Communication Aware Actor Allocation of Stream Programs

Vitaly Nikolenko vitaly.nikolenko@sydney.edu.au,
S.M. Farhad smfarhad@it.usyd.edu.au,
Bernhard Scholz bernhard.scholz@sydney.edu.au

October 16, 2012

Abstract

Sequential programming languages are not well-suited for multi-cores, providing insufficient parallel hardware abstraction, which greatly hampers performance and portability of software on multi/many-core architectures. It is left to the programmer to identify parallelism in programs, which is tedious and error-prone. This problem is paraphrased by the term "*The Parallel Programming Gap*", expressing the steady increase in number of cores in processors over time, and the slow adoption of new programming models to harvest effectively the computational power of parallel cores.

Stream programming is a parallel paradigm that allows to express data and task parallelism in a natural and intuitive way. It is used in applications that deal with regular sequences of data such as multimedia, graphics, signal processing and networking applications. In the stream programming model, computations are expressed by a collection of actors that are connected by data channels. A major challenge with stream programs is to load-balance actors among available processor cores. Even without considering communication costs of data channels, the problem is known to be NP-hard.

In this talk, we present an approximation algorithm for placing stream programs on processor cores that takes communication costs of data channels into account. Unlike load-balancing heuristics, our approximation algorithm runs in polynomial time with solutions within a factor of $\log_2 n$ of the optimal solution, where n is the number of actors in a stream program. We conduct experiments across a range of StreamIt benchmarks and demonstrate that the observed approximation ratio of an instance is less than or equal to 2.