SWE-X10: An Actor-Based and Locally Coordinated Solver for the Shallow Water Equations

Alexander Pöppl   Michael Bader
Technical University of Munich. Germany
{poeppl, bader}@in.tum.de

Abstract

We present an X10 software package for the solution of the shallow water equations, a set of equations commonly used to simulate tsunami and flooding events. The software uses an actor-oriented approach to obtain a communication scheme that does not rely on central coordination. Instead, each actor only communicates with its neighbors. We evaluated the package via scaling tests on single-place shared memory as well as multi-place distributed memory system configurations, and found it to perform comparably to prior implementations based on C++, OpenMP and MPI.

Categories and Subject Descriptors G.1.0 [*Numerical Analysis*]: Parallel Algorithms; G.1.10 [*Numerical Analysis*]: Applications

Keywords Actor Model, Parallel Algorithms, Shallow Water Equations, APGAS

1. Introduction

The shallow water equations are commonly used in the simulation of tsunami events, e.g. the *GeoClaw* package by LeVeque et al. (2011). They belong to the class of two-dimensional simulation problems. Hence, it is possible to solve realistic scenarios with a moderate amount of resources. However, the parallelization of the solution process is non-trivial as communication between different patches with neighboring cells is required between each time step. This makes the shallow water equations an interesting subject for research in the field of parallel algorithms. The Asynchronous Partitioned Global Address Space (APGAS) paradigm allows us to view the whole computation as one program instead of a set of processes running in parallel. This paradigm is also used in conjunction with resource-aware computing (Bungartz et al. 2013). In contrast to a prior approach based on work by Breuer and Bader (2012), our code features an actor-based, local coordination scheme without centralized control of the simulation. The current state may be viewed as a step towards a fully asynchronous local time stepping scheme. Compared to traditional approaches, we may view the parts running in parallel on a higher level, and may hence quickly explore different software configurations such as variations in the granularity of places or the distribution of simulation data, both of which are not as easily achievable with applications based on OpenMP and MPI.

X10'16, June 14, 2016, Santa Barbara, CA, USA
ACM. 978-1-4503-4386-2/16/06...$15.00
http://dx.doi.org/10.1145/2931028.2931034

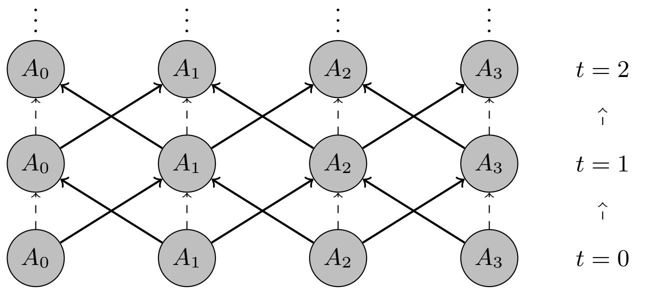

Figure 1. Actors advancing each other. Each actor has a left and a right neighbor, and in order to compute the next time step, they need to receive data from these first.

2. Numerical Model and System Design

The shallow water equations are a set of hyperbolic partial differential equations. They are derived from the Navier-Stokes equations by averaging over the unknowns in the third dimension, i.e. the depth of the water. Let h denote the water height, u and v the velocities in the remaining two spacial dimensions. $S(t, x, y)$ stands for an optional source term that enables the modelling of additional effects such as Coriolis forces and friction or bathymetry of the ocean floor. The notation $[f]_a$ is short for partial derivative of f in regards to a.

$$\begin{bmatrix} h \\ hu \\ hv \end{bmatrix}_t + \begin{bmatrix} hu \\ hu^2 + \frac{1}{2}gh^2 \\ huv \end{bmatrix}_x + \begin{bmatrix} hv \\ huv \\ hv^2 + \frac{1}{2}gh^2 \end{bmatrix}_y = S(t, x, y)$$

The numerical approach implemented by the SWE-X10 package is based on the original SWE package presented in (Breuer and Bader 2012). To progress the simulation, a Riemann problem has to be solved for each of the cell borders. This complicates parallelization, as data at the boundaries of each patch needs to be exchanged before the computation of each time step.

We use the actor model and the *ActorX10* library described in Roloff et al. (2016). We subdivide the simulation domain, represented by a two-dimensional array of unknowns for h, hu, hv and b, into equally sized, rectangular patches and assign one actor to every patch. Each of these has an incoming and one outgoing channel for each of its direct top, bottom, left and right neighbor. The channels are used to exchange cell information at the boundary of each patch between each step. Before an actor can compute a new time step, the results from the neighbors' previous step need to be received. Thereafter, new fluxes are computed, the unknowns are updated according to a precomputed time delta and the new values at the boundary are sent to the actor's neighbors. This behavior is illustrated in Figure 1 for four actors arranged in a linear fashion.

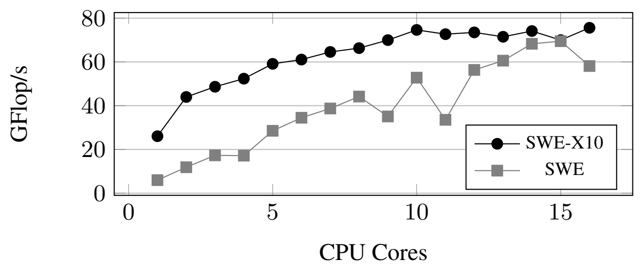

Figure 2. Single node performance test with 1024^2 simulated cells per CPU core. Our code is able to reach about 75 GFlop/s, which translates to about 23% of the peak performance of the node. The gray line depicts the single-node performance of SWE.

3. Results

We performed two benchmark tests to demonstrate the performance characteristics of the SWE-X10 application. Initially, we evaluated the performance on a single place by varying the number of CPU cores contributing to the execution of the simulation. Furthermore, we performed a weak-scaling test where we executed the simulation on configurations spanning multiple places on several nodes in a cluster. The target system for the performance evaluation is a cluster of 28 dual-socket Xeon E5-2670 (Sandy Bridge EP) nodes. Each node has a peak single precision floating point performance of 332.8 GFlop/s and a STREAM Triad performance of 60.8 GB/s. Both tests utilize the HLLE solver (Harten et al. 1997; Einfeldt 1988), an autovectorizing C++ Riemann solver. The scenario for the tests is a Radial Dam Break Scenario. As a comparison, we provide alternate performance results from SWE (Breuer and Bader 2012), a package that utilizes OpenMP and MPI for parallelization.

In the single-place test, we ran a simulation on n physical processor cores, with 1024^2n cells in the simulation domain, distributed onto $4n$ actors, each with 512^2 cells. We observed the best performance with four actors per core, presumably due to the effects of SMT. The results of the test are given in Figure 2. They show the floating point performance to be at 25 GFlop/s for a single core simulation run and indicate a saturation at 10 cores with a performance of 75 GFlop/s. This corresponds to 23% of the theoretical peak performance of a single Xeon E5-2670 node. A single execution of the HLLE-Solver entails 135 floating point operations, and a memory transfer volume of 44 Bytes. From these numbers, and the performance characteristics of th CPU, we deduce that execution of the code is memory-bound.

In the multi-place performance evaluation, we assigned a single place with 32 actors, each with a patch of 512×512 cells, to one socket. The test was executed in configurations ranging from one socket with 8 cores up to 16 nodes with a total of 256 cores. Its results are illustrated in Figure 3. For simulation runs up to 8 nodes, we observe ideal scaling behavior, afterwards the performance gains are slightly sub-linear for both implementations. This might be due to performance fluctuations inherent to the cluster we performed the measurements on.

4. Conclusion and Future Work

We presented a software package for the simulation of the shallow water equations based on an actor-based and locally coordinated control scheme. We showed that the code scales to at least 256 cores on 16 nodes of a cluster based on dual socket 8-core Sandy Bridge nodes. Using native SIMD-based Riemann solvers, we managed to reach a peak performance of 1.2 TFlop/s, which indicates a good scaling behavior relative to a single-node configuration with 16 physical CPU cores. Our implementation compares favorably to

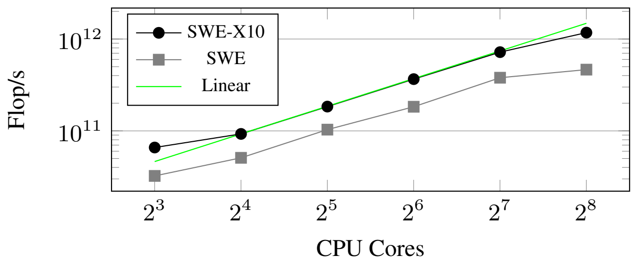

Figure 3. Weak scaling test encompassing a work load of four patches with 512^2 cells per physical CPU core. Scaling was performed from 8 cores on a single socket up to 256 cores on 16 nodes. The green line denotes linear scaling behavior relative to a single node, while the black line displays our actual measurements. The gray line denotes the performance of SWE.

SWE, a prior implementation based on MPI and OpenMP, which reached a peak performance of 464 GFlop/s under equal conditions.

In future work, we aim to extend the code to run in heterogeneous environments containing accelerators such as CUDA-capable GPUs or Xeon Phi coprocessors. This that will necessitate actors with non-uniform patch sizes in order to accommodate the different performance characteristics. Furthermore, we plan to allow actors to dynamically choose from different multiples of the base time step. This results in an asynchronous, local time stepping scheme without the need for central communication. Finally, we are exploring delayed activation. There, the actors gradually start computing updates as the wave propagates across the simulation domain.

Acknowledgments

This work was supported by the German Research Foundation (DFG) as part of the Transregional Collaborative Research Centre "Invasive Computing" (SFB/TR 89).

References

A. Breuer and M. Bader. Teaching parallel programming models on a shallow-water code. In *Proceedings of the 2012 11th International Symposium on Parallel and Distributed Computing*, ISPDC '12, pages 301–308, Washington, DC, USA, 2012. IEEE Computer Society. ISBN 978-0-7695-4805-0. doi: 10.1109/ISPDC.2012.48. URL `http://dx.doi.org/10.1109/ISPDC.2012.48`.

H.-J. Bungartz, C. Riesinger, M. Schreiber, G. Snelting, and A. Zwinkau. Invasive computing in hpc with x10. In *Proceedings of the Third ACM SIGPLAN X10 Workshop*, X10 '13, pages 12–19, New York, NY, USA, 2013. ACM. ISBN 978-1-4503-2157-0. doi: 10.1145/2481268.2481274. URL `http://doi.acm.org/10.1145/2481268.2481274`.

B. Einfeldt. On godunov-type methods for gas dynamics. *SIAM Journal on Numerical Analysis*, 25(2):294–318, 1988. doi: 10.1137/0725021. URL `http://dx.doi.org/10.1137/0725021`.

A. Harten, P. D. Lax, and B. Leer. *Upwind and High-Resolution Schemes*, chapter On Upstream Differencing and Godunov-Type Schemes for Hyperbolic Conservation Laws, pages 53–79. Springer Berlin Heidelberg, Berlin, Heidelberg, 1997. ISBN 978-3-642-60543-7. doi: 10.1007/978-3-642-60543-7_4. URL `http://dx.doi.org/10.1007/978-3-642-60543-7_4`.

R. J. LeVeque, D. L. George, and M. J. Berger. Tsunami modelling with adaptively refined finite volume methods. *Acta Numerica*, 20:211–289, 5 2011. ISSN 1474-0508. doi: 10.1017/S0962492911000043. URL `http://journals.cambridge.org/article_S0962492911000043`.

S. Roloff, A. Pöppl, T. Schwarzer, S. Wildermann, M. Bader, M. Glaß, F. Hannig, and J. Teich. ActorX10: An actor library for X10. In *Proceedings of the Sixth ACM SIGPLAN X10 Workshop (X10)*. ACM, June 2016.