Emulations Between QSM, BSP and LogP: A Framework for General-Purpose Parallel Algorithm Design

Vijaya Ramachandran, Brian Grayson, and Michael Dahlin
The University of Texas at Austin
{vlr@cs, bgrayson@ece, dahlin@cs}.utexas.edu URL: www.cs.utexas.edu/users/vlr/pub.html

1 Introduction

BSP [8], LogP [2], and QSM [3] are general-purpose parallel models that seek to achieve the right balance of simplicity, accuracy and broad applicability in the modeling of parallel machines. We show that these models are essentially interchangeable by giving *work-preserving* emulations with small slowdown between these models. The one mis-match we have is between *stalling* and *nonstalling* LogP, for which we expose an error in an earlier paper [1] that claimed that stalling LogP and BSP are essentially equivalent models.

The QSM is a shared-memory model (in contrast to BSP and LogP), and it has fewer parameters. Thus we argue that it is the model of choice for algorithm design. We formulate a suitable cost metric for QSM algorithms, and we describe simple efficient QSM algorithms for prefix sums, sample sort and list ranking. Experimental evaluation of these algorithms indicates that the QSM cost metric is a good predictor of performance for problem sizes that arise in practice.

2 Models

The *QSM (Queuing Shared Memory)* model [3] consists of a number of identical processors that communicate by reading and writing shared memory. Processors execute a sequence of synchronized phases, each consisting of an arbitrary interleaving of shared memory reads, shared memory writes, and local computation, subject to the restriction that (i) the values returned by shared-memory reads issued in a phase cannot be used in the same phase and (ii) the same shared-memory location cannot be both read and written in the same phase. A phase takes time $\max(m_{op}, g \cdot m_{rw}, \kappa)$, where m_{op} is the maximum number of local operations, m_{rw} is the maximum number of shared-memory read-writes issued by any processor, and κ is the maximum number of requests to any single memory location. Here g is the *gap parameter*, which captures the delay incurred by limited bandwidth to shared-memory. The *s-QSM (Symmetric QSM)* is the same as the QSM, except that a phase takes time $\max(m_{op}, g \cdot m_{rw}, g \cdot \kappa)$.

The BSP (*Bulk Synchronous Parallel*) [8] and LogP [2] each model a parallel machine as a collection of processor-memory units with no global shared memory. The processors are interconnected by a network whose performance is characterized by a gap parameter g and a latency parameter l (in LogP) or synchronization parameter L (in BSP). The LogP model also models the per-message overhead o for sending and receiving messages, and it limits network congestion by requiring that no more than l/g messages be in transit to a given destination processor in any interval of length l.

3 Emulation Results

The results on work-preserving emulations between models are tabulated in Table 1 with new results printed within a box. We focus on *work-preserving* emulations. An emulation is work-preserving if the processor-time bound on the emulating machine is the same as that on the machine being emulated, to within a constant factor. The ratio of the running time on the emulating machine to the running time on the emulated machine is the *slowdown* of the emulation. Typically, the emulating machine has a smaller number of processors and takes proportionately longer to execute. For instance, the entry in Table 1 for the emulation of s-QSM on BSP states that there is a randomized work-preserving emulation of s-QSM on BSP with a slowdown of $O(L/g + \log p)$. This means that, given a p-processor s-QSM algorithm that runs in time t (and hence with work $w = p \cdot t$), the emulation algorithm will map the p-processor s-QSM algorithm on to a p'-processor BSP, for any $p' \leq p/((L/g) + \log p)$, to run on the BSP in time $t' = O(t \cdot (p/p'))$ w.h.p. in p.

We mention three aspects of these emulation results, which are described in [7]. First, we have developed new, work-preserving emulations of QSM, s-QSM and BSP on LogP; previously known emulations [1] required sorting and increased both time and work by a logarithmic factor. Second, we have provided new analysis of the known emulation of LogP on BSP [1]; we have provided a counter-example to the claim that this emulation holds for the stalling LogP model, and

Slowdown of Work-Preserving Emulations (sublogarithmic factors have been rounded up for ease of display)

Emulated Model (p processors)	Emulating Model			
	BSP	LogP (stalling)	s-QSM	QSM
BSP		$\boxed{O(\log^4 p \ + \ (L/g)\log^2 p)}$	$O(\lceil \frac{g \log p}{L} \rceil)$	$O(\lceil \frac{g \log p}{L} \rceil)$
LogP (nonstalling)	$\boxed{O(L/l) \ \ (det.)^{\dagger}}$	1 (det.)	$\boxed{O(\lceil \frac{g \log p}{l} \rceil)}$	$\boxed{O(\lceil \frac{g \log p}{l} \rceil)}$
s-QSM	$O((L/g) + \log p)$	$\boxed{O(\log^4 p \ + \ (l/g)\log^2 p)}$		1 (det.)
QSM	$O((L/g) + g \log p)$	$\boxed{O(\log^4 p + (l/g)\log^2 p + g \cdot \log p)}$	$O(g)$ (det.)	

Table 1: *All results are randomized and hold w.h.p. except those marked as 'det.', which are deterministic emulations. The boxed results (those in which the LogP model is either the emulated or the emulating model) are the new results reported in this paper (and described in detail in [7]). The remaining results are in [3, 6].*

†This result is given in [1] but is stated erroneously that it holds for stalling LogP programs. We provide a counterexample in [7].

we observe that the original non-work-preserving emulation may be trivially extended to be work-preserving. Third, we observe that known emulations of message passing on shared memory require sorting and multiple-compaction, making emulations of BSP or LogP algorithms on any shared-memory interface complex. On the other hand QSM and s-QSM have simple emulations on message-passing interfaces as modeled by BSP and LogP (and also, of course, on shared-memory).

Since both message-passing and shared-memory are widely-used in practice, we suggest that a high-level general-purpose model should be one that maps on to both in a simple way. QSM and s-QSM give us this feature. Additionally, they have a smaller number of parameters, and do not have to keep track of the layout of data across shared memory. Hence they are ideal models for general-purpose parallel algorithm design.

4 QSM cost metric and basic QSM algorithms

Our cost metric for a QSM algorithm asks to (1) minimize work, (2) minimize the number of phases, and (3) maximize parallelism, subject to the above requirements.

The motivation for the second metric (which is the new one) is the following. One major simplification made by the QSM models is that it does not incorporate an explicit charge for latency or the synchronization cost at the end of each phase. The total time spent on synchronizations is proportional to the number of phases in the QSM algorithm. Hence minimizing the number of phases in an s-QSM algorithm minimizes the hidden overhead due to synchronization. In particular it is desirable to obtain an algorithm for which the number of phases is independent of the input size n as n becomes large. All of the algorithms we present have this feature.

We are interested in the design of algorithms to be used in practice. We expect problems sizes to be fairly large in practice, hence maximizing parallelism is a secondary goal in our cost metric.

In [7, 4] we describe the algorithms that lead to the results stated below, together with implementation results and comparison of implemented performance to QSM and BSP predictions. The upper bounds in the following theorems hold for both QSM and s-QSM, and optimality (for s-QSM) is denoted using a Θ; the matching lower bounds are in [3, 5]. Here n is the input size and P is the number of processors.

Theorem 4.1. *Algorithm Prefix Sums runs with $\Theta(gn)$ work, $\Theta(\frac{\log n}{\log(n/P)})$ phases and $\Theta(g \log n)$ time (thus $P \leq n/\log n$).*

Theorem 4.2. *Algorithm Sample Sort runs with $\Theta(gn + n \log n)$ work, and $\Theta(1)$ phases whp when $P \leq \sqrt{n/\log n}$.*

Theorem 4.3. *Algorithm List Rank runs with $\Theta(gn)$ work and $O(\log P)$ phases whp when $P = O(n^r)$, $r < 1$.*

References

[1] G. Bilardi, K. T. Herley, A. Pietracaprina, G. Pucci, P. Spirakis. *BSP vs LogP*, ACM SPAA, 25–32, 1996.

[2] D. Culler, R. Karp, et al. *LogP: Towards a realistic model of parallel computation*, PPoPP, 1–12, 1993.

[3] P. B. Gibbons, Y. Matias, and V. Ramachandran. *Can a shared-memory model serve as a bridging model for parallel computation?* ACM SPAA, 72–83, 1997.

[4] B. Grayson, M. Dahlin, V. Ramachandran, *Experimental evaluation of QSM: A simple shared-memory model*, TR-98-21, CS department, UT-Austin, 1998.

[5] P. D. MacKenzie and V. Ramachandran. *Computational bounds for fundamental problems on general-purpose parallel models.* ACM SPAA, 152–163, 1998.

[6] V. Ramachandran. *A general purpose shared-memory model for parallel computation.* IMA Vol. 105 in Math. and Its Appl., Springer-Verlag, in press.

[7] V. Ramachandran, B. Grayson, M. Dahlin, *Emulations between QSM, BSP, and LogP: A framework for general-purpose parallel algorithm design.* TR-98-22, CS department, UT-Austin, 1998.

[8] L. G. Valiant. *A bridging model for parallel computation.* CACM, 33(8):103–111, 1990.