Models for Parallel Computation *

Susanne E. Hambrusch
Department of Computer Sciences
Purdue University
West Lafayette, IN 47907, USA
seh@cs.purdue.edu

Parallel computing must meet seemingly contradictory goals: software development should be carried out in an architecture- and technology-independent environment, while both algorithms and system software should take full advantage of the features of the underlying parallel architecture. At the same time, application programmers cannot be expected to be parallel processing experts. Usability, scalability, and portability are thus central issues in high performance computing. A recognized important step towards achieving these goals is the development of a computational model which

- accurately reflects constraints of existing machines,
- has broad applicability with respect to existing and future machines, and
- allows accurate prediction of performance.

Such a computational model should provide a bridge between software and hardware and fulfill the same function as the von Neumann model for sequential computation.

In recent years parallel models have been proposed refined, and evaluated. While no model seems to yet fulfill the stated goals, significant progress has been made. This session contains three papers providing different viewpoints on parallel models. In the following we give a brief description of the development of parallel models and the papers appearing in this session. The discussion makes references to various models, but the list is not intended to be exhaustive.

The PRAM model, a simple shared memory model, was introduced in [9] and it has proven useful in identifying the inherent parallelism in problems. The PRAM has allowed the development of architecture-independent parallel algorithms at a time when parallel platforms were changing rapidly [15, 17, 22]. Some PRAM algorithms have been successfully translated into efficient distributed memory algorithms. For other, especially ones with highly data-dependent communication patterns, efficient adaptations have proven difficult. The PRAM model is widely considered to not match the essential features of existing parallel machines. The main objections include the lack of being able to account for communication latency as well as memory and network contention.

The paper by Gibbons argues the important role the PRAM model still plays [11]. He describes a new shared memory model, the Queued Shared Memory Model (QSM) which accounts for limited communication bandwidth by evaluating the maximum memory

*Research supported in part by ARPA under contract DABT63-92-C-0022ONR. The views and conclusions contained in this paper are those of the authors and should not be interpreted as representing official policies, expressed or implied, of the U.S. government.

contention in each phase of the algorithm. Gibbons shows that the QSM model has an efficient emulation on the BSP model and he demonstrates that the QSM model avoids some of the pitfalls distributed memory models exhibit when applied to shared memory machines.

Simultaneously to the development of algorithms for the PRAM model, parallel algorithms for network models were developed [18, 20]. In addition to an impressive selection and variety of algorithms for particular networks, important results about the equivalence of networks and the computational power of networks were obtained. However, designing a parallel algorithm for a particular network is generally viewed as the wrong direction with respect to usability and portability. The development of a uniform model for general purpose parallel computation was proposed.

The BSP model proposed by Valiant, [24], was one of the first parallel models accounting for communication and abstracting the architectural features in a small number of parameters. In the BSP model, a parallel algorithms consists of a sequence of supersteps. During a superstep, a processor can perform local computation and global communication in the form of send and receive operations. Consecutive supersteps are synchronized by a barrier synchronization. The running time of a superstep includes two parameters: L, the synchronization period and g, the ratio of computation to communication throughput. The fundamental primitives of the BSP models are global barrier synchronization and the routing of arbitrary messages. The paper by Cheatham describes H-BSP, a general purpose parallel computing environment based on the BSP model and intended for developing transportable programs [6]. The H-BSP environment consists of BSP-L, a higher level programming language, compiler tools, communication and synchronization libraries, and application programs.

Another parallel model having received considerable attention is the LogP model [7]. Compared to the BSP model, the LogP model has a more constrained message-passing mechanism and it lacks explicit synchronization. Four parameters, L, o, g, and P, capture the communication cost of an algorithm: communication latency L, overhead o, gap g, and number of processors P.

A number of researchers have argued that the BSP model allows a more convenient programming abstractions compared to the LogP model and that it is better suited as a computational model for parallel computation [5, 21]. In particular, the recent paper by Bilardi et al. presents a quantitative comparison of BSP and LogP models [5]. They develop cross simulations between BSP and LogP and conclude that the BSP model seems to allow slightly greater power and greater simplicity and portability. The BSP simulation in the LogP model is not straightforward and the slowdown experienced may be significant in practice. However, from an asymptotic point of view the two models are shown to be equivalent.

The paper by JáJá argues that both the BSP and the LogP model are not suitable as a bridging model for parallel computation since they require parameters to be adjusted from one parallel platform to the next [16]. The adjusting of parameters could hinder the development of tools and methods that achieve high performance across different platforms. JáJá proposes a parallel model which views a parallel algorithms as a sequence of local computation steps interleaved with global communication primitives. Communication primitives include operations like block transfer, broadcast, scatter, gather, and transpose [3].

A metric determines the cost of a communication primitive. It uses as parameters the amount of data transmitted or received, the latency of the network, and a parameter measuring the time required to inject or receive a word from the network.

Extensions and modifications to the PRAM, BSP, and LogP models have been proposed. In particular, [1, 10, 14, 23] are examples of models based on the PRAM and [2, 4, 12, 19] are examples of models based on abstract network models using point-to-point message sending. Additional references can be found in the subsequent papers.

Each one of three papers included in this session makes its conclusion and prediction as to future computational models and their success. In summary, researchers believe that a number of promising developments have helped the formulation of a parallel model. One of these is the acceptance of emerging standards like MPI [8]. Another one is a potential convergence of parallel architectures [13]. The challenges for an ultimately successful model include, in addition to the already stated needs,

- developing a model that can adapt to communication and memory requirements arising in networks of workstations,
- modeling I/O in addition to communication and computation,
- developing a model for asynchronous parallel computation.

References

[1] A. Aggarwal, A.K. Chandra, M. Snir, "On Communication Latency in PRAM Computations," *Proceeding of 1st ACM SPAA*, pp 11-21, 1989.

[2] A. Alexandrow, M. Ionescu, K. Schauser, C. Scheiman, "LogGP: Incorporating Long Messages into the LogP Model," *Proceedings of 7th ACM SPAA*, pp 95-105, 1995.

[3] D.A. Bader, J. JáJá, "Practical Parallel Algorithms for Dynamic Data Redistribution," Techn. Report, CS-TR-3494, University of Maryland, 1995.

[4] A. Bar-Noy, S. Kipnis, "Designing Broadcasting Algorithms in the Postal Model for Message-Passing Systems," *Proceedings of 4-th ACM Symp. on Parallel Algorithms and Architectures*, pp. 13-22, 1992.

[5] G. Bilardi, K. Herley, A. Pietracaprina, G. Pucci, P. Spirakis, "BSP vs LogP,", *Proceedings of 8th Annual ACM Symposium on Parallel Algorithms and Architectures*, July 1996.

[6] T. Cheatham, "Linguistic Constructs for BSP Style Programming," same proceedings.

[7] D. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, T. von Eicken, "LogP: Towards a Realistic Model of Parallel Computation," *Proceedings of 4-th ACM SIGPLAN Symp. on Principles and Practices of Parallel Programming*, pp. 1-12, 1993.

[8] J.J. Dongarra, R. Hempel, A.J.G. Hey, D.W. Walker. "A Proposal for a User-level, Message Passing Interface in a Distributed Memory Environment", Technical Report TM 12231, Oak Ridge National Laboratory, 1993.

[9] S. Fortune, J. Wyllie, "Parallelism in Random Access Machines," *10-th ACM Symposium on Theory of Computing*, pp 114-118, 1978.

[10] P.B. Gibbons, "A More Practical PRAM Model," *Proceedings of 1989 ACM Symposium on Parallel Algorithms and Architectures*, pp. 158-168, 1989.

[11] P.B. Gibbons, "What Good Are Shared-Memory Models," same proceedings.

[12] S.E. Hambrusch, A. Khokhar, "C^3: A Parallel Model for Coarse-grained Machines", *Journal on*

Parallel and Distributed Computing, Vol. 32, Nr. 2, pp 139–154, 1996.

[13] J.L. Hennessy, "Architectural Convergence and its Applications", in *Developing Computer Science Agenda(s) for High-Performance Computing*, Editor: Uzi Vishkin, ACM Press, pp 72–77, 1994.

[14] T. Heywood and S. Ranka, "A Practical Hierarchical Model of Parallel Computation: I. The Model," *Journal of Parallel and Distributed Computing*, Vol. 16, pp. 212-232, 1992.

[15] J. JáJá, *An Introduction to Parallel Algorithms*, Addison-Wesley, Reading, MA 1992.

[16] J. JáJá, "On Combining Technology and Theory in search of a Parallel Computation Model," same proceedings.

[17] R.M. Karp and V. Ramachandran, "Parallel Algorithms for Shared-Memory Machines," *in Handbook of Theoretical Computer Science*, J. van Leeuwen, Editor, MIT Press/Elsevier, 1990.

[18] F. Thomson Leighton, *Introduction to Parallel Algorithms and Architectures: Arrays . Trees . Hypercubes*, Morgan Kaufmann, 1992.

[19] P. Liu, W. Aiello, S. Bhatt, "An Atomic Model for Message Passing," *Proceedings of 5-th ACM Symp. on Parallel Algorithms and Architectures*, pp. 154-163, 1993.

[20] M.J. Quinn, *Parallel Computing, Theory and Practice*, McGraw-Hill, 1994.

[21] V. Ramachandran, "High Performance Computing Agenda: Converging on a Model for Parallel Machines," in *Developing Computer Science Agenda(s) for High-Performance Computing*, Editor: Uzi Vishkin, ACM Press, pp 129-130, 1994.

[22] J.R. Reif, Editor, *Synthesis of Parallel Algorithms*, Morgan Kaufmann, 1993.

[23] P. de la Torre and C.P. Kruskal, "Towards a Single Model of Efficient Computation in Real Parallel Machines," *Future Generation Computer Systems*, Vol. 8, pp. 395-408, 1992.

[24] L.G. Valiant, "A Bridging Model for Parallel Computation," *Communications of the ACM*, Vol. 33, No. 8, pp. 103-111, 1990.