Performance Optimization of a Multiresolution Compressible Flow Solver

Currently, biotechnological and biomedical procedures such as lithotripsy or histotripsy are used successfully in therapy. In these methods, compressible multiphase flow mechanisms, such as shock-bubble interactions are utilized. However, the underlying physics of the processes involved are not fully understood. To get deeper insights into these processes, numerical simulations are a favorable tool. In recent years, powerful numerical methods which allow for accurately simulating discontinuous, compressible multiphase flows have been developed. The immense numerical cost of these methods, however, limits the range of applications. To simulate three-dimensional problems, modern high-performance computing (HPC) systems are required and need to be utilized efficiently in order to obtain results within reasonable times. The sophisticated simulation environment "ALIYAH," developed at the Chair of Aerodynamics and Fluid Mechanics, combines advanced numerical methods—including Weighted Essentially Non-Oscillatory (WENO) stencils and sharp-interface treatment (Level-Set) in a Multiresolution Finite-Volume Framework with Total-Variation-Diminishing (TVD) Runge-Kutte (RK) time integration—to solve the Euler equations for compressible multiphase problems.

Exemplarily, the simulation result of a collapsing gas bubble near a deformable gelatin interface is shown in Figure 1. This configuration mimics the dynamics of an ultrasound-induced gas bubble near soft tissue as model for *in vivo* cavitation effects. The bubble collapse is asymmetrical and induces a liquid jet towards the gelatin that eventually ruptures this material. The detailed understanding of such phenomena is the overall scope of our research.

The baseline version of ALIYAH runs a block-based MR algorithm as described in [5]. The code is shared-memory parallelized using Intel Threading Building Blocks (TBB). The performance crucial (parallelizable) loops are distributed among the threads using the TBB affinity partitioner. Thus, the load is dynamically

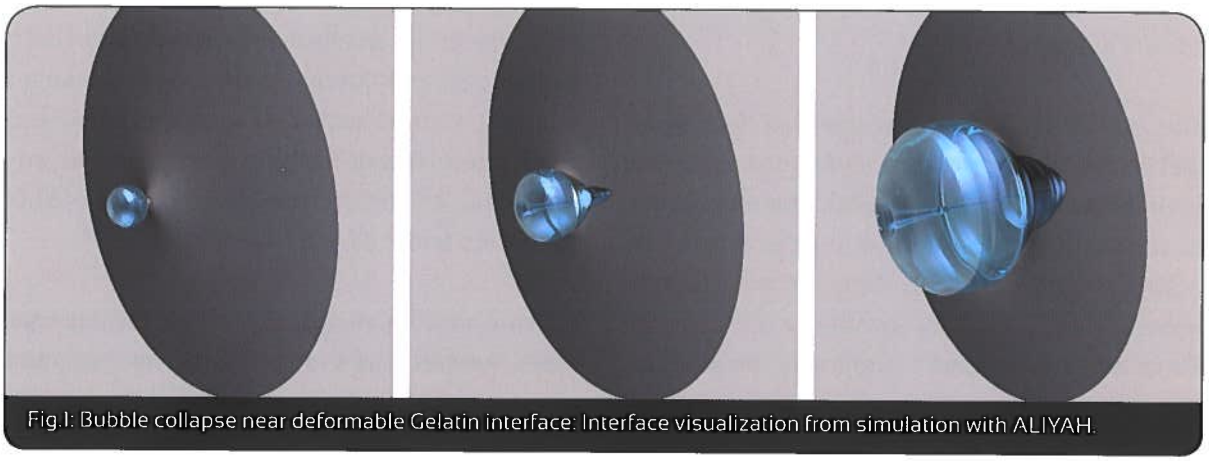

Fig.1: Bubble collapse near deformable Gelatin interface: Interface visualization from simulation with ALIYAH.

re-evaluated every time the algorithm reaches a certain function.

Much of the computational cost in the considered simulation comes from the modeling of the interface between fluids. In our approach the interface is modeled by a conservation ensuring scalar level set function [1], and the interactions across the interfaces need to be considered; this is done with an acoustic Riemann solver which includes a model for surface tension [3]. For the non-resolvable structures—i.e., droplets, bubbles, or filaments with diameters close to the cell size of the finite volume mesh—scale separation of [4] is used.

Performance and scalability test cases

The simulation tests were performed for two cases: A small generic case ("synthetic case"), which executes all methods described in the previous section but with a coarse resolution of only 4096 cells, and the second case ("restart case"), which is a real-application case with a high resolution in all three spatial dimensions. Due to its long run time, only one timestep of this case is analyzed.

The restart case scenario uses an axis-symmetric model, to simulate cylindrical channel geometries in a Cartesian grid. The simulation is conducted with a quarter-model of the full problem; i.e., the Y- and Z-planes are cut into halves with imposed symmetry conditions. Since a full simulation's runtime is too large to be profiled, the measurements are obtained for just one timestep on the coarsest level. To still capture a relevant and representative timestep, the simulation is advanced until time ts = 3.16μs without profiling the code. The corresponding physical state of the bubble break-up is shown in Figure 1.

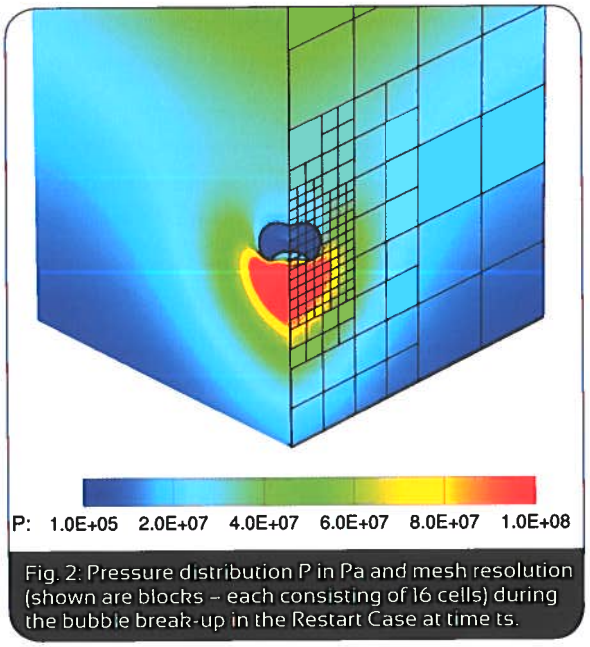

Fig. 2: Pressure distribution P in Pa and mesh resolution (shown are blocks – each consisting of 16 cells) during the bubble break-up in the Restart Case at time ts.

Code analysis

We conduct our analysis and optimization on a dual-socket Intel Xeon E5-2697 v3 (codenamed Haswell). Computational results are presented for an Intel Haswell system at 28 cores. The processor has 2.6 GHz frequency, 32 KB/256 KB L1/L2 caches and 2.3 GB RAM per core.

With the baseline version of the code the two test-cases—restart case and synthetic case, described above—were simulated in a wall clock time of 589 seconds and 666 seconds, respectively.

To find promising starting points for code optimization, a node-level analysis is performed using the Intel VTune Amplifier. To reduce the amount of collected information the Amplifier analysis as well as all subsequent optimization runs are performed using eight threads. The hotspot analysis for the restart case is presented in Figure 2 and for the small synthetic case in Figure 3.

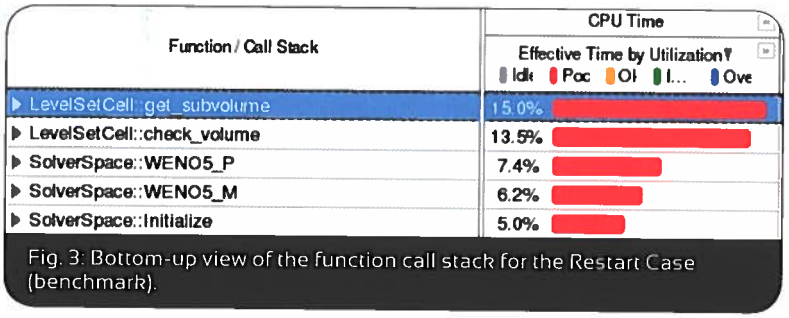

Fig. 3: Bottom-up view of the function call stack for the Restart Case (benchmark).

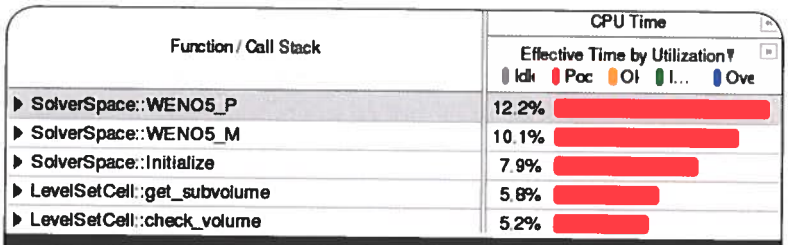

Fig. 4: Bottom-up view of the function call stack the Synthetic Case.

One can clearly identify the functions get_subvolume, check_volume, and WENO5_* as the hotspots. The optimization of WENO5_* requires only small reorganization of the corresponding source code. In contrast to the WENO methods, the time spent in the get_subvolume function does not increase linearly with the problem size (c.f., relative time spent for the small synthetic case and the larger restart case). Hence, a focus is laid on the non-straight-forward optimization of the get_subvolume and check_volume functions.

An essential ingredient to utilize HPC architectures efficiently is the usage of single instruction multiple data (SIMD) instructions in the computationally intensive parts of the code. SIMD instructions allow processing of multiple pieces of data in a single step, speeding up throughput for many tasks. Compilers can auto-vectorize loops that are considered safe for vectorization. In the case of the here-used Intel compiler version 16.0, this happens at default for optimization levels -O2 or higher.

To analyze the auto-vectorized code the Intel Advisor XE tool is used. The analysis revealed the functions listed in Figure 4 to be the most time consuming non-vectorized ones. In the figure, "self time" represents the time spent in a particular program unit and "total time" includes "self time" of the function itself and "self time" of all functions that were called from within this function. As seen, the function get_subvolume, which is called recursively from the function get_volume, is the most time-consuming non-vectorized function. In contrast to compilers assumption, the examination of get_subvolume's source code reveals no crucial dependency problems.

Function Call Sites and Loops		Vector Issues	Self Time	Total Time
[loop in LevelSetCell::get_subvolume at level_set_cell.cpp:602]	☐	1 Assumed dependency prese...	231.177s	3810.722s
[loop in [TBB Dispatch Loop at custom_scheduler.h:413]	☐		228.349s	231.398s
[loop in Meshlevel::Update_tags(Meshlevel&)::{lambda(tbb::bl...	☐	1 Assumed dependency prese...	126.556s	126.556s
[loop in SolverSpace::Initialize at solver_space.cpp:182]	☐	1 Opportunity for outer loop ve...	125.308s	125.308s
[loop in BlockAndPackageTags::UpdatePackageInterfacePro ...	☐	1 Assumed dependency prese...	68.673s	68.673s
[loop in BlockAndPackageTags::UpdatePackageInterfacePro ...	☐	1 Assumed dependency prese...	60.704s	60.704s
[loop in LevelSetCell::get_subvolume at level_set_cell.cpp:601]	☐	1 Assumed dependency prese...	45.179s	3855.900s
[loop in LevelSetCell::get_subvolume at level_set_cell.cpp:600]	☐	1 Assumed dependency prese...	39.763s	3895.664s
[loop in LevelSetNode::GetIncrementForReinitialization at lev...	☐	3 Assumed dependency prese...	38.644s	38.644s

Fig. 5: Survey analysis of the vectorization in the baseline version of ALIYAH.

Results

Since it is a recursive call automatic vectorization or OpenMP-SIMD, annotations cannot be applied directly to the body of the function get_subvolume. Moreover, due to the presence of the relatively large amount of nested loops with small trip counters the declaration of get_subvolume as "vectorizable" is not an optimal strategy in this case. On Haswell, SIMD instructions process four elements (double precision) at once. This means loops with a trip counter of two underutilize the vector registers by a factor of two. It appears OpenMP-SIMD is not able to collapse the two nested loops and apply vectorization automatically. As auto-vectorization fails even with the usage of OpenMP paragmas we follow the more aggressive approach, described below.

The function get_subvolume performs temporary subdivisions of the cubic grid cells based on linear interpolation to approximate the volume one phase occupies. Due to the recursive call with a local stopping criterion the data flow in each local volume evaluation is complex. To apply SIMD vectorization, we combine linear interpolation on several elements into one call. This is profitable since the operation on two neighbor grid points is the same, albeit with different data from the vector. We program vectorized loops directly using Intel AVX instructions.

The explicit SIMD vectorization with intrinsics allows us to reduce the number of micro-operations from 185 for the baseline version down to 88. The block throughput is also reduced from 48 cycles to 24 cycles. The total time spent in the get_subvolume function is reduced by a factor of 0.7, which means a gain in performance of 40%. CPU time of the two functions get_subvolume and check_volume after optimization is reduced by a factor of 0.5 compared to the baseline version. Moreover, the wallclock time of the AVX version is reduced to 531 sec and 558 sec for the restart case and the synthetic case, respectively. For the whole simulation this corresponds to a speedup of 11% for the restart case and 19% for the synthetic cases, correspondingly.

Acknowledgment

The authors gratefully acknowledge the Kompetenznetzwerk für wissenschaftliches Höchstleistungsrechnen in Bayern for the KONWIHR-III funding. S. Adami and N.A. Adams gratefully acknowledge the funding from the European Research Council (ERC) under the European Union's Horizon 2020 research and innovation program (grant agreement No 667483).

References

[1] X. Y. Hu, B. C. Khoo, N. A. Adams, and F. L. Huang: "A conservative interface method for compressible flows," J. Comput. Phys., vol. 219, no. 2, pp. 553–578, Dec. 2006.

[2] R. P. Fedkiw, T. D. Aslam, B. Merriman, and S. Osher, "A Non-oscillatory Eulerian Approach to Interfaces in Multimaterial Flows (The Ghost Fluid Method)," J. Comput. Phys., vol. 152, pp. 457–492, 1999.

[3] R. Saurel, S. Gavrilyuk, and F. Renaud: "A multiphase model with internal degrees of freedom: application to shock–bubble interaction," J. Fluid Mech., vol. 495, pp. 283–321, 2003.

[4] J. Luo, X. Y. Hu, and N. A. Adams: "Efficient formulation of scale separation for multi-scale modeling of interfacial flows," J. Comput. Phys., vol. 308, pp. 411–420, Mar. 2016.

[5] L. H. Han, X. Y. Hu, and N. A. Adams: "Adaptive multi-resolution method for compressible multi-phase flows with sharp interface model and pyramid data structure," J. Comput. Phys., vol. 262, pp. 131–152, Apr. 2014.

Written by Nils Hoppe[1], Igor Pasichnyk[2], Stefan Adami[1], Momme Allalen[3], and Nikolaus A. Adams[1]

[1] Lehrstuhl für Aerodynamik und Strömungsmechanik, Technische Universität München, Boltzmannstraße 15, 85748 Garching

[2] IBM Deutschland GmbH, Boltzmannstraße 1, 85748 Garching

[3] Leibniz-Rechenzentrum der Bayerischen Akademie der Wissenschaften, Boltzmannstraße 1, 85748 Garching

Contact: momme.allalen@lrz.de

Performance Evaluation of a Parallel HDF5 Implementation to Improve the Scalability of the CFD Software Package MGLET

This paper presents a performance evaluation for an implementation in parallel HDF5 inside the MGLET code.

The computational fluid dynamics (CFD) code "MGLET" is designed to precisely and efficiently simulate complex flow phenomena within an arbitrarily shaped flow domain. MGLET is capable of performing direct numerical simulation (DNS) as well as large eddy simulation (LES) of complex turbulent flows. It employs a finite-volume method to solve the incompressible Navier–Stokes equations for the primitive variables (i.e. three velocity components and pressure), adopting a Cartesian grid with staggered arrangement of the variables. The time integration is realised by an explicit third-order low-storage Runge–Kutta scheme. The pressure computation is decoupled from the velocity computation by the fractional time-stepping, or Chorin's projection method. Consequently, an elliptic Poisson equation has to be solved for each Runge–Kutta sub-step.

The current version of MGLET utilises a parallel adaptation of Gauss-Seidel solver as well as Stone's Implicit Procedure (SIP) within the multigrid framework, both as the smoother during the intermediate steps and the solver at the coarsest level. Such separate usage is justified by the fact that the former is very effective in eliminating low-frequency error predominant over the successive coarsening stage of multigrid algorithms, whereas the latter can be used to solve the Poisson problem at the coarsest level with a broad spectrum of residual error. Geometrically complex surfaces, or arbitrarily curved, can be represented by an immersed boundary method (IBM). MGLET offers several sub-grid scale models for LES simulation, such as Smagorinsky's model, two versions of the dynamic formulations and the WALE model. MGLET is written in FORTRAN and the parallelisation strategy is based on Message Passing Interface (MPI).

The code is currently being used by several research groups: At the Chair of Hydromechanics of the Technical University of Munich, for instance, turbulent flow through complex geometries, flow in porous media, and fibre suspensions in fluid media have been investigated using MGLET. The groups of Prof. Helge Andersson and Prof. Bjørnar Pettersen (both NTNU Trondheim) use the code to predict and analyse bluff-body flows primarily using DNS and IBM. At the Institute for Atmospheric Physics (DLR Oberpfaffenhofen), aircraft wake vortices are investigated, including their interaction with atmospheric boundary layers and ground effects. These applications demonstrate MGLET's excellent numerical efficiency and adaptability to the diverse hydrodynamic problems.

Continuous improvement of its parallel scalability has been, and will remain, critically important for the MGLET development programme, as it allows us to simulate ever-more realistic and engineering-relevant turbulent flows at an adequate resolution of motion. For example, there is a trend towards higher Reynolds numbers, more