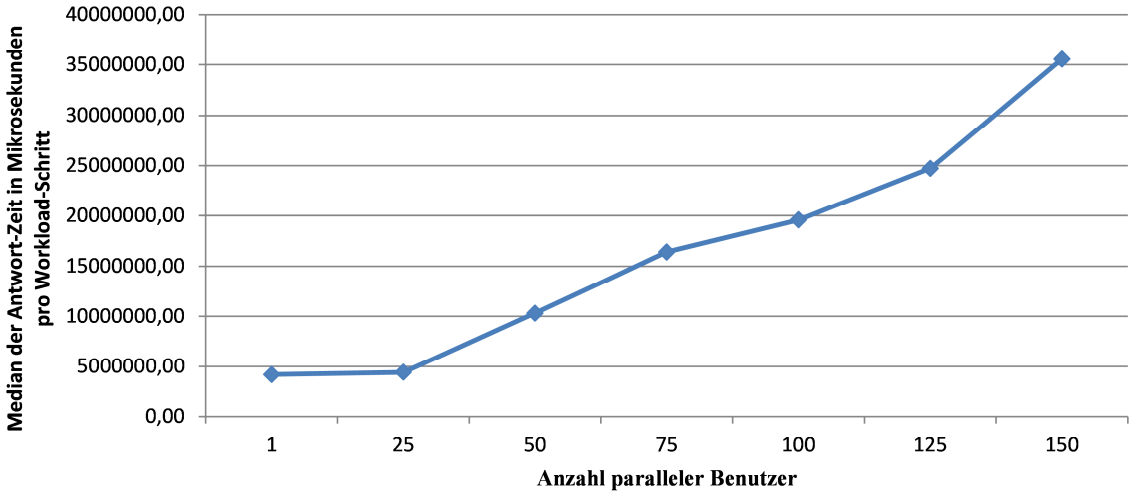


Abbildung 6-9: Antwortzeiten in Mikrosekunden in Abhängigkeit der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Die Entwicklung der Antwortzeit zeigt ab einer Anzahl von 25 parallelen Benutzern stark an. Diese Steigerungsrate wird dann tendenziell bis zu der maximal getesteten Menge an parallelen Benutzern beibehalten. Die Entwicklung der einzelnen für dieses unerwartete Ergebnis verantwortlichen Komponentenzeiten wird nun in den folgenden Abschnitten vorgestellt.

6.3.3.2. Pufferzugriffe

Die Qualität der Puffer und die daraus resultierenden Anfragen stellen eine wichtige Einflussgröße auf die Antwortzeit dar. Aus dem in Abschnitt 6.3.3.1 dargestellten Verhalten der Antwortzeit und der Konfiguration des SAP-WebAS-ABAP in diesem Szenario sollte sich eine nahezu konstante Anzahl an Pufferzugriffen ergeben. Abbildung 6-10 beweist diese Annahme. Da lediglich die Antwortzeit der Datenbank ein lastabhängiges Verhalten aufweist, ergeben sich daraus keine Auswirkungen auf die Anzahl der Pufferzugriffe. Diese sind unabhängig von der Anzahl paralleler Benutzer nahezu konstant.

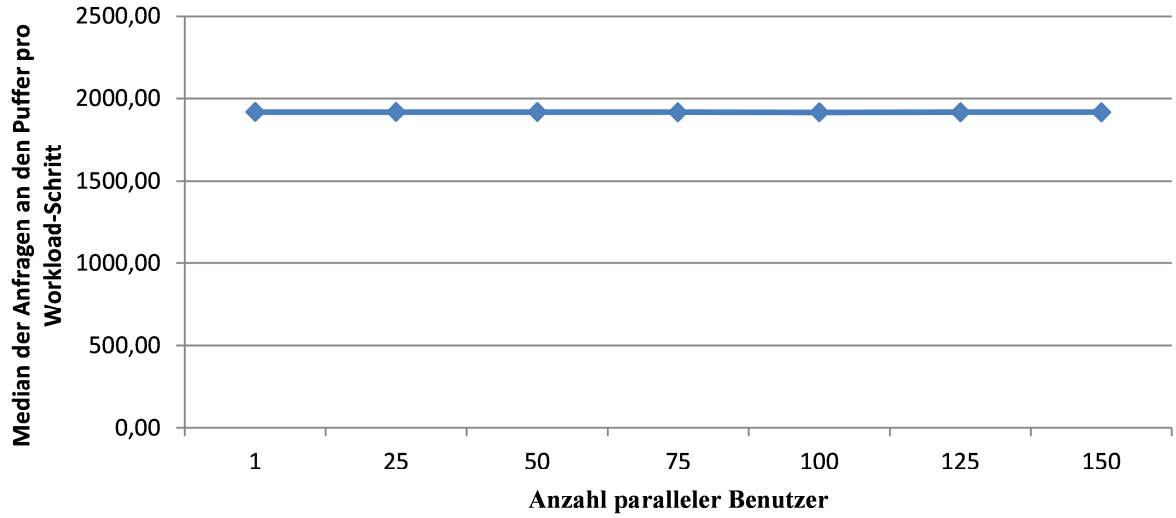


Abbildung 6-10: Anzahl der Pufferzugriffe in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

6.3.3.3. Datenbankanfragen

Wie bereits erwähnt, zeigt die Datenbank in diesem Szenario ein von der Last abhängiges Verhalten. Die, in Abbildung 6-11 dargestellte, Anzahl von Datenbankanfragen pro Workload-Schritt ist dabei keine Ursache für eine Veränderung des Antwortzeitverhaltens der Datenbank.

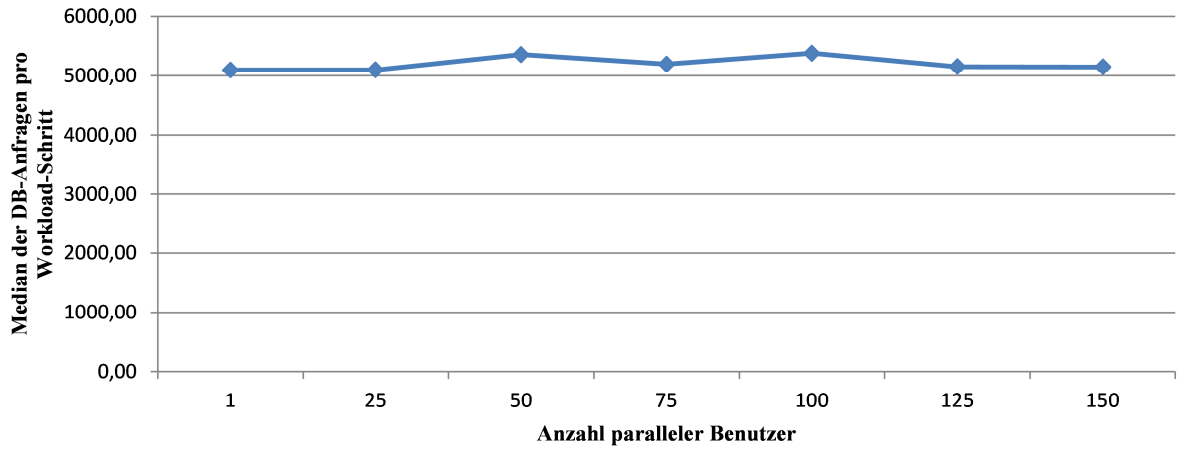


Abbildung 6-11: Median der Datenbankanfragen pro Aufruf in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Unabhängig von der verschiedenen Anzahl paralleler Benutzer bleibt diese nahezu konstant. Die Anzahl der Zugriffe auf die Datenbank steigt also nur aufgrund der wachsenden Anzahl von zu prozessierenden ABAP-Programmen, welche durch die steigende Anzahl von parallelen Benutzern verursacht wird.

Diese Steigerung der Last führt, wie in Abbildung 6-12 dargestellt, zu einer stark ansteigenden Antwortzeit der Datenbank pro Anfragen innerhalb eines Workload-Schrittes.

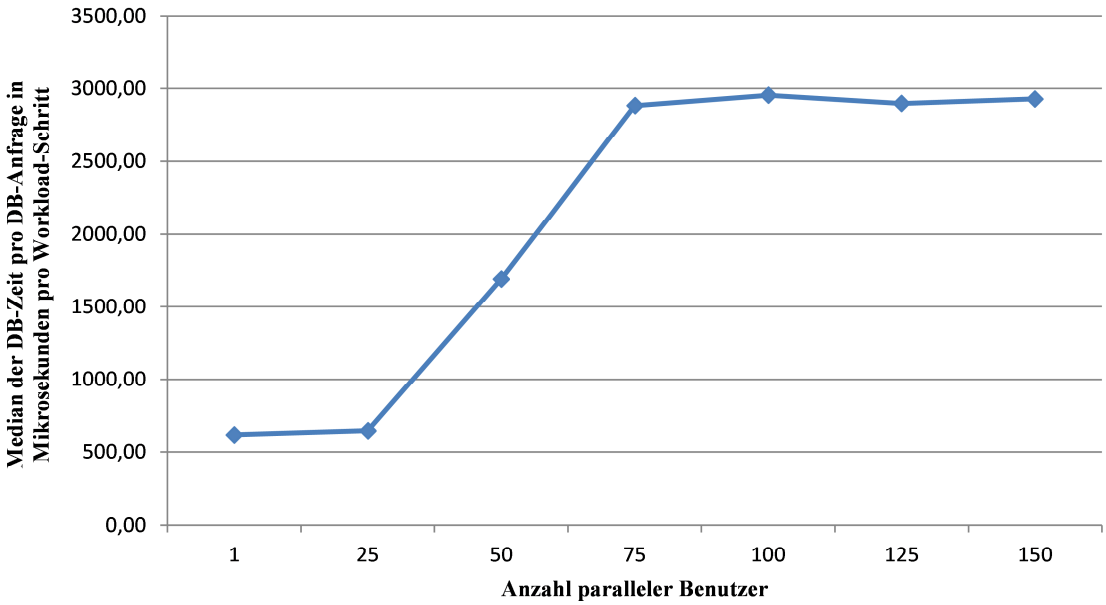


Abbildung 6-12: Median der Zeit pro Datenbankaufruf in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Ab einer Anzahl von 25 Benutzern steigt die Antwortzeit bis zu 75 parallelen Benutzer um ca. 600 Prozent, um dann wieder, wie erwartet, einen nahezu konstanten Verlauf aufzuzeigen.

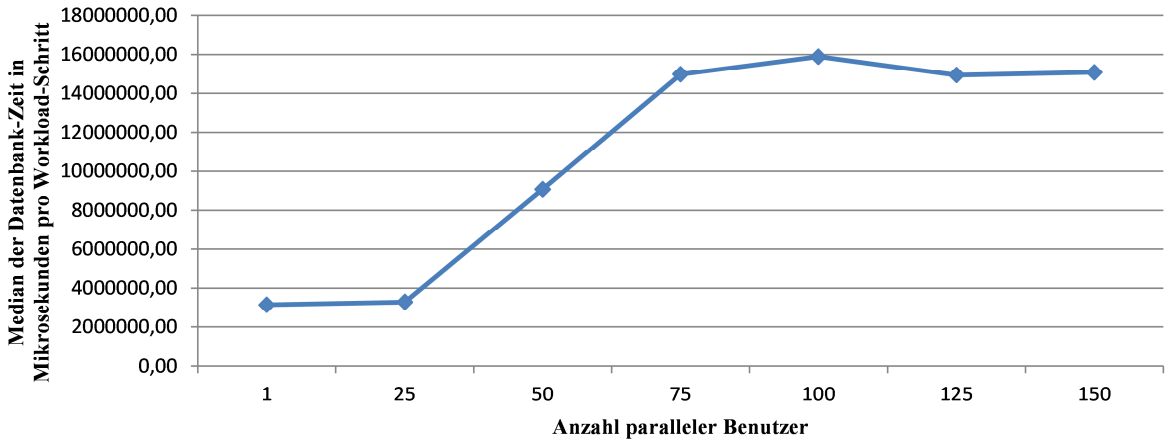


Abbildung 6-13: Median der DB-Zeit pro Workload-Schritt in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Der Anstieg der Antwortzeit pro Datenbankanfrage spiegelt sich direkt proportional in dem, in Abbildung 6-13 dargestellten, Median der Gesamtdatenbankzeit pro Workload-Schritt wieder.

6.3.3.4. Enqueue-Verhalten

Wie bereits dargestellt, sind Datenbankanfragen innerhalb von Sperrobjekten, welche durch den Enqueue-Server verwaltet werden, gekapselt.

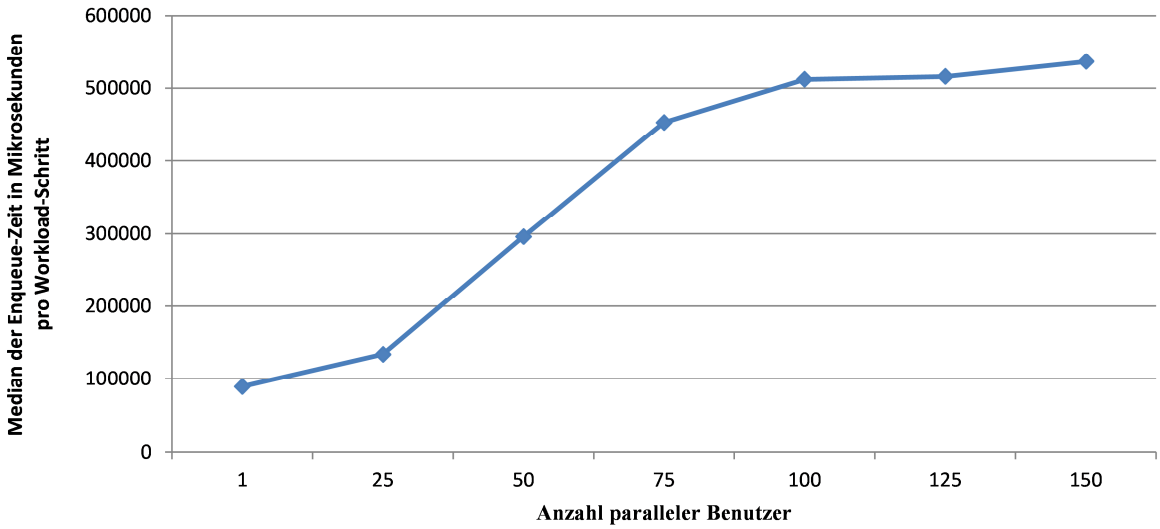


Abbildung 6-14: Median der Wartezeit in Mikrosekunden pro Work-Prozess in Abhängigkeit der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Diese Aussage wird durch das Diagramm in Abbildung 6-14 und der Entwicklung der Datenbankzeiten in Abbildung 6-11 untermauert. Die Enqueue-Zeiten steigen ebenfalls ab 25 parallelen Benutzern überdurchschnittlich an. Diese starke Steigerungsrate sinkt ab ca. 75 parallelen Benutzern deutlich. Ab 100 parallelen Benutzern wird die Enqueue-Zeit, analog zu den bereits dargestellten Messergebnissen der Datenbankzeiten, wieder nahezu konstant.

Dies hängt auch mit dem Erreichen des maximalen Durchsatzes des Systems zusammen. Da es 90 Dialog-Work-Prozesse enthält, sind ab etwas über 90 parallelen Benutzern alle Work-Prozesse belegt. Trotz konstanter Datenbankzeiten ab ca. 75 parallelen Benutzern steigt die Enqueue-Zeit bis zum Erreichen der 100 Benutzergrenze nochmals leicht an. Dieser Anstieg ist dem Enqueue-Verhalten geschuldet, welches auch eine Steigerung zwischen einem und 25 parallelen Benutzern verursacht. Ab 100 parallelen Benutzern ist dann die Enqueue-Zeit ebenfalls konstant, da das Queueing stark von der Anzahl der parallel bearbeiteten Anfragen abhängt. Diese Anzahl ist erreicht, wenn alle Dialog-Work-Prozesse einer Instanz ausgelastet sind. Aufgrund der in diesem Szenario konfigurierten Anzahl von Work-Prozessen ist die Instanz bei ca. 100 parallelen Benutzern vollständig ausgelastet und damit die maximale Ausprägung der Queueing-Zeit erreicht. Leichte Schwankungen in der gemessenen Queueing-Zeit ergeben sich aus der Tatsache, dass die Verwendung unterschiedlicher Sperrobjekte ebenfalls Auswirkungen auf die Queueing-Zeit hat.

6.3.3.5. Dispatcher

Trotz der in diesem Szenario stark ansteigenden Datenbankzeit zeigt der Anteil der Wartezeit in der Queue des Dispatchers in diesem Bereich keine starken Veränderungen. Wie in Abbildung 6-15 dargestellt, wird durch das Erreichen der maximalen Kapazität bei ca. 100 Benutzern dieser Anteil der Antwortzeit stark vergrößert.

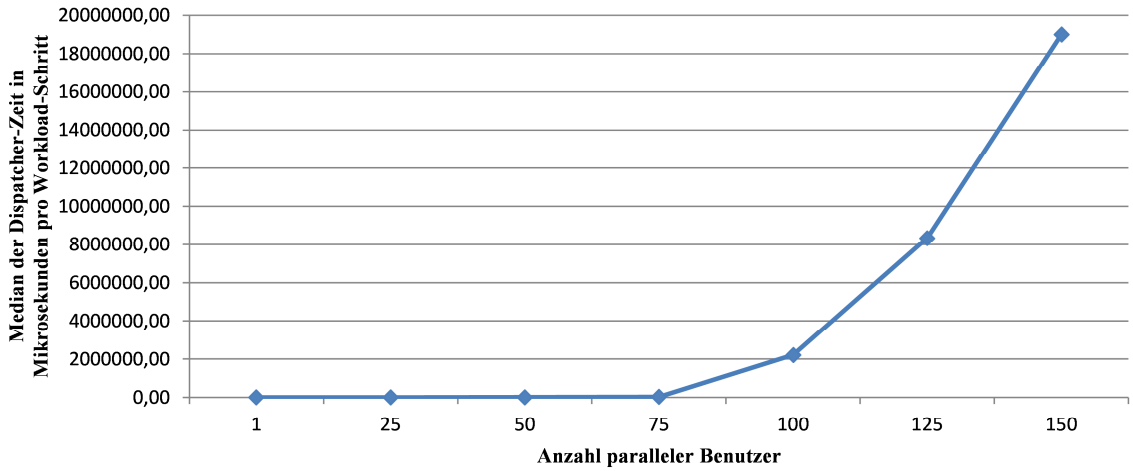


Abbildung 6-15: Median der Wartezeit in der Dispatcher Queue pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

6.3.3.6. CPU Zeiten

Die in der Beschreibung des Szenarios getroffene Aussage von ausreichenden CPU-Ressourcen belegt Abbildung 6-16.

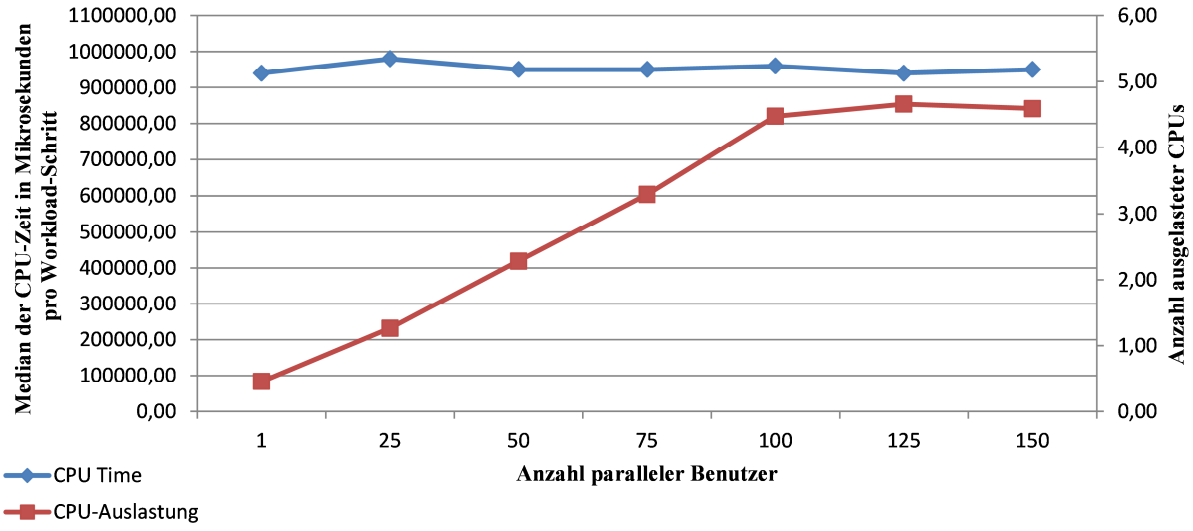


Abbildung 6-16: Median der benötigten CPU Zeit pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

In diesem Diagramm sind sowohl die nahezu konstante DB-Zeit pro Workload-Schritt, als auch die Anzahl der durchschnittlich verwendeten Prozessoren aufgezeichnet. Dies zeigt deutlich, dass die zur Verfügung stehenden sechs CPUs nicht vollständig benötigt werden und damit kein CPU-Engpass vorhanden ist.

6.3.4. Szenario 2

Das zweite Szenario stellt die Situation eines SAP-ERP-Systems mit nicht ausreichend vorhandenen CPU-Ressourcen dar. Dieses Szenario wurde gewählt, um die Modellierung des CPU-Verbrauchs zu evaluieren und daraus mit Hilfe der Simulation die benötigte Anzahl von CPU-Ressourcen zu bestimmen. Im Gegensatz zum ersten Szenario wurde hier die Konfiguration der Datenbank an den Workload angepasst, sodass deren Verhalten nicht von der Last abhängig ist.

6.3.4.1. CPU-Zeiten

In diesem Szenario stehen anstelle der benötigten 4,75 CPUs (6.3.3.6.) 3 CPUs zur Verfügung.

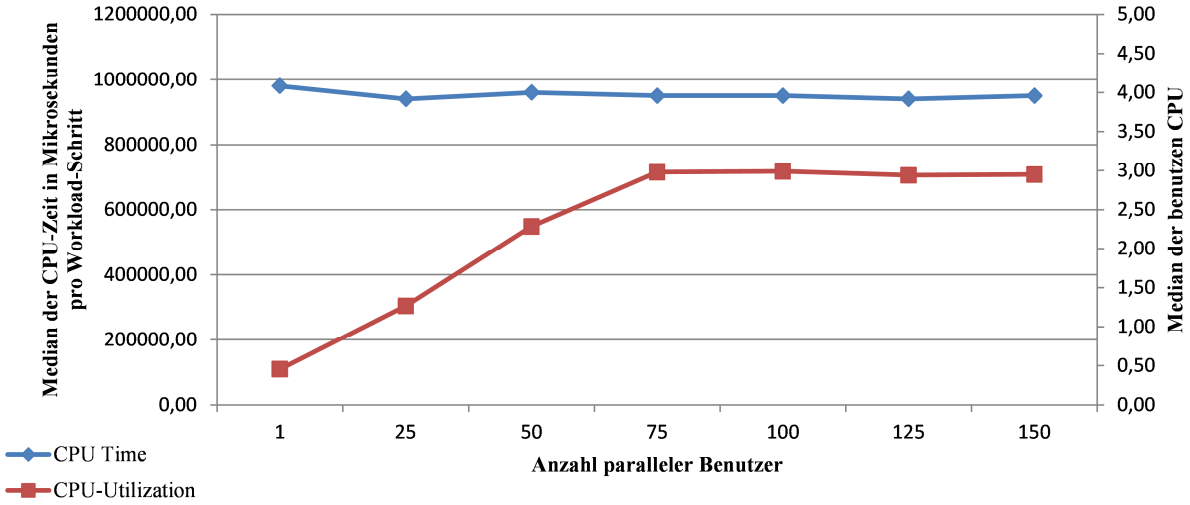


Abbildung 6-17: Median der benötigten CPU Zeit pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Abbildung 6-17 zeigt den Durchschnitt der benötigten CPU-Zeit pro Workload-Schritt in Mikrosekunden. Dieser ist unabhängig von der Anzahl der konfigurierten CPUs, wie aus dem Vergleich der Messungen der CPU-Zeit der anderen Szenarien (Abbildung 6-16 und Abbildung 6-32) hervorgeht. Die zweite Kurve in dieser Abbildung zeigt den Median des gemessenen CPU-Verbrauchs. Dieser entwickelt sich konstant steigend, bis die Anzahl der verfügbaren CPUs bei ca. 75 parallelen Benutzern erreicht ist. Aufgrund der leicht gesenkten Steigungsrate zwischen 50 und 75 parallelen Benutzern ist darauf zu schließen, dass der Bedarf bereits bei 75 parallelen Benutzern etwas höher ist als CPU-Ressourcen zur Verfügung stehen.

6.3.4.2. Antwortzeiten

Aufgrund des im vorangegangenen Abschnitt beschriebenen Verhaltens des CPU-Verbrauchs sollte die Antwortzeit für bis zu 50 parallelen Benutzern nicht durch die CPU-Konfiguration beeinflusst sein.

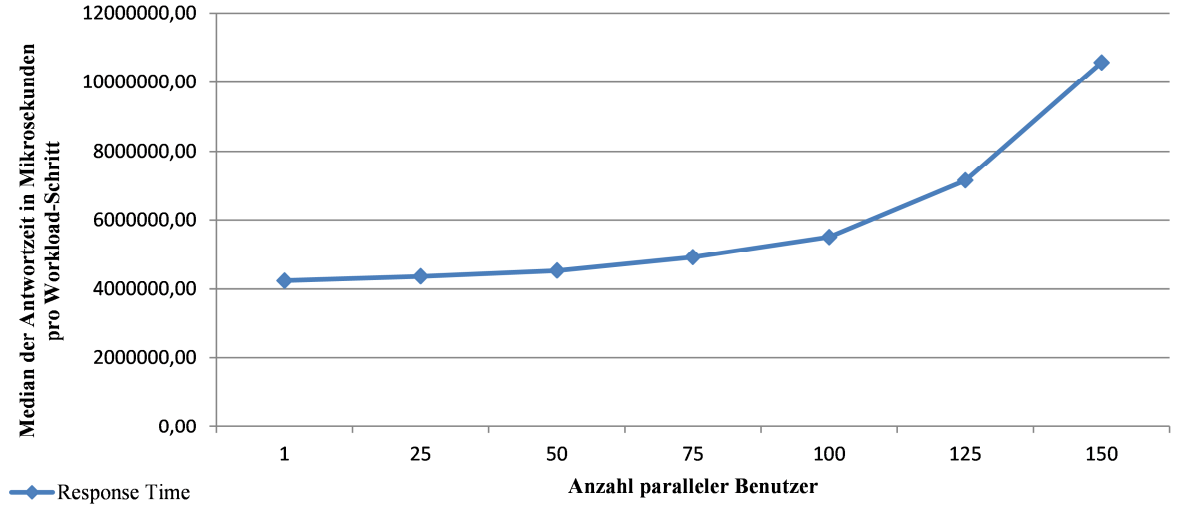


Abbildung 6-18: Median der Antwortzeit pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Wie in Abbildung 6-18 dargestellt ist, zeigt die Gesamtantwortzeit das prognostizierte Verhalten. Die zuvor konstante Steigerungsrate erhöht sich leicht zwischen 50 und 75 parallelen Benutzern, da 75 parallele Benutzer bereits etwas mehr an CPU-Ressourcen benötigen als vorhanden sind. Dies setzt sich fort, bis die maximale Kapazität des Systems erreicht ist und steigt dann aufgrund von Ressourcen-Engpässen und Dispatching-Zeit stark an.

6.3.4.3. Pufferzugriffe

Da in diesem Szenario keine Änderungen an der Konfiguration der Puffer vorgenommen wurden, sollten diese von der Verknappung der CPU-Ressourcen nicht betroffen sein.

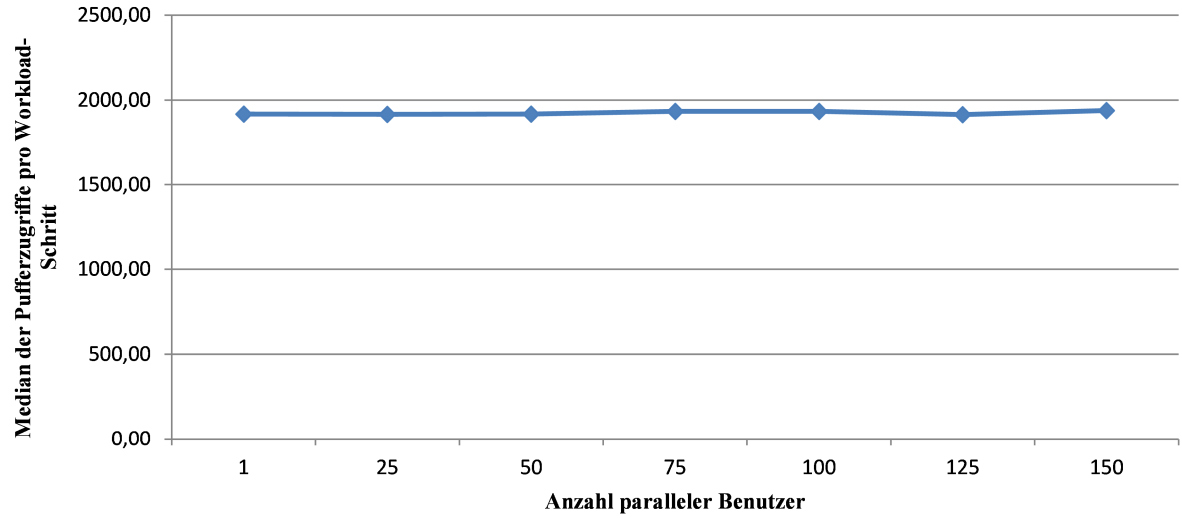


Abbildung 6-19: Median der Pufferzugriffe pro Aufruf in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Die Unabhängigkeit der Pufferzugriffe von den verfügbaren CPU-Ressourcen ist in Abbildung 6-19 dargestellt. Unabhängig von der Anzahl paralleler Benutzer und der Anzahl verfügbarer CPUs bleibt die Anzahl der Zugriffe auf den Puffer nahezu konstant.

6.3.4.4. Enqueue-Verhalten

Analog zu der Betrachtung der Pufferzugriffe 6.3.4.3. sollte auch die Entwicklung der Enqueue-Zeiten über die steigende Anzahl von parallelen Benutzern keine Auswirkungen der CPU-Knappheit aufweisen, da diese hauptsächlich Update-Anweisungen an die Datenbank synchronisieren. Diese Anfragen an die Datenbank sind aufgrund der in der vorliegenden Arbeit gewählten, verteilten Installation nicht von der CPU-Knappheit betroffen, da SAP-WebAS-ABAP und die Datenbank auf verschiedenen logischen Partitionen installiert sind.

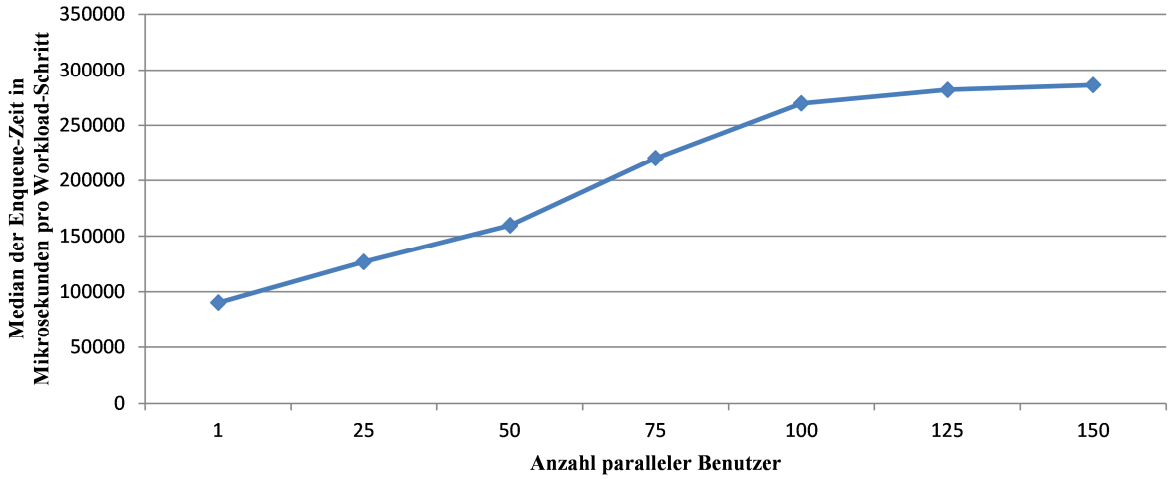


Abbildung 6-20: Median der Wartezeit im Enqueue pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Die Unabhängigkeit der Enqueue-Zeiten von der CPU-Knappheit zeigt Abbildung 6-20. Diese wächst mit konstanter Steigerungsrate, bis das maximale Queueing, vgl. 6.3.3.4. erreicht ist. Da der Enqueue-Server an sich CPU-Ressourcen benötigt, folgt aus dem Ergebnis der Messung in Abbildung 6-20, dass der Hauptanteil der Enqueue-Zeit durch das Warten auf eine Sperre verursacht wird, was keine CPU-Ressourcen benötigt. Das wirkliche Setzen der Sperre durch den Enqueue-Server-Prozess benötigt Rechenleistung und ist von der CPU-Knappheit ab 75 parallelen Benutzern betroffen. Eine detaillierte Analyse der Enqueue-Vorgänge mit Hilfe des Performance-Traces (vgl. 4.2.6) zeigt die minimale Ausführungszeit des Setzens einer Sperre und damit analog zu Abbildung 6-20 den geringen Anteil des Enqueue-Servers an der Enqueue-Zeit.

6.3.4.5. Datenbankanfragen

Wie bereits dargestellt, ist die Datenbank auf einer separaten LPAR installiert und damit nicht direkt von der Verminderung der zur Verfügung stehenden CPU-Ressourcen betroffen. Um indirekte Belastungen zu untersuchen, werden im Folgenden relevante Kennzahlen für die Beurteilung der Datenbank-Performance dargestellt und analysiert.

Abbildung 6-21 zeigt dabei die durchschnittliche Zeit von Datenbank-Anfragen pro Workload-Schritt auf. Diese beinhaltet keine Einflüsse der CPU-Knappheit.

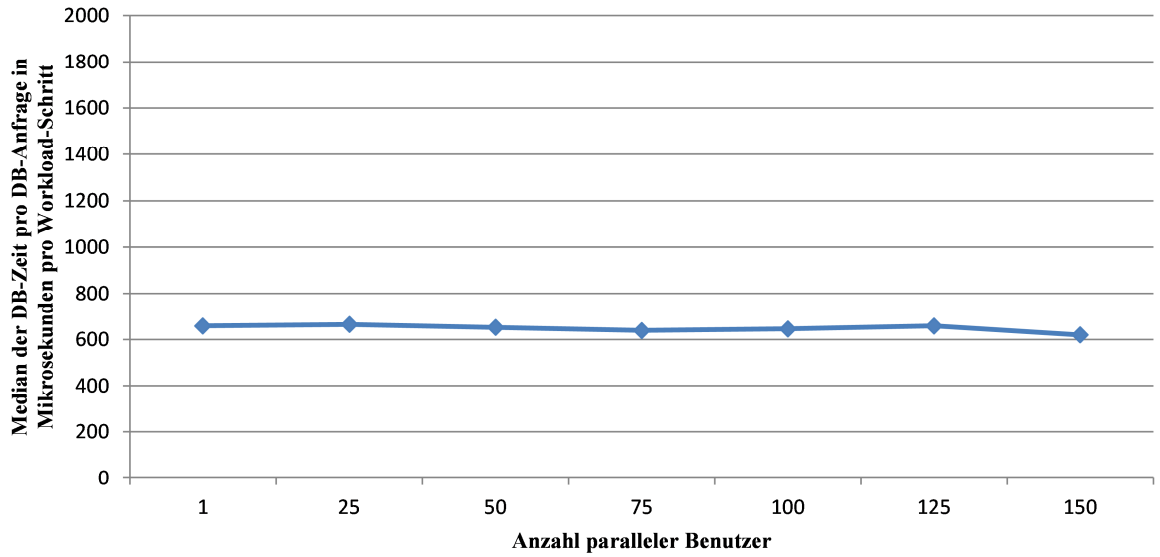


Abbildung 6-21: Median der Datenbankzeit pro Datenbankanfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Analog zu der Antwortzeit der Datenbank werden durch die Reduzierung der CPUs keine Einflüsse auf die Anzahl der Datenbankzugriffe (Abbildung 6-22) erzeugt.

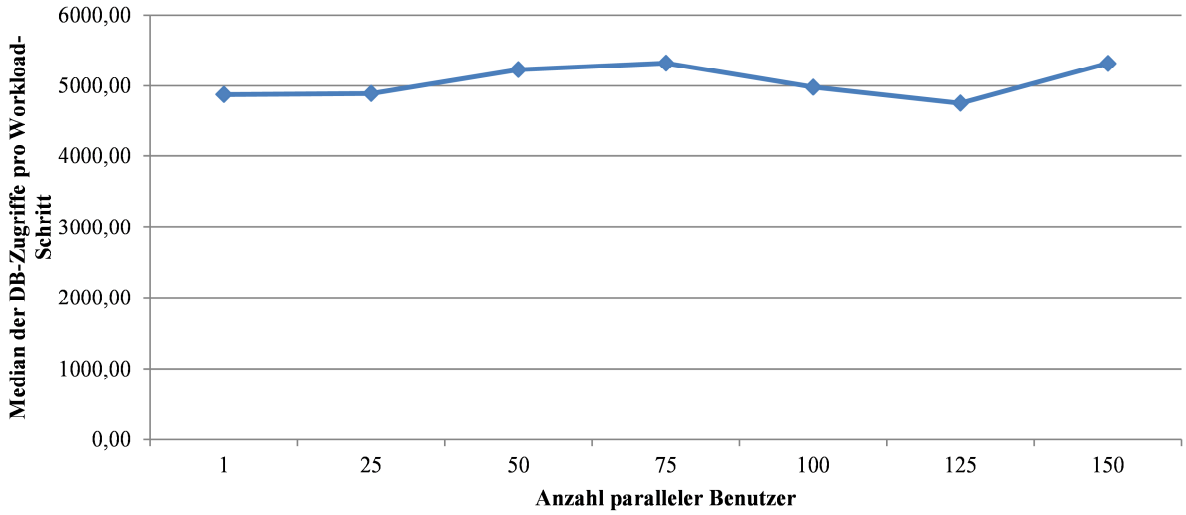


Abbildung 6-22: Median der Anzahl von Datenbankanfragen pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Die gesamte Datenbankzeit pro Workload-Schritt zeigt, wie in Abbildung 6-23 dargestellt, zwischen 25 und 75 parallelen Benutzern einen leichten Anstieg. Da dieser Wert jedoch sehr nahe an Werten unter CPU-Knappheit (150 parallele Benutzer) und mit ausreichenden Ressourcen (25 parallele Benutzer) liegt kann aus Abbildung 6-23 keine indirekte Beeinflussung der Datenbankzeit eines Workload-Schritts von der CPU abgeleitet werden.

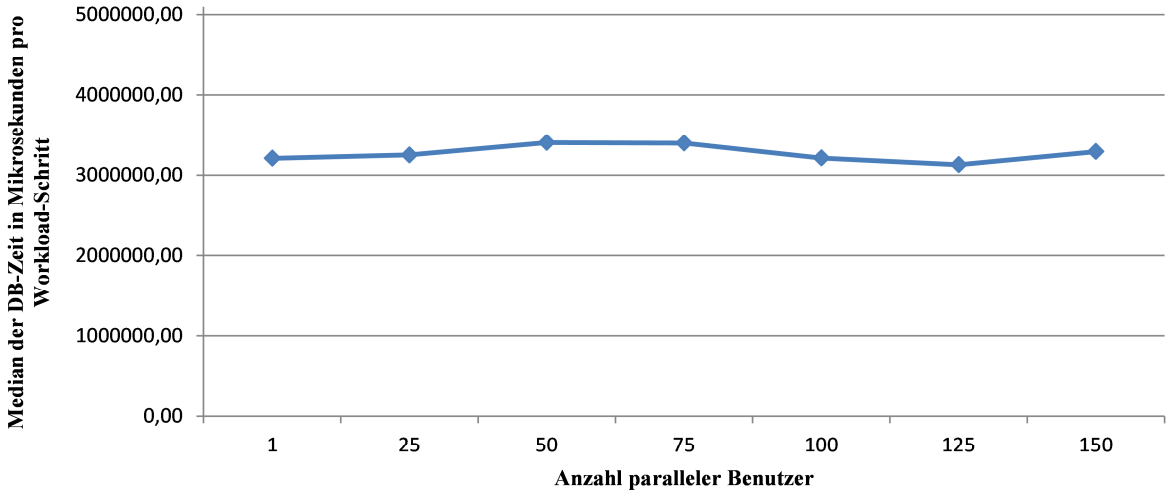


Abbildung 6-23: Median der DB-Zeit pro Workload-Schritt in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

6.3.4.6. Dispatcher

Der Anteil der Antwortzeit, welcher durch die Wartezeit in der Queue des Dispatcher entsteht, sollte bis zum Erreichen der vollen Auslastung der Work-Prozess-Kapazitäten keine Beeinflussung durch die CPU-Knappheit erfahren.

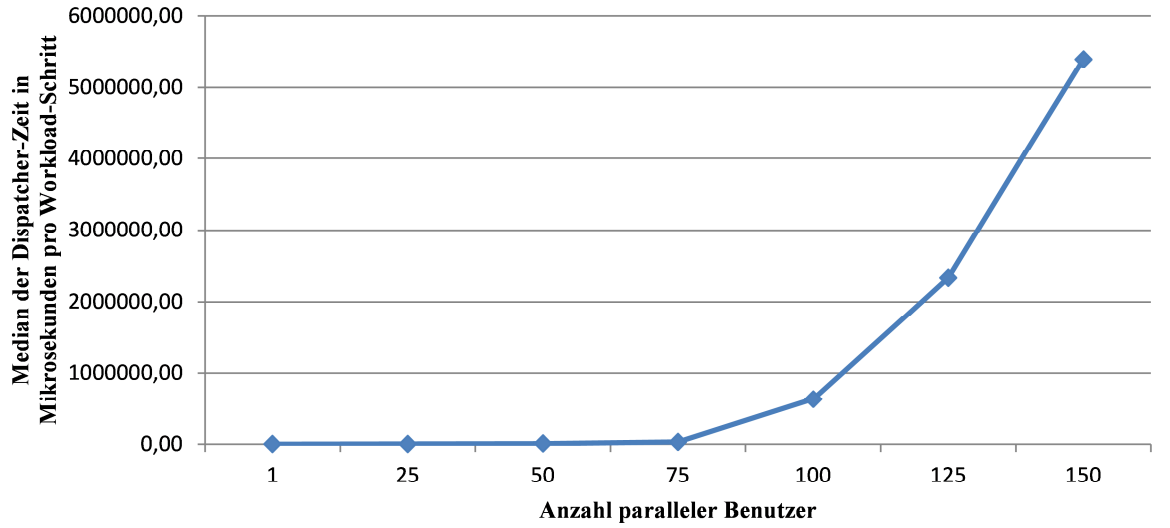


Abbildung 6-24: Median der Wartezeit im Dispatcher pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Jedoch steigt dieser Zeitanteil bereits bei 75 parallelen Benutzern leicht an. Da in diesem Bereich grundsätzlich kaum Dispatchern-Zeit anfällt, zeigt die CPU-Knappheit geringe Auswirkungen, welche aufgrund der minimalen Ausprägung der Dips-Waite-Time unter dieser Systemlast zu einer sichtbaren Beeinträchtigung führt. Die Ausprägung dieser Zeit unter Voll-Last des Systems zeigt jedoch die Auswirkung der CPU-Knappheit, welche zu

einer längeren Ausführungszeit der ABAP-Programme führt und damit zu einer längeren Wartezeit in der Queue des Dispatchers.

6.3.5. Szenario 3

Im dritten Szenario sind ausreichend CPU-Ressourcen konfiguriert. Um die Modellierung und dessen Vorhersagekraft bzgl. der Verwendung der Puffer und dessen Mechanismen zu evaluieren, wurde die Größe des Puffers halbiert. Damit steht nun die Hälfte an Speicher und Entrys zur Verfügung.

6.3.5.1. Datenbankanfragen

Um die Aussagekraft der Messergebnisse zu präzisieren, wurde bei der Konfiguration der Datenbank für dieses Szenario darauf geachtet, dass deren Antwortzeit trotz der höheren erwarteten Last nahezu konstant bleibt und keine lastabhängigen Abweichungen enthält.

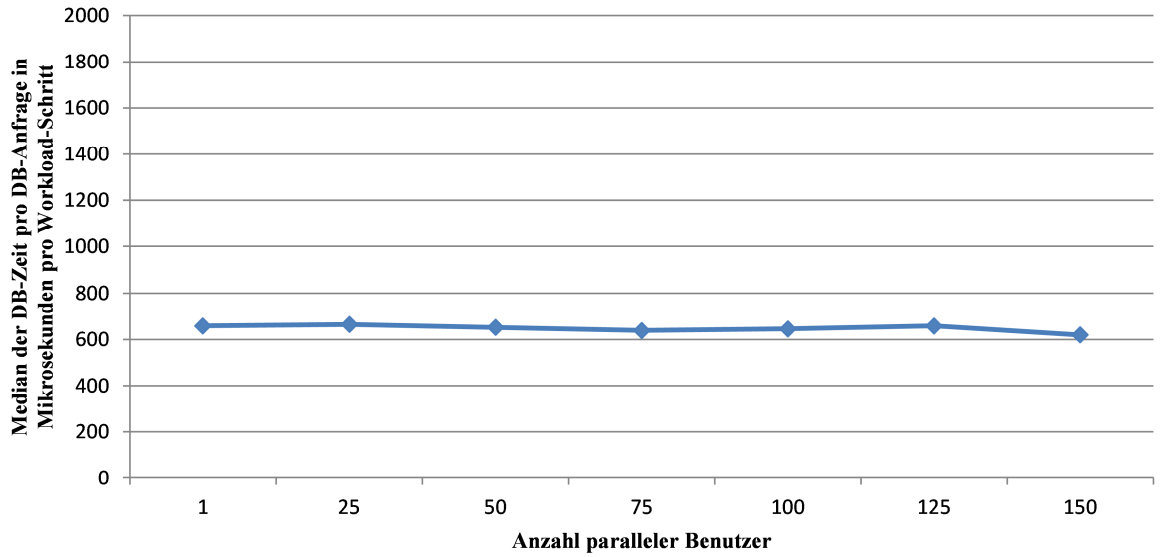


Abbildung 6-25: Median der Datenbankzeit pro Datenbankanfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Wie in Abbildung 6-25 dargestellt, zeigt die Datenbank, wie beabsichtigt, eine konstante Antwortzeit pro Anfrage in Abhängigkeit der steigenden Anzahl von parallelen Benutzern. Damit wird sichergestellt, dass die in diesem Szenario zu untersuchenden Auswirkungen von Pufferverdrängungen auf die Performance des SAP-WebAS-ABAP nur durch die im Modell abgebildete zusätzliche Anzahl von Datenbank-Anfragen, aufgrund von Verdrängungen gepufferter Objekte, verursacht wird. Damit fokussiert sich die Messung auf die in der vorliegenden Arbeit beabsichtigte Modellierung der Tabellenpuffer des SAP-WebAS-ABAP und vermindert den Einfluss der Antwortzeit der als Black-Box modellierten Datenbankschicht stark.

Da in diesem Szenario die Puffer zu gering dimensioniert sind, kommt es zu Verdrängungen von Inhalten aus den Puffern. Diese verdrängten Objekte müssen dann bei der nächsten ihnen geltenden Anfrage erneut von der Datenbank in die Puffer geladen werden. Dies verursacht zusätzliche Last auf die Datenbank in Form einer erhöhten Anzahl von Zugriffen.

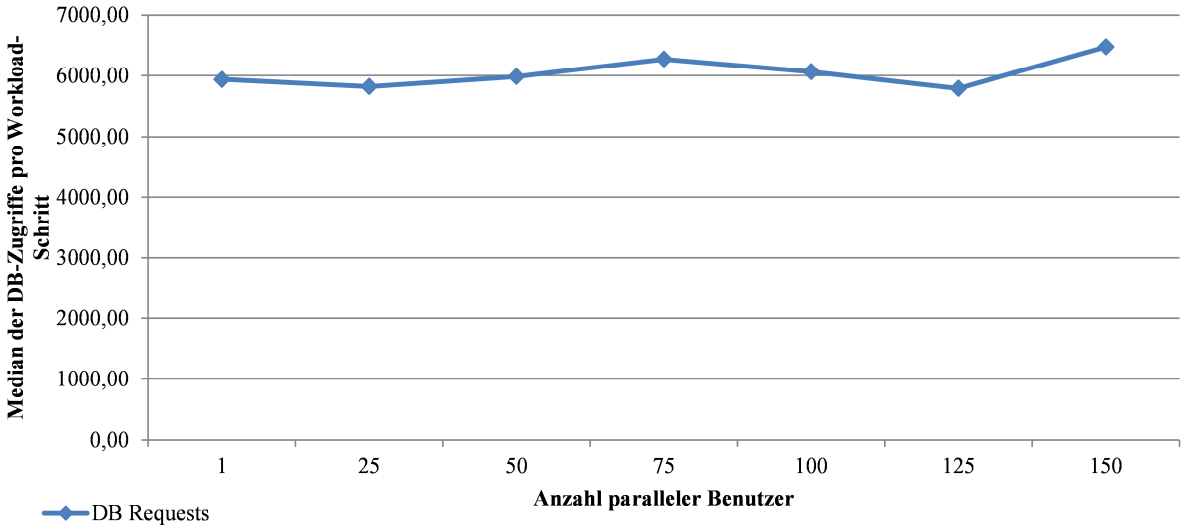


Abbildung 6-26: Median der Anzahl von Datenbankanfragen pro Workload-Schritt in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Ein Vergleich der Anzahl von Datenbankanfragen pro Workload-Schritt in diesem Szenario (Abbildung 6-26) mit der entsprechenden Anzahl aus den vorangegangenen Szenarien (Abbildung 6-22) zeigt eine konstante Erhöhung um ca. 20 Prozent, welche benötigt werden, um verdrängte Inhalte nachzuladen.

Trotz unveränderter Antwortzeiten der Datenbank steigt nun die Gesamtdatenbankzeit pro Workload-Schritt.

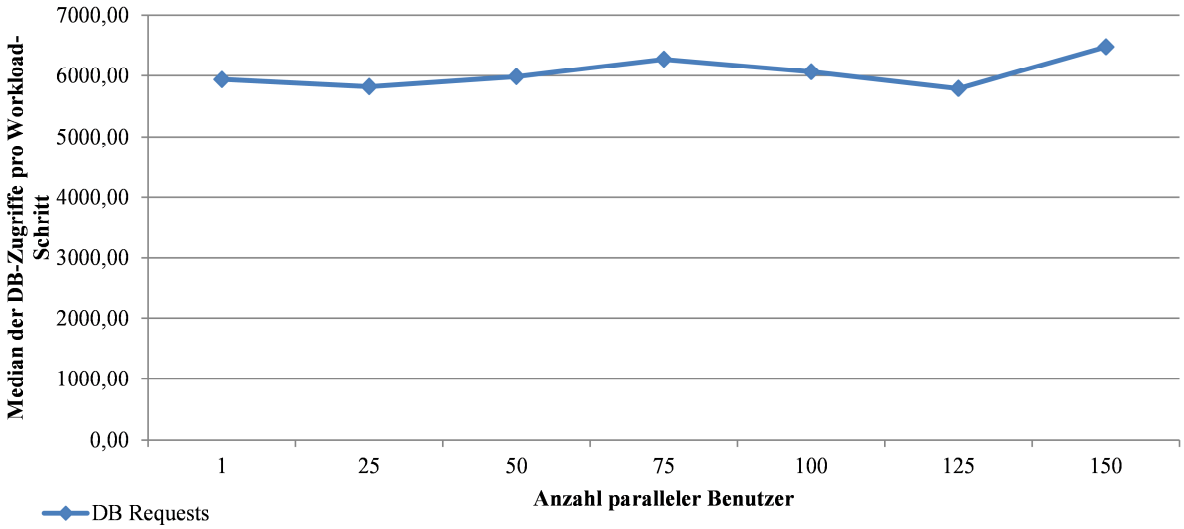


Abbildung 6-27: Median der DB-Zeit pro Workload-Schritt in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Abbildung 6-27 zeigt die Datenbankzeit pro Workload-Schritt in Abhängigkeit der Anzahl von parallelen Benutzern. Im Vergleich mit der Datenbankzeit bei ausreichenden Pufferressourcen (Abbildung 6-23) weist diese trotz unveränderter Antwortzeit pro Datenbankanfrage eine Erhöhung um das bereits ermittelte Verhältnis von 20 Prozent aufgrund von Pufferverdrängungen auf.

6.3.5.2. Antwortzeiten

Wie bereits erwähnt, führt die Erhöhung der Datenbankzeit pro Workload-Schritt zu einer Erhöhung der Gesamtantwortzeit des SAP-ERP-Systems.

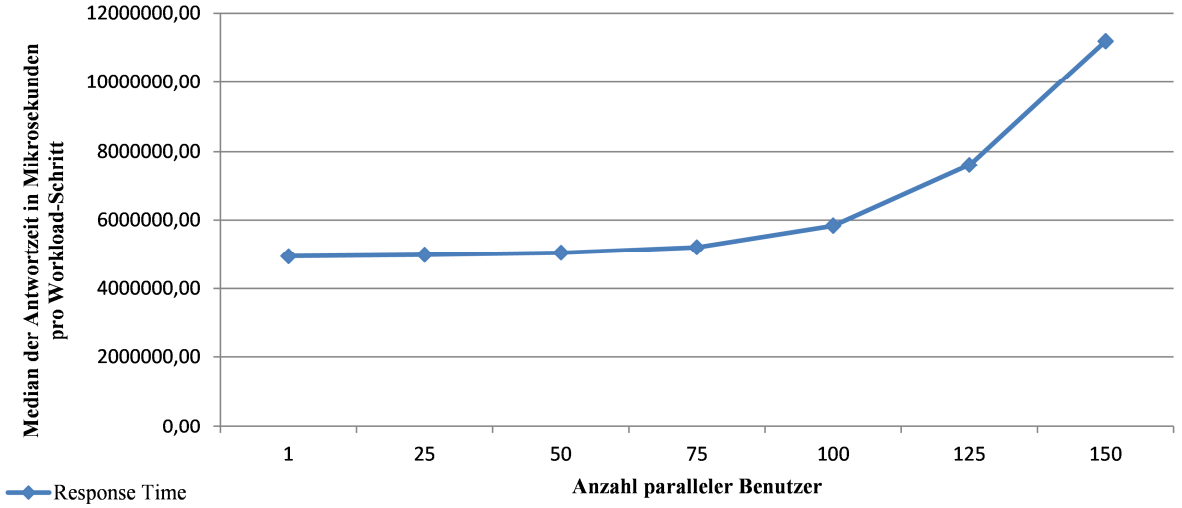


Abbildung 6-28: Median der Antwortzeit pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Ein Vergleich zwischen Abbildung 6-18 und Abbildung 6-28 zeigt die deutlich erhöhte Antwortzeit des SAP-WebAS-ABAP. Um andere Komponenten als Ursache ausschließen zu können, werden diese in den folgenden Abschnitten dargestellt.

6.3.5.3. Pufferzugriffe

Werden verdrängte Pufferinhalte durch ein ABAP-Programm angefragt, so wird der Inhalt von der Datenbank in den Puffer geladen und daraus beantwortet.

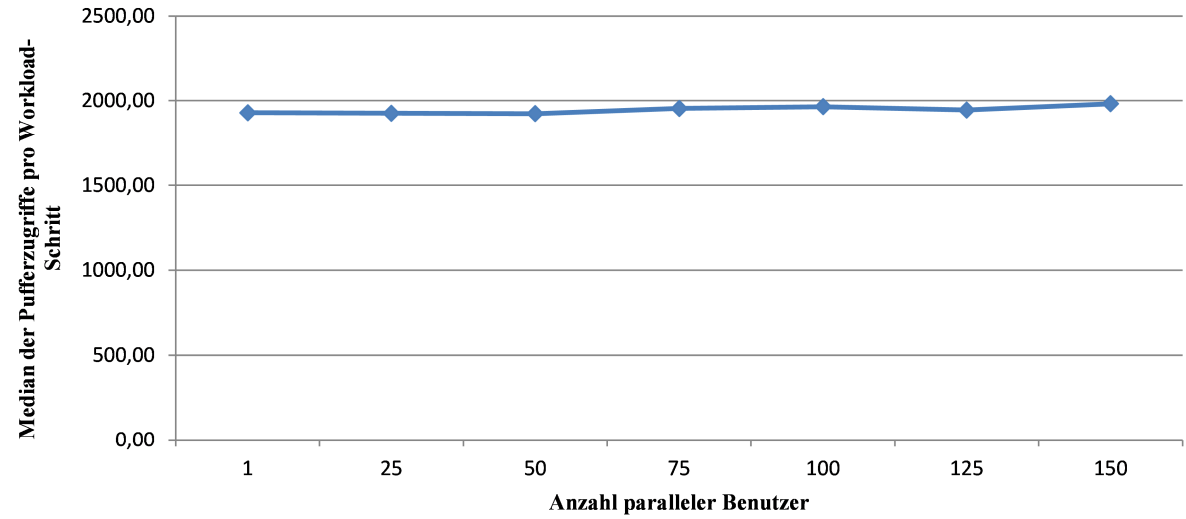


Abbildung 6-29: Median der Pufferzugriffe pro Aufruf in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Aus diesem Grund bleibt die in Abbildung 6-29 dargestellte Anzahl von Pufferzugriffen pro Workload-Schritt unverändert.

6.3.5.4. Enqueue-Verhalten

Ebenso zeigt das in Abbildung 6-30 dargestellte Verhalten der Enqueue-Zeiten keine Veränderung zu dem vorangegangenen Szenario mit ausreichenden Pufferressourcen.

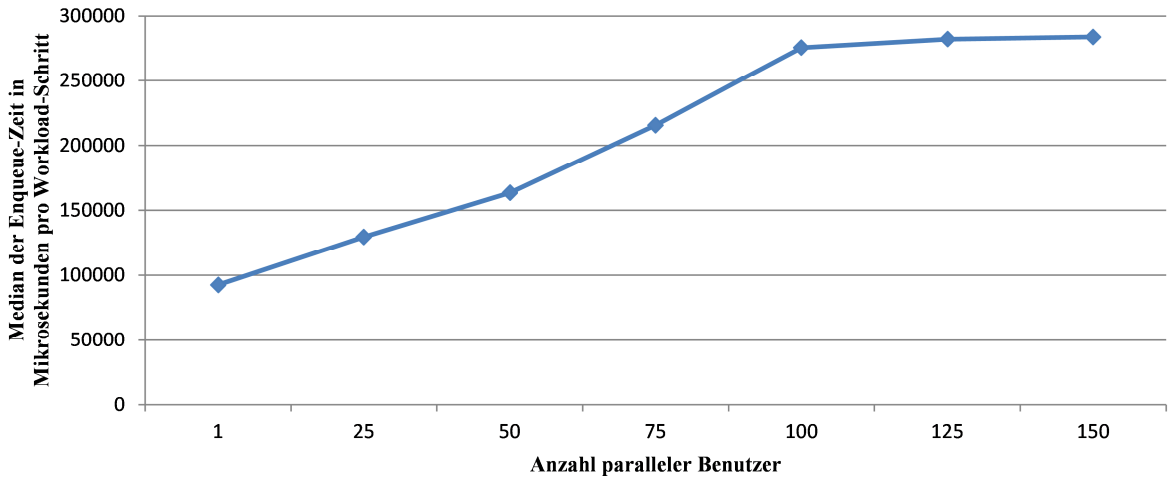


Abbildung 6-30: Median der Wartezeit im Enqueue pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

6.3.5.5. Dispatcher

Die Zeit pro Anfrage in der Queue des Dispatchers ist, analog zu Szenario zwei, bis zum Erreichen der maximalen Kapazität von Work-Prozessen unverändert und nicht von der Erhöhung der Gesamtantwortzeit beeinflusst.

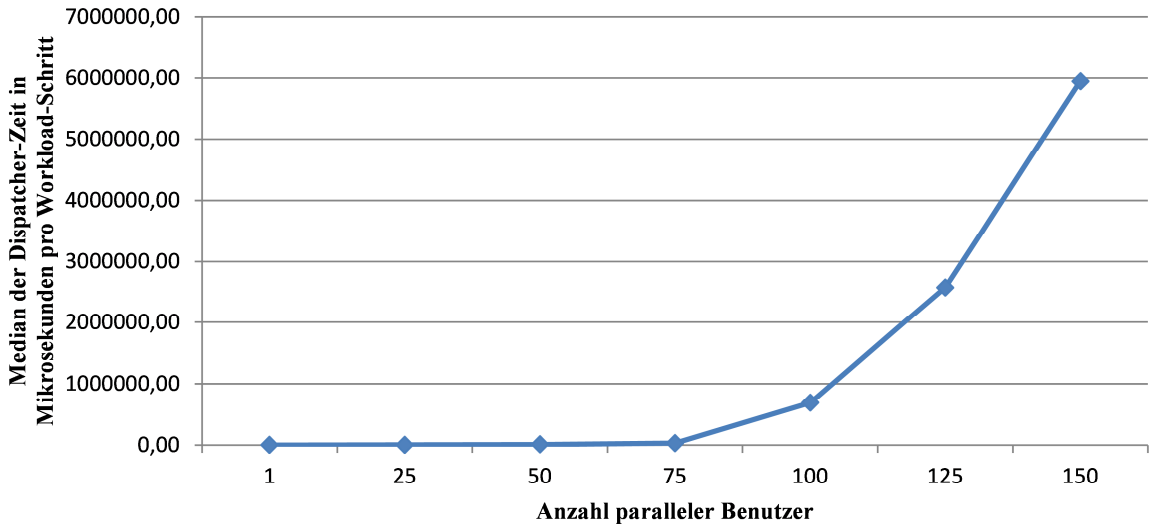


Abbildung 6-31: Median der Wartezeit im Dispatcher pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Der Vergleich von Abbildung 6-24 in Szenario 2 und der Entwicklung der Dispatcher-Zeit in diesem Szenario (Abbildung 6-31) beweist diese Vermutung. Trotz der erhöhten Datenbankzeit bleibt diese bis zum Erreichen der vollen Auslastung der verfügbaren Work-Prozesse unbeeinflusst. Ist diese Grenze jedoch erreicht, steigt die Antwortzeit überdurchschnittlich. Der Vergleich zeigt, dass der Einfluss durch die Pufferverdrängung unter diesem Workload etwas stärker ausgeprägt ist als der Einfluss durch die CPU-Knappheit.

6.3.5.6. CPU-Zeiten

Die erhöhte Anzahl von Datenbankzugriffen führt neben der Erhöhung der Gesamtantwortzeit auch zu erhöhten CPU-Bedarf. Da anstelle von Hauptspeicherzugriffen Anfragen an die Datenbank gestellt werden, muss dies zu einer erhöhten CPU-Nutzung führen.

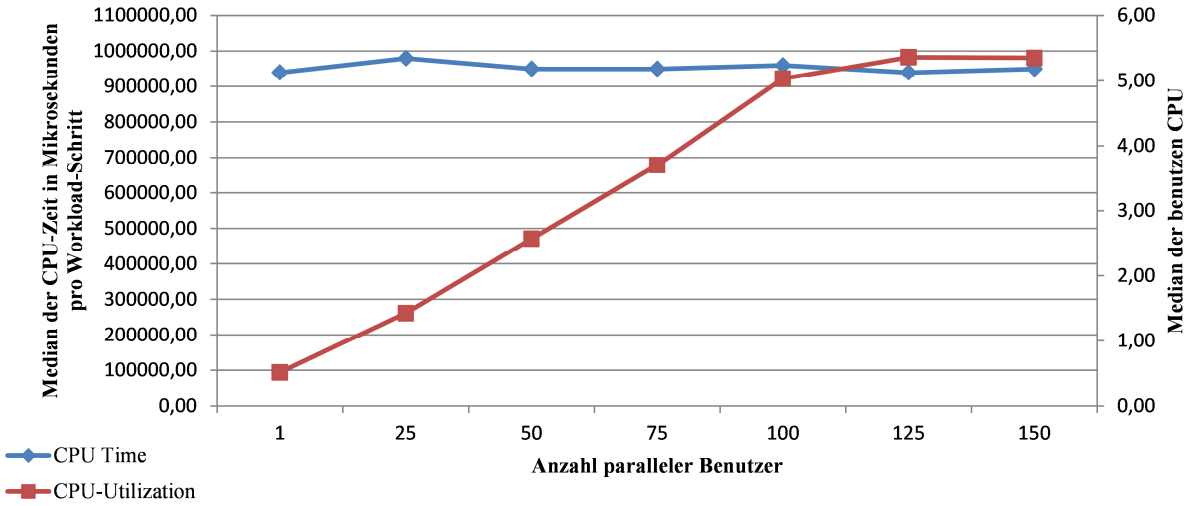


Abbildung 6-32: Median der benötigten CPU-Zeit pro Anfrage in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Abbildung 6-32 zeigt diese Auswirkungen auf die CPU-Nutzung. Im Vergleich zum ersten Szenario steigt der CPU-Bedarf. Da andere Komponenten kein abweichendes Verhalten aufzeigen, kann dieser gemessene Mehrverbrauch an CPU-Ressourcen der erhöhten Anzahl von Datenbankabfragen zugeordnet werden.

6.3.6. Vergleich der Messergebnisse mit bestehenden Arbeiten

Die Messergebnisse der vorliegenden Arbeit zeigen, dass die Antwortzeit bis zum Erreichen der Überlast-Grenze kontinuierlich mit der Anzahl der parallelen Benutzer auf dem System steigt. Ist diese Grenze überschritten, erhöht sich die Steigung der Antwortzeit-Kurve und zeigt schließlich eine exponentielle Charakteristik, welche auf Systemeinflüsse in der Überlastsituation zurückzuführen sind.

Diese Effekte zeigen sich auch in den Ergebnissen von Jehle (2010) und Boegelsack (2010). Abbildung 6-33 zeigt das von Jehle (2010) gemessene Antwortzeitverhalten eines auf dem SAP-WebAS-Java basierenden SAP-NetWeaver-Portal-Systems bei einem Grenzlastversuch. Dabei zeigt die Charakteristik Ähnlichkeiten zu den Messungen der vorliegenden Arbeit. Bei

Erreichen einer Grenze (in diesem Szenario 47 Benutzer) steigt die Antwortzeit des Systems in einer virtuellen Maschine sprunghaft an (Jehle 2010).

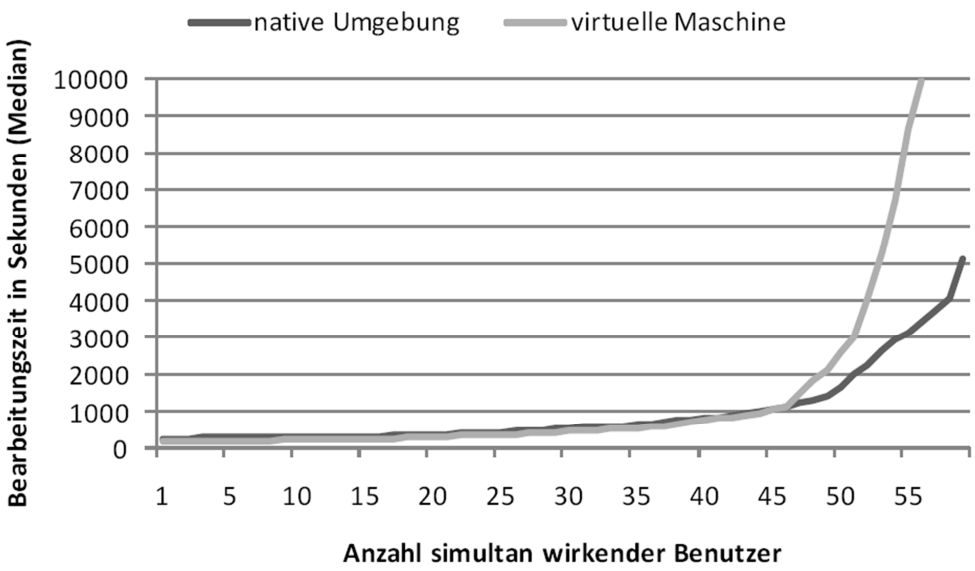


Abbildung 6-33: Bearbeitungszeit während des Grenzlastversuchs
Quelle: (Jehle 2010)

Ein ähnliches Verhalten zeigen die Messergebnisse bei (Boegelsack 2010). In Abbildung 6-34 wird der Gesamtdurchsatz des „Zachmanntests“ (Boegelsack 2010), eines synthetischen Benchmarks, unter verschiedenen Systemkonfigurationen in nativem und virtualisiertem Betrieb dargestellt. Auf der x-Achse ist die Anzahl der zur Verfügung stehenden Work-Prozesse im Laborsystem (SAP-WebAS-ABAP) aufgetragen. Die y-Achse zeigt den maximalen Durchsatz der jeweiligen Systemkonfiguration. Trotz der nicht kontinuierlich verlaufenden Ausprägungen des Durchsatzes zeigt sich in allen Konfigurationen, dass dieser mit wachsender Systemlast ansteigt bis er bei Erreichen einer gewissen, konfigurationsabhängigen Grenze zur Überlast wieder absinkt bzw. konstant bleibt. Ein Vergleich der Leistungsdaten bzgl. der Anzahl der installierten Systeme in Abbildung 6-34 zeigt zudem, dass die Leistung stärker absinkt, je mehr Prozesse, also Systeme, auf der Hardware-Plattform vorhanden sind. Dies deckt sich mit den Messungen in der vorliegenden Arbeit und zeigt den starken Einfluss der Hardware- und Betriebssystemressourcen.

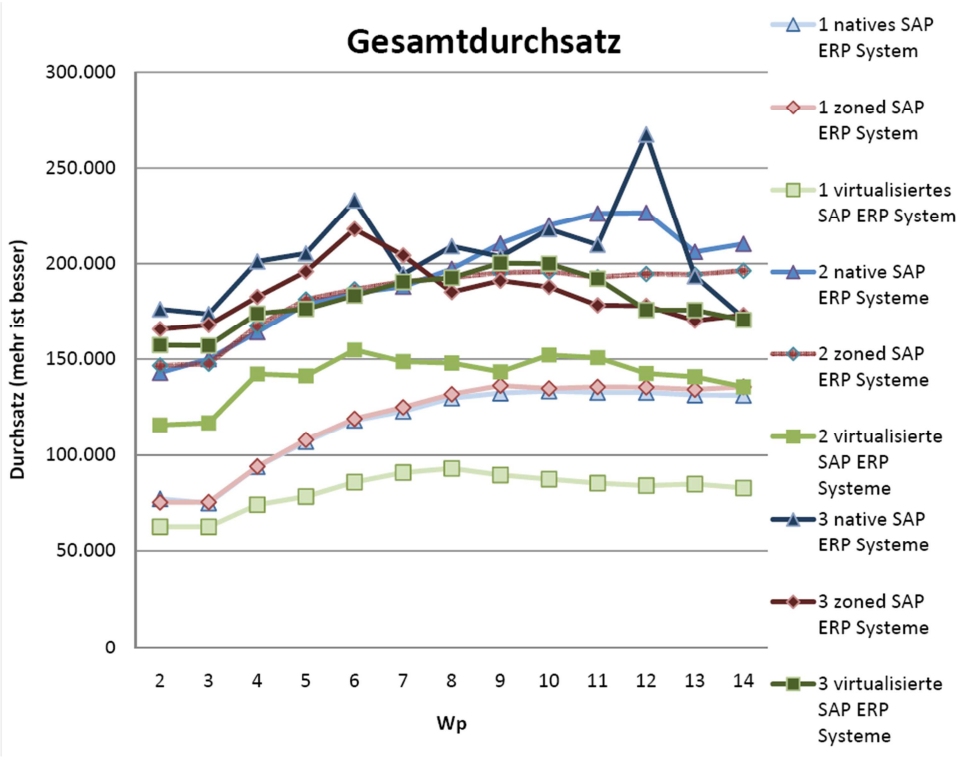


Abbildung 6-34: Skalierung bei Betrachtung des Gesamtdurchsatzes
Quelle: (Boegelsack 2010)

6.4. Simulation

Um die Genauigkeit des erstellten Modells anhand eines Vergleichs von Simulations- und Messwerten zu evaluieren, müssen Workload und Systemkonfiguration so aufeinander abgestimmt sein, dass Veränderungen in den Simulations- und Messvorgängen einer Ursache des Settings zugeordnet werden können.

6.4.1. Simulationsablauf

Zusätzlich zur kontrollierten Veränderung der Variablen im Sinne einer Sensitivitätsanalyse (Saltelli 2008) muss ebenso die statistische Aussagekraft der erhobenen Daten gewährleistet sein. Dies wird durch einen strukturierten Ablauf der Simulationsvorgänge gesichert.

6.4.1.1. Szenarien

Die Durchführung der Simulation ist auf oberster Ebene, wie bereits dargestellt, in 3 Szenarien unterteilt. Die Szenarien unterscheiden sich nicht im Workload, sondern in der Konfiguration der SAP-Systeme. Dabei wird zwischen Szenario 1 und Szenario 2 nur eine Variable, nämlich die Anzahl der zur Verfügung stehenden Prozessoren, von 6 auf 1 reduziert. Ebenso wird zwischen Szenario 1 und Szenario 3 nur eine Variable, die Größe des zur Verfügung stehenden Hauptspeichers, verändert. Die Verwendung verschiedener Szenarien und die Erhöhung der Lastintensität durch eine steigende Anzahl parallel wirkender Benutzer dient der Überprüfung der Auswirkungen verschiedener Parametrisierungen des Modells durch den Vergleich mit den gemessenen Performance-Werten.

6.4.1.2. Experiment-Sequenzen

Innerhalb eines jeden Szenarios wird eine Sequenz von Experimenten durchgeführt. Jedes Experiment unterscheidet sich zu dessen Vorgänger durch eine Erhöhung der Lastintensität in Form der Anzahl von parallelen Benutzern. Diese Anzahl wird in jedem Experiment sukzessive um 25 von 1 bis 150 parallelen Benutzern erhöht. In den vorangegangenen Testmessungen befand sich das System bei einer Menge von ca. 150 parallelen Benutzern bereits in einer Überlastsituation, welche sich durch einen sehr großen Anteil (>40 Prozent) an Verbindungsabbrüchen kennzeichnete. Aus diesem Grund wurde diese Menge als Obergrenze für die Anzahl paralleler Benutzer festgelegt, da die Genauigkeit der Simulation durch die zunehmende Häufigkeit von nicht-deterministischen und damit nicht modellierten Vorgängen mit steigender Überlast stark absinkt.

6.4.1.3. Experiment-Wiederholungen

Um die bereits besprochenen statistischen Verfahren aufgrund der hohen Variabilität der Messwerte anwenden zu können, ist eine mehrfache Ausführung eines jeden Experiments nötig. Aus diesem Grund wurde in Anlehnung an Jehle (2010) die Durchführung eines jeden Experiments 30-mal wiederholt.

6.4.2. Durchführung der Simulationen mit „lqsim“

Da die Anzahl der Benutzer innerhalb der Parametrisierung des Modells konfiguriert wird, ist jedes Experiment durch ein eigenes Modell in Form einer Textdatei auf Dateisystemebene abgelegt. Im Rahmen der vorliegenden Arbeit wurden 24 verschiedene Modelle generiert und parametrisiert. Wie bereits für die Messungen dargestellt, wurde jedes Experiment auch 30-mal simuliert.

Für die Durchführung der Simulation wurden folgende Übergabe-Parameter des Tools „lqsim“ verwendet:

- automatic blocking
- run-time
- precision
- skip

Das Tool „lqsim“ führt in einem Simulationslauf die Simulation mehrmals durch. Die Anzahl der Durchläufe kann entweder manuell oder automatisch mit Hilfe des Parameters „automatic blocking“ bestimmt werden. Die in dieser Arbeit verwendete automatische Bestimmung der benötigten Anzahl von Blocks basiert auf der Auswertung der Simulationsergebnisse nach jedem „block“. Trifft das durchschnittliche 95 Prozent Konfidenzniveau der simulierten Werte den mittels „precision“ übergebenen Wert, so wird die Simulation gestoppt. Ist dies nicht der Fall, so wird ein weiterer „block“ ausgeführt. Die maximale Anzahl von ausgeführten Blocks beträgt 30, nach dieser Anzahl von Ausführungen wird die Simulation gestoppt, unabhängig davon, welche Genauigkeit erzielt wurde (Franks et al. 2010). Der Parameter „skip“ kann benutzt werden, um für den Simulator eine Art Einschwingphase zu konfigurieren, da er nur Simulationsergebnisse für die Berechnung der Genauigkeit

verwendet, welche nach dieser Zeitspanne erzeugt wurden. Da die Genauigkeit des Simulationsergebnisses (bei konvergierenden Modellen) mit der Laufzeit zunimmt, kann diese Laufzeit (pro „block“) durch den Parameter „run-time“ bestimmt werden. Nach Franks et al. (2010) sollte die „block“-Laufzeit ungefähr das 10.000-fache der längsten Service-Zeit des Modells betragen.

6.4.3. Interpretation der Simulationsergebnisse

Die Ergebnisse eines Simulationslaufs werden vom Simulationstool „lqsim“ in Form einer Textdatei zurückgegeben (Franks et al. 2010). In diesem Abschnitt werden nun die einzelnen Teile der Rückgabe dargestellt und deren Interpretation diskutiert.

6.4.3.1. Allgemeine Informationen

Am Anfang der Ausgabe werden zuerst allgemeine Informationen über den Simulationslauf wiedergegeben. Dazu zählen:

- Verwendete Version des Simulators
- Name Eingabedatei
- Name Ausgabedatei
- Datum und Uhrzeit des Simulationslaufs
- Anzahl der benötigten statistischen Blöcke
- Dauer des Simulationslaufs insgesamt und pro Block
- Maximales Konfidenzintervall
- Die für die Berechnung der Zufallszahlen verwendete „Seed“

Da die Ergebnisse der Simulationsläufe in eine Datenbank eingelesen wurden, dienten diese allgemeinen Informationen anhand des Namens der Eingabedatei für die Zuordnung zu einem Experiment und der strukturierten Ablage der Dateien auf Dateisystemebene. Zudem ermöglichen die Informationen über die Anzahl der benötigten statistischen Blöcke sowie das maximal erreichte Konfidenzintervall Rückschlüsse auf das Konvergenzverhalten des Modells sowie auf die Varianz der Ergebnisse.

6.4.3.2. Statische Informationen

Die Rückgabedatei des Simulationstools beinhaltet einen Bereich mit statischen Informationen. Dieser enthält die in der Parametrisierung des Modells festgelegten Werte zu den folgenden Eigenschaften:

- „Task information“
- „Entry execution demands“
- „Service time limit“

- „Entry think times“
- „Phase type flags“

Der Abschnitt „Task information“ beinhaltet die Informationen über sämtliche Tasks des simulierten Modells, dazu zählen die folgenden Eigenschaften:

- Task Name:
  Dies ist der eindeutige Name des Tasks innerhalb eines LQN-Modells.
- Type:
  Damit wird der Typ des Tasks bezeichnet. Zur Verfügung stehen „client“ und „server“. Ein Entry, welcher sowohl Anfragen an andere Entrys stellt als auch Anfragen anderer Entrys sendet, wird als „server“ bezeichnet.
- Copies:
  In dieser Spalte wird die Multiplizität des Tasks wiedergegeben. Dies gilt ebenso im Falle von „replicated“ Tasks.
- Proc Name:
  Mit diesem Attribut wird die Zuordnung des Tasks zu einer Prozessor-Komponente des LQN-Modell dargestellt.
- Group name:
  Da der LQN-Formalismus und das Simulationswerkzeug die Gruppierung von Tasks und Entrys unterstützen, ist eine eventuelle Gruppenzuordnung ebenfalls Bestandteil der Beschreibung eines Tasks.
- Pri:
  Um Anfragen innerhalb eines LQN-Modells zu priorisieren, werden diese Prioritäten bei der Modellierung als Eigenschaften der Tasks abgebildet und damit ebenfalls in der Task-Beschreibung aufgeführt.
- Entry List:
  Diese Eigenschaft gibt eine, durch Leerzeichen getrennte, Aneinanderreihung aller zu einem Task gehörenden Entrys aus.

Um die parametrisierten Ausführungszeiten der jeweiligen Entrys für die Auswertung zur Verfügung zu stellen, werden diese in der Sektion „Entry execution demands“ mit ausgegeben. Für jeden Entry wird aufgelistet, zu welchem Task dieser gehört und die Ausführungszeit einer jeden Phase.

Die Struktur des Absatzes „Service time limit“ folgt dem Aufbau der Vorhergehenden. Die parametrisierte „max_service_time“ beschreibt die vom Modellierer vorgesehene maximale Servicezeit. Die Simulationsergebnisse werden mit dieser Servicezeit verglichen, um die Wahrscheinlichkeit für ein Übertreten dieser Zeit im Abschnitt „Probability maximum service time exceeded“ zurückgeben zu können.

Für jeden Entry in dem für die Simulation verwendeten LQN-Modell werden im Abschnitt „Entry think times“ die parametrisierten Wartezeiten zwischen zwei Anfragen von Referenz-Tasks dargestellt. Da auch die Thinktime separat für jede Phase definiert werden kann, wird dies für jede Phase, sofern verwendet, ausgegeben.

Im letzten Bereich der statischen Informationen werden im Abschnitt „Phase type flags“ die modellierten Typen jeder Phase eines Entrys dargestellt. Das „lqsim“-Tool bietet die Möglichkeit von deterministischen und stochastischen Phasen.

6.4.3.3. Simulierte Servicezeiten

Im Gegensatz zu den parametrisierten Servicezeiten in Abschnitt „Entry execution demands“ enthält die Sektion „Measured Entry execution demands“ die während der Simulation tatsächlich aufgetretenen Servicezeiten eines Tasks. Wie bereits dargestellt, wird bei der Ausführung eines Simulationslaufs eine statistische Verteilung der Servicezeiten eines Entrys angenommen. Basierend auf dieser Verteilung wird die Servicezeit bei einem Aufruf errechnet und verwendet. Die durchschnittliche Servicezeit eines Entrys weicht damit von der parametrisierten Servicezeit ab.

6.4.3.4. Aufrufverhalten:

Die durchschnittliche Anzahl von Anfragen eines Entrys an einen anderen wird für jeden einzelnen Aufruf im Abschnitt „Mean number of rendezvous from entry to entry“ dargestellt. Eine Anfrage wird dabei eindeutig durch die Nennung des Quell-und Ziel-Entrys definiert. Für jede Verbindung zweier Entrys wird die durchschnittliche Anzahl von Anfragen pro Phase zurückgegeben.

Die durchschnittliche Zeit, die Anfragen in der Warteschlange eines Tasks während der Simulation verbringen, wird im Abschnitt „Mean delay for a rendezvous“ dargestellt. Die Darstellung der mittleren Queueing-Zeit geschieht dabei aufgeteilt nach den im Modell verwendeten Phasen eines Quell-Entrys.

Um die tatsächliche Ausprägung der Wartezeiten präziser zu beschreiben, wird im Abschnitt „Variance of delay for a rendezvous“ noch deren Varianz pro Verbindung und Phase aufgelistet.

6.4.3.5. Antwortzeiten:

Im Gegensatz zu den Servicezeiten, welche für die Parametrisierung des Modells Verwendung finden, werden in den Abschnitten „Service times“ und „Service time variance (per phase) and squared coefficient of variation (over all phases)“ die durch die Simulation erzeugten Antwortzeiten dargestellt.

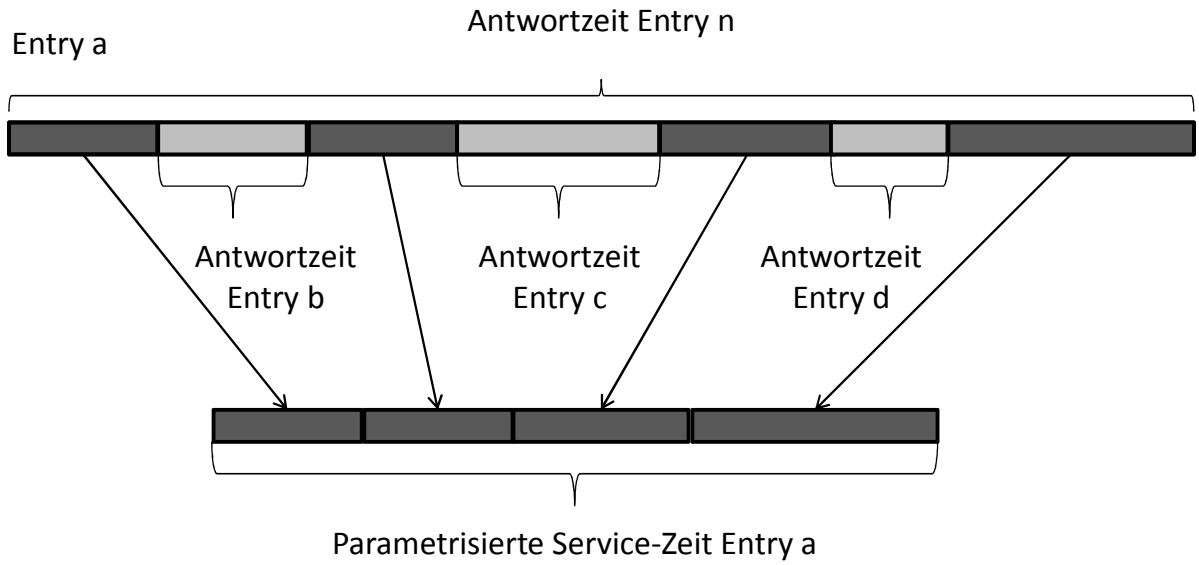


Abbildung 6-35: Gegenüberstellung Servicezeit und "gemessene" Servicezeit
Quelle: Eigene Darstellung

Abbildung 6-35 soll den Unterschied zwischen den verschiedenen Bedeutungen des Begriffs „service time“ veranschaulichen. Bei der Parametrisierung wird unter Servicezeit die Menge an Zeiteinheiten verstanden, welche ein Task für die interne Ausführung seiner Routinen benötigt. Diese Interpretation geschieht aus der Sicht des „Servers“. Bei der Auswertung der Simulationsergebnisse wird der Begriff hingegen aus der Sicht des „Clients“ interpretiert. Damit ist die Antwortzeit eines Tasks gemeint, welche sich, wie in der oberen Hälfte von Abbildung 6-35 dargestellt, aus der parametrisierten Servicezeit für die Durchführung der Task-internen Routinen und der Antwortzeit der gestellten Anfragen zusammensetzt. Diese Antwortzeit ist Ergebnis des Simulationslaufs.

Die Antwortzeiten werden zusätzlich zu deren Durchschnitt noch durch ihre Varianz in jeder verwendeten Phase und dem quadrierten Variationskoeffizienten über alle Phasen hinweg beschrieben.

6.4.3.6. Service-Level

Wie bereits erwähnt, kann mit Hilfe des Simulators auch die Wahrscheinlichkeit errechnet werden, dass eine zuvor parametrisierte maximale Antwortzeit überschritten wird. Diese Wahrscheinlichkeiten werden für jeden Entry im Absatz „Probability maximum service time exceeded“ ausgegeben. Ist für einen Entry keine maximale Antwortzeit definiert, so ist die Wahrscheinlichkeit gleich Null.

6.4.3.7. Durchsatz und Auslastung

Neben der Simulation der Antwortzeiten können LQN-Modelle auch für die Analyse des Durchsatzes und der Auslastung einzelner Komponenten verwendet werden. Diese Informationen gibt das Simulationswerkzeug im Abschnitt „Throughputs and utilizations per phase“ für die Entrys zurück. Dabei wird für jeden Entry der Durchsatz in Anfragen pro Zeiteinheit ausgegeben, während die prozentuale Auslastung für jede Phase separat dargestellt ist. Für die Beschreibung der CPU-Verwendung im Abschnitt „Utilization and waiting per phase for processors“ wird für jede CPU aufgeschlüsselt, welcher Anteil der Auslastung

durch einen bestimmten Entry verursacht wurde sowie die Wartezeit des Entrys auf die Ressourcen des Prozessors. Zusätzlich wird dabei noch die globale Auslastung eines jeden Prozessors geliefert.

6.5. Vergleich der Mess- und Simulationsergebnisse

Um die Tragfähigkeit des Modells und dessen Parametrisierung zu evaluieren, werden nun in diesem Kapitel die in einer Laborumgebung gemessenen Performance-Werte eines realen Systems mit den Ergebnissen der Simulation verglichen. Dies geschieht am Beispiel von drei verschiedenen Szenarien (6.3). Die Parametrisierung des Modells erfolgte wie in Abschnitt 5.3 beschrieben, anhand von Lastdaten, welche aus der wiederholten Ausführung und Vermessung des Systems unter dem in Absatz (6.1) vorgestellten Workload gewonnen wurden. Die Anzahl der Wiederholungen für die Parametrisierung des Modells wurde ermittelt, indem die Varianz nach jeder Wiederholung berechnet und mit dem Ergebnis aus der vorhergehenden Iteration verglichen wurde. Die Wiederholung wurde beendet, sobald die Differenz weniger als 1 Prozent betrug. Da die Datenbank als Blackbox im Modell abgebildet ist, wurden für die Parametrisierung der Datenbankkomponenten Performance-Werte verwendet, welche unter der zu simulierenden Last erzeugt wurden.

6.5.1. Szenario 1

Dieses Szenario zeichnet sich dadurch aus, dass der SAP-WebAS mit ausreichenden Ressourcen an CPU und Puffern konfiguriert ist. Um die Mechanismen der Sperrobjekte darzustellen, wurde die Datenbank so konfiguriert, dass sie ein signifikantes Verhalten aufweist, welches sich bei korrekter Modellierung auf die Antwortzeiten der modellierten Sperrobjekte auswirken muss.

6.5.1.1. Kapselung von DB-Anfragen durch Sperrobjekte

Wie bereits dargestellt, werden Datenbankabfragen innerhalb von Sperrobjekten gekapselt, um den wechselseitigen Zugriff auf Datenbankobjekte zu regeln. Dies stellt einen wichtigen Einflussfaktor auf die Antwortzeit eines SAP-ERP-Systems dar. Wie in Abbildung 6-36 dargestellt, ist die Datenbank in diesem Szenario so konfiguriert, dass sich deren Antwortzeit pro Datenbankanfrage ab einer Anzahl von 25 parallelen Benutzer stark erhöht.

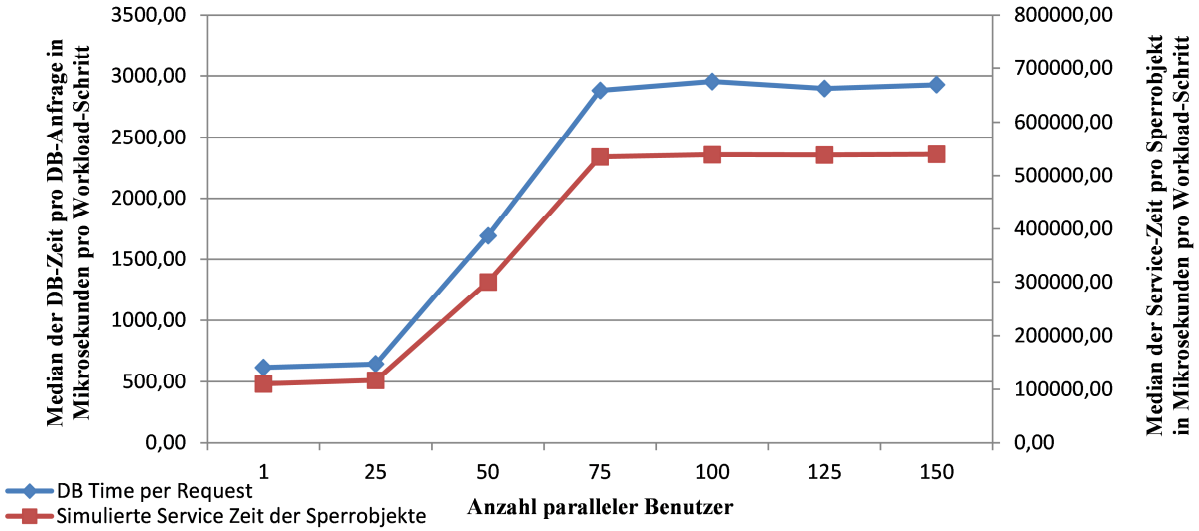


Abbildung 6-36: Vergleich der Datenbankzeit pro Request und der Service-Zeit der Enqueue-Sperrobjekte in Mikrosekunden abhängig von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Die Modellierung der Sperrobjekte bzgl. der Kapselung von Datenbankanfragen, wie in Abbildung 5-9 dargestellt, kann durch die simulierte Servicezeit der Sperrobjekte im Vergleich mit der Dauer von Datenbankanfragen pro Request evaluiert werden. Bildet das Modell die Kapselung der Datenbankanfragen durch die Sperrobjekte ab, so muss sich die Servicezeit der Sperrobjekte analog zur Servicezeit der Datenbankanfragen verändern. Abbildung 6-36 zeigt, dass beide Kurven die gleiche Charakteristik aufweisen. Die simulierte Antwortzeit der Sperrobjekte steigt an den gleichen Punkten. Damit kann gezeigt werden, dass im Modell Datenbankanfragen innerhalb von Sperrobjekten gekapselt werden.

6.5.1.2. Verwendung von Enqueue-Objekten

Im vorhergehenden Abschnitt konnte gezeigt werden, dass die Kapselung von Datenbankanfragen innerhalb der Sperrobjekte hinreichend abgebildet wurde.

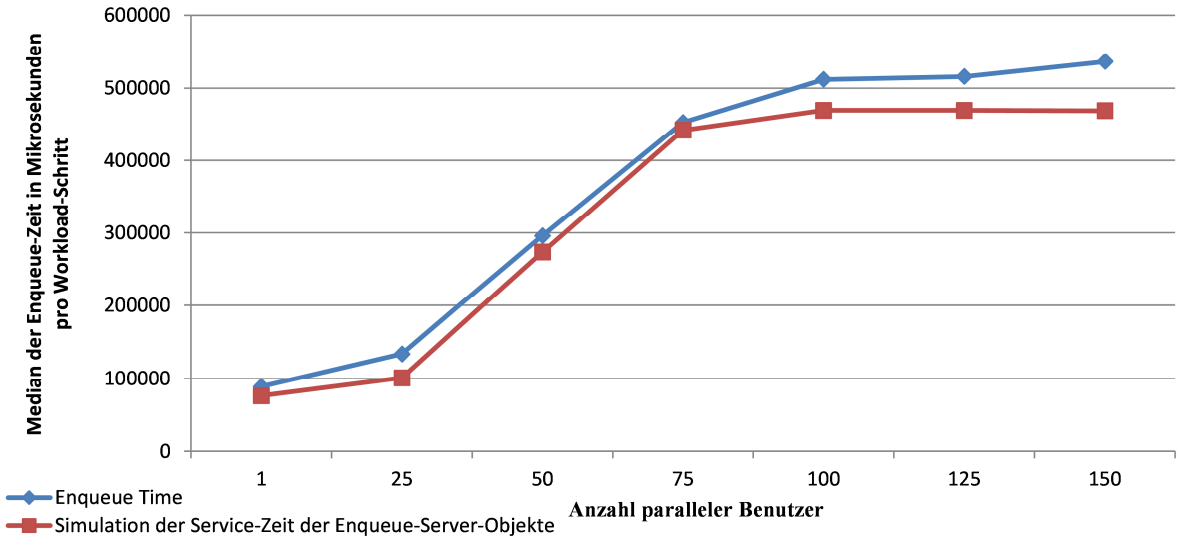


Abbildung 6-37: Vergleich der gemessenen und simulierten Enqueue-Wait-Time in Mikrosekunden abhängig von der Anzahl paralleler Benutzer

Quelle: Eigene Darstellung

Eine weitere für die Performance kritische Eigenschaft des Sperrmechanismus besteht in der wechselseitigen Sperrung von Datenbankobjekten. Um diese Eigenschaft zu evaluieren, muss die simulierte Zeit, welche ein Work-Prozess auf das Setzen einer Sperre warten muss, mit der real gemessenen Zeit verglichen werden. Dieser Vergleich ist in Abbildung 6-37 dargestellt. Die beiden im Diagramm abgebildeten Verläufe von gemessener und simulierter Enqueue-Wait-Time zeigen wiederum die gleiche Charakteristik. Durch die längere Bearbeitungsdauer der Datenbankanfragen wird ab einer Anzahl von ca. 25 parallelen Benutzern die Dauer einer Sperre erhöht, sodass die einzelnen Work-Prozesse dementsprechend länger warten müssen, bis die innerhalb der Sperre gekapselten Datenbankabfragen beendet sind und die Sperre gelöst werden kann.

6.5.2. Szenario 2

In diesem Szenario wurden die zur Verfügung stehenden CPU-Ressourcen im Vergleich zu Szenario 1 deutlich reduziert. Anstelle der für den Workload benötigten ca. fünf CPUs stehen dem SAP-WebAS-ABAP in diesem Szenario drei CPUs zur Verfügung.

6.5.2.1. CPU-Auslastung

Um die Modellierung der CPU-Nutzung zu evaluieren, wird im folgenden Abschnitt die gemessene CPU-Auslastung (4.1.2) mit der simulierten Auslastung verglichen. Abbildung 6-38 zeigt, dass die CPU bei der Simulation etwas später die volle Auslastung erreicht, als dies im Laborsystem aufgezeichnet wurde. Diese Differenz entsteht durch nicht modellierte Einflussfaktoren auf die CPU-Belastung. Da die im SAP-System gemessene CPU-Zeit pro Workload-Schritt gleich bleibt und diese Einflüsse unter hoher Last in Szenario 1 nicht aufgetreten sind, wurde das Betriebssystem hinsichtlich möglicher Einflüsse untersucht

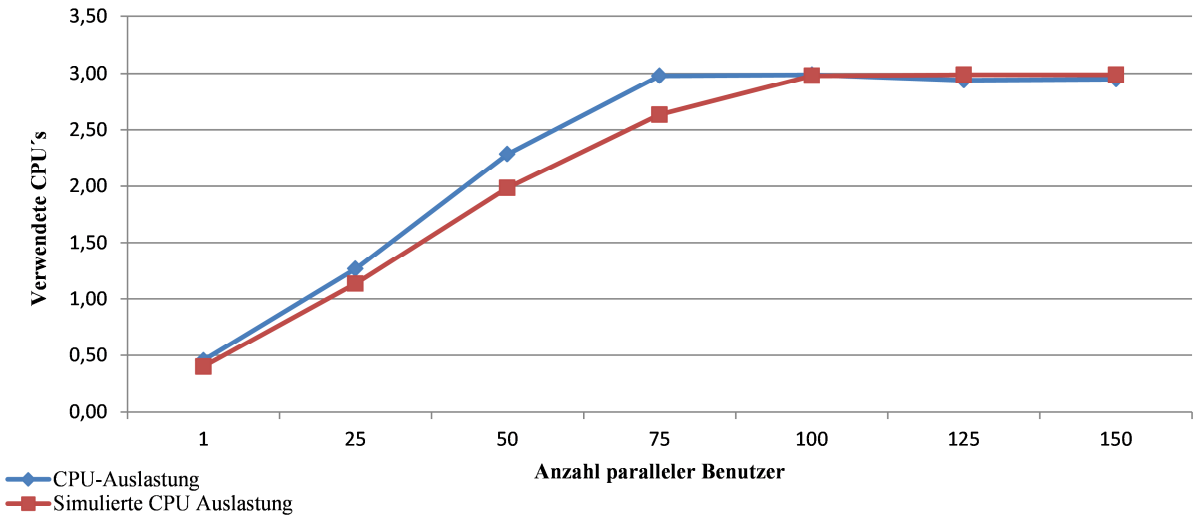


Abbildung 6-38: Vergleich der gemessenen und simulierten CPU-Verwendung in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Wie Jehle (2010) anhand von Messungen in einer Laborumgebung aufzeigt, kommt es bei CPU Engpässen zu einer Reduzierung der CPU-Ressourcen, welche für Benutzeranfragen zur Verfügung stehen. Dies geschieht bereits bei einer CPU-Auslastung von ca. 75 Prozent. Abbildung 6-38 zeigt analog zu dieser Aussage einen leicht überdurchschnittlichen Anstieg bei einer Auslastung von ca. 2,4 CPUs, was ca. 75 Prozent entspricht. Damit wird für die Bearbeitung der Benutzeranfragen mehr CPU benötigt, und das Betriebssystem erreicht die volle CPU Auslastung bereits bei weniger parallelen Benutzern als es das Ergebnis der Simulation zeigt.

6.5.3. Szenario 3

Um die Modellierung der Verdrängung von Inhalten aus den Tabellenpuffer zu evaluieren, wurden in diesem Szenario 3 die zur Verfügung stehenden CPU Ressourcen wieder auf 6 CPUs erhöht, jedoch die Größe der Tabellenpuffer um 50 Prozent reduziert. Dies führt, wie in den Messungen in Abschnitt 6.3.5 dargestellt, zu einer erhöhten Anzahl von Zugriffen auf die Datenbank und damit zu einer Erhöhung der Gesamtantwortzeit. In den folgenden Unterkapiteln werden nun die Ergebnisse der Simulation mit den Messwerten verglichen, um die Modellierung von Swaps (6.5.3.1), der daraus resultierenden Antwortzeit (6.5.3.2) und der zu überprüfenden maximalen Antwortzeit (6.5.3.3) zu evaluieren.

6.5.3.1. Anzahl DB-Anfragen

Wenn Inhalte aus den Tabellenpuffern verdrängt werden, so müssen diese Inhalte beim nächsten Zugriff aus der Datenbank nachgeladen werden. Dies führt zu einer höheren Last auf der Datenbank in Form einer größeren Anzahl von Zugriffen.

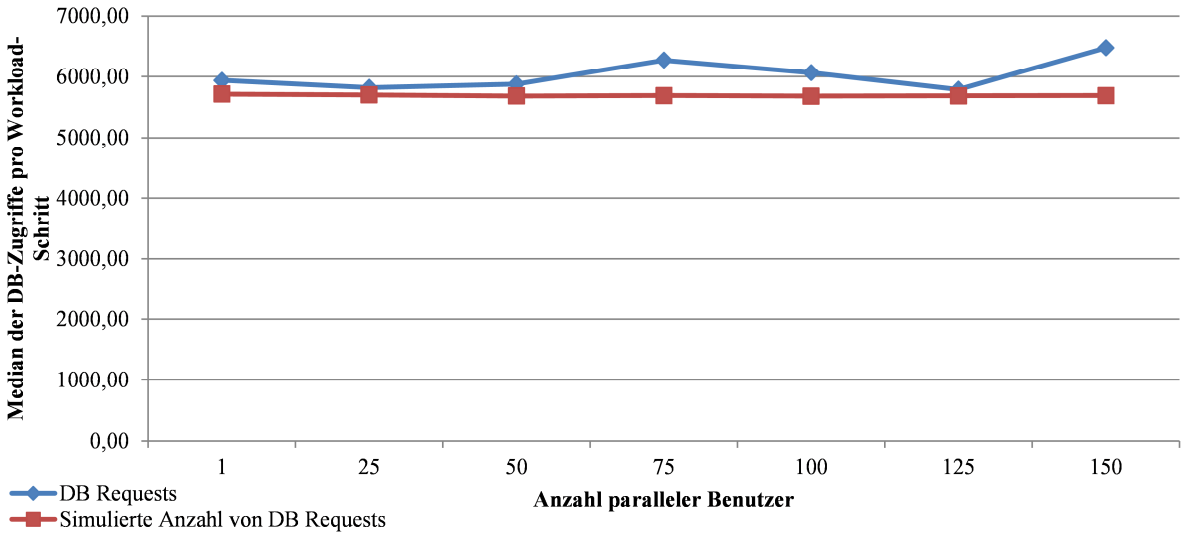


Abbildung 6-39: Vergleich der gemessenen und simulierten Anzahl von Anfragen auf die Datenbank in Abhängigkeit der Anzahl von parallelen Benutzern
Quelle: Eigene Darstellung

Wie aus dem Vergleich der Messwerte mit Abbildung 6-11 und der Beschreibung in Abschnitt 6.3.3.3 hervorgeht, ist die Anzahl der Datenbankanfragen pro Workload-Schritt in diesem Szenario um ca. 1000 Anfragen höher als bei ausreichender Konfiguration der Puffer. Abbildung 6-39 stellt den Median der gemessenen und simulierten Anzahl von DB-Zugriffen pro Workload-Schritt dar. Dieses Diagramm zeigt, dass die simulierte Anzahl von

Datenbankzugriffen geringer ausgeprägt ist als die gemessene Anzahl. Jedoch zeigen auch die simulierten Ergebnisse eine signifikant größere Anzahl von Datenbankzugriffen als die durch Messung erhobenen Werte aus Abbildung 6-11. Daraus kann gefolgert werden, dass die Modellierung der Pufferzugriffe und die Parametrisierung der Weiterleitungswahrscheinlichkeit das tatsächliche Verhalten der Puffermechanismen sehr gut abbilden.

6.5.3.2. Antwortzeit

Bisher wurden einzelne Aspekte, welche Einfluss auf die Gesamtantwortzeit des SAP-WebAS-ABAP haben, evaluiert. Ziel der Arbeit ist es jedoch die Entwicklung der Performance eines SAP-ERP-Systems anhand der Antwortzeit zu simulieren. Die Evaluation dieser Eigenschaft geschieht anhand des Vergleichs von real gemessenen Antwortzeiten mit Hilfe der STAD Performance Records mit den Ergebnissen der Simulation. Die Antwortzeit wird in der Ausgabe des Simulationstools anhand der Service Times der Client-Komponenten wiedergegeben.

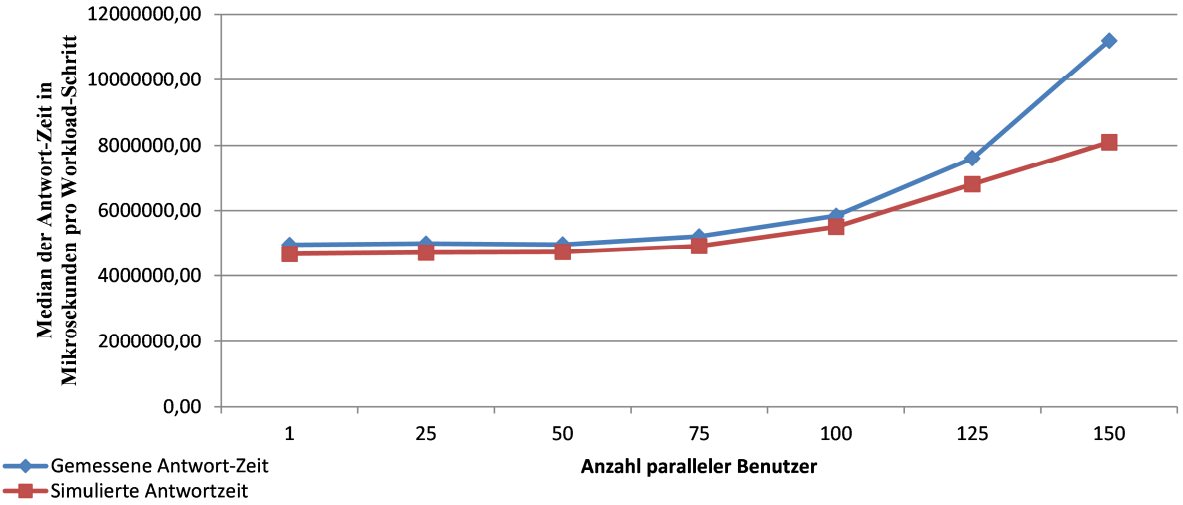


Abbildung 6-40: Vergleich der gemessenen und simulierten Antwortzeit in Abhängigkeit der Anzahl paralleler Benutzer
Eigene Darstellung

Wie in Abbildung 6-40 dargestellt, bildet das in der vorliegenden Arbeit eingeführte Modell der Antwortzeit eines SAP-ERP-Systems und dessen Parametrisierung das tatsächliche Antwortzeitverhalten außerhalb des Hochlastbereichs sehr gut ab. Bis zu einer Last von ca. 100 parallelen Benutzern (bei 92 konfigurierten Dialog-Work-Prozessen) zeigt die Simulation eine geringe Abweichung. Erst ab Erreichen dieser Grenze laufen Messung und Vorhersage signifikant auseinander. Dies ist Einflüssen des zugrunde liegenden Stacks aus Betriebssystem, Datenbank und Hardware geschuldet, welche so im Simulationsmodell aufgrund des nichtdeterministischen Verhaltens nicht berücksichtigt werden konnten.

6.5.3.3. Maximale Antwortzeit

Mit Hilfe der bei der Parametrisierung des Modells verwendeten maximalen Antwortzeit der Client-Komponente können nun in diesem Fall mehrere Anwendungszwecke verfolgt werden.

- Modellierung der im SAP-System konfigurierten maximalen Antwortzeit eines Aufrufs, welche bei Überschreitung abgebrochen wird

- Untersuchung von SLA Verletzungen

Neben der Untersuchung der Wahrscheinlichkeit für Verbindungsabbrüche aufgrund der Überschreitung einer maximalen Antwortzeit ist die Wahrscheinlichkeit für die Verletzung eines Service-Level-Agreements, welches z.B. im Rahmen eines Hosting-Vertrages vereinbart wurde, sehr oft von großer Bedeutung. Je nach gewünschtem Zweck kann die Parametrisierung der „max_median_time“ im Modell demnach durch die im System konfigurierte maximale Antwortzeit oder durch den zu überprüfende Service-Level erfolgen.

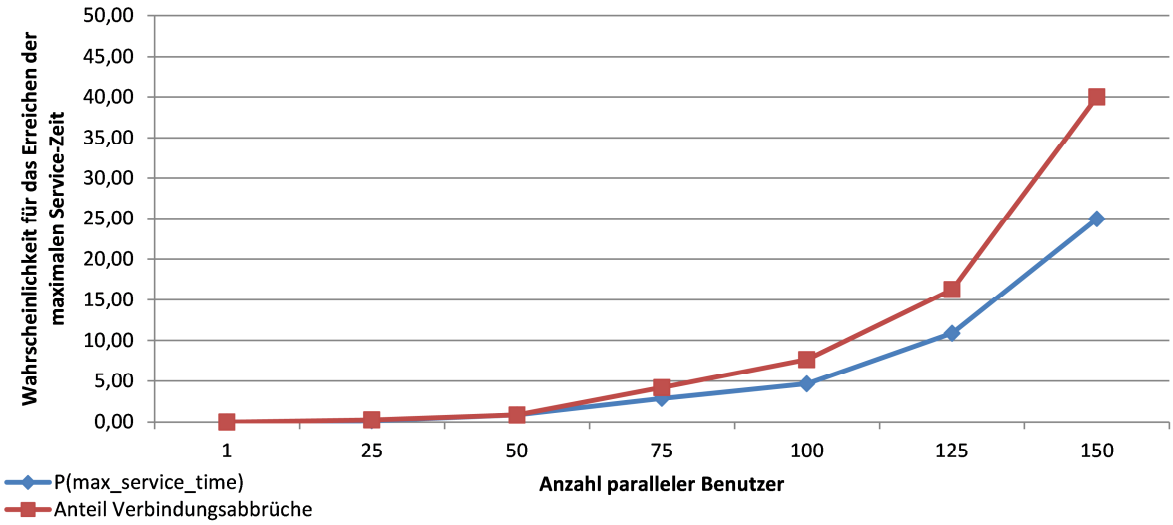


Abbildung 6-41: Vergleich der simulierten Wahrscheinlichkeit und der gemessen Häufigkeit von Überschreitungen der maximalen Antwortzeit in Abhängigkeit von der Anzahl paralleler Benutzer
Quelle: Eigene Darstellung

Abbildung 6-41 zeigt die Gegenüberstellung der Häufigkeit der tatsächlich aufgetretenen Verbindungsabbrüche aufgrund des Überschreitens einer maximalen Servicezeit mit der berechneten Wahrscheinlichkeit aus dem Simulationslauf. Für die Erstellung dieser Grafik wurde für die Parametrisierung des Modells die im SAP-ERP-System konfigurierte maximale Antwortzeit verwendet. Analog zu den bisher vorgestellten Ergebnissen zeigt auch der Verlauf dieser beiden Kurven eine anfangs geringe Abweichung zwischen Mess- und Simulationsergebnissen. Mit steigender Last wächst die gemessene Häufigkeit von Verbindungsabbrüchen stärker als die durch das Modell errechnete Wahrscheinlichkeit. Die Ursache für diese Abweichung erschließt sich direkt aus der Evaluation der Vorhersage der gesamten Antwortzeit. Da diese, wie in Abbildung 6-40 und Abschnitt 6.5.3.2 dargestellt, ebenfalls mit steigender Last eine steigende Differenz von Mess- und Simulationswerten aufweist, ergibt sich diese Differenz zwangsweise ebenso zwischen den erhobenen Mess- und Simulationswerten. Jedoch zeigt der Vergleich ebenso, dass die Kurve der Simulationswerte eine identische Charakteristik aufweist. Beide Kurven steigen anfangs konstant an. Die Steigung des Anstiegs erhöht sich bei beiden Kurven ab einer Last von 50 parallelen Benutzern. Erst bei Erreichen der maximalen Kapazität von ca. 100 Benutzern verlieren die Kurven ihre konstante Steigung und entwickeln sich exponentiell.

6.5.4. Fazit

Die im Rahmen dieser Arbeit identifizierten Quellen für eine Verringerung der Performance konnten im Modell umgesetzt werden. Die Genauigkeit der Simulation sinkt mit Annährung an die maximale Kapazität an parallelen Benutzern. Jedoch konnte in den verschiedenen Szenarien, welche unterschiedliche Eigenschaften der Antwortzeitkomponenten eines SAP-ERP-Systems unter Last gesetzt haben, gezeigt werden, dass die Kurven von Mess- und Simulationsergebnissen eine identische Charakteristik besitzen.

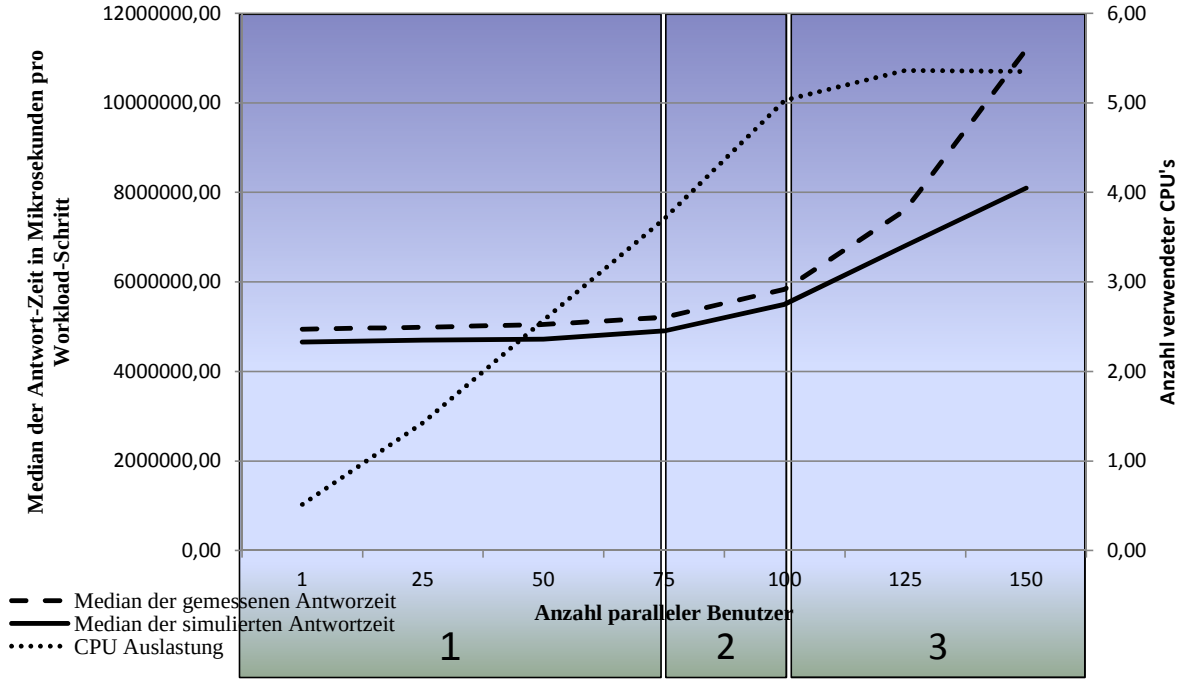


Abbildung 6-42: Vergleich der gemessenen und simulierten Antwortzeiten in Abhängigkeit der Anzahl paralleler Benutzer unterteilt in Lastphasen
Quelle: Eigene Darstellung

Ähnlich zu den Ergebnissen von Rolia et al. (2009a), der zuverlässige Simulationsergebnisse bis zu einer CPU-Auslastung von 76 % verzeichnet, zeigt der Vergleich zwischen gemessener und simulierter Antwortzeit in Abbildung 6-42 einen starken Einfluss der CPU-Auslastung auf die Antwortzeiten bei ca. 80 %. Dies deckt sich mit den praktischen Erfahrungen der Zuteilung von CPU-Ressourcen auf die Prozesse der Anwendung und der Selbstverwaltung des Betriebssystems, welche in den folgenden Absätzen weiter charakterisiert wird.

Abbildung 6-42 zeigt zudem eine Einordnung der verschiedenen Lastsituationen in Bereiche. Diese Einteilung basiert auf der Charakteristik der gemessenen und simulierten Werte der Antwortzeit unter einer steigenden Anzahl von parallelen Benutzern. Bezüglich der Lastsituation können diese Bereiche wie folgt charakterisiert werden:

- Niederige bis mittlere Systemlast – niedrige bis mittlere CPU-Last
- Mittlere Systemlast – hohe CPU-Last
- Hohe Systemlast – hohe CPU-Last

In Phase 1 ist, aufgrund der geringen Anzahl von parallelen Benutzern, der Anteil der Wartezeit auf Sperrobjekte gering, steigt jedoch konstant. Die benötigten CPU-Ressourcen stehen zur Verfügung, sodass an dieser Stelle keine Warte- bzw. Systemverluste auftreten. Die simulierten Werte der Antwortzeit liegen in diesem Bereich nahe an den durch Messung erhobenen Werten.

In Phase 2 erreicht der Bedarf an CPUs die Grenze von ca. 80 Prozent. An dieser Stelle beginnt, wie bereits erwähnt, die CPU Rechenzeit für die eigene Verwaltung zu benötigten, sodass weniger Ressourcen für die Bearbeitung der Benutzerprozesse zur Verfügung stehen. Die Antwortzeit des SAP-Systems steigt dadurch in dieser Phase stärker an. Da das Verhalten der CPU nicht Teil des Modells ist und dieses in diesem Bereich nicht vorhersagbar ist, steigt in dieser Phase die Differenz zwischen den gemessenen und simulierten Werten.

Ist die Kapazität des SAP-ERP-Systems an parallelen Benutzern erreicht, so müssen diese auf einen freien Work-Prozess warten. Zusammen mit dem steigenden Bedarf an CPU-Ressourcen und dem steigenden System-Anteil der verwendeten CPUs beginnt die Antwortzeit in diesem Bereich sehr stark zu steigen. Aufgrund der bereits in Phase 2 beschriebenen Effekten und dem im Überlastbereich nicht mehr deterministischen Verhalten des unterliegenden System-Stacks steigt die Differenz der Simulationswerte zu den Messwerten in dieser Phase stark an.

6.6. Eingrenzung des Lastbereichs für verlässliche Performance-Vorhersagen

Am Beispiel eines SAP-NetWeaver-Portal-Systems teilt Jehle (2010) die verschiedenen Zustände der Systemlast eines SAP-WebAS-Java-Systems in 4 Phasen ein:

- Phase 1:
  Die Bearbeitungszeit steigt moderat und konstant an.

- Phase 2:
  Die Bearbeitungszeit zeigt einen nicht linearen Anstieg. Die obere Grenze dieser Phase liegt bei zwei Drittel der verfügbaren Ressourcen.

- Phase 3:
  Die Berechnung der Entwicklung der Bearbeitungszeit liefert kaum noch zuverlässige Ergebnisse. Die obere Grenze dieser Phase ist gekennzeichnet durch eine Annäherung an die maximale Bearbeitungskapazität.

- Phase 4:
  In dieser Phase ist keine zuverlässige Berechnung der Entwicklung der Bearbeitungszeit möglich. Durch die Annäherung bzw. Überschreitung der maximalen Bearbeitungskapazität des Systems treten Timeouts auf.

Diese Einteilung kann durch die in der vorliegenden Arbeit durchgeführten Messungen bestätigt werden. Die Simulationsgenauigkeit in den verschiedenen Phasen wird im folgenden Absatz beschrieben.

Eine Analyse der Simulationsergebnisse in den betrachteten Lastszenarien, zeigt, dass die Lastsituationen eines SAP-System bezüglich der Genauigkeit der Simulation in verschiedene Bereiche einer Matrix eingeteilt werden kann. Die Achsen werden von System- bzw. Host-Auslastung gebildet. Abbildung 6-43 zeigt die Entwicklung der Simulationsgenauigkeit in den verschiedenen Kombinationen aus System und Hostauslastung.

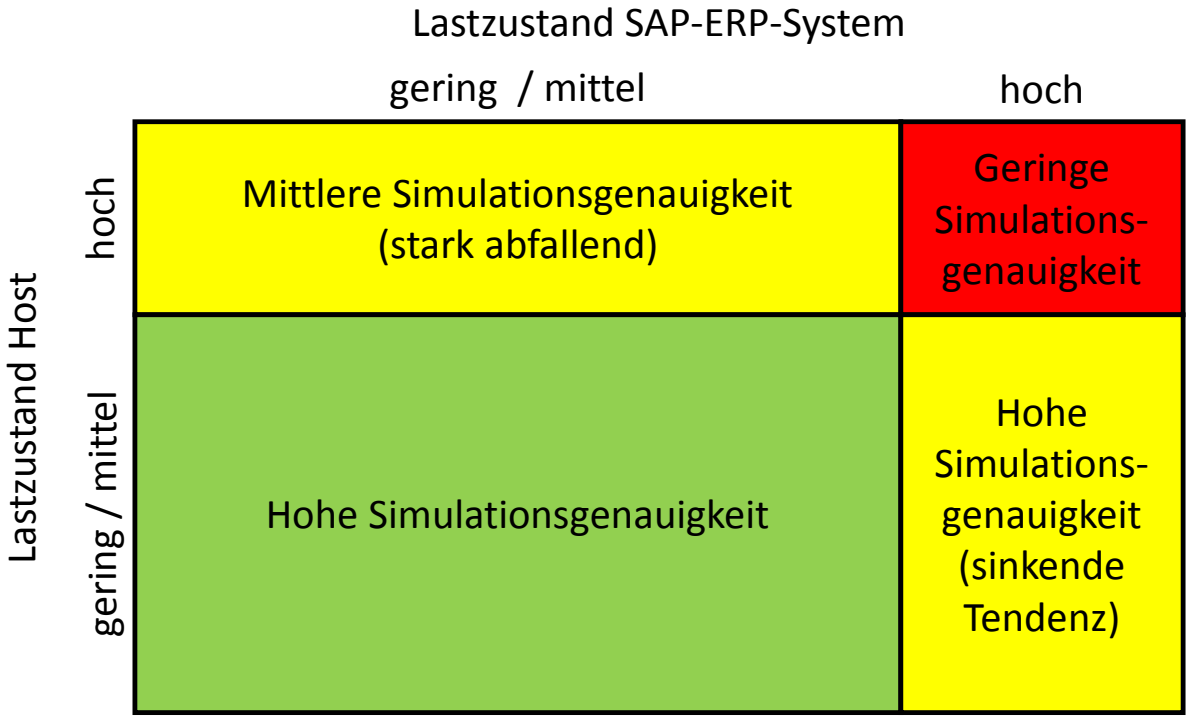


Abbildung 6-43: Bewertung der Simulationsgenauigkeit in Abhängigkeit des Lastzustandes
Quelle: Eigene Darstellung

Quadrant 1 beschreibt hierbei eine Lastsituation, in welcher sowohl der Host als auch das SAP-System an der Obergrenze der möglichen Leistung angelangt sind. Mit steigender Auslastung des Hosts sinkt die Genauigkeit der Simulationsergebnisse des SAP-Systems, siehe Quadrant 2. Durch die hohe Skalierbarkeit und Stabilität des SAP-System zeigen die Simulationsergebnisse in Quadrant 3 trotz hoher Auslastung des SAP-Systems eine gute Genauigkeit. Die volle Genauigkeit der Simulation und damit des vorgestellten Modells werden in Quadrant 4 erreicht, in welchem sich sowohl Host als auch SAP-System in einem niedrigen bis mittlerem Lastniveau befinden.

Durch die Fokussierung im Modell auf den WebAS-ABAP und das nicht-deterministische Verhalten des Betriebssystems im Überlastbereich haben die Performance-Veränderungen im Betriebssystem Einfluss auf die Genauigkeit der Simulationsergebnisse. Abbildung 6-44 zeigt die Entwicklung der CPU-Anteile für System- und Benutzerprozesse unter wachsender Last. Verursacht durch häufige Prozesswechsel wird für die Verwaltung der Prozesse auf Betriebssystemebene mehr CPU benötigt, welche damit dem SAP-System nicht mehr zur Verfügung steht. Durch Messung im Labor konnte festgestellt werden, dass dies bereits bei einer CPU-Auslastung von ca. 60 Prozent beginnt. Mit steigender Belastung des Hosts werden die, den Benutzerprozessen zur Verfügung stehenden und frei nutzbaren CPU-Ressourcen immer geringer. Dieser Abfall ist betriebssystem- und lastspezifisch und kann

nicht analog zur Parametrisierung des Modells des SAP-Systems aus Situationen mit geringer Last ermittelt werden.

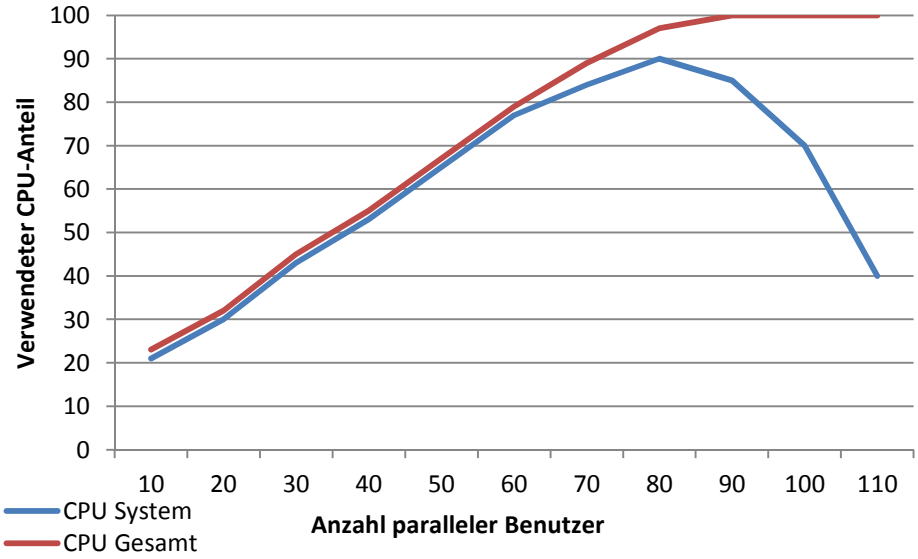


Abbildung 6-44: Entwicklung der CPU Benutzung durch System-Task
Quelle: Eigene Messung

Um eine größtmögliche Genauigkeit der Simulation zu erreichen, ist es daher nötig, die Grenzen zwischen den einzelnen Quadranten zu bestimmen. Dies kann sowohl durch Messung als auch aufgrund der genauen Simulationsergebnisse in Bereichen niedriger bis mittlerer Host-Auslastung durch die Simulation selbst geschehen.

Die Performance-Analyse des Testsystems zeigt, wie bereits erwähnt, als Grenze für eine steigende Ungenauigkeit der Simulationsergebnisse eine Grenze von ca. 60 Prozent der CPU Auslastung im Betriebssystem. Liefert die Simulation eine Auslastung der CPU, welche höher als 60 Prozent ist, sind die vorhergesagten Antwortzeiten als Minimalwerte der tatsächlichen Antwortzeit zu interpretieren.

Der Übergang von Hoch- zu Überlast zeigt sich bei der Betrachtung des SAP-Systems, wenn der Durchsatz des Systems geringer ist als die Ankunftsrate neuer Anfragen. Die Simulationsergebnisse zeigen in diesem Bereich einen linearen Anstieg der Antwortzeiten, wohingegen die Antwortzeit der Messwerte konstant auf einem Maximalwert verbleibt. Dies geschieht aufgrund von Timeouts der Anfragen. Damit ist die Phase der Überlast des SAP-Systems charakterisiert durch das Auftreten von Timeouts aufgrund der zu langen Wartezeit in der Dispatcher-Queue oder durch die Überfüllung derselben. Sowohl die Höhe des Timeouts als auch die Größe der Dispatcher-Queue sind im SAP-System konfiguriert und können daher direkt dem System entnommen werden. Durch die Analyse der Ausgabedatei kann die in der Simulation erreichte Größe der Dispatcher Queue ermittelt werden.

Die simulierte Wartezeit im Dispatcher hingegen wird nicht direkt durch den Simulator bereitgestellt, sondern ergibt sich aus der simulierten Servicezeit des Client-Tasks abzüglich der Summe der Servicezeiten der „Workloadstep“-Entrys. Damit kann das Auftreten von Timeouts und damit die Überlast des SAP-Systems ermittelt werden, indem die Auslastung der Entrys des Tasks „Workloadsteps“ bei 100 Prozent ist, und entweder die ermittelte

Verweildauer in der Dispatcher Queue größer ist als im realen System konfiguriert oder die Größe der Queue des „Workloadsteps“ Tasks die Systemkonfiguration übersteigt.

6.7. Aussagen mittels Simulation

Neben der Vorhersage der Antwortzeit in verschiedenen Lastsituationen sind durch die Modellierung und die aus der Simulation resultierenden Informationen weitere Aussagen über die Performance, das Sizing von SAP-ERP-Systemen, sowie die Analyse der ABAP-Programme hinsichtlich gegenseitiger Wechselwirkungen möglich.

6.7.1. Planung von CPU-Ressourcen

Um Systeme bzgl. Performance-Anforderungen gegenwärtiger oder zukünftiger Lastprofile auszulegen, werden Informationen über die unter einem definierten Lastprofil benötigten Systemressourcen benötigt. Bei der Auswahl von Hardware sind vor allem die folgenden Aspekte zu berücksichtigen:

- Anzahl der Prozessoren
- Größe des Arbeitsspeichers

6.7.1.1. Ermittlung der benötigten CPU-Ressourcen

Die Ermittlung der Anforderungen an die Größe des Arbeitsspeichers wird im Abschnitt über die Erkenntnisse aus der Analysephase (6.8.2) für die Tabellenpuffer dargestellt. Das Vorgehen für die Ermittlung der benötigten Anzahl an Prozessoren hängt grundsätzlich von dem Ziel der Analyse ab:

- Anzahl der Prozessoren für eine gewisse Antwortzeit
- Maximale Beschleunigung der Antwortzeiten

Bei beiden Zielen wird bei der Parametrisierung des Modells die Anzahl an verfügbaren Prozessoren durch die Erhöhung der Multiplizität schrittweise pro Experiment erhöht. Bei dieser Durchführung bleiben die Anzahl der parallelen Benutzer sowie die Parametrisierung der Tabellenpuffer unverändert. Das Ergebnis dieser Analyse ist ein Diagramm der Antwortzeit und der Wahrscheinlichkeit des Überschreitens einer maximalen Antwortzeit in Abhängigkeit der Anzahl an verfügbaren Prozessoren. Abbildung 6-45 zeigt exemplarisch ein durch die Auswertung von Simulationsergebnissen erhaltenes Diagramm. Dabei wurde das Modell mit einer maximalen Antwortzeit von 2 Sekunden parametrisiert. Diese Zeitspanne wurde so gewählt, da sie auch bei der Ermittlung der Leistungsfähigkeit von Hardware-Systemen mit Hilfe des SAP-SD-Benchmarks Anwendung findet.

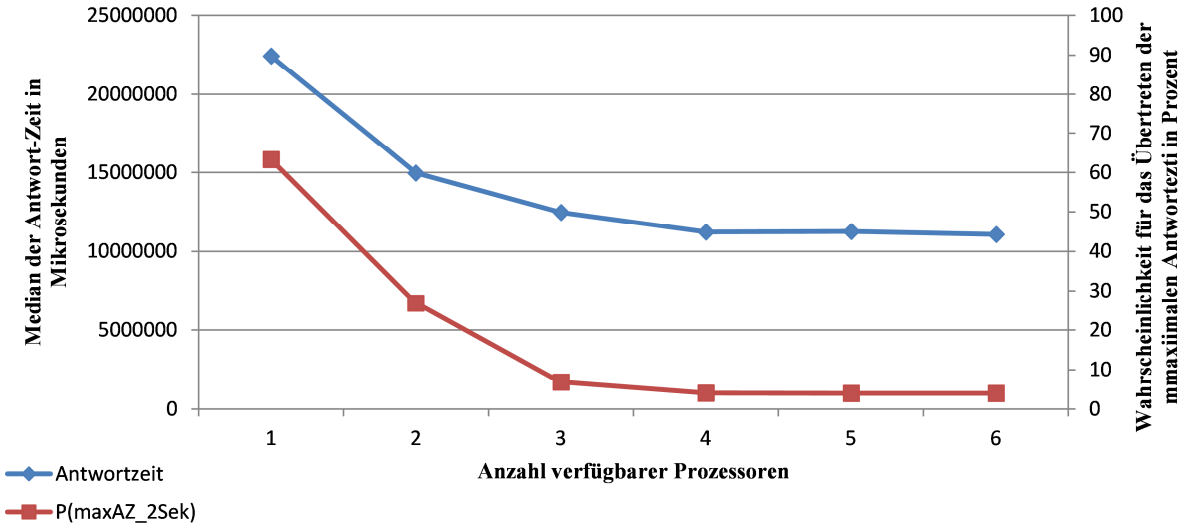


Abbildung 6-45: Ermittlung der Anzahl benötigtet Prozessoren
Quelle: Eigene Darstellung

Mit diesen Informationen kann nun auf Basis der Simulation ermittelt werden, welche Anzahl an Prozessoren für die Erfüllung der Zielfunktion benötigt werden.

Soll die Anzahl an Prozessoren für die Erfüllung eines Service Level Agreements (SLA) verwendet werden, so müssen sowohl die Antwortzeit als auch die durch Simulation ermittelte Wahrscheinlichkeit des Überschreitens der maximalen Antwortzeit betrachtet werden. In Abbildung 6-45 wäre der angenommene SLA im Durchschnitt bereits mit 2 verwendeten Prozessoren erreicht. Die Wahrscheinlichkeit für eine Verletzung des SLAs beträgt jedoch noch rund 25 Prozent. Es liegt nun im Ermessen des Betreibers, ob 2 Prozessoren bei einer Wahrscheinlichkeit einer SLA-Verletzung von ca. 25 Prozent, 3 Prozessoren bei ca. 7 Prozent verwendet oder die Wahrscheinlichkeit mit dem Einsatz von 4 Prozessoren auf unter 5 Prozent senkt.

Bei der Analyse der minimal möglichen Antwortzeit, soweit abhängig von der CPU, ist vor allem die Antwortzeit in Abhängigkeit von der Anzahl verwendeter Prozessoren zu betrachten, da eine derartige Analyse nur sinnvoll ist, wenn die CPU den zu erwartenden Flaschenhals darstellt. Wie in Abbildung 6-45 zu sehen ist, sinkt die durchschnittliche Antwortzeit kontinuierlich, um dann zwischen 3 und 4 verwendeten Prozessoren die minimale Ausprägung zu erlangen. Eine Konfiguration von Prozessoren über diesen Punkt hinaus bewirkt keine Verbesserung der Antwortzeit.

6.7.1.2. Bewertung der Ergebnisse

Wie in Abschnitt 6.5.2.1 diskutiert, bildet das Modell nur die CPU-Ressourcen ab, welche durch das SAP-System benötigt werden. Abbildung 6-38 zeigt eine Differenz zwischen der im Labor gemessenen CPU-Auslastung und der Auslastung, welche durch Simulation ermittelt wurde. Für die Konfiguration der Hardware muss analog zu anderen Sizing-Verfahren ein Sicherheitspuffer veranschlagt werden, um den CPU-Bedarf abzudecken, welcher nicht durch das SAP-ERP-System erzeugt wurde.

6.7.2. Anzahl Prozesse

Da eine Instanz des SAP-WebAS-ABAP nur insgesamt 100 Prozesse beinhalten kann, stellt sich die Frage der richtigen Aufteilung der Prozesse auf Dialog-Work- und Verbucher-Prozesse. Zu wenige Prozesse können dabei Einbußen an Antwortzeit bzw. Durchsatz mit sich bringen, wohingegen zu viele Prozesse Ressourcen verschwenden. In den folgenden Abschnitten, wird nun dargestellt, wie Simulation verwendet werden kann, um dieses Verhältnis zu optimieren.

6.7.2.1. Dialog-Work-Prozesse

Die Anzahl an verfügbaren Dialog-Work-Prozessen ist für die Gewährleistung der gewünschten Performance von großer Wichtigkeit. Da die Antwortzeit nicht indirekt-proportional mit der Anzahl der verfügbaren Dialog-Work-Prozesse sinkt, ist die, für die gewünschte Antwortzeit benötigte, Anzahl an verfügbaren Prozessen nur sehr schwierig vorher zu sagen.

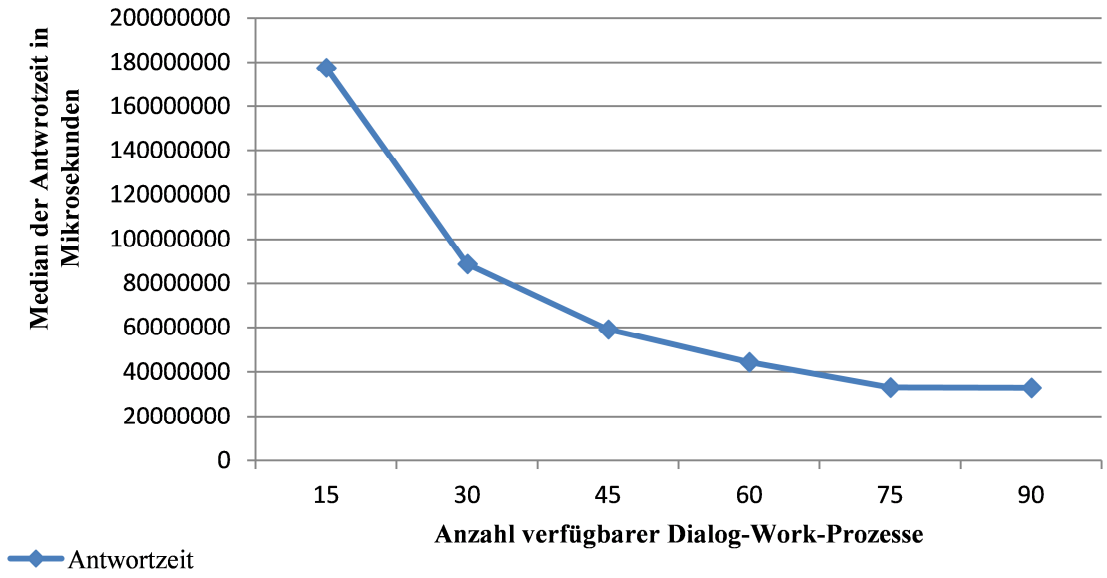


Abbildung 6-46: Simulierte Entwicklung der Gesamtantwortzeit in Abhängigkeit der Anzahl verfügbarer Dialog-Work-Prozesse
Quelle: Eigene Darstellung

Abbildung 6-46 zeigt die Entwicklung der Antwortzeit bei 75 parallelen Benutzern in Abhängigkeit von der Anzahl an verfügbaren Dialog-Work-Prozessen. Durch die stetige Erhöhung der verfügbaren Dialog-Work-Prozesse kann die Ausprägung der Antwortzeit gesenkt werden. Ab 90 verfügbaren Prozessen tritt keine Verbesserung der Performance mehr ein. Wie bereits dargestellt, sinkt die Antwortzeit nicht indirekt proportional mit der steigenden Anzahl an verfügbaren Dialog-Prozessen. Dies ist den internen Vorgängen des SAP-WebAS-ABAP geschuldet. Mechanismen, wie die Enqueue-Verwaltung oder die Pufferung von Datenbank-Objekten, gewinnen mit steigender Anzahl von parallelen Verarbeitungen an Gewicht und beeinflussen die Antwortzeit.

Mit Hilfe der Simulation können diese Auswirkungen in die Berechnung der benötigten Ressourcen einbezogen werden. Damit kann eine verlässlichere Prognose für die Konfiguration des Systems und der damit verbundenen Antwortzeiten erreicht werden.

Um die zur Verfügung stehende, maximale Anzahl von Prozessen richtig zu verteilen, muss neben der Analyse des Antwortzeitverhaltens in Abhängigkeit der Anzahl verfügbarer Dialog-Prozesse auch das Durchsatz-Verhalten der Verbucher-Prozesse berücksichtigt werden.

6.7.2.2. Verbucher-Prozesse

Da Verbucher-Prozesse asynchron aufgerufen werden, hat deren Anzahl keine direkte Auswirkung auf die Antwortzeit. Vielmehr ist bei dieser Art von Prozessen der Durchsatz zu betrachten, um die durch den Verbucher zu schreibenden Daten zeitnah persistent auf der Datenbank abzulegen.

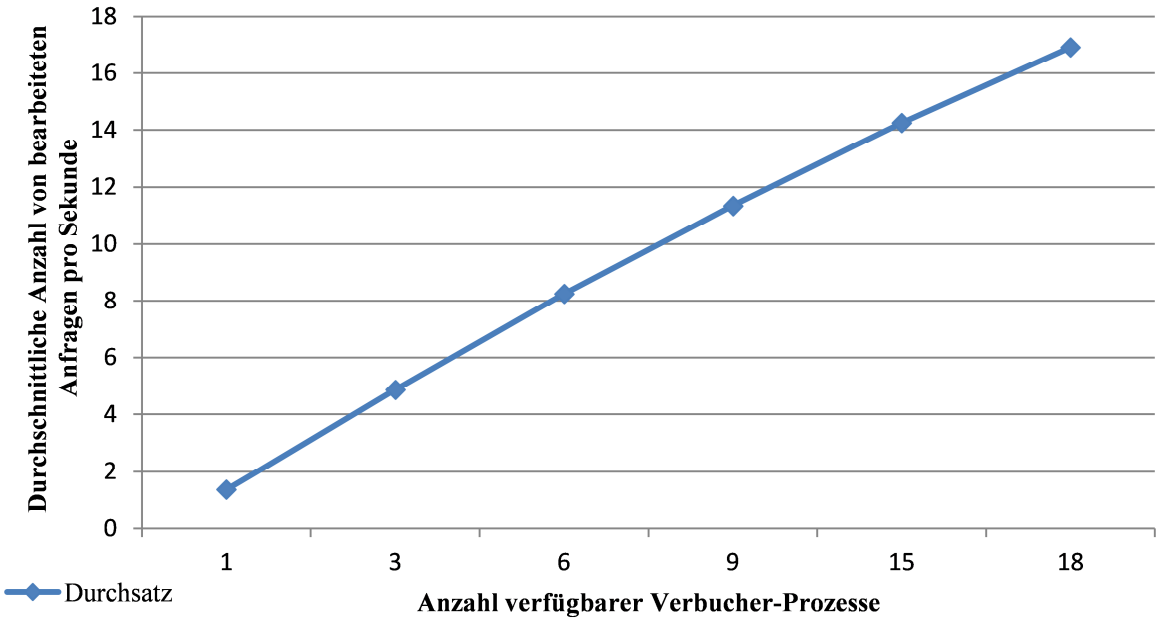


Abbildung 6-47: Simulierte Entwicklung des Durchsatzes von Verbucher-Aufträgen in Abhängigkeit der Anzahl verfügbarer Verbucher-Prozesse
Quelle: Eigene Darstellung

Abbildung 6-47 zeigt die Entwicklung des Durchsatzes an Verbucher-Aufträgen in Abhängigkeit der Anzahl verfügbarer Verbucher-Prozesse. Treten keine wechselseitigen Sperren (vgl. 6.7.3) auf, so steigert sich der Durchsatz mit der Anzahl an verfügbaren Prozessen. Die Tendenz der Steigungsrate in Abbildung 6-47 ist nicht direkt proportional zu der Anzahl der Verbucher-Prozesse, da es aufgrund der bei Verbuchungs-Aufträgen häufig verwendeten exklusiven Schreibsperren zu Enqueue-Wait-Situationen kommt.

Aufgrund dieser Tatsache kann der Durchsatz in Abhängigkeit von der Anzahl der Verbucher-Prozesse nur sehr schwierig ohne eine Simulation der Aktivitäten eines Workloads vorher bestimmt werden. Die Anwendung von Simulation bietet dagegen eine Alternative zu der in der Praxis verwendeten, schrittweisen Optimierung der konfigurierten Anzahl von Verbucher-Prozessen. Da bei jeder Veränderung dieser Anzahl die Instanz neu gestartet werden muss, kann die Simulation des Durchsatzes den Sizing-Prozess bei einem veränderten

Workload und/oder dem GoLive eines neuen Systems vereinfachen. Aufgrund der bereits in den verschiedenen Abschnitten diskutierten Genauigkeit der Simulationsergebnisse kann mit diesem Vorgehen nicht die optimale Konfiguration ermittelt werden. Jedoch bieten die Ergebnisse der Simulation einen sehr guten Näherungswert für die initiale Konfiguration der Instanzen eines SAP-WebAS-ABAP nach einer Veränderung des Workloads oder einer Neuinstallation.

6.7.2.3. Mengenverhältnis Dialog-Work-Prozesse zu Verbucher-Prozessen

Um die in einer Instanz maximal mögliche Anzahl von Prozessen auf Dialog-Work- und Verbucher-Prozesse zu verteilen, ist nach der Abschätzung des zu erwartenden Workloads die Entwicklung von Antwortzeit und Durchsatz in Abhängigkeit der jeweils zur Verfügung stehenden Menge von Prozessen zu betrachten.

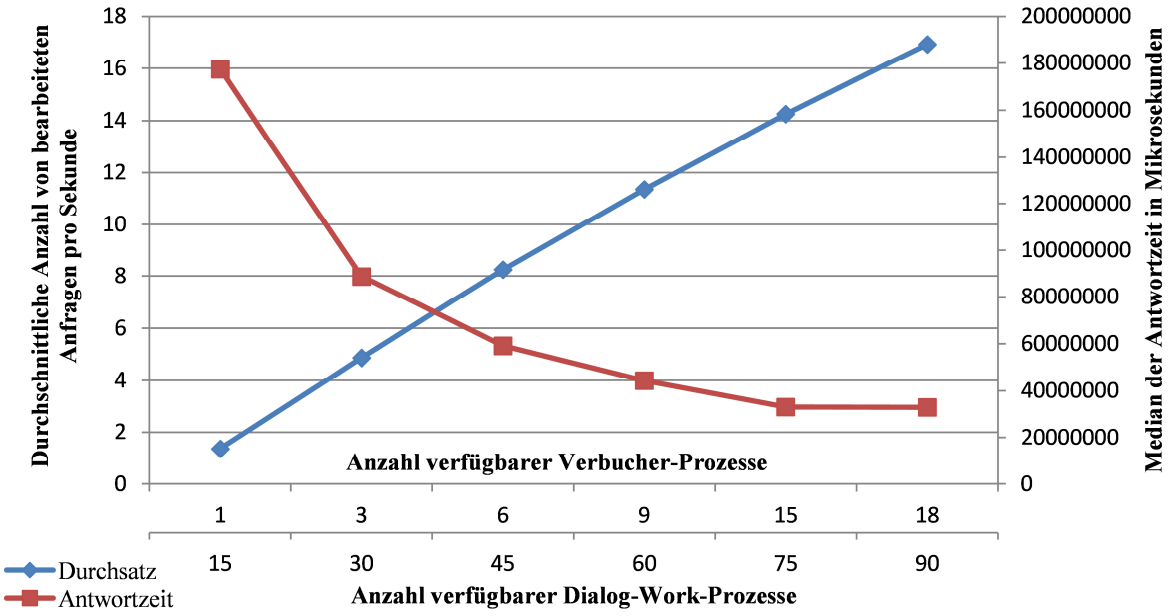


Abbildung 6-48: Gegenüberstellung von Antwortzeit und Durchsatz in Abhängigkeit der Anzahl an jeweiligen Prozessen
Quelle: Eigene Darstellung

Stehen nicht genügend Prozesse zur Verfügung, so muss entweder eine zusätzliche Instanz installiert oder es müssen die Auswirkungen von zu wenig konfigurierten Prozessen abgewogen werden. Abbildung 6-48 zeigt innerhalb eines Diagramms die Entwicklung von Antwortzeit und Durchsatz in Abhängigkeit der jeweils konfigurierten Anzahl von Dialog-Work- und Verbucher-Prozessen. Diese Informationen können einen Anhaltspunkt für die Konfiguration und der daraus resultierenden Performance eines SAP-WebAS-ABAP liefern. Da neben den Dialog-Work- und den Verbucher-Prozessen noch weitere Prozesse in einem System vorhanden sein müssen, stehen pro Instanz ca. 90 Prozesse zur Verfügung, welche auf Dialog- und Verbucher-Prozesse verteilt werden können. Im Beispiel, welches für diesen Abschnitt gewählt wurde, kann eine Anzahl von 75 Dialog-Work-Prozessen gewählt werden, da sich die Antwortzeit ab diesem Wert nur noch sehr gering verbessert. Damit können 15 Verbucher-Prozesse konfiguriert werden. Bei einer durchschnittlichen Antwortzeit von 30 Sekunden und 75 konfigurierten Dialog-Work-Prozessen müssen in diesem Beispiel durchschnittlich 9 Verbucher-Anfragen pro Sekunde bearbeitet werden. Durch die

Darstellung der Auswirkungen von verschiedenen Anzahlen konfigurierter Prozesse auf die Performance kann die Simulation die initiale Konfiguration benötigter Prozesse deutlich verbessern.

6.7.3. Hot-Spot-Analyse

Für die im vorhergehenden Kapitel angesprochene Optimierung des Verhältnisses von Dialog-Work-Prozessen zu Verbucher-Prozessen ist eine fehlerfreie Implementierung der verwendeten ABAP-Programme Voraussetzung. Ist dies nicht gegeben, können Flaschenhälse entstehen, die mit den bekannten Mitteln sehr aufwendig zu identifizieren sind. In diesem Kapitel wird ein Vorgehen eingeführt, welches auf Basis der Simulationsergebnisse die Analyse von Flaschenhälsen unterstützt und deren Identifikation erleichtert.

6.7.3.1. Hinweise auf Flaschenhälse

Bestehen in einem SAP-WebAS ABAP Flaschenhälse im Programmcode, so basieren diese häufig aus ineffizienten ABAP-Programmen, die hohe CPU Ressourcen benötigten. Dies wird bereits bei der Analyse der STAD Records sichtbar, da hier eine ungewöhnlich hohe CPU-Zeit verzeichnet ist. Diese Art von Flaschenhälsen ist leicht identifizierbar.

Wird die Skalierung jedoch von nicht optimal verwendeten Sperrobjekten oder deren Wechselwirkungen beeinträchtigt, so sind diese Flaschenhälse sehr schwer zu identifizieren, da sie erst im Betrieb bei hoher Last auftreten.

Blockieren sich zwei oder mehrere Sperrobjekte, bremst dies den Durchsatz und erhöht dementsprechend die Antwortzeit. Ist der Durchsatz einer Instanz zu niedrig, wird häufig die Anzahl der zur Verarbeitung verwendeten Prozesse erhöht. Bei der Suche nach einem Workload für die Evaluation des in der vorliegenden Arbeit eingeführten Modells ist dieser Fall aufgetreten. Nach der Durchführung eines Experiments waren asynchron aufgerufene Verbucher aktiv. Die in dem Experiment erzeugten Verbucher-Aufrufe benötigten ca. 120 Sekunden über die Dauer des Experiments hinaus, bis diese abgearbeitet waren.

Die Abhängigkeit der Ausführungszeit für den innerhalb des Experiments entstanden Workload von der Anzahl der konfigurierten Verbucher-Prozesse zeigt Abbildung 6-49. Trotz einer signifikanten Erhöhung der Anzahl von Verbucher-Prozessen konnte die Ausführungszeit nicht vermindert werden. Die Analyse der STAD-Records zeigte einen sehr hohen Enqueue-Wait-Anteil der Antwortzeit. Um nun den für die Sperre verantwortlichen Sperrbaustein zu identifizieren, müssen die innerhalb des Verbucher-Programms verwendeten Bausteine analysiert werden. Dies kann, wie im folgenden Abschnitt beschrieben, mit Hilfe des in dieser Arbeit vorgestellten Performance-Modells und den daraus erhaltenen Simulationswerten geschehen.

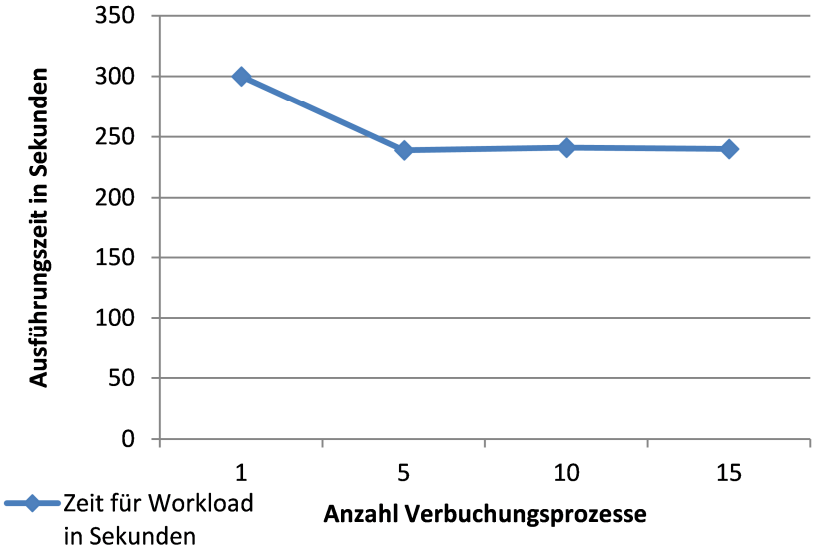


Abbildung 6-49: Dauer aller Verbuchungs-Prozesse während einem Experiment
Quelle: Eigene Darstellung

6.7.3.2. Analyse der Sperrobjekte

Da Flaschenhälse eine geringere Durchsatz-Kapazität aufweisen als die übrigen Komponenten der Architektur, ist deren Auslastung stärker ausgeprägt. Da die Ausgabe des Simulationstools, wie in Abschnitt 6.4.3, die Auslastung („Utilization“) einer jeden Modellkomponente während des Experiments enthält, kann für die modellierten Sperrobjekte ein Diagramm, wie in Abbildung 6-50 exemplarisch dargestellt, erstellt werden.

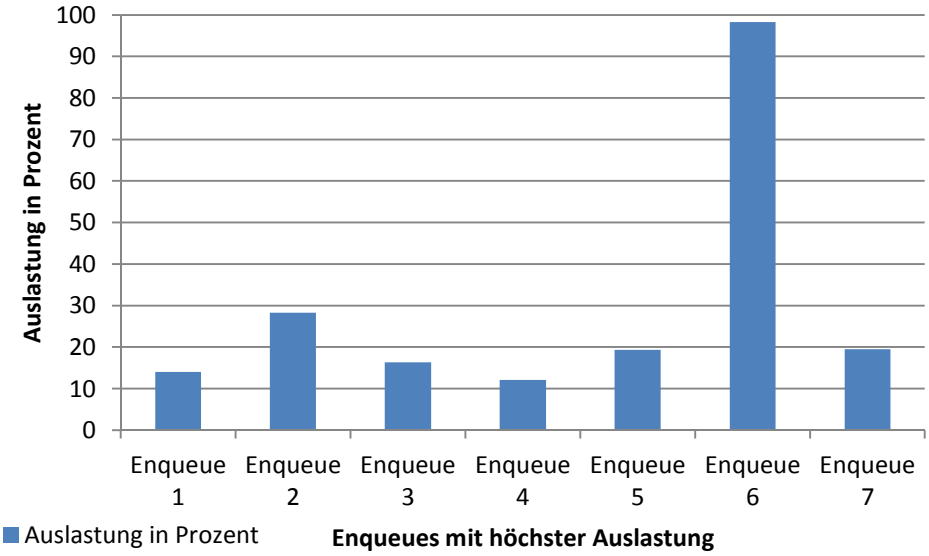


Abbildung 6-50: Auslastung der Sperrobjekte in Prozent
Quelle: Eigene Darstellung

Wie das Diagramm in Abbildung 6-50 zeigt, können mit Hilfe der Simulationsergebnisse Flaschenhälse auf Basis der Sperrobjekte sehr einfach ermittelt werden. Um diese Informationen zu erhalten, müssen die Modellkomponenten ermittelt werden, welche die Sperrobjekte im Modell abbilden. Für alle Komponenten des Modells enthält der Abschnitt „Throughputs and utilizations per phase“ deren Auslastung.

6.7.3.3. Analyse wechselseitiger Sperren

Neben der nicht optimalen Verwendung von Sperrbausteinen können Flaschenhälse auf Enqueue-Ebene auch noch durch Wechselwirkungen an sich korrekt verwendeter Sperrobjekte entstehen. Aufgrund von Wechselwirkungen, wie in Abbildung 3-6 dargestellt, können mehrere Sperrobjekte sich gegenseitig blockieren, wenn sie versuchen, eine Sperre auf den gleichen Datenbankbereich zu setzen.

Um den betroffenen Datenbankbereich zu identifizieren, können die innerhalb eines Workloads verwendeten Sperrobjekte bezüglich gemeinsam verwendeter Datenbankbereiche analysiert und verglichen werden. Durch die Verwendung eines Performance-Modells und dessen Simulation, können diejenigen Datenbankbereiche identifiziert werden, für welche mehrere Sperrobjekte versuchen, Sperren zu setzen.

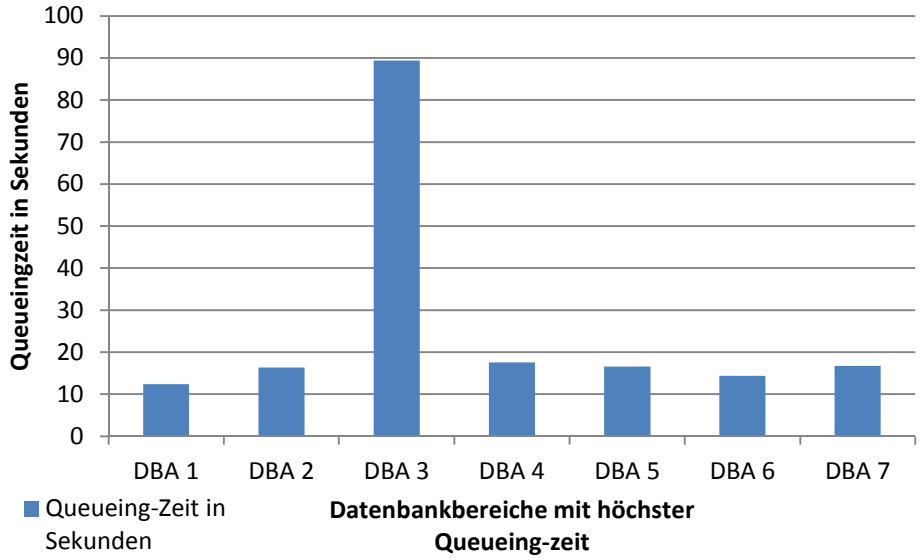


Abbildung 6-51: Auslastung der Datenbank-Semaphore in Prozent
Quelle: Eigene Darstellung

Abbildung 6-51 zeigt die aus der Ausgabe des Simulationstools erhaltenen Werte für die Queueing-Zeit der Semaphor-Komponenten im Modell. Diese Semaphor-Objekte wurden für die Modellierung von wechselseitigen Sperren in das Modell eingefügt. Da in der Ausgabe des Simulationswerkzeuges nur der „Mean delay for a rendezvous“ zurückgegeben wird, muss die Wartezeit aller Verbindungen auf die Semaphor-Task der Datenbankbereiche errechnet werden. Die Verzögerungen werden durch die folgenden Informationen eindeutig spezifiziert:

- Task Name:
  Dies bezeichnet den Namen des Task, zu welchem der „Source Entry“ gehört.

- Source Entry:
  Client Entry der Verbindung, bei welcher der „delay“ auftrat.

- Target Entry
  Server Entry der Verbindung.

Um nun die Verzögerungen zu erhalten, welche die Semaphor-Tasks der einzelnen Datenbankbereiche verursacht haben, muss der Abschnitt „Mean delay for a rendezvous“ der Ausgabe des Simulationstools nach allen Einträgen durchsucht werden, deren „Target Entry“ ein Semaphor-Entry ist. Das Ergebnis dieser Suche muss nach den Semaphor-Entrys sortiert werden, und die Einträge für jeden Entry müssen aufaddiert werden. Damit kann, wie in Abbildung 6-51 exemplarisch dargestellt, für jeden Datenbankbereich die Verzögerung identifiziert werden, und damit können der oder die von wechselseitigen Sperren betroffenen Datenbankbereiche identifiziert werden.

6.7.4. Entwicklung der Datenbanklast

Wachsen SAP-Systeme, so wird dieses Wachstum auf Seiten des SAP-WebAS-ABAP durch die zusätzliche Installation von Dialog-Instanzen umgesetzt. Dadurch wird die Last über mehrere Instanzen hinweg verteilt. Da diese Dialog-Instanzen auch auf verschiedenen Rechnern verteilt sein können, kann so durch Integration neuer Hardware die Leistungsfähigkeit des SAP-WebAS-ABAP einfach erhöht werden.

Dies gilt so nicht für die Datenbank. Diese wird auf einem zentralen Server mit einer großen Anzahl an Prozessoren und einer großen Menge an Arbeitsspeicher vorgehalten. Da Server dieser Größe eine beträchtliche Investition darstellen, ist es für den effizienten Betrieb von SAP-ERP-Systemen wichtig, abzuschätzen zu können, welche Kapazität (Beantwortung von Anfragen) die Datenbank und damit die Unterliegende Hardware besitzen muss. Diese Kapazität stellt sich aus den folgenden Bestandteilen zusammen:

- Arten von Anfragen („insert“, „update“, „delete“ und „select“)
- Anzahl der jeweiligen Anfragen pro Datenbankbereich
- Anzahl an abgefragten Datensätze

Einige Datenbankhersteller bieten auch die Möglichkeit, Datenbanken zu partitionieren. Damit kann eine Datenbank über mehrere physikalische Server hinweg verteilt werden.

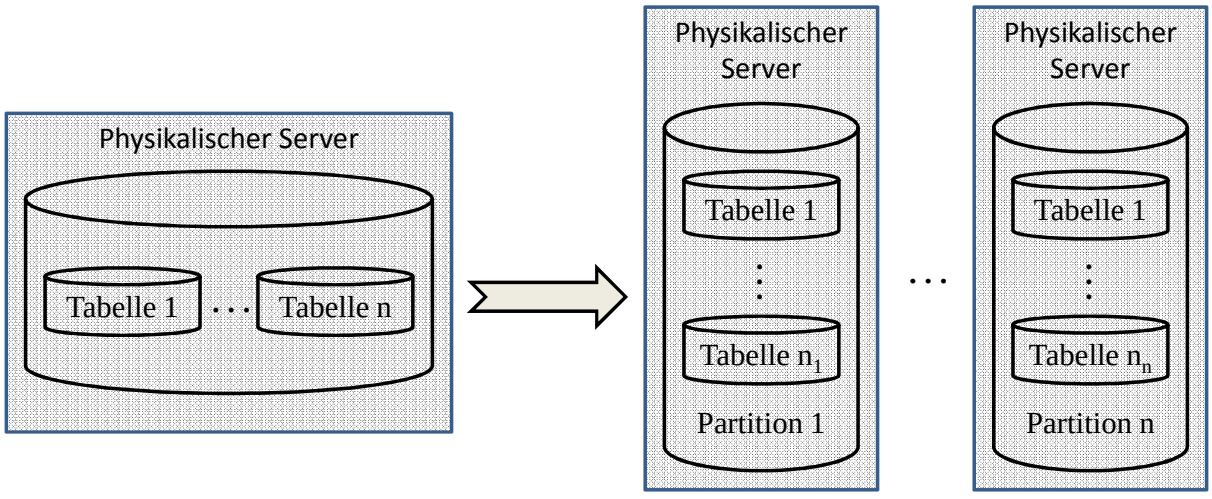


Abbildung 6-52: Partitionierung einer Datenbank
Quelle: Eigene Darstellung

Damit kann die Performance der Datenbank durch Hinzunahme zusätzlicher Hardware erhöht werden. Jedoch wird die Datenbank nicht repliziert auf verschiedenen Hosts abgelegt, sondern mit der Granularität von Tabellen auf die Server verteilt. Um die Last, welche die Datenbank abzuarbeiten hat, effizient auf die zur Verfügung stehenden physikalischen Ressourcen zu verteilen, sind ebenso die bereits im vorherigen Abschnitt dargestellten Informationen über die zu erwartenden Anfragen und deren Intensitäten nötig.

6.7.4.1. Anzahl und Qualität der Querys

Grundsätzlich kann die einfache Anzahl der Datenbankanfragen über die Analyse des SAP-SQL-Performance-Trace ermittelt werden (4.2.7). Um jedoch diese Genauigkeit zu erhöhen und die zusätzlichen Datenbankanfragen, welche über Verdrängungen und Invalidierungen gepufferter Objekte erzeugt werden, ebenso zu ermitteln, kann das Ergebnis der Simulation verwendet werden.

Ebenso kann aus dem Simulationsergebnis die Qualität der Querys identifiziert werden. Dies geschieht durch Analyse des Abschnittes „Mean number of rendezvous from entry to entry“. Dieser Abschnitt beinhaltet für jede Verbindung den Namen des Tasks des anfragenden Entrys, den Namen des Entrys selbst, die Identifikation des Ziel-Entrys sowie die durchschnittliche Anzahl von Anfragen von Quell-Entry an Ziel-Entry pro Phase.

Da in der Modellierung der Datenbankschicht (5.2.7.2) pro Datenbankbereich jeweils ein Entry für die Anfragetypen „select“, „select single“, „insert“, „update“ und „delete“ existiert, können nun die in der Simulation aufgetretenen Anfragetypen und deren Anzahl pro Datenbankbereich ermittelt werden. Dies geschieht durch Identifikation und Aufsummierung aller Anfragen, welche auf den gleichen Entry, welcher durch Datenbankbereich und Anfragetyp eindeutig definiert ist, gerichtet sind.

6.7.4.2. Gewichtung der Querys mit der durchschnittlichen Anzahl von abgefragten Datensätzen

Da sowohl bei der Partitionierung von Datenbanken und bei der Beschreibung des zu erwartenden Lastprofils neben Anzahl, Typ und Zielbereich von Datenbankanfragen auch die Menge an zu übertragenden Datensätzen eine Rolle spielt, müssen die bereits ermittelten Informationen mit der Anzahl der Anfragen angereichert werden.

Die Gesamtanzahl an übertragenen Datensätzen ist in den STAD-Records der betreffenden ABAP-Programme aufgelistet. Jedoch wird dabei nur nach den Anfragetypen differenziert. Die jeweiligen Datenbankbereiche werden hier nicht berücksichtigt. Diese Information kann über die Analyse der STAD-Performance-Trace zu den aus der Simulation ermittelten Werten hinzugefügt werden. Wie aus der Beschreibung des SAP-SQL-Trace (4.2.7) hervorgeht, können dessen Einträge zu Sequenzen (4.2.7.4) zusammengefasst werden. Pro Sequenz können damit der abgefragte Datenbankbereich sowie die Anzahl der übermittelten Datensätze bestimmt werden. Die Sequenzen müssen den, aus den Simulationsergebnissen ermittelten, Kombinationen aus Datenbankbereich und Anfragetyp zugeordnet und aufsummiert werden. Diese Summe kann nun durch die Anzahl der ermittelten Sequenzen pro Datenbankbereich und Anfragetyp dividiert werden, um die durchschnittliche Anzahl an übertragenen Datensätzen pro Datenbankbereich und Anfragetyp zu erhalten.

Mit diesen Informationen kann nun ein Lasttest parametrisiert werden, um die Leistungsfähigkeit einer (partitionierten) Datenbank bzgl. einer zukünftig zu erwartenden Last zu evaluieren. Im Falle von partitionierten Datenbanken können diese Informationen zusätzlich für die Verteilung der Tabellen auf die einzelnen Partitionen verwendet werden, um Tabellen mit sehr hohen Zugriffsraten oder einer sehr großen Anzahl zu übertragender Datensätze auf eine separate Partition zu verlagern.

6.8. Aussagen aus der Analyse

Neben den Informationen, welche durch Simulation erlangt werden können, ergeben sich bereits aus den in dieser Arbeit vorgestellten Analyse- und Messwerkzeugen nützliche Hinweise für die Konfiguration und die Analyse von ABAP-Programmen. In den folgenden Abschnitten wird nun zunächst die Analyse von ABAP-Programmen hinsichtlich der Invalidierung von gepufferten Objekten dargestellt. Den Abschluss dieses Kapitels bildet ein Vorgehen für die Ermittlung der benötigten Dimensionen der Tabellenpuffer.

6.8.1. Häufigkeit von Invalidierungen

Werden Datenbankobjekte, welche sich im Puffer befinden, verändert, so wird dies in der Datenbank gespeichert, und die zugehörigen Puffereinträge werden invalidiert. Dies sollte grundsätzlich nicht vorkommen, da dies nach der Karenzzeit ein Nachladen der Objekte in den Puffer verursacht und damit die Anzahl der Datenbankzugriffe erhöht. Bei häufigem Auftreten von Invalidierungen kann dies starke, negative Auswirkungen auf die Performance des Systems haben. Mögliche Lösungen für eine solche Situation ist eine Überarbeitung des ABAP-Programms oder das Entfernen der Tabelle aus der Pufferung.

Da solche Invalidierungen häufig in „Custom-Code“, also in von Kunden selbstentwickelten Programmen vorkommen, ist eine Überprüfung dieser Implementierungen vor dem operativen Gebrauch unumgänglich.

Mit Hilfe der in der vorliegenden Arbeit vorgestellten Analyse- und Messwerkzeuge kann dies automatisiert durchgeführt werden. Für die Parametrisierung des Modells wurde jeder Lastschritt mit Hilfe des SAP-Performance-Trace analysiert. Dieser Trace bietet die folgenden, für diesen Anwendungsfall relevanten, Informationen:

- Typ und Anzahl von SQL Zugriffen
- Beschreibung des, durch die SQL-Anfragen betroffenen, Datenbankbereichs

Damit können alle modifizierenden Datenbankzugriffe („Insert“, „Update“ und „delete“) identifiziert werden. Durch die Analyse des Kommandos kann aus den Spezifikationen der „where“-Klausel der betroffene Datenbankbereich identifiziert werden.

Um nun zu ermitteln, ob eine dieser Anfragen ein gepuffertes Objekt betrifft, müssen nun die im Puffer enthaltenen Objekte untersucht werden. Diese Informationen kann mit Hilfe des bereits vorgestellten Bausteins „SAPWL_TABSTAT_SINCE_STARTUP“ durchgeführt werden. Die Ausgabe dieses Bausteins enthält die folgenden Informationen für alle gepufferten Objekte:

- Zustand der gepufferten Objekte
- Beschreibung der Objekte anhand von Datenbankbereichen

Wird dieser Baustein also nach der Durchführung des Workloads aufgerufen, so werden die innerhalb des Workloads gepufferten Objekte dargestellt. Durch einen Vergleich der gepufferten Objekte mit den modifizierenden SQL-Anfragen anhand der Datenbankbereiche können so diejenigen SQL-Statements und deren aufrufende Programme identifiziert werden, welche Invalidierungen von Einträgen der Tabellenpuffer erzeugen.

Gegenüber statischen Analysen von ABAP-Programmen und DDIC-Objekten, bietet dieses Vorgehen den Vorteil, dass nur diejenigen SQL-Anfragen identifiziert werden, welche im realen Workload Invalidierungen hervorrufen. Da Tabellen auf verschiedene Arten gepuffert werden können, ist es bei „single record“- und „generic key“-Pufferung wichtig, ob die zu modifizierenden Datensätze tatsächlich im Workload verwendet werden und damit überhaupt im Puffer enthalten sind.

6.8.2. Benötigter Speicher für Tabellenpuffer

Wie in den vorangegangenen Kapiteln beschrieben, können zu gering dimensionierte Puffer erheblichen Einfluss auf die Performance des SAP-WebAS-ABAP haben. Die Ausprägung der benötigten Ressourcen ist jedoch zu großen Teilen abhängig vom prozessierten Workload. In diesem Abschnitt wird ein Vorgehen dargestellt, welches die Messung der benötigten Ressourcen in den Tabellenpuffern ermöglicht. Wie bereits dargestellt, wird die Kapazität der Tabellenpuffer durch die folgenden Dimensionen definiert:

- Größe in Kilobyte
- Maximale Anzahl von Entrys

Für den „single record“-Puffer ist ein Entry mit einem Datensatz gleichzusetzen, wohingegen ein Entry im „generic key“-Puffer als ein Datenbankbereich verstanden wird. Da innerhalb eines Workloads auch Daten erzeugt werden können, welche für die Pufferung vorgesehen sind, ist die Größe der Puffer nicht als statisch zu betrachten. Um die Puffer optimal an einen Workload anzupassen, muss zunächst der grundsätzliche Ressourcenbedarf ermittelt werden, um darauffolgend die Wachstumsrate durch den ausgeführten Workload zu bestimmen.

6.8.2.1. Grundsätzlicher Bedarf an Ressourcen

Der in Abschnitt 4.2.3 vorgestellt Baustein „SAPWL_TABSTAT_SINCE_STARTUP“ liefert alle zum Ausführungszeitpunkt im Puffer enthaltenen Objekte. Dies beinhaltet die folgenden Informationen:

- Anzahl Datenbankbereiche
- Anzahl Datensätze pro Bereich
- Größe des Bereichs in Kilobyte
- Art der Pufferung

Mit diesen Informationen kann nun die für die beiden Tabellenpuffer die Anzahl der benötigten Entrys, sowie die benötigte Menge an Speicher in Kilobyte ermittelt werden. Um diese Informationen zu erhalten, muss die Rückgabe des Bausteins nach einer einmaligen Durchführung des Workloads nach den verschiedenen Arten der Pufferung sortiert werden. Werden die Einträge „Größe in Kilobyte“ und „Anzahl Entrys“ für beide Puffer aufaddiert, erhält man den Mindestbedarf an Ressourcen für die Puffer. Da zusätzlich Speicher für die Verwaltung des Puffers selbst benötigt wird, müssen (den in dieser Arbeit durchgeführten Wiederholungen zu folge) ca. 20 Prozent des ermittelten Bedarfs zusätzlich konfiguriert werden.

6.8.2.2. Wachstumsrate

Um nun die Wachstumsrate und die damit verbundene zusätzliche Menge an Speicher und Entrys zu ermitteln, muss das Vorgehen aus Abschnitt 6.8.2.1 mehrere Male wiederholt werden. Aus der Differenz der jeweils ermittelten Werte können die folgenden Informationen zu den hinzukommenden Datensätzen errechnet werden:

- Anzahl der Datensätze
- Speicherbedarf der Datensätze

Da der Baustein „SAPWL_TABSTAT_SINCE_STARTUP“ nur jeweils die Größe eines Entrys zurückgibt, kann die durchschnittliche Größe eines Datensatzes aus der Division der Größe durch die Anzahl der enthaltenen Datensätze ermittelt werden. Mit dieser Information und der Anzahl von Datensätzen, welche pro Ausführung des Workloads zusätzlich gepuffert werden müssen, können die für die Tabellenpufferung benötigten Ressourcen in Abhängigkeit der prognostizierten Anzahl von Ausführungen des Workloads berechnet werden.

6.9. Fazit

In diesem Kapitel erfolgte die Evaluation des Modells. Dazu wurden zunächst Anforderungen an einen Workload erhoben und auf deren Basis ein solcher für die Evaluation in Form eines Vergleichs von Mess- und Simulationswerten konzipiert. Darauf aufbauend wurden verschiedene Last- bzw. Konfigurationsszenarien erstellt, welche eine aussagekräftige Evaluation des Modells gewährleisteten. Die erhaltenen Messwerte wurden vorgestellt und die Tragfähigkeit des Modells unter diesem Workload ermittelt. Damit wird Forschungsfrage 3 beantwortet, indem die Evaluation des Modells durchgeführt und dessen Tragfähigkeit unter einen Referenz-Workload nachgewiesen wurde.

Damit eröffnet sich die Möglichkeit, verschiedene Konfigurationen anhand von einzelnen Parametern der System- und Workload-Modellierung zu evaluieren. Um die identifizierten System-Komponenten bzw. deren negative Einflüsse auf die System-Performance zu überprüfen, kann das Modell in verschiedenen Parametrisierungen verwendet werden. Einflüsse, die bei Workloads mit nicht optimaler Unterstützung der parallelen Ausführung auftreten, können durch eine Änderung der Parametrisierung der Client-Komponenten im Modell nachgebildet werden. Dazu kann die Ausführungsreihenfolge der Workload-Schritte so angepasst werden, dass die Schritte nacheinander mit einer konstanten Thinktime durch die Benutzer aufgerufen werden. Weitere Variationsmöglichkeiten der Parametrisierung bestehen darin, innerhalb einer Simulationsausführung jeweils nur einen Workload-Schritt

durchzuführen. Damit können der gesamte Workload oder einzelne Schritte gezielt auf deren Verhalten bei massiv paralleler Ausführung analysiert werden.

Eine weitere Möglichkeit der Verwendung des Modells hinsichtlich der Überprüfung unterschiedlicher Parametrisierungen besteht in der Variation der Anzahl von verfügbaren CPUs und Work-Prozessen. Die einzelnen Vorgänge zur Bestimmung der benötigten Anzahl von CPUs- und Work-Prozessen wurden bereits in den Absätzen 6.7.1 und 6.7.2 dargestellt. Eine Kombination dieser beiden Vorgänge könnte die Ermittlung der Lastkapazitäten eines bestehenden Systems sowie die Planung der Hardware-Ressourcen für zu erwartende Lastanforderungen unterstützen.

Unabhängig von den verschiedenen Möglichkeiten der Parametrisierung zeigt sich, dass Software-Architekturen, welche eine (konfigurierbare) maximale Bearbeitungskapazität unterstützen, deutliche Vorteile bzgl. der Skalierbarkeit und des Performanceverhaltens in Hoch- bzw. Überlastsituationen aufweisen können. Die Ergebnisse der in der vorliegenden Arbeit durchgeführten Messungen unterstützen diese Aussage. Während bei einem SAP-ERP-System die maximale Kapazität der parallel bearbeiteten Anfragen durch die Anzahl der konfigurierten Dialog-Work-Prozesse begrenzt und damit die Auswirkungen von Überlastsituationen auf die Performance gesteuert werden können, zeigt sich das Gegenteil auf Seiten des Betriebssystems. Hier bedient die CPU eine nicht begrenzte Anzahl von Prozessen durch eine hohe Frequenz von Kontextwechseln scheinbar gleichzeitig. Dies bringt einen großen Verwaltungsaufwand für die einzelnen Prozesse mit sich, welcher in Überlastsituationen einen großen Anteil der verfügbaren Ressourcen benötigt. Damit kann aus den Ergebnissen der vorliegenden Arbeit eine konfigurierbare, aber stringente Begrenzung der maximalen Anzahl von parallel bearbeiteten Anfragen für Software-Architekturen abgeleitet werden.

7. Zusammenfassung und Ausblick

Ziel der vorliegenden Arbeit ist es, die Performance unter einer steigenden Anzahl von parallelen Benutzern durch Modellierung und Simulation zu prognostizieren. Die Arbeit unterliegt der Annahme, dass SAP-ERP-Systeme in der Praxis zwar unterschiedliche Programme ausführen, diese jedoch alle die durch den SAP-Kernel bereitgestellte Infrastruktur aus Puffern, Sperrverwaltung usw. verwenden. Die Durchführung der Arbeit wurde durch 3 Forschungsfragen geleitet, deren Beantwortung im folgenden Unterkapitel dargestellt wird. Die Arbeit endet mit der Darstellung der Limitationen des präsentierten Ansatzes sowie Ansatzpunkten für künftige Forschungsvorhaben in diesem Bereich.

7.1. Zusammenfassung

Da die Bearbeitung des in der vorliegenden Arbeit beschriebenen Forschungsvorhabens grundsätzlich in 3 Forschungsfragen aufgeteilt ist, werden die in dieser Arbeit erlangten Ergebnisse entsprechend den Forschungsfragen präsentiert.

> Forschungsfrage 1:
>
> Welche Systemkomponenten haben Einfluss auf die Performance und wie wirken sich deren internen Vorgänge auf die Performance eines SAP-ERP-Systems aus, welche Performancedaten werden von einem SAP-ERP-System aufgezeichnet und wie können diese für eine externe Analyse extrahiert werden?

In Forschungsfrage 1 wurde der Einsatzbereich eines SAP-ERP-Systems analysiert, um eine Metrik für die Performance-Messung zu identifizieren. Da es sich bei dieser Art von System vor allem um transaktionsorientierte Systeme zur Bearbeitung von Benutzeranfragen handelt, wurde für die vorliegende Arbeit die Antwortzeit als Bewertungsgröße für die System-Performance gewählt. Durch eine ausführliche Analyse der Komponenten von Antwortzeit und Bearbeitung von Benutzeranfragen konnten diese bzgl. der Integration in das Performance-Modell evaluiert werden. Dabei wurden die Komponenten nicht nur nach deren maximal möglichen Einfluss auf die Antwortzeit bewertet, sondern vor allem nach deren Variabilität während eines Workloads. Aus diesem Grund wurde neben den Lastschritten und der Lastintensität der Fokus der Modellierung auf die Tabellenpuffer- und Sperrverwaltung gelegt. Obwohl zu geringe Ressourcen im „Extended Memory“ ebenso schwerwiegenden, negativen Einfluss auf die System-Performance haben können, wurden diese nicht modelliert da die ausreichende Konfiguration bereits durch die einmalige Vermessung des Workloads bestimmt werden kann und dies keiner Simulation bedarf. Im Gegensatz dazu können als Beispiel die Auswirkungen ineffizient gesetzter Sperren durch einmalige Ausführung kaum bestimmt werden. Deren Beeinflussung der Antwortzeit wird erst bei mittlerer bis hoher Last und teilweise erst in Wechselwirkung mit anderen Programmen erkennbar. Nach der Auswahl der zu modellierenden Komponenten wurde innerhalb eines weiteren Kapitels erarbeitet, wie die benötigten Daten aus dem System extrahiert werden können, damit diese für die Parametrisierung und Evaluation des Modells zur Verfügung stehen. Dazu wurden in einem ersten Schritt die Informationen über die Zustände der Objekte im Puffer, die Verwendung von Sperren innerhalb eines Programms sowie dessen Zugriffe auf Datenbank und Puffer innerhalb des Systems identifiziert. In einem zweiten Schritt wurden Bausteine ausfindig

gemacht und teilweise modifiziert, sodass diese Informationen automatisch der in der vorliegenden Arbeit implementierten Analysekomponente zur Verfügung standen. Einzig die Performance-Traces konnten in dieser Implementierung nicht automatisch zugänglich gemacht werden, sodass ein manueller Weg über Export auf Dateisystemebene und Import in das Analyse-Tool konzeptioniert und implementiert wurde. Die folgenden Informationen wurden in der vorliegenden Arbeit automatisch aus dem System extrahiert bzw. ermittelt:

- Aktuelle Zustände gepufferter Inhalte
- STAD Performance Records
- Analyse der Puffer- und Sperrobjektbeschreibung
- Benötigte Speicherressourcen pro Workload-Schritt

Mit diesen Informationen ist es möglich, die Auswirkungen eines jeden Workload-Schritts auf gepufferte Ressourcen, auf die Datenbank und deren Sperranfragen zu analysieren. Mit Hilfe der STAD-Performance-Records konnten die Ergebnisse der Simulation mit den Performance-Werten des realen Systems im Labor verglichen werden.

Forschungsfrage 2:

Welche technischen und logischen Systemkomponenten müssen modelliert und wie müssen diese parametrisiert werden, um ausreichend genaue Simulationsergebnisse zu erzielen?

Diese Forschungsfrage zielt auf die Konzeption des Artefakts. Dazu wurden in einem ersten Schritt die Anforderungen an die Modellierung der einzelnen Komponenten bzgl. der abzubildenden internen Abläufe erhoben. Dies sind vor allem die Eigenschaften, welche in Forschungsfrage 1 ermittelt und beschrieben wurden. In einem zweiten Schritt wurde dann eine Modellierung entwickelt, welche zusätzlich zu den bereits identifizierten technischen Systemkomponenten logische Konstrukte beinhaltet. Dies gilt vor allem für die Umsetzung von Sperrverwaltungsmechanismen, welche nach Franks (2011) noch nicht ausreichend durch die zur Verfügung stehenden LQN-Modellkomponenten unterstützt werden. Die Modellierung dieser Systemkomponente konnte nur durch Einfügen zusätzlicher logischer Modellkomponenten erreicht werden. Ebenso stehen für die Modellierung von Puffern, welche durch die Dimensionen Speichergröße und Anzahl der Einträge definiert sind, keine direkten Umsetzungsmöglichkeiten in der LQN-Syntax zur Verfügung. Da LQN-Modellkomponenten, mit Ausnahme der sehr einfachen Semaphore, keine Zustandsspeicherung beinhalten, können damit Pufferverdrängungen nicht direkt modelliert werden. Dies wurde durch die Ermittlung und Parametrisierung einer vom Workload abhängigen Verdrängungswahrscheinlichkeit gelöst.

Diese Forschungsfrage dient der Demonstration und Evaluation der Wirkungsweise des entwickelten Artefakts. Dazu wurden zuerst die Anforderungen an einen Workload für die Evaluation erhoben. Dabei wurde neben der Vergleichbarkeit der Ergebnisse auf die Nutzung

der durch den SAP-Kernel bereitgestellten Infrastruktur fokussiert, da dies eine entscheidende Gemeinsamkeit aller SAP-ERP-Systeme auf Basis des SAP-WebAS-ABAP darstellt.

> Forschungsfrage 3:
>
> Welche Unterschiede treten zwischen den Simulationsergebnissen und den im Labor gemessenen Werten auf, und was sind die Ursachen hierfür?

Um diese Vergleichbarkeit zu gewährleisten, muss neben dem dedizierten Ablauf des Geschäftsprozesses auch die gleiche Verteilung der sogenannten „Think time“ zwischen zwei Anfragen eines Benutzers, und die garantiert gleiche Anzahl an parallelen Benutzern gewährleistet sein. Zu diesem Zweck wurde mittels des Axis2-Frameworks (Apache-Software-Foundation 2009) ein auf JAVA (Sun 2004) und SOAP (W3C 2004) basierender Lastgenerator entwickelt. Dieser baut auf einer Client-Server-Architektur auf, sodass die Lasterzeugung auf mehrere Hosts verteilt werden kann. Dabei kann die Anzahl paralleler Benutzer durch die Implementierung eines Benutzers innerhalb eines Threads (Sun 2004) sowie der Erwartungswert der exponentiell-verteilten „Think time“ beliebig konfiguriert werden. Auf Basis der bereits in Forschungsfrage 1 ermittelten Extraktionsmechanismen von Performance-Werten in die implementierte Analysekomponente konnte damit für diesen Workload ein Modell erstellt und parametrisiert werden.

Die Tragfähigkeit des Modells wurde anhand des Vergleichs der Simulationsergebnisse mit den Messwerten durchgeführt. Insgesamt zeigte die Evaluation, dass die Vorhersage der Antwortzeit unter einer steigenden Anzahl von parallelen Benutzern im Niedrig- und Mittellastbereich sehr gut war. Die Abweichungen im Hochlastbereich konnten auf Ursachen im Betriebssystem zurückgeführt werden. Durch die Simulation des CPU-Verbrauchs, welcher indirekt Ursache für die Abweichung war, konnte die Grenze zum Hochlastbereich durch die Anzahl der parallelen Benutzer vorhergesagt werden. Analog dazu konnte innerhalb der gewählten Szenarien die Modellierung der Anzahl von Abbrüchen indirekt durch deren Wahrscheinlichkeit evaluiert werden. Die Gegenüberstellung von relativer Häufigkeit der gemessenen Abbrüche aufgrund von Überschreitungen der konfigurierten maximalen Antwortzeit mit der durch Simulation ermittelten Wahrscheinlichkeit zeigte analog zur Antwortzeit sehr gute Annäherungen im Mittel- und Hochlastbereich.

Im Detail zeigte sich, dass die Modellierung der Sperrverwaltung bzgl. der wechselseitigen Sperren aussagekräftige Ergebnisse liefert. Damit können neben der Auswirkung auf die Antwortzeit ebenso Sperrobjekte mit hohen Wartezeiten identifiziert werden. Dies ist entweder auf ineffiziente Verwendung oder Programmierung zurückzuführen und kann nicht durch eine Modifikation der Konfiguration verändert werden. Dieses Vorgehen stellt eine Analyse dar, die in dieser Art und Weise so bisher nicht durchgeführt werden konnte. Zusätzlich konnte die Umsetzung des Modells der Sperrverwaltung die Kapselung von Datenbankanfragen innerhalb von Sperrobjekten zeigen. So konnte die Integration der Auswirkungen unterschiedlicher Antwortzeiten von Datenbankanfragen auf die Sperrdauer und damit auf die Gesamtantwortzeit gezeigt werden.

Wie bereits erwähnt, konnten Verdrängungen gepufferter Objekte nur indirekt über eine Weiterleitungswahrscheinlichkeit modelliert werden. Durch die Wahl einer geeigneten

Wahrscheinlichkeit konnte das Modellierungskonzept bestätigt werden. Es zeigte sich, dass bei zu gering dimensionierten Puffern in der Simulation die Anzahl der Anfragen an die Datenbank gegenüber den Szenarien mit ausreichend konfigurierten Puffern analog zu den Messergebnissen angestiegen ist.

7.2. Annahmen und Limitationen

Es gibt verschiedene Abstraktionsschichten, auf denen SAP-ERP-Systeme modelliert und simuliert werden können. Während viele wissenschaftliche Arbeiten sich mit der betriebswirtschaftlichen Performance dieser Systeme beschäftigen, zielt die vorliegende Arbeit auf die Modellierung und Simulation der technischen Performance eines SAP-ERP-Systems.

Dem in der vorliegenden Arbeit entwickelten Performance-Modell eines SAP-ERP-Systems liegt die grundsätzliche Annahme zugrunde, dass sich existierende Systeme zwar in den implementierten und ausgeführten Programmen unterscheiden, die Laufzeitumgebung dieser Programme jedoch in allen Systemen identisch ist. Dies ist dadurch begründet, dass alle ABAP-Programme nur innerhalb des SAP-Kernels ausgeführt werden, und dieser die in dieser Arbeit modellierten Mechanismen zur Pufferung und Sperrverwaltung bereitstellt. Damit ist das Modell aufgrund der Fokussierung auf die Architektur des ERP-Systems der Firma SAP nicht allgemeingültig für ERP-Systeme. Durch die einheitlichen Infrastruktur-Komponenten des SAP-Kernels kann jedoch das in der vorliegenden Arbeit konzipierte Artefakt als Grundlage für die Modellierung und Simulation von beliebigen SAP-ERP-Systemen auf Basis des SAP-WebAS-ABAP dienen. Gültigkeit der Modellierung von Systemkomponenten, im Besonderen der Puffer- und Sperrverwaltungsmechanismen, konnte dabei in der Evaluation der vorliegenden Arbeit gezeigt werden.

Neben der Vorhersage der Performance unter einer steigenden Anzahl von Benutzern kann der Simulationsansatz auch für gewisse Aspekte der Kapazitätsplanung und dem Vergleich verschiedener Systemkonfigurationen verwendet werden. Aufgrund der Vielzahl von Konfigurationsparametern und der daraus resultierenden Anzahl von Kombinationsmöglichkeiten erlaubt das entwickelte Artefakt keine Optimierungsvorgänge, basierend auf Simulation und Vergleich aller möglichen Konfigurationen.

Das in der vorliegenden Arbeit entwickelte und evaluierte Artefakt fokussiert auf den SAP-WebAS ABAP und dessen interne Abläufe. Die Modellierung, Simulation und Evaluation der Performance des unterliegenden Datenbanksystems ist nicht Ziel dieser Arbeit. Für die Parametrisierung der Datenbankkomponenten wurden die jeweiligen Messergebnisse der einzelnen Konfigurationen verwendet. Für die praktische Anwendbarkeit stellt dies eine Einschränkung dar, da die Performance des Datenbanksystem ex ante ermittelt werden muss. Da aus der Vermessung, Analyse und Simulation der Workload-Schritte jedoch die SQL-Anfragen, welche an das Datenbanksystem gestellt werden, ermittelt werden können, ist es möglich, auf Basis dieser Informationen einen Lastgenerator zu erstellen, welcher die flexible Ermittlung der Performance der Datenbankkomponenten für die Parametrisierung des Modells ermöglicht.

Die Anwendung von Simulationstechniken generell und damit diese Arbeit im Speziellen unterliegen Einschränkungen bzgl. der Aussagekraft und der Anwendbarkeit in einzelnen

Szenarien. Wie bereits dargestellt, sinkt die Genauigkeit der Vorhersage im Hochlastbereich stark ab. Dies ist zu einem großen Teil durch Einflussfaktoren außerhalb des SAP-Systems verursacht und nur durch Lasttests der Hard- und Software zu ermitteln.

7.3. Ausblick

Der auf Basis des konzipierten Artefakts entwickelte Prototyp konnte in der vorliegenden Arbeit bzgl. der aufgestellten Anforderungen positiv evaluiert werden. Die bereits erwähnte Einschränkung der fehlenden Datenbanksubmodelle ist ein weiterer großer Bereich für zukünftige Forschung. Dies könnte auf mehrere Arten gelöst werden. Zum einen besteht die Möglichkeit, analog zum LQN-Modell des SAP-WebAS-ABAP ein Modell für die Datenbankkomponenten zu erstellen. Dabei ist jedoch zu beachten, dass dieses Modell spezifisch bzgl. des Datenbanksystems ist. Da der SAP-WebAS-ABAP mehrere Datenbanksysteme unterstützt, wäre es nötig, Submodelle für diese verschiedenen Datenbanksysteme zu erstellen. Ein weiteres mögliches Forschungsgebiet ist die Parametrisierung der Datenbankmodellkomponenten durch historische Daten. Dazu könnte der LQN-Simulator so erweitert werden, dass die Servicezeit einer Komponente als mehrdimensionale Funktion parametrisiert werden kann. Damit wäre es möglich, Servicezeiten in Abhängigkeit von Anzahl paralleler Anfragen und Art der Anfragen in das Modell zu integrieren und so eine lastabhängige Servicezeit in die Simulation zu integrieren. Dazu könnten historische Daten mittels multidimensionaler Regressionsverfahren wie den in (Tertilt/Krcmar 2011; Tertilt et al. 2010) beschriebenen evolutionären Algorithmen analysiert und die Servicezeiten der Datenbankkomponente lastabhängig parametrisiert werden.

Für eine einfache und vollautomatisierte Anwendung sind noch einige Entwicklungen und Erweiterungen nötig. Dazu zählt das automatische Auslesen der SAP-Performance-Traces. Diese konnten in der vorliegenden Arbeit bereits im Zuge der Workload-Vermessung automatisiert, parametrisiert, gestartet und gestoppt werden. Für die Auswertung mussten diese jedoch manuell in Dateien auf Betriebssystemebene exportiert werden, um dann mittels der dafür entwickelten Komponente analysiert zu werden. Durch die Entwicklung eines remotefähigen Funktionsbausteins könnten diese manuellen und damit zeitaufwendigen Arbeitsschritte eliminiert, und damit Vermessung, Analyse, Modellerstellung, Parametrisierung und Auswertung automatisiert ausgeführt werden. Damit könnte das in der vorliegenden Arbeit entworfene Artefakt innerhalb eines Tools durch Administratoren verwendet werden, ohne erforderliche Modellierungs- und Simulationskenntnisse.

Ein weiteres Anwendungsfeld für künftige Forschung ist die Entwicklung eines generischen Lastgenerators für Datenbankinstanzen von SAP-ERP-Systemen. Aus der in der vorliegenden Arbeit konzipierten Systemanalyse und –Simulation können Art, Anzahl und Verteilung der SQL-Anfragen eines Workloads an die Datenbank ermittelt werden. Auf dieser Basis wäre es zu untersuchen, in wieweit ein generischer Lastgenerator die tatsächlich erzeugte Last und vor allem die dadurch hervorgerufenen Antwortzeiten der Datenbank nachstellen kann.

Das in der vorliegenden Arbeit eingeführte Submodell der Puffermechanismen basiert auf der Ermittlung der Weiterleitungswahrscheinlichkeit von Datenbankanfragen in Abhängigkeit von Puffergröße und deren Häufigkeit. Um die internen Vorgänge der Puffer detailliert abbilden zu können, sollte erforscht werden, ob und wie der LQN-Formalismus und die dafür implementierten Simulationstools erweitert werden könnten, um die Größe von Puffern

(Anzahl der Einträge und Speichermenge) modellieren zu können. Dabei wären neben den Dimensionen der Puffergröße ebenso die verschiedenen Verdrängungsmechanismen, wie im Falle der SAP-Tabellenpufferung, dem LRU-Mechanismus, zu berücksichtigen.

Literaturverzeichnis

Anderson, G.W.; Mißbach, M. (2005): Last-Testing und Performance-Tuning. SAP Press, Bonn 2005.

Apache-Software-Foundation (2009): Apache Axis2 User's Guide. In: http://axis.apache.org/axis2/java/core/docs/userguide.html, zugegriffen am 12.12.2011.2011.

Bankhofer, U.; Vogel, J. (2008): Datenanalyse und Statistik: Eine Einführung für Ökonomen im Bachelor. Gabler Verlag 2008.

Barham, P.; Dragovic, B.; Fraser, K.; Hand, S.; Harris, T.; Ho, A.; Neugebauery, R.; Pratt, I.; Warfield, A. (2003): Xen and the Art of Virtualization. In: 19th ACM Symposium on Operationg Systems Principles Hrsg., Bolton Landing, NY 2003, S. 164-177.

Barnett, V.; Lewis, T. (1994): Outliers in statistical data. In: International Journal of Forecasting, Vol. 12 (1994) Nr. 1, S. 2.

Bause, F. (1993): Queueing Petri Nets: A Formalism for the Combined Qualitative and Quantitative Analysis of Systems. In: 5th International Workshop on Petri Nets and Performance Models (IEEE) Hrsg. IEEE Computer Society, Toulouse, Frankreich 1993, S. 14-23.

Becker, B. (1993): Statistik. Oldenbourg, München, Wien 1993.

Boegelsack, A. (2010): Performance und Skalierung von SAP ERP Systemen in virtualisierten Umgebungen, University of Technology Munich 2010.

Boegelsack, A.; Gradl, S.; Mayer, M.; Krcmar, H. (2009): SAP MaxDB Administration. Galileo Press 2009.

Bolch, G.; Greiner, S.; de Meer, H.; Trivedi, K.S. (2006): Queueing Networks and Markov Chains: Modeling and Performance Evaluation with Computer Science Applications. John Wiley and Sons, Hoboken, NJ, USA 2006.

Bortz, J.; Döring, N. (1995): Forschungsmethoden und Evaluation. Für Sozialwissenschaftler. 2. vollst. überarb. Aufl., Springer-Verlag GmbH, Berlin, Heidelberg 1995.

Briggs, R.O. (2006): On theory-driven design and deployment of collaboration systems. In: Int. J. Hum.-Comput. Stud., Vol. 64 (2006) Nr. 7, S. 573-582.

Broy, M. (1997): Informatik. Eine grundlegende Einführung: Band 1: Programmierung und Rechnerstrukturen. Springer, Berlin, Heidelberg 1997.

Chen, P.M.; Patterson, D.A. (1994): A new approach to I/O performance evaluation: self-scaling I/O benchmarks, predicted I/O performance. In: ACM Trans. Comput. Syst., Vol. 12 (1994) Nr. 4, S. 308-339.

Chen, S.; Moreland, D.; Nepal, S.; Zic, J. (2008): Yet Another Performance Testing Framework. *Proceedings of the 19th Australian Conference on Software Engineering* (S. 170-179): IEEE Computer Society.

Claus, V.; Schwill, A. (2003): Duden. Informatik. 1. Bd., Bibliographisches Institut, Mannheim 2003.

Datta, A.; Ioannou, P.A. (1994): Performance Analysis and Improvement in Model Reference Adaptive Control. In: IEE Transactions on Automatic Control, Vol. 39 (1994) Nr. 12, S. 18.

Fahrmeier, L.; Künstler, R.; Pigeot, I.; Tutz, G. (1999): Statistik - Der Weg zur Datenanalyse. Springer, München 1999.

Ferschl, F. (1970): Markovketten. Springer-Verlag, Heidelberg 1970.

Franks, G. (2011): Simulating layered queueing networks with passive resources. *Proceedings of the 2011 Theory of Modeling & Simulation Symposium: DEVS Integrative M&S Symposium* (S. 8-15). Boston, Massachusetts: Society for Computer Simulation International.

Franks, G.; Hubbard, A.; Majumdar, S.; Neilson, J.; Petriu, D.; Rolia, J.; Woodside, M. (1996a): A Toolset for Performance Engineering and Software Design of Client-Server Systems. In: Performance Evaluation, Vol. 24 (1996a), S. 117-135.

Franks, G.; Majumdar, S.; Neilson, J.; Petriu, D.; Rolia, J.; Woodside, M. (1996b): Performance Analysis of Distributed Server Systems. In: Computer Hrsg. Citeseer, 1996b, S. 15-26.

Franks, G.; Maly, P.; MWoodside, M.; Petriu, D.; Hubbard, A. (2010): Layered Queueing Network Solver and Simulator User Manual. Department of Systems and Computer Engineering, Carleton University, Ottawa, Canada 2010.

Gat, E. (1998): Three-layer architectures. In: Artificial intelligence and mobile robots: case studies of successful robot systems. Hrsg.: Kortenkamp, D.; Bonasso, R.; Murphy, R. MIT Press, 1998.

Gibson, J.C. (1970): The Gibson Mix (TR 00.2043). IBM Systems Development Division, Poughkeepsie, NY, USA 1970.

Gradl, S.; Mayer, M.; Danciu, A.; Escrihuela, R.; Wittges, H.; Krcmar, H. (2011a): Measuring the Buffer Occupation of SAP ERP System Applications. In: International Conference on Enterprise Information Systems Hrsg., Beijing, China 2011a, S. 266-272.

Gradl, S.; Mayer, M.; Danciu, A.; Wittges, H.; Krcmar, H. (2011b): Understanding the Performance Behavior of a SAP ERP System for the Use of Queuing Models. In: 10th International Conference on Modeling and Applied Simulation Hrsg., Rome 2011b.

Gross, D. (2008): Fundamentals of queueing theory. Wiley 2008.

Group, R.T.a.D.S. (2011): Real-Time and Distributed Systems Group. In: zugegriffen am 12.07.2011.2011.

Heinrich, L.J.; Lehner, F. (2005): Informationsmanagement. Planung, Überwachung und Steuerung der Informationsinfrastruktur. 8. vollst. überarb. u. erg. Aufl., Oldenbourg, München 2005.

Hellermann, H. (1975): Computer System Performance. McCraw-Hill Book Company, New York 1975.

Hevner, A.; March, S.; Park, J.; Ram, S. (2004): Design Science in Information Systems Research. In: MIS Quarterly, Vol. 28 (2004) Nr. 1, S. 30.

Hoetzel, A.; Benhaim, A.; Griffiths, N.; Holliday, C. (1998): Benchmarking in Focus. IBM 1998.

Hu, L.; Gorton, I. (1997): Performance Evaluation for Parallel Systems: A Survey (9707). University of New South Wales, Sydney, Australia 1997.

Hubbard, A. (1997): S P E X - Software Performance Experiment Driver. In: http://www.sce.carleton.ca/rads/lqns/lqn-documentation/spex.txt, zugegriffen am 12.07.2011.2011.

IBM (2011a): iostat Command. In: http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp?topic=%2Fcom.ibm.aix.cmds%2Fdoc%2Faixcmds5%2Ftopasout.htm, zugegriffen am 08.07.2011.2011.

IBM (2011b): mpstat Command. In: http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp?topic=%2Fcom.ibm.aix.cmds%2Fdoc%2Faixcmds5%2Ftopasout.htm, zugegriffen am 08.07.2011.2011.

IBM (2011c): Topas. In: http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp?topic=%2Fcom.ibm.aix.cmds%2Fdoc%2Faixcmds5%2Ftopas.htm, zugegriffen am 08.08.2011.2011.

IBM (2011d): topas CEC Analyser. In: https://www.ibm.com/developerworks/wikis/display/WikiPtype/topas+CEC+Analyser, zugegriffen am 08.07.2011.2011.

IBM (2011e): topasout Command. In: http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp?topic=%2Fcom.ibm.aix.cmds%2Fdoc%2Faixcmds5%2Ftopasout.htm, zugegriffen am 08.07.2011.2011.

IBM (2011f): topasrec Command. In: http://publib.boulder.ibm.com/infocenter/aix/v6r1/index.jsp?topic=%2Fcom.ibm.aix.cmds%2Fdoc%2Faixcmds5%2Ftopasrec.htm, zugegriffen am 07.08.2011.2011.

Jaenecke, P. (1982): Grundzüge einer Meßtheorie. In: Journal for General Philosophy of Science, Vol. 13 (1982) Nr. 2, S. 234-279.

Jain, R. (1991): The Art of Computer Systems Performance Analysis: Techniques for Experimental Design, Measurement, Simulation, and Modeling. Wiley/Interscience, New York, NY, USA 1991.

Jehle, H. (2010): Performance Measurement of an SAP Enterprise Portal System in a Virtualized Environment, University of Technology Munich 2010.

Jenq, B.C.; Kohler, W.H.; Towsley, D. (1988): A queueing network model for a distributed database testbed system. In: Software Engineering, IEEE Transactions on, Vol. 14 (1988) Nr. 7, S. 908-921.

Jonkers, H. (1994): Queueing models of parallel applications: the Glamis methodology. In: Proceedings of the 7th International Conference on Computer Performance Evaluation Hrsg. Springer-Verlag New York, Inc., Secaucus, NJ, USA 1994, S. 123-138.

Keller, H.; Kruger, S. (2002): ABAP Objects: Introduction to Programming SAP Applications. Addison-Wesley Professional 2002.

Kemmesies, O. (2009): Prozeßmodellierung und Parameteridentifikation von Mehrphasenströmungsprozessen in porösen Medien. Grin Verlag 2009.

Krcmar, H. (2010): Informationsmanagement. 5. vollst. überarb. u. erw. Aufl., Springer-Verlag, Berlin, Heidelberg 2010.

Krüger, G. (2006): Handbuch der Java-Programmierung. Addison-Wesley, München 2006.

Lazowska, E.D.; Zahorjan, J.; Graham, G.S.; Sevcik, K.C. (1984): Quantitative System Performance: Computer System Analysis Using Queueing Network Models. Prentice-Hall, Englewood Cliffs, NJ, USA 1984.

Leimbach, T. (2008): The SAP Story: Evolution of SAP within the German Software Industry. In: IEEE Ann. Hist. Comput., Vol. 30 (2008) Nr. 4, S. 17.

Lilja, D.J. (2000): Measuring Computer Performance - A Practitioner's Guide. Cambridge University Press, Cambridge 2000.

Lilja, D.J.; Yi, J.J. (2006): Statistical Techniques for Computer Performance Analysis. CRC Press Taylor & Francis Group, Boca Raton 2006.

Little, J.D.C. (1961): A Proof for the Queueing Formula L = lamda * W. In: Operations Research, Vol. 9 (1961) Nr. 3, S. 383-387.

March, S.; Smith, G. (1995): Design and natural science research on information technology. In: Decision Support Systems - Special issue on WITS '92, Vol. 15 (1995) Nr. 4, S. 251-266.

Marquard, U.; Götz, C. (2008): SAP Standard Application Benchmarks - IT Benchmarks with a Business Focus. In: (Vol. 5119). Hrsg.: Kounev, S.; Gorton, I.; Sachs, K. Springer Berlin / Heidelberg, 2008, S. 4-8.

Mascarenhas, E.; Knop, F.; Rego, V. (1995): ParaSol: a multithreaded system for parallel simulation based on mobile threads. *Proceedings of the 27th conference on Winter simulation* (S. 690-697). Arlington, Virginia, United States: IEEE Computer Society.

Menascé, D.A.; Almeida, V.A.F. (2002): Capacity Planning for Web Services: Metrics, Models, and Methods. Prentice-Hall, Englewood Cliffs, NJ 2002.

Menascé, D.A.; Almeida, V.A.F. (2000): Scaling for E-Business: Technologies, Models, Performance, and Capacity Planning. 1st. Aufl., Prentice Hall 2000.

Menascé, D.A.; Dowdy, L.W.; Almeida, V.A.F. (2004): Performance by Design: Computer Capacity Planning By Example. 1st. Aufl., Prentice Hall 2004.

Meyer, E.; Guicking, D. (1974): Schwingungslehre. Vieweg-Verlag, Braunschweig 1974.

Nudd, G.R.; Kerbyson, D.J.; Papaefstathiou, E.; Perry, S.C.; Harper, J.S.; Wilcox, D.V. (2000): Pace - A Toolset for the Performance Prediction of Parallel and Distributed Systems. In: International Journal of High Performance Computing Applications, Vol. 14 (2000).

Nunamaker, J.F.; Dennis, A.R.; Valacich, J.S.; Vogel, D.; George, J.F. (1991): Electronic meeting systems. In: Commun. ACM, Vol. 34 (1991) Nr. 7, S. 40-61.

Omari, T.; Franks, G.; Woodside, M.; Pan, A. (2007): Solving Layered Queueing Networks of Large Client-Server Systems with Symmetric Replication. In: The Journal of Systems and Software, Vol. 80 (2007), S. 18.

Orth, B. (1974): Einführung in die Theorie des Messens. Kohlhammer, Stuttgart 1974.

Park, A.; Becker, J.C. (1990): IOStone: A Synthetic File System Benchmark. In: Computer Architecture News, Vol. 18 (1990) Nr. 2, S. 45-52.

Parupudi, M.; Winograd, J. (1972): Interactive Task Behavior in a Time-Sharing Environment. In: ACM Annual Conference Hrsg., Boston, MA 1972, S. 680-692.

Petriu, D. (1994): Approximate Mean Value Analysis of Client-Server Systems with Multi-Class Requests. In: ACM SIGMETRICS Conference on Measurement and Modeling of Computer Systems Hrsg., Nashville, TN 1994, S. 77-86.

Pfeffers, K.; Tuunanen, T.; Gengler, C.; Rossi, M.; Hui, W.; Virtanen, V.; Bragge, J. (2006): THE DESIGN SCIENCE RESEARCH PROCESS: A MODEL FOR

PRODUCING AND PRESENTING INFORMATION SYSTEMS RESEARCH. In: First International Conference on Design Science Research in Information Systems and Technology Hrsg.: Chatterjee, S.; Hevner, A., Claremont 2006, S. 83-106.

Prechelt, L. (2001): Kontrollierte Experimente in der Softwaretechnik: Potenzial und Methodik. Springer, Berlin 2001.

Reilly, E.D. Power user. *Encyclopedia of Computer Science* (S. 1419-1419): John Wiley and Sons Ltd.

Risse, T. (2006): Design and Configuration of Distributed Job Processing Systems. Thesis (PhD), Technische Universität Darmstadt 2006.

Rolia, J.; Casale, G.; Krishnamurthy, D.; Dawson, S.; Kraft, S. (2009a): Predictive modelling of SAP ERP applications: challenges and solutions. *Proceedings of the Fourth International ICST Conference on Performance Evaluation Methodologies and Tools* (S. 1-9). Pisa, Italy: ICST (Institute for Computer Sciences, Social-Informatics and Telecommunications Engineering).

Rolia, J.; Casale, G.; Krishnamurthy, D.; Dawson, S.; Kraft, S. (2009b): Predictive Modelling of SAP ERP Applications: Challenges and Solutions. In: ROSSA 2009 Hrsg. ACM Press, Pisa, Italy 2009b.

Rolia, J.A.; Servcik, K.C. (1995): The Method of Layers. In: IEEE Transactions on Software Engineering, Vol. 21 (1995) Nr. 8, S. 689-700.

Royalty, C. (2009): Dialog Processing in the SAP System. In: http://sapwhiz.com/printdoc/908/, zugegriffen am 23.06.2010.2010.

Saltelli, A. (2008): Global Sensitivity Analysis: The Primer. John Wiley 2008.

Sandberg, R.; Goldberg, D.; Kleiman, S.; Walsh, D.; Lyon, B. (1985): Design and Implementation of the Sun Network Filesystem Hrsg. USENIX Association, Berkeley, CA 1985.

SAP (2012a): ABAP Programming (BC-ABA). In: http://help.sap.com/saphelp_47x200/helpdata/en/d3/2e974d35c511d1829f0000e829fbfe/frameset.htm, zugegriffen am 25.03.2012.

SAP (2012b): SAP University Alliances. In: http://www.sap.com/corporate-de/sustainability/csr/education/alliance.epx, zugegriffen am 10.03.2012.2012.

SAP (2011a): Funktionsweisedes SAP Memory-Management-Systems. In: http://help.sap.com/saphelp_nwes73/helpdata/DE/49/32eff3e92e3504e10000000a421937/content.htm, zugegriffen am 12.07.2011.2011.

SAP (2011b): Hinweis 26171 - Mögliche Eingaben im Befehlsfeld ("OK-code"). In: https://websmp230.sap-ag.de/sap/bc/bsp/spn/sapnotes/index2.htm?numm=26171, zugegriffen am 10.02.2011.2011.

SAP (2011c): Kumulation von Sperren. In: http://help.sap.com/saphelp_nwmobile711/helpdata/de/47/dc35ac5bc33b8be10000000a421937/content.htm, zugegriffen am 12.07.2011.2011.

SAP (2011d): Optimistische Sperren. In: http://help.sap.com/saphelp_nwmobile711/helpdata/de/47/dc36275bc33b8be10000000a421937/content.htm, zugegriffen am 12.07.2011.2011.

SAP (2011e): Pufferkomponenten. In: http://help.sap.com/saphelp_nw70ehp2/helpdata/de/c4/3a6dc8505211d189550000e829fbbd/content.htm, zugegriffen am 30.09.2011.2011.

SAP (2011f): SAP-Kalenderpuffer. In: http://help.sap.com/saphelp_nw70ehp2/helpdata/de/c4/3a6e3d505211d189550000e829fbbd/content.htm, zugegriffen am 12.07.2011.2011.

SAP (2011g): SAP-Sperrkonzept. In: http://help.sap.com/saphelp_nwmobile711/helpdata/de/47/daeac909dd3020e10000000a42189d/content.htm, zugegriffen am 12.07.2011.2011.

SAP (2011h): SAP Standard Application Benchmarks. In: http://www.sap.com/solutions/benchmark/index.epx, zugegriffen am 20.08.2011.2011.

SAP (2011i): sapmssy0.

SAP (2011j): Sperrtabelle. In: http://help.sap.com/saphelp_erp60_sp/helpdata/de/12/038337fb02735ce10000009b38f8cf/content.htm, zugegriffen am 12.07.2011.2011.

SAP (2011k): Übersicht über den SAP Web AS ABAP. In: http://help.sap.com/saphelp_dimp50/helpdata/DE/fc/eb2e97358411d1829f0000e829fbfe/content.htm, zugegriffen am 12.07.2011.2011.

SAP (2011l): Welche Tabellen sollten gepuffert werden? In: http://help.sap.com/SAPHELP_NW04/HELPDATA/DE/CF/21F26B446011D189700000E8322D00/CONTENT.HTM, zugegriffen am 10.02.2011.2011.

SAP (2009): sappfpar - Version 7.01.

SAP (2008a): SAPLSSQ0 - Version 7.01.

SAP (2008b): sapmseds - Version 7.01.

SAP (2008c): SAPWL_TABSTAT_SINCE_STARTUP - Version 7.01.

SAP (2007): Hinweis 1063061 - Erläuterung zur Responsetime (Antwortzeit) in STAD/ST03. In: https://websmp130.sap-ag.de/sap/bc/bsp/spn/sapnotes/index2.htm?numm=1063061, zugegriffen am 14.08.2011.2011.

Schendera, C. (2007): Datenqualität mit SPSS. Oldenbourg Wissenschaftsverlag 2007.

Schneider, T. (2008): SAP-Performanceoptimierung. 5. Aufl., Galileo Press, Bonn, Boston 2008.

Schreiber, H.; Thomas, B.; Wolf, F. (1974): Beschreibung eines synthetischen Jobmix für verschiedene Benchmark-Tests GI-NTG Fachtagung Strucktur und Betrieb von Rechensystemen. In: (Vol. 8). Hrsg.: Leilich, H. Springer Berlin / Heidelberg, 1974, S. 218-232.

Sheikh, F.; Woodside, M. (1997): Layered analytic performance modelling of a distributed database system. In: Distributed Computing Systems, 1997., Proceedings of the 17th International Conference on Hrsg., 1997, S. 482-490.

Shein, B.; Callahan, M.; Woodbuy, P. (1989): NFSStone - A Network File Server Performance Benchmark. In: USENIX Summer Tech. Conf. Hrsg., Baltimore, MD 1989, S. 269-275.

Shousha, C.; Petriu, D.; Jalnapurkar, A.; Ngo, K. (1998): Applying Performance Modelling to a Telecommunication System. In: First International Workshop on Software and Performance Hrsg., Santa Fe, New Mexico 1998, S. 1-6.

Silver, M.; Markus, M.L.; Beath, C. (1995): The Information Technology Interaction Model: A Foundation for the MBA Core Course. In: MIS Quarterly, Vol. 19 (1995) Nr. 3, S. 30.

Simon, H. (1996): The sciences of the artificial. MIT Press 1996.

Spillner, A.; Linz, T. (2007): Basiswissen Softwaretest. Dpunkt-Verlag, Berlin 2007.

Sun (2004): Java(TM) 2 Platform Standard Edition 5.0 API Specification. In: http://docs.oracle.com/javase/1.5.0/docs/api/overview-summary.html, zugegriffen am 12.10.2011.2011.

Tertilt, D.; Krcmar, H. (2011): Generic Performance Prediction for ERP and SOA Applications. In: 19th European Conference on Information Systems Hrsg., Helsinki, Finnland 2011.

Tertilt, D.; Leimeister, S.; Gradl, S.; Mayer, M.; Krcmar, H. (2010): Towards an evolutionary model generation for ERP performance simulation. *International Conference on Intelligent Systems and Agents*. Freiburg.

Ufimtsev, A. (2006): Vertical Performance Modelling and Evaluation of Component-Based Software Systems, National University of Ireland 2006.

Versick, D. (2010): Verfahren und Werkzeuge zur Leistungsmessung, -analyse und -bewertung der Ein-/Ausgabeeinheiten von Rechensystemen, Universität Rostock 2010.

W3C (2004): SOAP Specifications. In: http://www.w3.org/TR/soap/, zugegriffen am 12.12.2011.2011.

Weidner, S. (2006): Integrations-Fallstudie PP (SAP ECC 6.0). In: http://portal.ucc.uni-magdeburg.de/irj/go/km/docs//documents/SAP%20UA%20Portal%20Documents/Members%20Area/Trainings_Courses/2009-07-13_60de_pp_fallstudie.pdf, zugegriffen am 14.06.2011.2011.

Woodside, C.M.; Neilson, J.E.; Petriu, D.C.; Majumdar, S. (1995): The Stochastic Rendezvous Network Model for Performance of Synchronous Client-Server-like Distributed Software. In: IEEE Transactions on Computers, Vol. 44 (1995) Nr. 1, S. 20-34.

Woodside, M. (2002): Tutorial Introduction to Layered Modeling of Software Performance. Carleton University, Ottawa, Canada 2002.

Xu, J.; Oufimtsev, A.; Woodside, M.; Murphy, L. (2005): Performance modeling and prediction of enterprise JavaBeans with layered queuing network templates. In: SIGSOFT Softw. Eng. Notes, Vol. 31 (2005) Nr. 2, S. 5.