Analyzing the Impact of Parallel Programming Models in Forthcoming CMP Architectures

Iván Pérez*, Emilio Castillo†‡, Ramón Beivide*, José Luis Bosque*,
Enrique Vallejo*, Miquel Moretó†‡, Mateo Valero†‡
*Universidad de Cantabria †Barcelona Supercomputing Center ‡Universidad Politécnica de Catalunya

Nowadays, the design of the memory hierarchy is one of the most relevant subjects of research in Computer Architecture because of the quickly grow of concurrent compute units in the same chip. The distribution of shared memory levels, the design of scalable cache coherence protocols or the on-chip interconnection network (NoC) are some of the most studied topics of this area.

In the same way, at the software layer, parallel programming models based on tasks are gathering strength to face up to traditional ones based on threads. Easing programming, fine grain synchronization and tracking data flow dependencies are the fundamental reasons for this change.

There are previous works that study these two programming paradigms such as [2] which compares the scalability of the PARSEC benchmark suite for both models in real machines. None of them studies the penalties that the memory system brings in the performance of the programing model. Therefore, the goal of this study lies in evaluate the impact of the memory system on the programming model performance to take architectural conclusions, in particular focusing on the NoC. For this purpose, we use the PARSEC benchmarks in their pthreads/OpenMP and OmpSs versions and a simulation environment made up by Gem5 [1], Ruby and Garnet. This environment allow us to measure performance metrics such as execution times, memory bandwidths, idle times, network latencies or the amount of injected traffic, for example, defining the most recent or alternative architectures with high degree of detail.

To assess the sensitivity of the programming model to the NoC performance, we compare two network topologies: a complete graph, that gives the most performing results of latency and throughput but far from being implementable in a chip with tens of cores; and a mesh, which is the most common topology in up-to-date many-core processors. Figure 1 shows preliminary results of normalized execution times of some PARSEC benchmarks. These simulations use: 64 x86 out-of-order CPUs at 2 GHz; a first level of private caches with 32KB for instructions and 64KB for data; a second cache level shared and distribute among cores with a size of 32MB; 16 memory controllers; a MESI coherence protocol; and the two topologies mentioned before, a complete graph and a mesh, both operating at 1GHz. The results show differences between both programming models, with OmpSs showing highest performance. Comparing both topologies, OmpSs presents lower sensitivities than pthreads for ferret (28.8% and 41.5% respectively) and bodytrack (43.3% and 48.3 respectively) and slighly greater for blacksholes (10.6% and 8.8% respectively). This could mean that OmpSs is more tolerant to network performance.

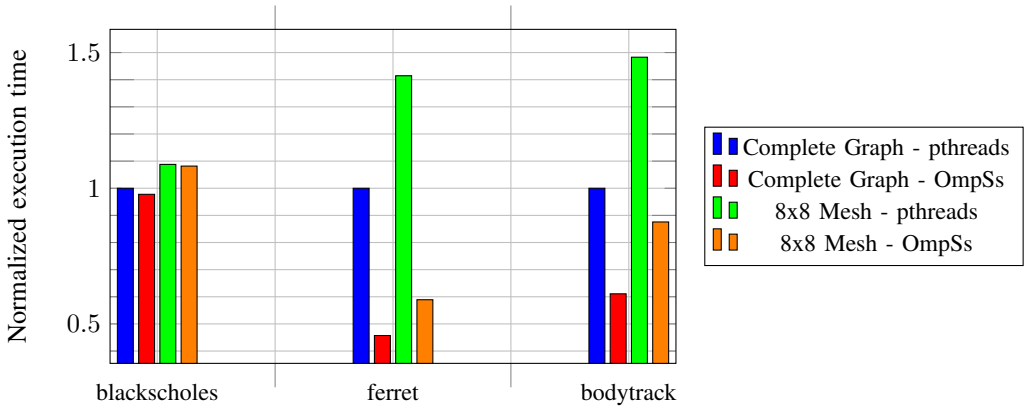

Figure 1. Normalized execution times to Complete Graph and pthreads.

REFERENCES

[1] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K. Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R. Hower, Tushar Krishna, Somayeh Sardashti, Rathijit Sen, Korey Sewell, Muhammad Shoaib, Nilay Vaish, Mark D. Hill, and David A. Wood. The gem5 simulator. *SIGARCH Comput. Archit. News*, 39(2):1–7, August 2011.

[2] Dimitrios Chasapis, Marc Casas, Miquel Moretó, Raul Vidal, Eduard Ayguadé, Jesús Labarta, and Mateo Valero. Parsecss: Evaluating the impact of task parallelism in the parsec benchmark suite. *ACM Trans. Archit. Code Optim.*, 12(4):41:1–41:22, December 2015.