

Memory Analysis and Performance Modeling for HPC Applications on Embedded Hardware via Instruction Accurate Simulation

Alexander Ditter^{1(✉)}, Dominik Schoenwetter¹, Anton Kuzmin¹,
Dietmar Fey¹, and Vadym Aizinger²

¹ Chair of Computer Science 3 (Computer Architecture),
Friedrich-Alexander-Universität Erlangen-Nürnberg (FAU), Erlangen, Germany
{alexander.ditter, dominik.schoenwetter,
anton.kuzmin, dietmar.fey}@fau.de

² Chair of Applied Mathematics (AM1), Friedrich-Alexander-Universität
Erlangen-Nürnberg (FAU), Erlangen, Germany
aizinger@math.fau.de

Abstract. The efficient usage and development of embedded multi- and many-core systems is an important field of research for years and decades. In the last decade, utilizing embedded and especially low-power architectures for high performance computing (HPC) applications became an important part of research. The reason for this are the constantly increasing energy costs along with an increasing awareness of energy consumption in general. As suitable low-power HPC architectures are not yet available at a larger scale, simulation of new and appropriate architectures becomes an important factor for the success of low-power systems and clusters. In order to speed up simulation, at the cost of accuracy, different levels of abstraction were introduced. Currently the class of instruction accurate simulations seems to yield the best trade-off between speed and precision, especially when simulating complex multi- and many-core systems. In this paper we present our investigations about the accuracy of the state-of-the-art instruction accurate embedded multi- and many-core simulation environment Open Virtual Platforms (OVP) in comparison to an actual embedded hardware system from Altera. Our investigations include the actual usage of the same operating system running on both, the target hardware and the simulation as well as serial and parallel software benchmarks. We analyze the current accuracy of the simulation environment with respect to a performance model, based on the execution time of the simulation and the actual embedded hardware system. Using the instruction accurate simulation technology from OVP is for the simulation of embedded/low-power HPC hardware architectures and applications. Furthermore, we point out first steps towards further possibilities for obtaining a better performance model by the use of our simple memory model.

1 Introduction

Single-core processors have been replaced by multi-core, and in many areas of *high performance computing* (HPC) to a significant extend by many-core processors, such as *general purpose GPUs* (GPGPUs) or Intel's *many integrated core* (MIC) system.

While this development has progressed for well over a decade it has just reached its slow pick-up phase in the field of embedded low-power systems during the last years. Yet, the same restrictions and requirements, e.g., the thermal power wall along with decreasing structures in the manufacturing process and that have been driving this conversion for desktop, server and HPC systems become more and more important for many applications in the embedded domain. The increasing cost for single-core systems, due to the low remaining stocks and production rates, but also and more importantly their lack of performance, becomes more and more of a hindrance, e.g., for driving assistance systems, where computational power is indispensable. Thus, the transition to multi-core systems was necessary and unavoidable. Currently, as more and more functionality is added to applications running on the embedded devices, as well the steadily increasing interest in utilizing them for HPC applications, the additional computing power is needed to satisfy the increasing performance requirements. This, in turn, leads to the development of interconnected multi-core architectures and cluster systems, mostly based on conventional and HPC network fabrics, as embedded hardware developers abandon the use of the more traditional bus systems. Even if such systems are currently not yet an established standard they will play an important role in most embedded low-power application domains in the future. Due to the increasing requirements for power and energy efficiency for HPC systems and architectures, a significant amount of research has been put into the utilization of the afore mentioned conventional multi- and many-core low-power architectures. Yet, as suitable low-power architectures for HPC systems currently are still only scarcely available, accurate simulations of these architectures are a key instrument for verification, memory analysis, performance modeling, prediction and evaluation. It may even be used to guide to development of new embedded hardware towards a more suitable design for HPC applications. Historically, simulation techniques can be categorized in different abstraction levels (cf. Table 1) [1], where the level of abstraction differentiates two main aspects: (i) the speed of the simulation and (ii) the accuracy of the results. In this classification the functional model corresponds to the highest level of abstraction, resulting in the lowest simulation accuracy, but the highest simulation speed. The lowest level of abstraction is the physical level, where even physical effects, e.g., of transistor gate delays, are considered. Naturally, the physical level has the worst simulation speed, yet the highest accuracy of simulation results. The functional level does not distinguish between hard- and software. Thus, this level is too abstract for evaluating combined soft- and hardware aspects. Yet, the instruction accurate level does distinguish between hard- and software. Furthermore, it is even possible to obtain information about non-functional properties like time and energy. At the same time the simulation speed of instruction accurate simulations is very fast, as compared to all more accurate modeling techniques (cycle accurate, RTL and physical level). In terms of simulation speed, the cycle accurate simulation level is very slow in comparison to the instruction accurate one. Weaver and McKee [2] showed that discrepancies of hours up to days are possible and the simulation results are not mandatory more accurate. As a consequence, cycle accurate simulations are not an option for the simulation of multi- or large many-core systems. Ideally the simulation of large embedded multi-core systems, has to be both: accurate and fast. Due to the large number of processors, peripherals and network interconnects in such a system,

Table 1. Level of abstraction decreases from top to bottom, while accuracy increases.

Modeling technique	Modeling language
Functional Model	MATLAB/Simulink
Instruction Accurate Model	Instruction Set Simulator (ISS)
Transactional Level without Time (Programmers View without Time)	Un-Timed SystemC
Transactional Level with Time (Programmers View with Time)	Timed SystemC
Cycle Accurate Model	C model
Register Transfer Level (RTL)	HDL like VHDL
Physical Level	SPICE

especially the simulation speed is an important aspect. A state-of-the-art instruction accurate simulation technology is called *Open Virtual Platforms* (OVP) and provided by Imperas [3]. This simulation environment shows excellent properties concerning to simulation speed. Also, the scalability of the simulation results for different numbers of cores was verified [4].

In this paper we investigate the accuracy of results from the state-of-the-art instruction accurate simulation environment OVP along with their applicability for the verification and evaluation of embedded HPC architectures and algorithms. Furthermore, we have developed an instrumentation mechanism for OVP that allows us to track, record and trace memory access patterns within the simulation of whole applications as well as functions and instructions. Using this technique, we can provide a better understanding for memory access patterns to analyze and compare different algorithms and hardware systems and advice software and hardware developers about potential improvements in their respective designs, allowing for an overall better hardware-software co-design. We extend this capability with a first simple memory model, allowing us to use this approach for a more accurate performance modeling than conventional instruction accurate simulation techniques. For our analysis and evaluation we compare the results of OVP to those obtained from our actual reference hardware, the Altera Cyclone V *system on chip* (SoC) (cf. Sect. 3.2), as we emulated the Altera SoC using OVP.

The rest of the Paper is structured as follows. Section 2 provides a comprehensive survey of related work in the field. Section 3 gives an in depth overview of the OVP simulation environment as well as the actual hardware system we compare our results against. Section 4 explains our instrumentation for tracing memory access in OVP along with the memory and performance modeling. Section 5 introduces the applications and benchmarks used for our investigations. Section 6 presents our findings and quantitative results along with a conclusion and an outlook on future work in Sect. 7.

2 Related Work

There is a significant body of research in the field of utilizing low-power architectures for HPC and in the optimization of energy efficiency for HPC applications. Rajovic et al. investigated the usage of low-power ARM¹ architectures and SoCs as means to reduce the cost of HPC [5]. They conclude that low-power ARM-based SoCs have promising characteristics for HPC. In 2013, Goeddeke et al. presented a paper on energy-to-solution comparisons between different architectures for different classes of numerical methods for partial differential equations [6]. They 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 based on a setting with 96 ARM Cortex-A9 dual-core processors [7], instead of an x86-based cluster. The x86 cluster used for the reference measurements was the Nehalem sub-cluster of the LiDong machine provided by TU Dortmund [8]. A study comparing the performance as well as the energy consumption of different low-power and general-purpose architectures was published by Castro et al. [9]. Based on the Traveling-Salesman problem [10], they investigated time-to-solution and energy-to-solution for an Intel Xeon E5-4640 Sandy Bridge-EP with 8 cores, 16 threads, i.e., hyper-threading support, a low-power Kalray MPPA-256 many-core processor consisting of 256 user cores and 32 system cores [11], as well as for a low-power CARMA board from SECO [12], consisting of a NVIDIA Tegra 3 and a NVIDIA Quadro 1000M GPU. The results show, that the CARMA board and the MPPA-256 many-core processor achieve better results than the Xeon 5 measured in terms of energy to solution. With regard to the time-to-solution metric, the Xeon 5 performed better than the CARMA board but not as good as the low-power MPPA-256 many-core processor. A work considering low-power processors and accelerators in terms of energy aware HPC was published in 2013 [13]. There, a number of different HPC micro-benchmarks was used to determine the energy-to-solution. The architectures evaluated were NVIDIA Tegra 2 [14] and Tegra 3 [15] SoCs. The results show that drastic energy to solution improvements are possible on the newer Tegra 3 SoC in comparison to the Tegra 2 SoC (reduction of 67% on average). Furthermore, the authors conclude that the usage of integrated GPUs in low-power architectures, such as Tegra 2 and Tegra 3, can improve the overall energy efficiency. All presented investigations emphasize, that lowpower hardware architectures have promising characteristics for HPC. There is a range of tools that allow memory modeling and simulation, e.g., Gem5 [16], DiskSim [17] or DRAMSim2 [18]. The cycle accurate simulation environment Gem5 allows a comprehensive set of building blocks, ranging from caches, crossbars, to full-blown DRAM controllers. DiskSim is an accurate and highly-configurable disk system simulator. It was developed to support research into various aspects of storage subsystem architecture, including modules that simulate intermediate controllers, buses, device drivers, request schedulers, disk block caches, and disk array data organizations. DRAMSim2 is a cycle accurate memory system simulator. The goal of DRAMSim2 is to be an accurate DDR2/3 memory system model which can be used in tracebased and full system simulations.

¹ <http://www.arm.com/>.

3 Environment

3.1 Simulation Environment

The simulation technology from Open Virtual Platforms was developed for high performance simulation. The technology enables debugging applications, which run on the virtual hardware, as well as analysis of virtual platforms containing multiple processor and peripheral models. The OVP simulation technology is extensible. Furthermore, it provides the ability to create new processor models and other platform components by writing C/C++ code that uses the *application programming interface* (API) and libraries supplied as part of OVP [19]. The API defines a virtual hardware platform which is called ICM (Innovative CPUManager Interface). This API includes functions for setting up, running and terminating a simulation (*icmInitPlatform*, *icmSimulatePlatform*, *icmTerminate*), defining components for the simulation (e.g., *icmNewProcessor*) and loading application's executable (*icmLoadProcessorMemory*). Figure 1 shows an overview about how an OVP simulation with minimal effort works. Minimal effort means, that one processor has to be defined for the simulation. The example uses the language C for the hardware platform as well as the application to run on that platform. The simulator included in OVP is an instruction accurate simulator. This means, that the functionality of a processor's instruction execution is represented without regard to artifacts like pipelines. Instruction accurate simulation cannot make a clear statement about time spent during pipeline stalls, due to cache misses and other things that are not modeled, so any conversion to time will have limited accuracy compared to actual hardware. OVP multi-processor platforms are not working simultaneously. For efficiency, each processor advances a certain number of instructions in turn. So in multi-processor simulations a single processor is simulated until it has signaled that it has finished its quantum. The quantum is defined as the time period in which each component in turn simulates a certain number of instructions [19]. Simulated time is moved forward only at the end of a quantum. This can create simulation artifacts, for example where a processor spends time in a wait loop, while waiting for the quantum to finish. To avoid this the quantum has to be set very low value (potentially having a significant impact on the simulation performance) so that the measurements will not be affected by this simulation artifacts. This can be adjusted in the simulator settings [20]. The simulation environment can only provide the total amount of instructions that were 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 run time of the program. The OVP simulator provides the possibility for measuring instruction counts within a program. As a consequence, the instruction counts for specific code snippets can be recorded. On singlecore platforms, assuming that no time-controlled peripheral models are invoked, there is no need to set the quantum to one because the multi-core scheduling algorithm does neither affect nor intervene the simulation.

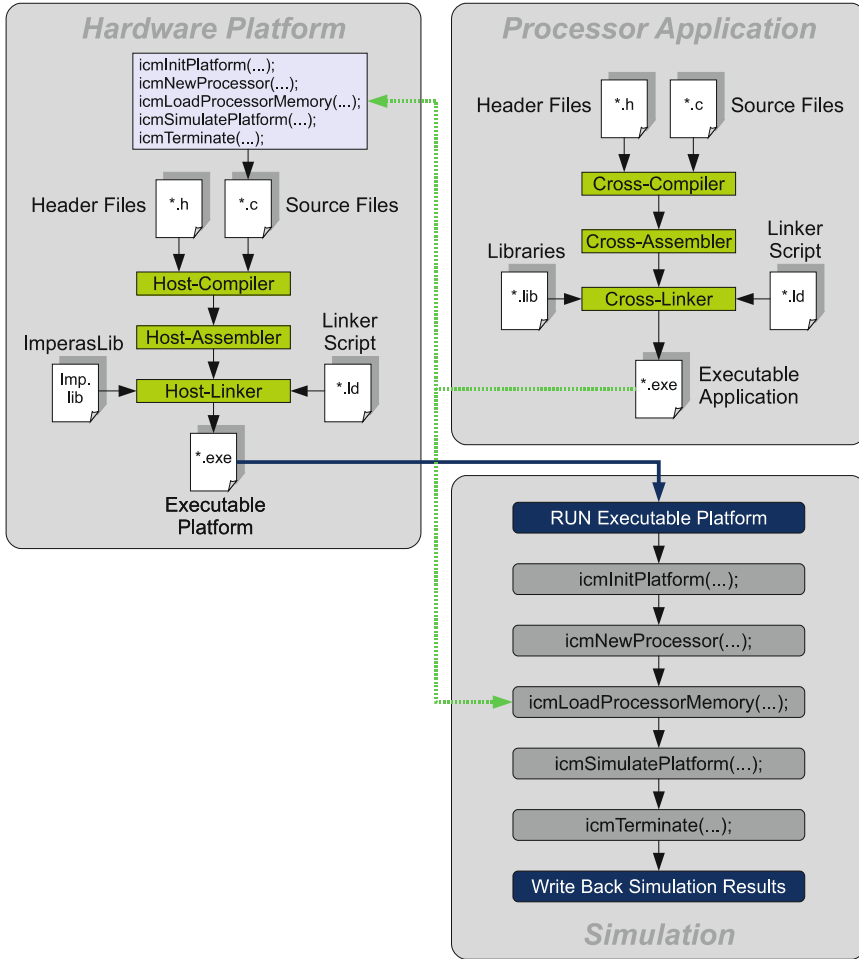


Fig. 1. Operating principle of open virtual platforms simulations.

3.2 Reference Hardware

Embedded system developers have to satisfy multiple requirements such as a high computational performance, support for a wide variety of communication interfaces and protocols, execution of complex signal processing algorithms in realtime, low power consumption. All these requirements have to be fulfilled with a very limited amount of resources. Given a long lifetime of the deployed systems and ever changing environmental conditions, the in-field support and upgrade is one of the most crucial requirements. A common solution to these demands is the extensive use of *field-programmable gate arrays* (FPGAs) for the hardware part and on the software side, relying on a processor architecture with a well-established and active ecosystem. Just a few years ago this approach implied at least two separate complex chips to be used in a single system. However the situation has recently changed and several FPGA vendors

came to market with integrated devices combining high-performance ARM CPUs, a fast memory controller along with a rich set of peripheral devices and a programmable logic unit. The integration of all these units into a single SoC provides developers with multiple benefits and thus, we expect such systems to become established in many embedded devices. For this reason we chose Altera's development kit board [21] with a Cyclone V SX SoC-FPGA as our reference hardware platform to carry out our benchmarking. This SoC-FPGA includes a *hard processor system* (HPS) consisting of *multiprocessor subsystem* (MPU), multiport SDRAM controller with support for *double data rate 2* (DDR2), DDR3 and low-power DDR2 memory, 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. On the board 1 GiB of DDR3 SDRAM is connected to the memory controller via a 40-bit data bus operating at 400 MHz for a total theoretical bandwidth of over 25.6 Gbps. The multiprocessor subsystem of the HPS includes two ARM Cortex-A9 MPCore 32-bit processors with a NEON SIMD co-processor and double-precision floating point unit (FPU) per processor, 32 KiB instruction and 32 KiB data level 1 (L1) caches and *memory management unit* (MMU) per processor, ARM level 2 (L2) cache controller and shared 512 KiB L2 cache. 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. Low-speed peripheral controllers reside on the level 4 (L4) AXI buses working in separate clock domains for efficient power management. Programmable logic part of the SoC is a high-performance 28 nm FPGA. The HPS and FPGA part of the chip are connected via high-bandwidth (> 125 Gbps) on-chip interfaces. All the benchmarks presented in this paper use only the HPS part of the SoC-FPGA, while 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 platform implements just a part of the actual Altera Cyclone V SoC [22]. Specific and not needed hardware parts, e.g., the FPGA block, are not implemented. Anyway, all required components for running a Linux kernel and guarantee correct hardware functionality for our test cases are available. Figure 2 shows the subset of implemented hardware components. The virtual hardware allows to boot the same Linux kernel (3.16.0) as the actual hardware does. As a consequence, the virtual and the actual hardware are binary compatible.

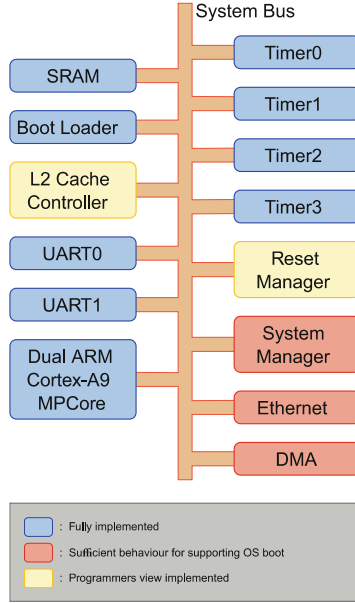


Fig. 2. Virtual Cyclone V SoC

4 OVP Instrumentation and Modeling

OVP, being an instruction accurate simulation environment, allows to track and trace each individual instruction in the program flow, we utilize this functionality to capture each memory access. For this purpose we designed a light weight library that allows to start and stop the recording of memory access instruction from within the respective application. This can simply be achieved by linking against our library. Since many HPC applications are either written in C/C++ as well as Fortran, especially for legacy application and as in our case the benchmark set, we have designed the library in C, allowing to interface with both programming languages without any additional requirements. The library offers the possibility to start and stop the recording of memory accesses, thus, allowing to restrict the data acquisition to the respective region of interest. This may be an entire application as well as a function call or an individual instruction. Currently the library is capable to record the total number of read and write accesses. We have also experimented with a more detailed recording, which is currently not incorporated any more, as it slows down the entire simulation by one to two orders of magnitude and creates memory access log files of several GiB within just a few minutes of host run time. Nonetheless, we plan to extend the current capabilities by employing techniques to reduce the memory consumption of trace files analogous to [23].

Using the results of our memory access recording library we obtained we designed a first very basic memory model for OVP, which considers and distinguishes the amount of read and write access to memory and weighs them with different factors. This makes sense, as it is much less likely for a write access to be delayed as much as a

read access, in case the data is not available in the cache. We use these factors to better estimate the performance, i.e., run time of the applications and benchmarks in Sect. 5.

5 Benchmarks and Applications

Our investigations are based on one real world HPC application, one real world but artificial problem and an extensive artificial benchmark set. All these individual benchmarks resemble typical computational fluid dynamics (CFD) applications and provide a wide range of typical HPC characteristics, such as compute and memory bound kernels. We evaluate our approach for the well known 3D shallow-water solver UTBEST3D, which is a typical high performance computing application and described in detail in Sect. 5.1 as well as the computation of mandelbrot sets. Additionally we use the NAS Parallel Benchmark suite (NPB). [24], i.e., the eight original benchmarks specified in NPB 1, consisting of five kernels and three pseudo applications. We cross-compiled the applications and benchmarks for both, the real hardware as well as the OVP simulation to ensure binary equality and thus, best possible comparability of the obtained results. For this, we used gfortran-arm-linux-gnueabi for the Fortran based and gcc-arm-linux-gnueabi for the C based benchmarks (both in version 4.8.2). The same binaries were used in the real and the virtual environment. The individual characteristics of UTBEST3D, the mandelbrot set and the NAS benchmarks are described in the following.

5.1 UTBEST3D – U3D

The numerical solution algorithm in the 3D shallow-water solver University of Texas Bays and Estuaries Simulator - 3D (UTBEST3D) considers the system of hydrostatic primitive equations with a free surface [25, 26]. A prismatic mesh is obtained by projecting a given triangular mesh in the vertical direction to provide a continuous piecewise linear representations of the topography and of the free surface. The vertical columns are then subdivided into layers. If a bottommost prism is degenerate, it is merged with the one above it. 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. The discontinuous Galerkin (DG) discretization is based on the local discontinuous Galerkin method [27] 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 method guarantees the elementwise conservation of all primary unknowns including tracers, supports an individual choice of the approximation space for each prognostic and diagnostic variable, demonstrates excellent stability properties, and possesses

proven local adaptivity skills. 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.

5.2 Mandelbrot Set – MB

A mandelbrot set is defined as the set of complex numbers in the complex plane where the sequence $c; c^2 + c; (c^2 + c)^2 + c; \dots$ does not approach infinity, even if the iteration counter tends to infinity [28]. Due to the characteristic of the sequence, it can be defined as a complex quadratic polynomial of the form $z_{n+1} = z_n^2 + c$; with $z_0 = 0$. Mandelbrot sets are often visualized by a mapping from the complex plane into an image representation. For this purpose the imaginary and real part of each complex number is considered as an image coordinate. Depending on how rapid the sequence and quadratic polynomial, of each pixel diverges, the corresponding pixel gets a defined color. If the sequence converges, the pixel is colored black. As each pixel can be computed independently and requires few memory accesses for every iteration, the mandelbrot set is a compute bound application.

5.3 NAS Parallel Benchmarks

The kernels considered in our benchmarking are CG (Conjugate Gradient with irregular memory access and communication), MG (Multi-Grid on a sequence of meshes, long- and short-distance communication), FT (discrete 3D fast Fourier Transform containing all-to-all communication), EP (Embarrassingly Parallel) and IS (Integer Sort with random memory access). LU (Lower-Upper Gauss-Seidel solver) are the defined pseudo applications, BT (Block Tri-diagonal solver) as well as, SP (Scalar Penta-diagonal solver).

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 [29]. The run time of this benchmark is dominated by the sparse matrix vector multiplication in the conjugate gradient subroutine. Due to the random pattern of nonzero entries of 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 the residual is evaluated and used to apply a correction to the current solution. The most expensive parts of this algorithm are evaluation of the residual and application of the smoother, both of which are stencil operations with constant coefficients for the specified problem. The update of a grid point require the values of neighboring points. Thus, even with an optimal implementation this requires in

between four and eight additional memory access operations per grid point. For constant stencil coefficients the run time is dominated by memory access and not 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 and multiplying the result by an exponential. Then an inverse FFT is used to recompute the original solution. Finally, a complex checksum is computed to verify the result [29]. The FFT is dominating the run time of this benchmark. As the implementation in the benchmark uses a blocked variant of the Stockham FFT. This procedure is bound by memory operations, but due to blocking the limiting factor is not directly 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 of many Monte Carlo simulations [29]. The EP benchmark is computationally expensive, complex operations such as computation of logarithms and roots make up a big portion of the total run time 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 is able to sort N keys in parallel. The keys are generated by a sequential key generation algorithm. The sorting operations performed in this kernel are important in particle method codes. Both, integer computation speed as well as communication performance are under test [29]. IS requires ranking of an unsorted sequence of N keys. The initial sequence of keys will be generated in a defined sequential manner, but the initial distribution of the keys can have a significant impact on the performance of this benchmark.

Lower Upper Benchmark – LU: LU uses a Gauss-Seidel solver for lower and upper triangular systems (regular-sparse, block size 5×5). It solves flows 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 implicit CFD algorithms [29].

Diagonal Block Matrix Benchmark – SP and BT: The SP benchmark solves multiple, independent systems of scalar, non diagonally dominant, pentadiagonal equations. BT, however, solves multiple and independent systems of non diagonally dominant, block tridiagonal equations with block size 5×5 . SP and BT are representative of computations associated with the implicit operators of CFD codes. Both, SP and BT, are similar in many respects, the essential difference is the communication to computation ratio [29].

6 Results

We have carried out extensive measurements for all NAS benchmarks, the mandelbrot set as well as UTBEST3D and have obtained the following results.

As described, we have analyzed all benchmarks with respect to their plain run time behavior. This means that we compare the results from test runs on the real hardware platform with the run times in the simulation environment. The observation is that generally the prospective run time in the simulation is faster than the execution on the real hardware. This is due to the fact that the simulation does not contain a proper model for cache and memory access penalties as is the case for real hardware. The simulator assumes a constant and too low latency for each memory access and thus yields a lower run time. The degree of deviation is directly coupled to the total amount of memory access within the individual benchmarks. As can be seen in Fig. 3 the total run time can be as much as 4.9 times slower than the prospected run time of the simulation. In our benchmark set this was the case for the CG benchmark of the NPB suite, i.e., a benchmark with low computational complexity. The high deviation becomes not directly visible, as we chose to display the ordinate in log scale to compensate the varying run times across the whole of our benchmarks. We also scaled the benchmarks to obtain an impression about impact on the relative error of the simulation and the execution on real hardware. As the NPB suite offers several problem size classes we analyzed the two smallest classes. This is on one hand due to the fact that the simulation requires about on order of magnitude more to finish than the execution on the real hardware. On the other hand the memory available on the SoC is limited and cannot fit certain benchmarks beyond this class into memory. The results for both classes were consistent with respect to number of memory accesses and the total run times.

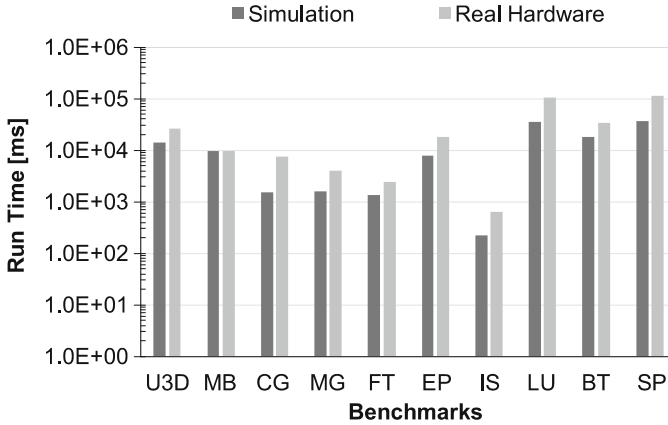


Fig. 3. Comparisons of benchmark run times on real hardware and in simulation; ordinate in log scale.

Furthermore, we carried out the benchmarks in parallel using OpenMP and observed a consistent speedup for both, the simulation as well as the execution on the real hardware. The prognosis of the speedup was slightly overshooting in seven out of ten cases, which was to be expected as the run times in the simulation are consistently faster than on the real hardware.

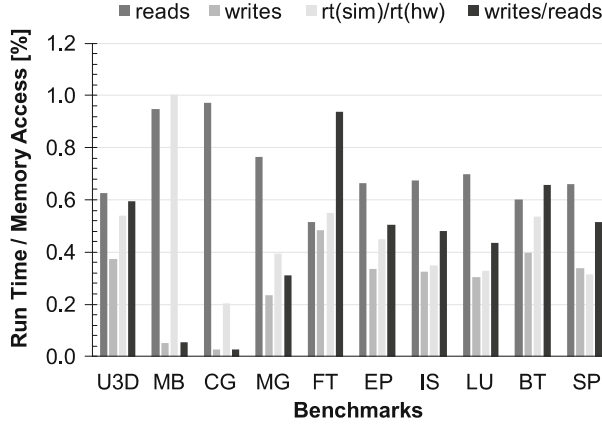


Fig. 4. Percentage and relation of read and write operations for each of the benchmarks, along with the quotient of the predicted run time of the simulation, $rt(sim)$ and the actual run time on the hardware, $rt(hw)$.

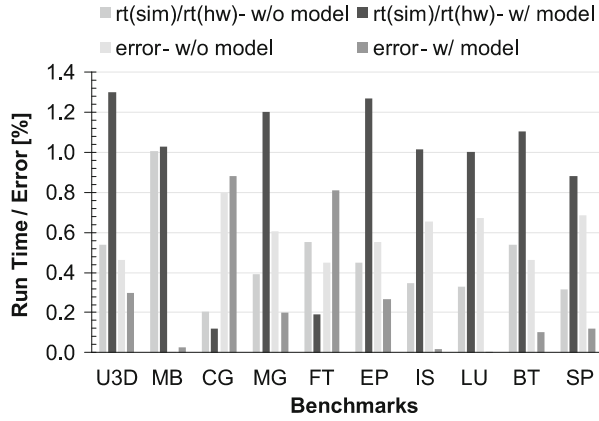


Fig. 5. Quotient of simulation run time, $rt(sim)$ and run time on real hardware, $rt(hw)$, in relation with the relative error with and without our memory model.

As we observed, the predicted run time in the simulation to be heavily dependent on the type of kernel and thus the amount of memory access, we instrumented the simulation model in a way that allows to measure the amount of reading and writing memory access of the benchmark runs. Based on our observation (cf. Fig. 4), we derived a memory model improving the prediction accuracy of the run time on the real hardware in seven out of ten cases (cf. Fig. 5) using function (1), which we derived on the basis of our data.

$$rt(hw) = \frac{e^{(1/m)^2}}{\ln(1/m)^m} \cdot rt(sim) \quad (1)$$

where: m is the m quotient of the #writes/#reads, $rt(hw)$ is the run time on real hardware, which is to be predicted from, $rt(sim)$, the run time obtained from the simulation.

Using this fitting function, the error margin reduces from a standard deviation of close to 0.4 in the case with no memory model to 0.2, when our model is used for the approximation of the run time on the real hardware.

7 Conclusions and Future Work

7.1 Conclusion

We carried out a comprehensive analysis of benchmarks on real and simulated hardware in order to analyze the accuracy of simulations with respect to the execution on real hardware platform. We found that a naive simulation model is not sufficient and must be extended with a proper memory model to match the anticipated results for real hardware runs. We developed such a model that allowed us to reduce the inaccuracy in the prediction of the run time on the real hardware based on the simulation. Yet, we show that the combination of an instruction accurate simulation is well suited as a basis for abstract simulations of multi-core embedded systems. Simulations can be carried out in an acceptable time frame and yield sufficiently accurate results.

7.2 Future Work

In addition to our first evaluation of one HPC application and benchmark a broader range especially of applications has to be evaluated with respect to their computational and memory requirements. Using these applications and benchmark information we want to extend our memory model with information about parallel execution using OpenMP as well as cache hierarchy its related latency, energy consumption and their interaction. Furthermore, we will expand our models and investigations by the ARMv8 architecture (64-bit). Because instruction accurate simulation itself can never provide precise timing information, beyond modeling it with one or several instructions, such as L1-/L2-Cache accesses, displacement, cache miss rates and page faults we want to introduce concepts of statistical memory modeling. An interesting concept about statistical memory modeling was published by Davy Genbrugge and Lieven Eeckhout in 2009 [30]. They extend statistical simulation methodology to model shared resources in the memory subsystem of multi-processors as shared caches, off-chip bandwidth and main memory. In a next step, we will examine how well suited this approach is for our work and if there is an additional benefit. An interesting article about energy modeling was published by Kerrison and Eder in 2015 [31]. They examine a hardware multi-threaded microprocessor and discusses the impact such an architecture has on existing software energy modeling techniques. Their multithreaded software energy model used with Instruction Set Simulation can yield an average error margin of less

than 7%. This model could be also of benefit for us, even if the ARM architecture we use does not support hardware multi-threading, because the base for the multi-threaded energy model was a single threaded one.

References

1. Köhler, C.: Enhancing Embedded Systems Simulation: A Chip-Hardware-in-the-Loop Simulation Framework. Vieweg+Teubner research. Vieweg+Teubner Verlag, Wiesbaden (2011)
2. Weaver, V.M., McKee, S.A.: Are cycle accurate simulations a waste of time? In: Proceedings of the 7th Workshop on Duplicating, Deconstructing, and Debunking, June 2008
3. Imperas Software Limited. Official Open Virtual Platforms Website. <http://www.ovpworld.org/>. Accessed 27 April 2015
4. Schoenwetter, D., Schneider, M., Fey, D.: A speed-up study for a parallelized white light interferometry preprocessing algorithm on a virtual embedded multiprocessor system. In: ARCS Workshops (ARCS), pp. 1–6, February 2012
5. Rajovic, N., Carpenter, P.M., Gelado, I., Puzovic, N., Ramirez, A., Valero, M.: 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 2013, pp. 40:1–40:12. ACM, New York (2013). <http://doi.acm.org/10.1145/2503210.2503281>
6. Göddeke, D., Komatitsch, D., Geveler, M., Ribbrock, D., Rajovic, N., Puzovic, N., Ramirez, A.: 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 (2013). <http://dx.doi.org/10.1016/j.jcp.2012.11.031>
7. Rajovic, N., Rico, A., Puzovic, N., Adeniyi-Jones, C., Ramirez, A.: Tibidabo: making the case for an ARM-based HPC System. Future Gener. Comput. Syst. **36**, 322–334 (2014). <http://www.sciencedirect.com/science/article/pii/S0167739X13001581>
8. ITMC TU Dortmund. Official LiDO Website. <https://www.itmc.uni-dortmund.de/dienste/hochleistungsrechnen/lido.html>. Accessed 26 March 2015
9. Castro, M., Franceschini, E., Ngele, T.M., Mehaut, J.-F.: 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 2013, pp. 5:1–5:8. ACM, New York (2013). <http://doi.acm.org/10.1145/2535753.2535757>
10. Applegate, D., Bixby, R., Chvátal, V., Cook, W.: The Traveling Salesman Problem: A Computational Study: A Computational Study. Princeton Series in Applied Mathematics. Princeton University Press, Princeton (2011). <http://books.google.de/books?id=zflm94nNqPoC>
11. KALRAY Corporation. Official KALRAY MPPA Processor Website. <http://www.kalrayinc.com/kalray/products/#processors>. Accessed 31 March 2015
12. NVIDIA Corporation. Official NVIDIA SECO Development Kit Website. <https://developer.nvidia.com/seco-development-kit>. Accessed 31 March 2015
13. Rajovic, N., Rico, A., Vipond, J., Gelado, I., Puzovic, N., Ramirez, A.: Experiences with mobile processors for energy efficient HPC. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2013, pp. 464–468. EDA Consortium, San Jose (2013). <http://dl.acm.org/citation.cfm?id=2485288.2485400>

14. NVIDIA Corporation. Official NVIDIA Tegra 2 Website. <http://www.nvidia.com/object/tegra-superchip.html>. Accessed 27 March 2015
15. NVIDIA Corporation. Official NVIDIA Tegra 3 Website. <http://www.nvidia.com/object/tegra-3-processor.html>. Accessed 27 March 2015
16. Binkert, N., Beckmann, B., Black, G., Reinhardt, S.K., Saidi, A., Basu, A., Hestness, J., Hower, D.R., Krishna, T., Sardashti, S., Sen, R., Sewell, K., Shoaib, M., Vaish, N., Hill, M. D., Wood, D.A.: The gem5 simulator. *SIGARCH Comput. Archit. News* **39**(2), 1–7 (2011). <http://doi.acm.org/10.1145/2024716.2024718>
17. Bucy, J.S., Schindler, J., Schlosser, S.W., Ganger, G.R.: The DiskSim Simulation Environment Version 4.0 Reference Manual, May 2008
18. Rosenfeld, P., Cooper-Balis, E., Jacob, B.: DRAMSim2: a cycle accurate memory system simulator. *Comput. Architect. Lett.* **10**(1), 16–19 (2011)
19. Imperas Software Limited, OVP Guide to Using Processor Models, Imperas Buildings, North Weston, Thame, Oxfordshire, OX9 2HA, UK, January 2015, version 0.5, docs@imperas.com
20. Imperas Software Limited, OVPSim and Imperas CpuManager User Guide, Imperas Buildings, North Weston, Thame, Oxfordshire, OX9 2HA, UK, January 2015, version 2.3.7, docs@imperas.com
21. Altera Corporation. Cyclone V SoC Development Kit User Guide. <https://www.altera.com/content/dam/altera-www/global/enUS/pdfs/literature/ug/ugcvsocdevkit.pdf>. Accessed 07 May 2015
22. Imperas Software Limited. Description of Altera Cyclone V SoC. <http://www.ovpworld.org/library/wikka.php?wakka=AlteraCycloneVHPS>. Accessed 29 April 2015
23. Janapsatya, A., Ignjatovic, A., Parameswaran, S., Henkel, J.: Instruction trace compression for rapid instruction cache simulation. In: Proceedings of the Conference on Design, Automation and Test in Europe, DATE 2007, pp. 803–808. EDA Consortium, San Jose (2007). <http://dl.acm.org/citation.cfm?id=1266366.1266538>
24. Hardman, J.: Official NAS Parallel Benchmarks Website. <http://www.nas.nasa.gov/publications/npb.html>. Accessed 23 April 2015
25. Dawson, C., Aizinger, V.: A discontinuous Galerkin method for three-dimensional shallow water equations. *J. Sci. Comput.* **22**(1–3), 245–267 (2005)
26. Aizinger, V., Proft, J., Dawson, C., Pothina, D., Negjusse, S.: A three-dimensional discontinuous Galerkin model applied to the baroclinic simulation of Corpus Christi Bay. *Ocean Dyn.* **63**(1), 89–113 (2013). <https://www.math.fau.de/fileadmin/am1/users/aizinger/AizingerPDPN2013.pdf>
27. Cockburn, B., Shu, C.-W.: The local discontinuous Galerkin method for time-dependent convection-diffusion systems. *SIAM J. Numer. Anal.* **35**(6), 2440–2463 (1998). <http://dx.doi.org/10.1137/S0036142997316712>
28. Branner, B.: The mandelbrot set. *Proc. Symp. Appl. Math.* **39**, 75–105 (1989)
29. 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.: The NAS parallel benchmarks - summary and preliminary results. In: Proceedings of the 1991 ACM/IEEE Conference on Supercomputing 1991, pp. 158–165. IEEE (1991)
30. Genbrugge, D., Eeckhout, L.: Chip multiprocessor design space exploration through statistical simulation. *IEEE Trans. Comput.* **58**(12), 1668–1681 (2009)
31. Kerrison, S., Eder, K.: Energy modeling of software for a hardware multithreaded embedded microprocessor. *ACM Trans. Embed. Comput. Syst.* **14**(3), 56:1–56:25 (2015). <http://doi.acm.org/10.1145/2700104>

Proceedings of the 2015 Federated Conference on
Software Development and Object Technologies

Janech, J.; Kostolny, J.; Gratkowski, T. (Eds.)

2017, XV, 386 p. 258 illus., Softcover

ISBN: 978-3-319-46534-0