Synthesizing Parallel Programming Models for Asymmetric Multi-core Systems

Dimitrios S. Nikolopoulos and Kirk W. Cameron
Department of Computer Science
Virginia Tech
dsn@cs.vt.edu, cameron@cs.vt.edu

Asymmetric multi-core processors integrating conventional with customized accelerator cores, have exhibited the potential to provide unprecedented performance for data-intensive applications. Although significant effort has been invested in parallel programming models that exploit a single form of parallelism, programming models that synthesize polymorphic parallelism and runtime systems that exploit multiple dimensions of concurrency and heterogeneity in parallel execution are rare. As a consequence, software developers lack both the programming abstractions and the methodologies for synthesizing programming models for polymorphic parallelism. This deficit may render software unable to exploit asymmetric multi-core processors, since polymorphic parallelism is essential to balance the supply of computation from conventional cores with the demand from accelerators [1]. Our research on runtime performance modeling and scheduling of dynamic polymorphic parallelism on the Cell Broadband Engine [1,2,3], derives a methodology for synthesizing polymorphic programming models for explicitly parallel programs *on-the-fly*. The proposed methodology off-loads the tasks of mapping algorithmic parallelism to the architecture and adapting parallel execution to the available resources from the programmer to the runtime environment. In addition to mapping and adaptation, the runtime environment performs a synthesis of execution, communication and synchronization schemes to balance parallel execution across the cores and the interconnection network of the processor. The runtime system leverages two complementary system software modules. The first module features an event-driven scheduler (EDTLP), which unifies the scheduling of different forms of algorithmic parallelism (e.g. task, data, pipeline), and systemic parallelism (e.g. concurrent DMA and communication requests), across the PPE and SPEs. The second module comprises a performance modeling and prediction framework, coined MMGP, which derives performance predictions on all feasible PPE/SPE core configurations and thereby drives the event-driven scheduler to near-optimal schedules. A snapshot of our results, illustrated in Figure 1, indicates the accuracy of MMGP in identifying the optimal forms and degrees of concurrency in realistic parallel applications on the Cell BE. Figure 1 illustrates the performance of two algorithms for parallel phylogenetic tree construction (RAxML and PBPI), along with the optimal operating points of these codes on an IBM BladeCenter QS20.

REFERENCES

[1]. Filip Blagojevic, Dimitrios S. Nikolopoulos, Alexandros Stamatakis, and Christos D. Antonopoulos. *Dynamic Multi-grain Parallelization on the Cell Broadband Engine*. Proc. of the 11th ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming, pp. 90-100, San Jose, CA, March 2007.

[2]. Filip Blagojevic, Alexandros Stamatakis, Christos D. Antonopoulos, and Dimitrios S. Nikolopoulos. *RAxML-CELL: Parallel Phylogenetic Tree Construction on the Cell Broadband Engine*. Proc. of the 21st International Parallel and Distributed Processing Symposium, Long Beach, CA, March 2007.

[3]. Filip Blagojevic, Xizhou Feng, Kirk Cameron, and Dimitrios S. Nikolopoulos. *Modeling Multi-Grain Parallelism on Heterogeneous Multi-core Processors: A Case Study of the Cell BE*. Proc. of the 2008 International Conference on High-Performance Embedded Architectures and Compilers, Göteborg, Sweden, February 2008.

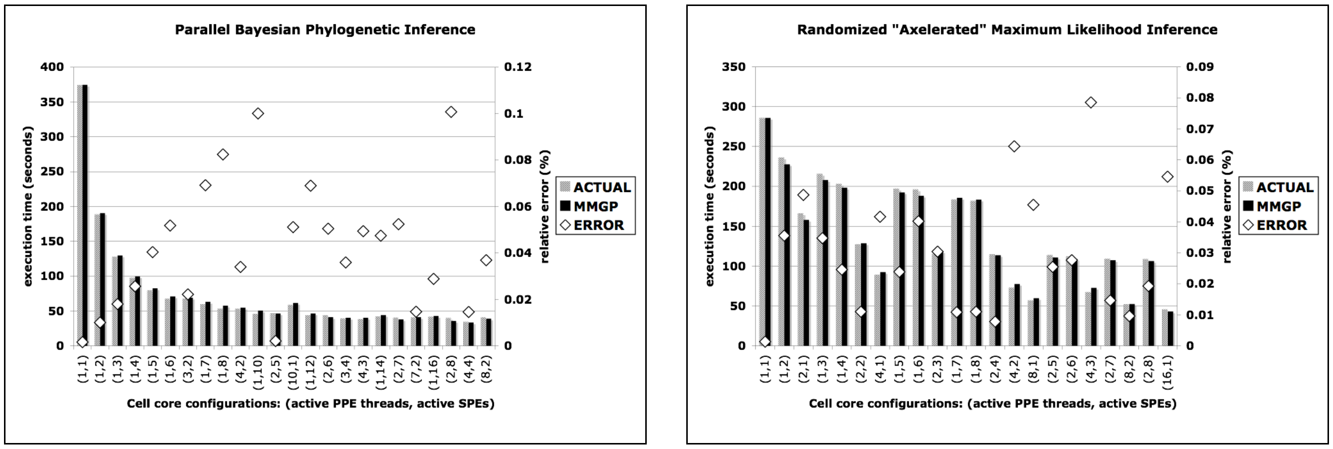

Figure 1: Actual and predicted execution times of PBPI (left) and RAxML (right) with all feasible core configurations on an IBM BladeCenter QS20, with two Cell BEs. Each configuration is represented as a pair (x, y), where x is the number of PPE threads and y the number of SPEs activated to execute the programs in the given configuration. MMGP accurately locates the optimal operating points of the benchmarks, despite the complexity of the layered parallel architecture of the blade. The prediction error of MMGP never exceeds 10% and typically lies between 1% and 5%.