Studying Asynchronous Shared Memory Computations

Simo Juvaste
University of Joensuu, Computer Science
juvaste@cs.joensuu.fi

Abstract

We present an experimental framework, F-PRAM model, for modeling, studying, and teaching the impacts of different properties of parallel computers. The performance model is a parameterized model similar to BSP [4] and LogP [1] with some additional parameters for more refined analysis when needed. To use the model, we present an emulation and experimentation system to study the impact of the parameters and communication network design decisions on application algorithm performance.

Even if the parameterized models suite well for algorithm development and analysis, they cannot, however, represent accurately a complex real parallel computer, for example, hierarchical structure of computing clusters. Consequently, we use simulated network as a basis for comparison. With the simulated network approach, we can simulate any network topology, link/node speed, and routing algorithm. Using the simulated system, we can execute either benchmarks to measure the values of F-PRAM parameters (such as latency and bandwidth), or application algorithms to analyze their performance.

The emulator takes as input the application algorithm (high level language), input data, machine parameters (such as number of processors and memory modules), other details (such as memory module latency and bandwidth), interconnection network topology and properties, routing algorithm (in C), length of buffers, and memory allocation (hash) scheme. During the execution, the program can do any output, but usually we are interested in the number of clock cycles needed for the execution. To use system efficiently, we have an automated measurement system that takes sets of configuration parameters and executes the simulation for every combination of the parameters, records the results, and visualizes the results with one or more graphs. For example, we select an application algorithm, input size, a number of processors, a set of different shapes of 3D mesh network, and mesh usage sparseness to see which is optimal for our algorithm. Or, we can select a set of hash algorithms and variations of application algorithm to see if some access patterns perform better than others.

Our programming model is simplified Modula-2 with additional par-do to divide the execution threads for parallel execution. Shared memory data (variables) must be asynchronously pre-fetched to local variables before usage. Similarly, shared memory is updated with asynchronous writes. Programmer has the responsibility of memory consistency. The programmer can use the values of model parameters in program to make program to adapt to machine properties. In case of simulated network, the routing algorithm can collect statistics of the delays, and adjust the parameters accordingly on runtime.

Our current set of example algorithms includes generic benchmarks (for latency and bandwidth), maximum finding, odd-even mergesort, matrix multiplication, matrix inversion, and image smoothing. Detailed descriptions of algorithms can be found in [2]. Our current set of network topologies include butterfly (deflection routing), hypercube, and 3D mesh/torus. We are planning to implement hierarchical memory structures, e.g., mesh of SMP nodes. Current version of the system is available at [3].

References