CS 388P Writeup

Jesus Gonzalez, Tomas McCandless

October 9, 2012

1 Summary of Contributions

2 Summary of Key Papers

2.1 Can a Shared-Memory Model Serve as a Bridging Model for Parallel Computation?

A bridging model for parallel computation is a model that "spans the range from algorithm design to architecture to hardware". [1] Such a model should provide an abstraction that is easy for algorithm designers to use, and should be realizable by hardware and system sofware at varying price/performance points.

Models of parallel computation should attempt to satisfy two goals that are in conflict: a model should be sufficiently abstract to allow algorithm designers to write programs that are simple and portable across architectures, and a model should also expose some low-level architectural details to allow for optimization. The PRAM model is both widely used and simple, yet it has been criticized for being too high-level and thus failing to accurately model parallel machines. Specifically, the PRAM does not model realities of current parallel machines, such as bandwidth limitations. Similarly, network-based models such as the hypercube have been criticized for being too low-level, failing to be widely reflective of the current technological state of parallel machines.

These shortcomings of existing models have led to the introduction of more intermediate models such as the BSP or LogP models.

The new model this paper introduces, the QSM, is meant to be an attractive candidate for a bridging model. The QSM consists of processors with private individual memory in addition to a global shared memory.

One goal for a higher-level shared-memory model should be allowing efficient emulation on lower-level models which may be more realistic.

It is mistaken to view "shared-memory model" as being synonymous with PRAM. The shared-memory abstraction refers to interprocessor communication. That is, a processor may have additional memory as part of its local state. While the PRAM does not model bandwidth limitations, a shared-memory model may be asynchronous, or have explicit charges for communication.

One of the primary motivations for positing a shared-memory model as a bridging model is the portability across memory architectures that accompanies such models. This is because the layout of memory is hidden by the model.

The QSM model consists of a number of processors, each with private memory, that communicate via reading and writing to shared memory locations. Each synchronized phase executed by the processors consists of an arbitrary interleaving of the following operations:

- shared-memory reads: processor P_i copies values from shared memory locations into private memory.
- shared-memory writes: processor P_i writes to w_i shared memory locations.
- local computation: processor P_i performs c_i RAM computations which

involve only private state and private memory.

The QSM has the following features which make it more attractive as a bridging model:

- shared-memory abstraction
- bulk-synchrony
- few parameters
- queue contention metric
- work-preserving emulation on BSP
- work-preserving emulation of BSP

The authors compare the QSM and the BSP in terms of their effectiveness as bridging models for parallel computation. The BSP consists of p processor/memory components that communicate by sending point-to-point messages. The network supporting this communication is by two parameters: g, which models bandwidth; and L, which models latency. A computation on the BSP is a series of supersteps, each of which is separated by bulk synchronization.

The authors give a work preserving emulation (an emulation is work-preserving if both models perform the same amount of work to within a constant factor) of the QSM on the BSP. However, since the QSM has fewer parameters than the BSP and does not deal directly with memory partitioning, we would expect it to be easier to design algorithms on the QSM. Designs can then be mapped onto the BSP with modest slowdown.

The (d, $\mathbf{x}$)-BSP is more detailed in its modeling of memory contention than the BSP, and is characterized by 5 parameters:

- p - number of processors
- g - bandwidth
- L - latency
- d - delay, models gap at the memory banks
- $\mathbf{x}$ - expansion, ratio of memory banks to processors

Developing a suitable model for parallel computation at a proper level of abstraction has remained a challenging problem for computer scientists. BSP and LogP have recently gained popularity as candidate bridging models. The QSM is an attractive candidate for a bridging model, in particular because it provides a simple shared memory abstraction and has a small number of parameters (p, the number of processors, and g, the bandwidth gap). Additionally, the QSM can be efficiently emulated on both the BSP and (d, $\mathbf{x}$)-BSP.

2.2 Emulations between QSM, BSP and LogP: A framework for general-purpose parallel algorithm design. [2]

The authors present work-preserving emulations with just a small slowdown between the geneal-purpose models: LogP, BSP and QSM. The relevance of this lies on the simplicity of the QSM model, which makes it easy to program

for and, yet, through emulation on more realistic models like BSP and LogP run an algorithm with just a logarithmic slowdown.

The authors present 3 algorithms for basic problems (prefix sums, sample sort and list ranking). All of them are optimal and simulations show that the predicted performance for each of them is accurate.

PRAM is a very simple model for parallel computer that abstracts too much parameters from real machines, which turns into apparently eficcient models that turn to run slowly on real machines. Therefore, exists the need of more realistic models that reflect with more fidelity a machine and yet is not to complicated so that implementation is kept simple.

2.2.1 Brief description of the models

Al the next models have work measured as the time-processors product.

BSP BSP (Bulk-Synchronous Parallel) takes 3 parameters:

- p processor/memory components: Components that communicate via point-to-point messages.
- Bandwidth g.
- Latency L.

A computation in this model consists of a sequence of supersteps, each separated by a bulk synchronization. In each superstep, a component can perform local computations, send and receive a set of messages. The cost T superstep is calculated as $T = max(w, gh, L)]$ and the overal time that takes an algorithm to run is the sum of T of every superstep.

LogP LogP (Latency, overhead, gap, Processors) takes 4 parameters and a derived value:

- Processors p: Number of processors.
- Latency l: Time taken by network to transmit a message from one processor to another is at most l.
- Gap g: Least units of time taken between sending or reception of a message by a processor.
- Overhead o: Time taken by a processor to send or receive a message.
- Capacity constraint: Maximum number of messagings heading to the same processor can't be larger than $\lceil l/g \rceil$

Once a processor has reached its capacity, another processor trying to send a message will stall. The time taken by a LogP algorithm is the longest time taken by any processor to end its operation.

QSM QSM (Queuing Shared Memory) consists of p processors that communicate through shared memory, but run computations on private memory. An algorithm consists of the analogous for a superstep on BSP, a synchronized phase. Concurrent reads and concurrent writes are permited on a single synchronized phase over the same shared-memory location, but not both. Metrics:

- Processors p.
- Maximum contention k: of a synchronized phase Largest number of processors reading or writing a location.

The time cost of a phase is $max(m_{op}, gm_{rw}, k)$, where m_{op} = maximum operations performed by any processor and m_{rw} =maximum read/write operations performed by any processor. The time taken by an algorithm is the sum of the cost of all of its phases. s-QSM is a variation of QSM where the time cost of a phase is $max(m_{op}, gm_{rw}, gk)$

3 Open Problems

References

[1] P. Gibbons, Y. Matias, V. Ramachandran. "Can a shared memory model serve as a bridging model for parallel computation?" Theory of Computing Systems Special Issue on papers from SPAA'97, vol. 32, no. 3, 1999, pp. 327-359

[2] V. Ramachandran, B.Grayson, M. Dahlin.(2003) "Emulations between QSM, BSP and LogP: A framework for general-purpose parallel algorithm design." Journal of Parallel and Distributed Computing, vol. 63, 2003, pp. 1175-1192.