ABSTRACTS HETEROPAR

DATARACEONACCELERATOR - A MICRO-BENCHMARK SUITE FOR EVALUATING CORRECTNESS TOOLS TARGETING ACCELERATORS

Adrian Schmitz, Joachim Protze, Lechen Yu, Simon Schwitanski, and Matthias S. Müller

The advent of hardware accelerators over the past decade has significantly increased the complexity of modern parallel applications. For correctness, applications must synchronize the host with accelerators properly to avoid defects. Considering concurrency defects on accelerators are hard to detect and debug, researchers have proposed several correctness tools. However, existing correctness tools targeting accelerators are not comprehensively and objectively evaluated since there exist few available micro-benchmarks that can test the functionality of a correctness tool.

In this paper, we propose DataRaceOnAccelerator (DRACC), a micro-benchmark suite designed for evaluating the capabilities of correctness tools for accelerators. DRACC provides micro-benchmarks for common error patterns in CUDA, OpenMP, and OpenACC programs. These micro-benchmarks can be used to measure the precision and recall of a correctness tool. We categorize all micro-benchmarks into different groups based on their error patterns, and analyze the necessary runtime information to capture each error pattern. To demonstrate the effectiveness of DRACC, we utilized it to evaluate four existing correctness tools: ThreadSanitizer, Archer, GPUVerify, and CUDA-MEMCHECK. The evaluation results demonstrate that DRACC is capable of revealing the strengths and weaknesses of a correctness tool.

APPLICATION TOPOLOGY DEFINITION AND TASKS MAPPING FOR EFFICIENT USE OF HETEROGENEOUS RESOURCES

Kods Trabelsi, Laïc Cudennec, and Rihab Bennour

Nowadays, high-performance computing (HPC) not only faces challenges to reach computing performance, it also has to take in consideration the energy consumption. In this context, heterogeneous architectures are expected to tackle this challenge by proposing a mix of HPC and low-power nodes. There is a significant research effort to define methods for exploiting such computing platforms and find a trade-off between computing performance and energy consumption. To this purpose, the topology of the application and the mapping of tasks onto physical resources are of major importance. In this paper we propose an iterative approach based on the exploration of logical topologies and mappings.

These solutions are executed onto the heterogeneous platform and evaluated. Based on these results a Pareto front is built, allowing users to select the most relevant configurations of the application according to the current goals and constraints. Experiments have been conducted on a heterogeneous micro-server using a video processing application running on top of a software-distributed shared memory and deployed over a mix of Intel i7 and Arm Cortex A15 processors. Results show that some counterintuitive solutions found by the exploration approach perform better than classical configurations.

TOWARD HETEROGENEOUS MPI+MPI PROGRAMMING: COMPARISON OF OPENMP AND MPI SHARED MEMORY MODELS

Lukasz Szustak, Roman Wyrzykowski, Kamil Halbiniak, Pawel Bratek

This paper introduces our research on investigating the possibility of using heterogeneous all-MPI programming for the efficient parallelization of real-world scientific applications on clusters of multicore SMP/ccNUMA nodes. The investigation is based on verifying the efficiency of parallelizing a CFD application known as MPDATA, which contains a set of stencil kernels with heterogeneous patterns. As the first step of the research, we consider the level of SMP nodes, and compare the performance achieved by the MPI Shared Memory model of MPI-3 versus the OpenMP approach. In contrast to other works, this paper aims to evaluate these two programming models in conjunction with the parallelization methodology recently proposed [1] for performance portable programming across multicore SMP/ccNUMA platforms. We show that the shared memory extension of MPI delivers portable means for implementing all steps of this methodology efficiently, to take advantages of emerging multicore ccNUMA architectures.

MULTICORE PERFORMANCE PREDICTION – COMPARING THREE RECENT APPROACHES IN A CASE STUDY

Matthias Lüders, Oliver Jakob Arndt, and Holger Blume

Even though parallel programs, written in high-level languages, are portable across different architectures, their parallelism does not necessarily scale after migration. Predicting a multicore-application's performance on the target platform in an early development phase can prevent developers from unpromising optimizations and thus significantly reduce development time. However, the vast diversity and heterogeneity of system-design decisions of processor types: from HPC and desktop PCs to embedded MPSoCs complicate the modeling due to varying capabilities. Concurrency effects (caching, locks, or bandwidth bottlenecks) influence parallel runtime behavior as well. Complex performance prediction approaches emerged, which can be grouped into: virtual prototyping, analytical models, and statistical methods. In this work, we predict the performance of two algorithms from the field of advanced driver-assistance systems in a case study. With the following three methods, we provide a comparative overview of state-of-the-art predictions: GEMS (virtual prototype), IBM Exabounds (analytical model), and an in-house developed statistical method. We first describe the theoretical background, describe the experimental- and model-setup, and give a detailed evaluation of the prediction. In addition, we discuss the applicability of all three methods for predicting parallel and heterogeneous systems.

EXPLOITING HISTORICAL DATA: PRUNING AUTOTUNING SPACES AND ESTIMATING THE NUMBER OF TUNING STEPS

Jaroslav Oľha, Jana Hozzová, Jan Fousek, and Jiri Filipvic

Autotuning, the practice of automatic tuning of code to provide performance portability, has received increased attention in the research community, especially in high performance computing. Ensuring high performance on a variety of hardware usually means modimodifications to the code, often via different values of selected set of parameters, such as tiling size, loop unrolling factor or data layout. However, the search space of all possible combinations of these parameters can be enormous. Traditional search methods often fail to find a well-performing set of parameter values quickly.

We have found that certain properties of tuning spaces do not vary much, when hardware is changed. In this paper, we demonstrate that it is possible to use historical data to reliable predict the nimber of tuning steps necessary to find a well-performing configuration, and reduce the size of the tuning space. We evaluate our hypotheses on a number of GPU-accelerated benchmarks written in CUDA and OpenCL.

ADVANCING AUTOMATIC CODE GENERATION FOR AGENT-BASED SIMULATIONS ON HETEROGENEOUS HARDWARE

Jiajian Xiao, Philipp Andelfinger, Wentong Cai, Paul Richmond, Alois Knoll, and David Eckhoff

The performance of agent-based simulations has been shown to benefit immensely from execution on hardware accelerator devices such as graphics processing units (GPUs). Given the increasingly heterogeneous hardware platforms available to researchers, it is important to enable modellers to target multiple devices using a single model specification, and to avoid the need for in-depth knowledge of the hardware. Further, key modelling steps such as the definition of the simulation space and the specification of rules to resolve conflicts among agents should be supported in a simple and generic manner, while generating efficient code. To achieve these goals, we extend the OpenABL modelling language and code generation framework by three aspects: firstly, a new OpenCL backend enables the co-execution of arbitrary agent-based models on heterogeneous hardware. Secondly, the OpenABL language is extended to support graph-based simulation spaces. Thirdly, we specify a generic interface for specifying conflict resolution rules. In a performance comparison to the existing OpenABL backends, we show that depending on the simulation model, the opportunity for CPU-GPU co-execution enables a speedup of up to 2.0 over purely GPU-based simulation.

OPTIMIZATION OF DATA-PARALLEL APPLICATIONS ON HETEROGENEOUS HPC PLATFORMS FOR DYNAMIC ENERGY THROUGH WORKLOAD DISTRIBUTION

Hamidreza Khaleghzadeh, Muhammad Fahad, Ravi Reddy Manumachu, and Alexey Lastovetsky

Energy is one of the most important objectives for optimization on modern heterogeneous high performance computing (HPC) platforms. The tight integration of multicore CPUs with accelerators such as graphical processing units (GPUs) and Xeon Phi coprocessors in these platforms present several challenges to optimization of multithreaded data-parallel applications for dynamic energy.

In this work, we formulate the optimization problem of data-parallel applications on heterogeneous HPC platforms for dynamic energy throughworkload distribution. We propose a solution method to solve the problem. It consists of a data-partitioning algorithm that employs load imbalancing technique to determine the workload distribution minimizing the dynamic energy consumption of the parallel execution of an application. The inputs to the algorithm are discrete dynamic energy profiles of individual computing devices, which are constructed using a practical approach. The approach accurately models the energy consumption by a hybrid scientific data-parallel application executing on a heterogeneous platform containing different computing devices such as CPU, GPU, and Xeon Phi.

We experimentally analyse the proposed algorithm using two multithreaded data-parallel applications, matrix multiplication and 2D fast Fourier transform. The load-imbalanced solutions provided by the algorithm achieve significant dynamic energy reductions (on the average 130% and 44%) compared to the load balanced solutions for the two applications.

SEARCH-BASED SCHEDULING FOR PARALLEL TASKS ON HETEROGENOUS PLATFORMS

Robert Dietze and Gudula Rünger

Scheduling is a widely used method in parallel computing, which assigns tasks to several compute resources of the parallel environments. In this article, we consider parallel tasks as the basic entities to be scheduled onto a heterogeneous execution platform consisting of multicores of different architecture. A parallel task has an internal potential parallelism which allows a parallel execution for example on multicore processors of different type. The assignment of tasks to different multicores of a heterogeneous execution platform may lead to different execution times for the same parallel tasks. Thus, the scheduling of parallel tasks onto a heterogeneous platform is more complex and provides more choices for the assignment and for finding the most efficient schedule. Search-based methods seem to be a promising approach to solve such complex scheduling problems. In this article, we propose a new task scheduling method HP* to solve the problem of scheduling parallel tasks onto heterogeneous platforms. Furthermore, we propose a cost function that reduces the search space of the algorithm. In performance measurements, the scheduling results of HP* are compared to several existing scheduling methods. Performance results with different benchmark tasks are shown to demonstrate the improvements achieved by HP*.

ADAPTATION OF WORKFLOW APPLICATION SCHEDULING ALGORITHM TO SERVERLESS INFRASTRUCTURE.

Maciej Pawlik, Pawel Banach, and Maciej Malawski

Function-as-a-Service is a novel type of cloud service used for creating distributed applications and utilizing computing resources. Application developer supplies source code of cloud functions, which are small applications or application components, while the service provider is responsible for provisioning the infrastructure, scaling and exposing a REST style API. This environment seems to be adequate for running scientific workflows, which in recent years, have become an established paradigm for implementing and preserving complex scientific processes.

In this paper, we present work done on adaptation of a scheduling algorithm to FaaS infrastructure. The result of this work is a static heuristic capable of planning workflow execution based on defined function pricing, deadline and budget. The SDBCS algorithm is designed to determine the quality of assignment of particular task to specific function configuration. Each task is analyzed for execution time and cost characteristics, while keeping track of parameters of complete workflow execution. The algorithm is validated through means of experiment with a set of synthetic workflows and a real life infrastructure case study performed on AWS Lambda. The results confirm the utility of the algorithm and lead us to propose areas of further study, which include more detailed analysis of infrastructure features affecting scheduling.

CCAMP: OPENMP AND OPENACC INTEROPERABLE FRAMEWORK

Jacob Lambert, Seyong Lee, Allen Malony, and Jeffrey S. Vetter

Heterogeneous systems have become a staple of the HPC environment. Several directive-based solutions, such as OpenMP and OpenACC, have been developed to alleviate the challenges of programming heterogeneous systems, and these standards strive to provide a single portable programming solution across heterogeneous environments. However, in many ways this goal has yet to be realized due to device-specific implementations and different levels of language support across compilers. In this framework we aim to analyze and address the different levels of optimization and compatibility between OpenACC and OpenMP programs and device compilers. We introduce the CCAMP framework, built on the OpenARC compiler, which implements language translation between OpenACC and OpenMP, with the goal of exploiting the maturity of different device-specific compilers to maximize performance for a given architecture. We show that CCAMP allows us to generate code for a specific device-compiler combination given a device-agnostic OpenMP or OpenACC program, allowing compilation and execution of programs with specific directives on otherwise incompatible devices. CCAMP also provides a starting point for a more advanced interoperable framework that can effectively provide directive translation and device, compiler, and application specific optimizations.