Cache Aware Instruction Accurate Simulation of a 3-D Coastal Ocean Model on Low Power Hardware

Dominik Schoenwetter[1], Alexander Ditter[1], Vadym Aizinger[2], Balthasar Reuter[2] and Dietmar Fey[1]
[1]*Chair of Computer Science 3 (Computer Architecture), Friedrich-Alexander University Erlangen-Nürnberg (FAU), Martensstr. 3, 91058, Erlangen, Germany*
[2]*Chair of Applied Mathematics (AM1), Friedrich-Alexander University Erlangen-Nürnberg (FAU), Cauerstr. 11, 91058, Erlangen, Germany*

Keywords: Environmental Modeling, Coastal Ocean Modeling and Simulation, Hardware Simulation, Hardware Virtualization, Low Power Architectures.

Abstract: High level hardware simulation and modeling techniques matured significantly over the last years and have become more and more important in practice, e.g., in the industrial hardware development and automotive domain. Yet, there are many other challenging application areas such as numerical solvers for environmental or disaster prediction problems, e.g., tsunami and storm surge simulations, that could greatly profit from accurate and efficient hardware simulation. Such applications rely on complex mathematical models that are discretized using suitable numerical methods, and require a close collaboration between mathematicians and computer scientists to attain desired computational performance on current micro architectures and code parallelization techniques to produce accurate simulation results as fast as possible. This complex and detailed simulation requires a lot of time during preparation and execution. Especially the execution on non-standard or new hardware may be challenging and potentially error prone. In this paper, we focus on a high level simulation approach for determining accurate runtimes of applications using instruction accurate modeling and simulation. We extend the basic instruction accurate simulation technology from OVP using cache models in conjunction with a statistical cost function, which enables high precision and significantly better runtime predictions compared to the pure instruction accurate approach.

1 INTRODUCTION

Nowadays, unfortunately, the number of catastrophic geophysical events like hurricanes or tsunamis is steadily increasing. An important tool to predict the implications of such natural disasters are accurate simulations with regional or global ocean models (two- or three-dimensional). To ensure in-time notification of to be affected regions, these simulations have to be carried out in real time and the simulation model has to offer sufficient spatial resolution to guarantee no misestimation in the affected regions. Moreover, endangered regions often do not have reliable communication and power infrastructure, which renders running such simulations and prediction on-site difficult or even impossible.

There is a number of state-of-the-art approaches for flood warning systems. Running simulations of such numerical solvers at a coarser grid resolution or relying on less accurate numerical methods are two of that. Two other approaches are to scan a database of precomputed scenarios, hoping to find a similar setting, or running the simulation not on-site but, e.g. in a cloud environment. However, each of these approaches increases the risk of predicting either too late or being too inaccurate, which can result in property damage or even in loosing lives. As a consequence, a minimum set of two requirements has to be fulfilled, namely:

- The hardware must be able to complete the simulation battery powered if the power infrastructure collapses.
- The computation must be performed on-site to guarantee in-time warning of inhabitants without the need of a reliable communication network to the rest of the world.

In 2015, we proposed a concept that enables the determination of suitable low power multi- and many-core architectures for tsunami and storm surge simulations fulfilling both of these requirements (Schoenwetter et al., 2015). The concept relies on the virtual envi-

Schoenwetter, D., Ditter, A., Aizinger, V., Reuter, B. and Fey, D.
Cache Aware Instruction Accurate Simulation of a 3-D Coastal Ocean Model on Low Power Hardware.
DOI: 10.5220/0006006501290137
In *Proceedings of the 6th International Conference on Simulation and Modeling Methodologies, Technologies and Applications (SIMULTECH 2016)*, pages 129-137
ISBN: 978-989-758-199-1

ronment Open Virtual Platforms (OVP) to investigate different hardware configurations, which enables emulation and simulation of different low power multi- and many-core hardware architectures on the instruction accurate level.

In this paper we investigate the accuracy of results from the state-of-the-art instruction accurate simulation environment OVP by means of comparing runtime predictions for the NAS Parallel Benchmarks with execution times on real hardware. We verify the applicability of our findings to real world applications on ARM[1] hardware by performing the same comparison using the 3-D shallow-water solver UTBEST3D (cf. Sec. 4.2).

Furthermore, we show how the accuracy with respect to non-functional metrics such as runtime can be improved by the addition of complex memory models. We present our developed memory and cache models that can be used in conjunction with OVP as well as a corresponding instrumentation mechanism to track, record, and trace accesses to these models within the simulation. Using this technique and models, we can offer a better understanding of memory access patterns, which allows to analyze and compare different algorithms and hardware systems. From this, software and hardware developers can derive potential improvements in their respective designs, allowing for an overall better hardware-software co-design. Furthermore, this approach permits a more accurate performance modeling than conventional instruction accurate simulation techniques. For our analysis and evaluation we compare the results from our reference hardware, the Altera Cyclone V *system on chip* (SoC) (cf. Sec. 3.2), to those obtained on the emulated ARM part of the Altera SoC using OVP.

The rest of the paper is organized as follows: Section 2 provides an overview of the state-of-the-art in the use of low power architectures for high performance computing (HPC) as well as approaches for fast simulation and modeling. A description of the simulation environment and used hardware is given in Section 3, followed by details on the used benchmarks and application in Section 4. Sections 5 and 6 present the instrumentation technique and obtained results. The paper concludes with a summary and outlook on future work.

2 RELATED WORK

In the last years, the research in the field of of low power architectures for high performance computing (HPC) constantly increased.

Rajovic et al. highlighted that low power ARM architectures have well suited characteristics for HPC (Rajovic et al., 2013). Their investigations focused on reducing power consumption.

A study that had a detailed look on energy-to-solution comparisons for different classes of numerical methods for partial different equations by using various architectures was published by Goeddeke et al. (Göddeke et al., 2013). The results showed that energy-to-solution and energy-per-time-step improvements up to a factor of three are possible when using the ARM-based Tibidabo cluster (Rajovic et al., 2014). This factor was determined by comparing the Tibidabo cluster to a Nehalem-based x86 subcluster of the LiDOng machine provided by TU Dortmund (ITMC TU Dortmund, 2015).

In 2013, Castro et al. (Castro et al., 2013) compared the energy consumption and the performance of different general-purpose as well as low power architectures. They investigated energy- and time-to-solution for the Traveling-Salesman problem on three architectures (Applegate et al., 2011), namely an Intel Xeon E5-4640 Sandy Bridge-EP, the low power Kalray MPPA-256 many-core processor (KALRAY Corp., 2015) and the low power CARMA board (NVIDIA Corp., 2015). In their study both, the CARMA board as well as the MPPA processor, achieved better energy-to-solution results than the Intel architecture.

Because of multi- and many-core hardware architectures, detailed levels of modeling and simulation are not an option, due to the fact that simulation of such large systems is very time-consuming. As a consequence, simulation approaches on higher levels of abstraction are more promising. One such approach is the statistical simulation, which measures and detects specific characteristics (branches, load/store, etc.) during the execution of a program and then generates a synthetic trace that guarantees syntactical correctness. Afterwards, the trace is simulated (Eeckhout et al., 2004). The synthetic trace is orders of magnitude smaller than the whole program. As a consequence, the simulation is much more faster.

A concept of extending statistical simulation by adding statistical memory modeling was put forward by Genbrugge and Eeckhout in 2009 (Genbrugge and Eeckhout, 2009). They model shared resources in the memory subsystem of multi-processors as shared caches, off-chip bandwidth and main memory.

An open-source hardware simulator for the x86 architecture that uses various abstraction techniques to provide accurate performance results is Graphite (Miller et al., 2010), in which all hardware

[1] http://www.arm.com/

models use further analytical timing models to guarantee accurate results. Using Graphite as the base, Carlson et al. (Carlson et al., 2011) developed Sniper that enhances the original simulator with an interval simulation approach, a more detailed timing model, and improvements concerning to operating system modeling. Thus, exploring homogeneous and heterogeneous multi- and many-core architectures is faster and more precise as in Graphite.

In the area of flood prediction and ocean modeling, a broad number of numerical models is available and actively used. Due to the large domain sizes and long simulation times, most applications make use of parallelization techniques to reduce the computation time, and for many of the established frameworks exist performance studies or performance models, e.g., POP (Worley and Levesque, 2004; Kerbyson and Jones, 2005), HYCOM (Wallcraft et al., 2005; Barker and Kerbyson, 2005), FVCOM (Cowles, 2008), HOMME (Nair et al., 2009), ADCIRC (Tanaka et al., 2011; Dietrich et al., 2012), MPAS (Ringler et al., 2013), or UTBEST3D (Reuter et al., 2015).

Most of these codes employ numerical discretizations of lower order, based on Finite Difference, Finite Element, or Finite Volume schemes. However, higher order numerical methods can be beneficial in the representation of convection dominated physical processes, as outlined in a current paper by Shu (Shu, 2016). One such numerical scheme is the local discontinuous Galerkin method, introduced by Cockburn and Shu (Cockburn and Shu, 1998) and applied to geophysical flows by Aizinger and Dawson (Aizinger and Dawson, 2002; Dawson and Aizinger, 2005).

3 ENVIRONMENT

3.1 Simulation Environment

The simulation technology from Open Virtual Platforms (OVP) runs unchanged binaries in an emulated environment described by virtual hardware models, which can contain multiple processors and peripheral models. Its instruction accurate simulator was developed with high simulation speeds in mind and allows to execute or debug applications (using the integrated GDB interface) in the virtual environment, or evaluate the virtual platform itself. OVP provides the ability to create new processor models and other platform components by writing C/C++ code using the *application programming interface* (API) and libraries supplied as part of OVP (Imperas Software Ltd., 2015).

The API defines a virtual hardware platform called ICM (Innovative CPUManager Interface), that includes functions for setting up, running and terminating a simulation (*icmInitPlatform*, *icmSimulatePlatform*, *icmTerminate*), defining components for the simulation (e.g., *icmNewProcessor*), and loading the application's executable (*icmLoadProcessorMemory*).

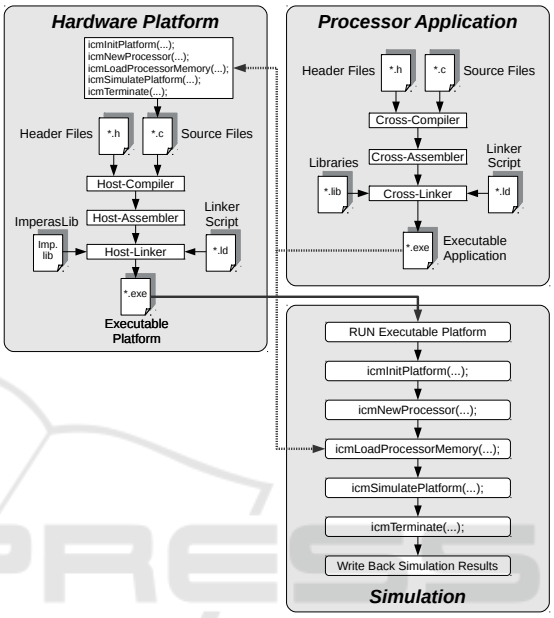


Figure 1: Operating Principle of Open Virtual Platforms Simulations.

A minimal setup for an OVP simulation requires the definition of one processor and an application that is to be run on the virtual platform. Figure 1 gives an example for such a setup using a processor model and application both provided in the C programming language.

OVP's instruction accurate simulator represents the functionality of a processor's instruction execution without accounting for such artifacts as pipelines. This is due to the fact that the provided instruction accurate simulation cannot make clear statements about the time spent during pipeline stalls since cache misses and other things are not modeled. Thus, conversions to runtimes will have limited accuracy when compared to actual hardware.

The simulation environment can only provide the total amount of instructions executed. Assuming a perfect pipeline, where one instruction is executed per cycle, the instruction count divided by the processor's instruction rate in *million instructions per second* (MIPS) yields the runtime of the program. To measure the instruction counts within specific code snippets of a larger application, the OVP simula-

Table 1: The specifications of the *hard processor system* on the Altera Cyclone V SoC.

Processor:	2x ARM Cortex-A9 @ 925 MHz
Co-processor:	2x NEON SIMD double-precision FPU
Caches per proc.:	32 KiB instruction, 32 KiB L1
Shared caches:	512 KiB L2 cache
Main memory:	1 GiB DDR3 SDRAM
Mem. interface:	40-bit bus @ 400 MHz (25.6 Gbps)

tor provides the possibility for measuring instruction counts in parts of a program.

3.2 Reference Hardware

We use Altera's development kit board (Altera Corp., 2016) with a Cyclone V SX SoC-FPGA as our reference hardware platform. This SoC-FPGA includes a *hard processor system* (HPS) consisting of *multiprocessor subsystem* (MPU), multi-port SDRAM controller, a set of peripheral controllers, and a high-performance interconnect. The memory controller supports command and data reordering, *error correction code* (ECC), and power management. Some relevant specifications of the HPS are listed in Tab. 1. The cache controller has a dedicated 64-bit master port connected directly to the SDRAM controller and a separate 64-bit master port connected to the system level 3 (L3) interconnect. All blocks of the HPS are connected with L3 multilayer AXI interconnect structure, and low-speed peripheral controllers reside on the level 4 (L4) AXI buses that work in separate clock domains for efficient power management.

The programmable logic part of the SoC is a high-performance 28 nm FPGA, which is connected to the HPS part of the board via high-throughput (125 Gbps) on-chip interfaces. All the applications presented in this paper make only use of the HPS part of the SoC-FPGA, the FPGA part is not used. The Cortex-A9 MPCore runs a Linux kernel version 3.16.0, and the user space software is an ARM Arch Linux distribution utilizing a rolling release model.

3.3 Virtual Hardware

The virtual hardware description represents just the relevant parts of the actual Altera Cyclone V SoC (Imperas Software Ltd., 2016) and neglects all units not required for the execution of our benchmarks and application. For example, the FPGA part of the board is neither considered nor implemented in the virtualization environment. Yet, all hardware components that are necessary to run a Linux kernel and provide correct hardware functionality in our test cases are virtualized. Fig. 2 depicts this subset of implemented hardware components.

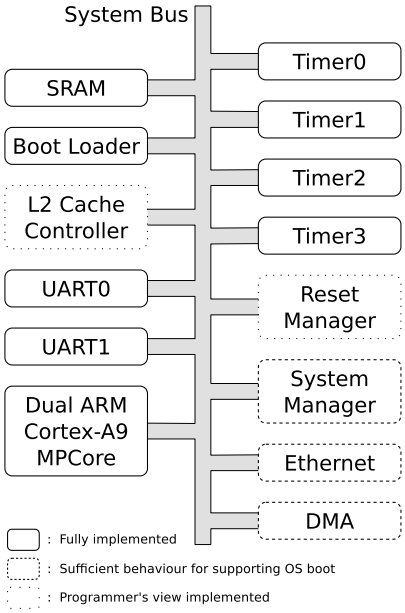


Figure 2: Schematic view of the implementation state of the virtual Cyclone V SoC in OVP. All relevant hardware components are sufficiently abstracted and implemented to boot a 3.16 Linux kernel.

The virtual hardware is capable of booting the same Linux kernel (3.16) as the real hardware. This is very important for our considerations, as it does guarantee binary compatibility, i.e., the identical compiled executable can be run on the real and virtualized hardware.

4 APPLICATION AND BENCHMARKS

Our investigations are based on an extensive artificial benchmark set representing a broad range of typical *computational fluid dynamics* (CFD) and HPC application characteristics, e.g., compute and memory bound kernels. For that we use the NAS Parallel Benchmark (NPB) suite (John Hardman, 2016), i.e., the eight original benchmarks specified in NPB 1 consisting of five kernels and three pseudo applications. The findings from these benchmarks are verified using a real world HPC application, the three-dimensional regional ocean model UTBEST3D.The individual characteristics of UTBEST3D and the NAS benchmarks are described in the following.

We cross-compile the application and benchmarks for both, real hardware as well as the OVP simulation to ensure binary equality and thus the best possible comparability of the obtained results. For this, we use gfortran-arm-linux-gnueabihf for the Fortan

based and gcc-arm-linux-gnueabihf for the C based benchmarks (both in version 4.8.2) and run the same binaries in the real and the virtual environment.

4.1 NAS Parallel Benchmarks (NPB)

Our benchmark suite consists of eight compute kernels, which are listed in Tab. 2 and described in more detail in the following.

Conjugate Gradient Benchmark – CG: This benchmark computes an estimate of the largest eigenvalue of a symmetric positive definite sparse matrix using the conjugate gradient method (Bailey et al., 1991). Its runtime is dominated by the sparse matrix-vector-multiplication in the conjugate gradient subroutine. Due to the random pattern of nonzero entries in the matrix this requires a high number of memory accesses, leading to a low computational intensity of this memory bound benchmark.

Multi Grid Benchmark – MG: The MG benchmark is based on a multigrid kernel, which computes an approximative solution of the three dimensional Poisson problem. In each iteration of the algorithm, the residual is evaluated and used to apply a correction to the current solution. Its most expensive parts are the evaluation of the residual and the application of the smoother, both of which are stencil operations with constant coefficients for the specified problem. The update of a grid point involves the values of neighboring points thus, even with an optimal implementation, this requires between four and eight additional memory access operations per grid point. For constant stencil coefficients, the runtime is dominated by memory access rather than the computational effort meaning that the MG benchmark is memory bound.

Fourier Transform Benchmark – FT: The FT benchmark solves a partial differential equation by applying a Fast Fourier Transform (FFT) to the original state array, multiplying the result by an exponential, and using an inverse FFT to recompute the original solution. Finally, a complex checksum is computed to verify the result (Bailey et al., 1991). The FFT implementation in the benchmark uses a blocked variant of the Stockham FFT and dominates the runtime of this benchmark. This procedure is bound by memory operations however, due to blocking, the limiting factor is not the memory but rather the cache bandwidth.

Embarrassingly Parallel Benchmark – EP: EP is an embarrassingly parallel kernel, which generates pairs of Gaussian random deviates and tabulates the number of pairs in successive square annuli, a problem typical for many Monte Carlo simulations (Bailey et al., 1991). The EP benchmark is computationally expensive, complex operations such as computation of logarithms and roots make up a big portion of the total runtime whereas only very few memory operations are necessary for both random number generation and calculation of the Gaussian pairs. The EP benchmark is compute bound.

Integer Sort Benchmarks – IS: This benchmark sorts N integer keys in parallel, which are generated by a sequential key generation algorithm. IS requires ranking of an unsorted sequence of N keys, for which the initial distribution of keys can have significant impact on the performance of the benchmark. Thus, the initial sequence of keys is generated in a defined sequential manner. The performed sorting operations are important in particle method codes and both, integer computation speed as well as communication performance are relevant (Bailey et al., 1991).

Lower Upper Benchmark – LU: LU uses a Gauss-Seidel solver for lower and upper triangular systems (regular-sparse, block size 5×5) resulting from a discretization of compressible Navier-Stokes equations in a cubic domain and implements several real-case features, e.g., a dissipation scheme. This benchmark represents the computations associated with the implicit operator of an implicit time-stepping algorithm (Bailey et al., 1991).

Diagonal Block Matrix Benchmark – SP and BT: The SP benchmark solves multiple, independent, non diagonally dominant, penta-diagonal systems of scalar equations. By contrast, BT solves multiple, independent, non diagonally dominant, block tri-diagonal systems of equations with block size 5×5. SP and BT are representatives of computations associated with the implicit operators of CFD codes and similar in many aspects with the essential difference being the communication to computation ratio (Bailey et al., 1991).

4.2 UTBEST3D

Our real-world application is UTBEST3D, a fully featured regional and coastal ocean model that, among other applications, can be used for flood prediction. with a numerical solution algorithm The mathematical model is the system of hydrostatic primitive equations with a free surface (Dawson and Aizinger,

Table 2: List of benchmarks in the NAS Parallel Benchmark suite with their respective characteristics.

Name	Description	Characteristic
CG	Conjugate Gradient with irregular memory access and communication	memory-bound
MG	Multigrid on a sequence of meshes, long- and short-distance communication	memory-bound
FT	Discrete 3-D Fast Fourier Transform containing all-to-all communication	cache-bandwidth bound
EP	Embarrasingly parallel benchmark generating random numbers	compute-bound
IS	Integer sort with random memory access	integer comp. / comm. speed
LU	Lower-upper Gauss-Seidel solver with blocking	CFD-kernel
BT	Block tri-diagonal solver	CFD-kernel
SP	Scalar penta-diagonal solver	CFD-kernel

2005; Aizinger et al., 2013; Reuter et al., 2015). The discretization is based on the *local discontinuous Galerkin* (LDG) method (Cockburn and Shu, 1998) that represents a direct generalization of the cell-centered finite volume method, the latter being just the piecewise constant DG discretization. One of the features of this method is a much smaller numerical diffusion exhibited by the linear and higher order DG approximations compared to the finite difference or finite volume discretization. The implementation guarantees the element-wise conservation of all primary unknowns, supports an individual choice of the approximation space for each prognostic and diagnostic variable, demonstrates excellent stability properties, and can use mesh adaptivity.

The underlying prismatic mesh is obtained by, first, projecting a given unstructured triangular mesh in the vertical direction to provide a continuous piecewise linear representation of the topography and the free surface. The vertical columns are then subdivided into layers. Due to the discontinuous nature of the approximation spaces, no constraints need to be enforced on the element connectivity. Hanging nodes and mismatching elements are allowed and have no adverse effects on stability or conservation properties of the scheme. This flexibility with regard to mesh geometry is exploited in several key parts of the algorithm: vertical mesh construction in areas with varying topography, local mesh adaptivity, and wetting/drying.

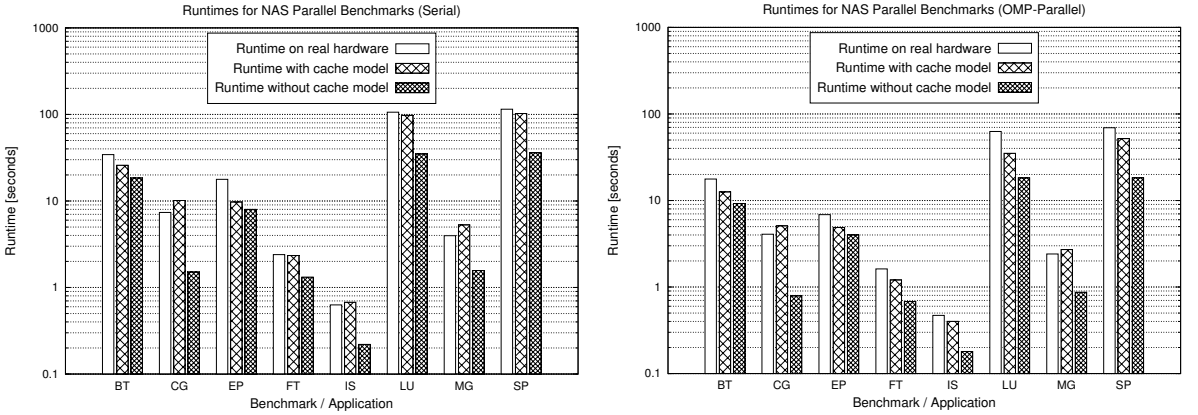
UTBEST3D is written in C++ to provide clean interfaces between geometrical, numerical, computational, and communication parts of the code. The object-oriented coding paradigm is designed to enable a labor efficient development lifecycle of the model. The programming techniques were carefully chosen and tested with the view of guaranteeing a smooth portability to different hardware architectures, operating systems, compilers, and software environments. It is parallelized using MPI and OpenMP, however, within this study only the serial and OpenMP-parallel versions are used. A detailed description of the numerical algorithm and the OpenMP-parallelization can be found in (Reuter et al., 2015).

As model setup we choose a tidal scenario in the Gulf of Mexico with an input mesh consisting of ca. 15 000 triangles and up to 10 layers, resulting in ca. 18 000 prismatic elements. The simulations are done using a barotropic model and an algebraic vertical eddy viscosity parameterization with a total of ca. 260 000 degrees of freedom. Simulated time varies between 0.0001 days (UTB SER S / UTB OMP S, cf. Sec. 6), 0.001 days (UTB SER M / UTB OMP M), and 0.01 days (UTB SER L / UTB OMP L).

5 OVP INSTRUMENTATION AND MODELING

The instruction accurate simulation environment of OVP allows to track and trace each individual instruction in the program flow. To utilize this functionality to capture memory accesses, we designed a light weight library that allows to start and stop the recording of memory access instructions from within the measured application to limit data acquisition to the relevant region of interest. This can be simply achieved by linking the application against our library and calling the start / stop-routines before and after the region of interest, respectively. Since most HPC applications are written in either C/C++ or Fortran, especially legacy applications use the Fortran programming language, we have implemented the library in C, allowing to interface with both programming languages and without any additional requirements.

Additionally to the development of our library, we extended and enhanced an existing OVP cache model for usage in conjunction with our library. That results in separated and configurable L1 and L2 cache models. Both models consider and distinguish the amount of cache read and write accesses. Due to further improvements of our model, we are now able to detect which SMP CPU triggered the read or write access for both, L1 and L2. By using a suitable cost function (cf. Sec. 6), the cache read and write accesses can be used to better estimate the runtime of the measured application.



(a) Serial execution of NAS Parallel Benchmark.

(b) OpenMP-Parallel execution of NAS Parallel Benchmark.

Figure 3: Comparison of runtimes for the execution on real hardware, along with OVP based simulations without a memory model and the estimated runtimes based on our new cache model. In all cases our cache model improves the overall accuracy, i.e., compared to the runtimes on real hardware, significantly.

6 RESULTS

6.1 Benchmarks and Runtime Estimates

We carried out extensive measurements for all NAS benchmarks for both, serial and OpenMP parallelized versions using problem class W, which is one of seven supported classes and has a fixed problem size for every benchmark. All benchmarks are analyzed with respect to their runtime behavior, i.e., we compare the results from runs on the real hardware platform with the runtimes in the simulation environment. For the simulation we use two types of models: (i) a basic instruction accurate model and (ii) our extended model including a level 1 (L1) and level 2 (L2) cache (cf. Sec. 5). We configure the L1 and L2 cache in our model in accordance with the reference hardware platform (cf. Sec. 3.2).

Without our cache model, the basic runtime $\mathrm{rt}_{\mathrm{basic}}$ is estimated by dividing the total number of recorded instructions from the instruction accurate simulation by the ARM Cortex-A9 CPU's clock speed

$$\mathrm{rt}_{\mathrm{basic}} = \frac{\#\mathrm{Instr}}{925\ \mathrm{MHz}}\,. \tag{1}$$

Our observation (cf. Fig. 3) is that, generally, the estimated runtime $\mathrm{rt}_{\mathrm{basic}}$ using the basic instruction accurate model in the simulation is lower than the execution on the real hardware. This is easily explained with the fact that the simulation does not account for the additional overhead connected to cache misses in real hardware. The simulator assumes a constant and thus generally too low latency for each memory access. In conjunction with our cache model, we use a modified runtime estimate $\mathrm{rt}_{\mathrm{cache}}$ by adding a penalty corresponding to the number of additionally required cycles associated with an L1 or L2 cache miss to the basic runtime estimate (1) and obtain the total runtime as

$$\mathrm{rt}_{\mathrm{cache}} = \mathrm{rt}_{\mathrm{basic}} + \mathrm{rt}_{\mathrm{pen}}\,. \tag{2}$$

Using the recorded number of L1 cache misses (L2 cache hits) and L2 cache misses we can derive a generic cost function

$$\mathrm{rt}_{\mathrm{pen}} = \frac{\#\mathrm{L1}_{\mathrm{miss}} \cdot 6 + \#\mathrm{L2}_{\mathrm{miss}} \cdot 88}{925\ \mathrm{MHz}}\,, \tag{3}$$

where the penalty of 6 cycles for an L1 cache miss (L2 cache hit) is based on the data sheet, stating a best case delay of 6 cycles for an L1 cache miss. The penalty of 88 cycles for L2 cache misses (data to be fetched from main memory) was determined by us via empirical testing.

When running the simulations with our cache model, the results become more accurate (cf. Fig. 3). In the best case, there is a deviation less than one percent (serial SP benchmark). The same holds for the case of OpenMP parallelization: the results are getting more precise for all benchmarks when compared to the basic instruction accurate simulations without a cache model. While compute-bound benchmarks (e.g., EP) still leave some room for improvement, scenarios that are dominated by memory access operations (such as memory bound benchmarks and CFD-kernels) gain the most from the cache model.

6.2 Application Results

We verify our runtime estimation model (2) by comparing the results obtained for the application

UTBEST3D on real hardware and in the simulation environment. We measure a serial and an OpenMP-parallel version of UTBEST3D with different simulation lengths, divided into the three classes S, M, and L (cf. Sec. 4.2). The results are shown in Fig. 4.

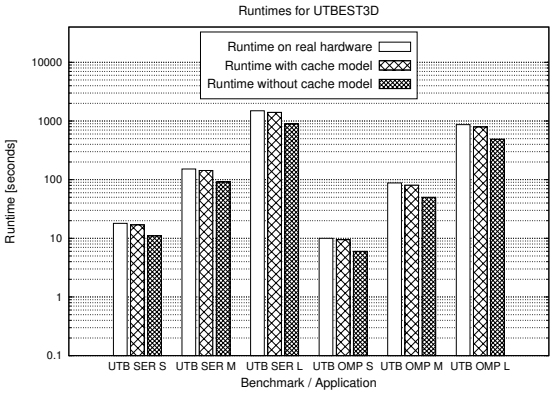


Figure 4: Comparison of runtimes for the execution on real hardware, along with OVP based simulations without a memory model and the estimated runtimes based on our new cache model for UTBEST3D.

Clearly, the runtime predictions for the real world application are more accurate when using the cache model with cost function (3) than the estimates obtained from the basic instruction accurate simulation. This confirms our findings from the benchmark runs.

7 CONCLUSION AND FUTURE WORK

In this paper, we investigate and improve the basic instruction accurate simulation technology from OVP in order to obtain more accurate results with respect to the runtime prediction of applications. Our results show that state-of-the-art instruction accurate simulation can be significantly enhanced by the use of hardware specific cache models. This is an important first step towards increasing the accuracy of the hardware simulation at little additional runtime overhead. It is especially important as both complexity as well as execution time for simulations of future software systems are expected to increase steadily. Furthermore, the complexity for simulations increases exponentially with the complexity and size of the simulated software and hardware.

We use the NAS Parallel Benchmark suite, a selection of individual serial and parallel kernels that contains a broad and representative set of applications corresponding to application classes in the HPC domain to quantify the improvements provided by our cache model. These findings are confirmed using the real-world application UTBEST3D.

As a next step, we are going to develop and analyze the impact of statistical pipeline models on the accuracy of simulations. Since each additional level of accuracy in the modeling phase corresponds to additional overhead in the runtime of the simulations, our goal is to find the sweet spot between simulation accuracy and runtime.

REFERENCES

Aizinger, V. and Dawson, C. (2002). A discontinuous Galerkin method for two-dimensional flow and transport in shallow water. *Advances in Water Resources*, 25(1):67–84.

Aizinger, V., Proft, J., Dawson, C., Pothina, D., and Negusse, S. (2013). A three-dimensional discontinuous Galerkin model applied to the baroclinic simulation of Corpus Christi Bay. *Ocean Dynamics*, 63(1):89–113.

Altera Corp. (2016). Cyclone V SoC Development Kit User Guide. https://www.altera.com/en_US/pdfs/literature/ug/ug_cv_soc_dev_kit.pdf. Last visit on 31.03.2016.

Applegate, D., Bixby, R., Chvátal, V., and Cook, W. (2011). *The Traveling Salesman Problem: A Computational Study: A Computational Study*. Princeton Series in Applied Mathematics. Princeton University Press.

Bailey, D. H., Barszcz, E., Barton, J. T., Browning, D. S., Carter, R. L., Dagum, L., Fatoohi, R. A., Frederickson, P. O., Lasinski, T. A., Schreiber, R. S., et al. (1991). The NAS parallel benchmarks – summary and preliminary results. In *Proceedings of the 1991 ACM/IEEE conference on Supercomputing*, pages 158–165. ACM.

Barker, K. and Kerbyson, D. (2005). A performance model and scalability analysis of the hycom ocean simulation application. In *Proc. IASTED Int. Conf. On Parallel and Distributed Computing*.

Carlson, T. E., Heirman, W., and Eeckhout, L. (2011). Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulations. In *International Conference for High Performance Computing, Networking, Storage and Analysis (SC)*, pages 52:1–52:12.

Castro, M., Francesquini, E., Nguélé, T. M., and Méhaut, J.-F. (2013). Analysis of computing and energy performance of multicore, numa, and manycore platforms for an irregular application. In *Proceedings of the 3rd Workshop on Irregular Applications: Architectures and Algorithms*, IA3 '13, pages 5:1–5:8, New York, NY, USA. ACM.

Cockburn, B. and Shu, C.-W. (1998). The Local Discontinuous Galerkin Method for Time-Dependent Convection-Diffusion Systems. *SIAM Journal on Numerical Analysis*, 35(6):2440–2463.

Cowles, G. W. (2008). Parallelization of the FVCOM coastal ocean model. *International Journal of High*

Performance Computing Applications, 22(2):177–193.

Dawson, C. and Aizinger, V. (2005). A discontinuous Galerkin method for three-dimensional shallow water equations. *Journal of Scientific Computing*, 22(1-3):245–267.

Dietrich, J., Tanaka, S., Westerink, J., Dawson, C., Luettich, R.A., J., Zijlema, M., Holthuijsen, L., Smith, J., Westerink, L., and Westerink, H. (2012). Performance of the unstructured-mesh, swan+adcirc model in computing hurricane waves and surge. *Journal of Scientific Computing*, 52(2):468–497.

Eeckhout, L., Bell, R. H., Stougie, B., De Bosschere, K., and John, L. K. (2004). Control flow modeling in statistical simulation for accurate and efficient processor design studies. In *Proceedings of the 31st Annual International Symposium on Computer Architecture, 2004*, pages 350–361. IEEE.

Genbrugge, D. and Eeckhout, L. (2009). Chip Multiprocessor Design Space Exploration through Statistical Simulation. *Computers, IEEE Transactions on*, 58(12):1668–1681.

Göddeke, D., Komatitsch, D., Geveler, M., Ribbrock, D., Rajovic, N., Puzovic, N., and Ramirez, A. (2013). Energy efficiency vs. performance of the numerical solution of PDEs: An application study on a low-power ARM-based cluster. *J. Comput. Phys.*, 237:132–150.

Imperas Software Ltd. (2015). *OVP Guide to Using Processor Models*. Imperas Buildings, North Weston, Thame, Oxfordshire, OX9 2HA, UK. Version 0.5, docs@imperas.com.

Imperas Software Ltd. (2016). Description of Altera Cyclone V SoC. http://www.ovpworld.org/library/wikka.php?wakka=AlteraCycloneVHPS. Last visit on 31.03.2016.

ITMC TU Dortmund (2015). Official LiDO website. https://www.itmc.uni-dortmund.de/dienste/hochleistungsrechnen/lido.html. Last visit on 26.03.2015.

John Hardman (2016). Official NAS Parallel Benchmarks Website. http://www.nas.nasa.gov/ publications/npb.html. Last visit on 12.04.2016.

KALRAY Corp. (2015). Official kalray mppa processor website. http://www.kalrayinc.com/kalray/ products/#processors. Last visit on 31.03.2015.

Kerbyson, D. J. and Jones, P. W. (2005). A performance model of the parallel ocean program. *International Journal of High Performance Computing Applications*, 19(3):261–276.

Miller, J., Kasture, H., Kurian, G., Gruenwald, C., Beckmann, N., Celio, C., Eastep, J., and Agarwal, A. (2010). Graphite: A distributed parallel simulator for multicores. In *IEEE 16th International Symposium on High Performance Computer Architecture (HPCA), 2010*, pages 1–12.

Nair, R., Choi, H.-W., and Tufo, H. (2009). Computational aspects of a scalable high-order discontinuous galerkin atmospheric dynamical core. *Computers & Fluids*, 38(2):309 – 319.

NVIDIA Corp. (2015). Official NVIDIA SECO development kit website. https://developer.nvidia.com/seco-development-kit. Last visit on 31.03.2015.

Rajovic, N., Carpenter, P. M., Gelado, I., Puzovic, N., Ramirez, A., and Valero, M. (2013). Supercomputing with commodity cpus: Are mobile SoCs ready for HPC? In *Proceedings of the International Conference on High Performance Computing, Networking, Storage and Analysis*, SC '13, pages 40:1–40:12, New York, NY, USA. ACM.

Rajovic, N., Rico, A., Puzovic, N., Adeniyi-Jones, C., and Ramirez, A. (2014). Tibidabo: Making the case for an ARM-based HPC system. *Future Generation Computer Systems*, 36(0):322 – 334.

Reuter, B., Aizinger, V., and Köstler, H. (2015). A multi-platform scaling study for an OpenMP parallelization of a discontinuous Galerkin ocean model. *Comput Fluids*, 117:325 – 335.

Ringler, T., Petersen, M., Higdon, R. L., Jacobsen, D., Jones, P. W., and Maltrud, M. (2013). A multi-resolution approach to global ocean modeling. *Ocean Modelling*, 69:211 – 232.

Schoenwetter, D., Ditter, A., Kleinert, B., Hendricks, A., Aizinger, V., Köstler, H., and Fey, D. (2015). Tsunami and Storm Surge Simulation using Low Power Architectures – Concept and Evaluation. In *SIMULTECH 2015 - Proceedings of the 5th International Conference on Simulation and Modeling Methodologies, Technologies and Applications*, pages 377–382.

Shu, C.-W. (2016). High order {WENO} and {DG} methods for time-dependent convection-dominated pdes: A brief survey of several recent developments. *Journal of Computational Physics*, 316:598 – 613.

Tanaka, S., Bunya, S., Westerink, J. J., Dawson, C., and Luettich, R. A. (2011). Scalability of an unstructured grid continuous galerkin based hurricane storm surge model. *J. Sci. Comput.*, 46(3):329–358.

Wallcraft, A., Hurlburt, H., Townsend, T., and Chassignet, E. (2005). 1/25 degree atlantic ocean simulation using hycom. In *Users Group Conference, 2005*, pages 222–225.

Worley, P. and Levesque, J. (2004). The performance evolution of the parallel ocean program on the cray x1. In *Proceedings of the 46th Cray User Group Conference*, pages 17–21.