Criticality-Aware Dynamic Task Scheduling for Heterogeneous Systems

Kallia Chronaki[*], Alejandro Rico[*], Rosa M. Badia[*†] and Eduard Ayguadé[*‡]
[*]Barcelona Supercomputing Center, Barcelona, Spain
[†]Research Institute (IIIA) - Spanish National Research Council (CSIC), Barcelona, Spain
[‡]Universitat Politècnica de Catalunya - BarcelonaTech, Barcelona, Spain
kallia.chronaki@bsc.es, alejandro.rico@bsc.es, rosa.m.badia@bsc.es, eduard.ayguade@bsc.es

Introduction

In the search of performance and energy efficiency, heterogeneous multi-core architectures are an appealing option for next-generation high-performance computing.

These architectures combine different types of processing cores designed at different performance and power optimization points, thus exposing a performance-power trade-off.

Current and future parallel programming models need to be portable and efficient when moving to such systems. Load balancing and scheduling are two of the main challenges in utilizing such heterogeneous platforms. The use of task-based programming models with dynamic scheduling is a way to tackle these challenges. Some of these programming models allow the specification of inter-task dependencies that enable automatic scheduling and synchronization by the runtime system.

OmpSs is a powerful task-based programming model with dependency tracking and dynamic scheduling. In this talk we will describe the OmpSs approach on scheduling dependent tasks onto the asymmetric cores of a heterogeneous system. The proposed dynamic scheduling policy uses information discoverable at runtime and reduces total execution time. It first prioritizes the newly-created tasks at runtime according to the shape of the task dependency graph; it then detects the longest path of the dynamic task dependency graph, and finally it assigns critical tasks to fast cores and non-critical tasks to slow cores.

Previous works on scheduling for heterogeneous systems by using task priorities require the prior knowledge of various parameters of the workload; for example the task execution time, which cannot be discoverable without profiling or the task dependency graph of the workload.

The experimental evaluation proves that our implementation speeds up the execution of four scientific kernels on the ARM big.LITTLE heterogeneous chip. Our proposal outperforms a dynamic implementation of the state-of-the-art Heterogeneous Earliest Finish Time scheduler by up to 1.15x, and the default breadth-first
OmpSs scheduler by up to 1.3x in the 8-core heterogeneous ARM big.LITTLE and up to 2.7x in a simulated 128-core chip.

Previous criticality-aware schedulers for heterogeneous systems are static and based on the knowledge of profiling information. Our proposal performs dynamic scheduling using information discoverable at runtime, is implementable and works without the need of an oracle or profiling.

In our evaluation using four dependency-intensive applications, our proposal outperforms a dynamic implementation of Heterogeneous Earliest Finish Time by up to 1.15x, and the default breadth-first OmpSs scheduler by up to 1.3x in a real 8-core heterogeneous platform and up to 2.7x in a simulated 128-core chip.

Related Work

Several previous works propose scheduling heuristics that focus on the critical path in a task dependency graph (TDG) to reduce total execution time [1, 2, 3, 4]. To identify the tasks in the critical path, most of these works use the concept of *upward rank* and *downward rank*. The upward rank of a task is the maximum sum of computation and communication cost of the tasks in the dependency chains from that task to an exit node in the graph. The downward rank of a task is the maximum sum of computation and communication cost of the tasks in the dependency chain from an entry node in the graph up to that task. Each task has an upward rank and downward rank for each processor type in the heterogeneous system, as the computation and communication costs differ across processor types.

The Heterogeneous Earliest Finish Time (HEFT) algorithm [4] maintains a list of tasks sorted in decreasing order of their upward rank. At each schedule step, HEFT assigns the task with the highest upward rank to the processor that finishes the execution of the task at the earliest possible time. Another work is the Longest Dynamic Critical Path (LDCP) algorithm [1]. LDCP also statically schedules first the task with the highest upward rank on every schedule step. The difference between LDCP and HEFT is that LDCP updates the computation and communication costs on multiple processors of the scheduled task by the computation and communication cost in the processor to which it was assigned.

The Critical-Path-on-a-Processor (CPOP) algorithm [4] also maintains a list of tasks sorted in decreasing order as in HEFT, but in this case it is ordered according to the addition of their *upward rank* and *downward rank*. The tasks with the highest *upward rank* + *downward rank* belong to the critical path. On each step, these tasks are statically assigned to the processor that minimizes the critical-path execution time.

The main weaknesses of these works are that (a) they assume prior knowledge of the computation and communication costs of each individual task on each processor type, (b) they operate statically on the whole dependency graph, so they do not apply to dynamically scheduled applications in which only a partial representation of the dependency graph is available at a given point in time, and (c) most of them use randomly-generated synthetic dependency graphs that are not necessarily representative of the dependencies in real workloads.

References

[1] M. Daoud and N. Kharma. Efficient Compile-Time Task Scheduling for Heterogemeous Distributed Computing Systems. In Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on, volume 1, pages 9, 2006.

[2] M. Hakem and F. Butelle. Dynamic Critical Path Scheduling Parallel Programs onto Multiprocessors. In In the Proceedings of the 19th IEEE International Parallel and Distributed Processing Symposium, IPDPS'05, pages 203b-203b, April 2005.

[3] C.-H. Liu, C.-F. Li, K.-C. Lai, and C.-C. Wu. A dynamic Critical Path Duplication Task Scheduling Algorithm for Distributed Heterogeneous Computing Systems. In Parallel and Distributed Systems, 2006. ICPADS 2006. 12th International Conference on, volume 1, pages 8, 2006.

[4] H. Topcuoglu, S. Hariri, and M.-Y. Wu. Performance-Effective and Low-Complexity Task Scheduling for Heterogeneous Computing. IEEE Transactions on Parallel and Distributed Systems, 13(3):260-274, Mar 2002.