Parallel Algorithms Column: On the Search for Suitable Models

Yossi Matias
Bell Laboratories
600 Mountain Avenue
Murray Hill NJ 07974
matias@bell-labs.com

A lovely cartoon in the *SIGACT News* Complexity Theory Column 9 [59] illustrates a parallel computing laboratory whose door is "closed due to bankruptcy". Contrary to this perception regarding the state of parallel computing (which may be a reflection of the funding for academic research in the field, as well as the failures of a number of supercomputer vendors), we see in recent years how parallel computing moves from the labs into the workplace in an increasing pace. The role of parallel computing in commercial applications is expected to increase dramatically, as more and more users are demanding access to databases and data warehouses containing hundreds of gigabytes to terabytes of data; see, e.g., the presentations in [24], and the articles [56, 62, 57]. As pointed out in the latter references, one of the main challenges facing parallel computing in the marketplace is enabling software that is reasonably simple to develop, and that is compatible and portable for the variety of available parallel systems.

A closely related issue has been the focus of much attention in the field of parallel algorithm design for the past few years: *what should be the computational model for the design of parallel algorithms.* In choosing a model for algorithm design we are facing an inherent conflict. On the one hand, we would like to have high-level models which abstract away many details of the parallel systems on which the algorithms are to be implemented. Parallel algorithms designed on such models are simple to describe and to analyse; furthermore, such algorithms are easily portable to many different platforms. On the other hand, by ignoring important details about the parallel system, the analysis of the algorithms may not adequately reflect the performance of their respective implementations. Hence, for performance considerations it may become essential to use lower-level models which take into account more details. However, efficient algorithms for low-level models may be harder to design and to analyze; furthermore, such algorithms are often not easily portable from one parallel system to another.

In the first *SIGACT News* Column on Parallel Algorithms [35], Goodrich reviewed a "classic" high-level model for parallel algorithm design—the PRAM—and mentioned some work on alternative, lower level "bridging" models, such as the BSP and LogP. The recent introduction of these latter models has attracted considerable attention, with further studies, both theoretically and experimentally. Some efforts to develop software systems based on bridging models have been initiated (see, e.g., [8, 13, 18, 22, 28, 36, 39, 54]). At the same time, there has been a continuing search for alternative suitable models.

Looking through the technical programs of the last few (SIGACT co-sponsored) ACM Symposia on Parallel Algorithms and Architectures (SPAA), one may observe that the model of computation assumed in the design of the presented algorithms may be one of several models (e.g., CRCW PRAM, QRQW PRAM, Bulk Synchronous (BSP) and its extensions, LogP and its extensions, Block Distributed Memory (BDM), Coarse Grained Multicomputer (CGM), external memory models, and low level models such as meshes, among others). In addition, there are works that study, theoretically or experimentally, the advantages and limitations of various models. This column provides a few pointers to some of the recent works relevant to the modeling of parallel computation.

We first briefly remind the reader some of the contentious issues in modeling parallel computing. During the past two decades, the parallel random access machine, or PRAM, has gained vast popularity in the theoretical study of parallel algorithms. The PRAM model, introduced in [27], is a simple model in which processors execute in lock-step and communicate by reading and writing in a shared memory. The cost metric for an algorithm running on the PRAM model is the number of processors, p, and the time, t. Alternatively, the cost metric may consist of the *work*, w, defined as the time-processor product $p \cdot t$, and the time. The PRAM is a rather natural extension of the well-accepted (serial) random access model (RAM) [4]. It has proven a convenient model to design and analyze parallel algorithms, and to develop a robust theory of parallel algorithms (see, e.g., [41, 45, 58]).

While the PRAM enables one to concentrate on the inherent parallelism of a problem, it abstracts away several issues that may be significant to the performance of parallel programs. Several works have considered the issues of asynchrony, memory contention, latency, memory granularity, bandwidth, and memory hierarchy. (See, e.g., [35, 33] for some references.) Alongside with the study of PRAM algorithms, there has been considerable research on particular network-based models. These models, in which processors send messages to and receive messages from other processors over the given network (e.g., arrays, trees, hypercubes), are more accurate in addressing the communication networks in parallel systems (see, e.g., [47]). However, algorithms developed on one network-based model may be too particular to that model, and not adequate to other models. Thus, these relatively low-level models often result with algorithms that are not portable.

This has prompted the introduction of *bridging* models [61, 63], which is an attempt to provide a unified model that would capture features common to many architectures, and that are significant to the performance of parallel programs. An algorithm designed on a bridging model should be readily implementable on a variety of parallel architectures, and its efficiency on the bridging model should be a good reflection of its actual performance.

Valiant has proposed as a bridging model the Bulk Synchronous Parallel (BSP) model [63, 64], which consists of p processor/memory components communicating by sending point-to-point messages. The interconnection network supporting this communication is characterized only by a per-processor throughput parameter g and a latency parameter L. A BSP computation consists of a sequence of "supersteps" separated by bulk synchronizations. In each superstep the processors can perform local computations and send and receive a set of messages. Messages are sent in a pipelined fashion, and messages sent in one superstep will arrive prior to the start of the next superstep. The time charged for a superstep is calculated as follows. Let w_i be the amount of local work performed by processor i in a given superstep, and let $w = \max_{i=1}^{p} w_i$. Let h be the maximum number of messages sent by any one processor or received by any one processor. Then the cost, t, of the superstep is defined to be $t = \max(w,\ g \cdot h,\ L)$. Intuitively the communication throughput parameter, g, is the best sustainable gap between message sends issued by each individual processor;

therefore $1/g$ represents the available bandwidth per processor.

More recently, Culler *et al* have proposed the LogP model [21, 23]. The LogP is an asynchronous model; in addition to latency (L) and gap parameters (g), that are quite similar to those of the BSP, it has an overhead parameter o, which is defined as the length of time that a processor is engaged in the transmission or reception of each message. There is also a capacity restriction of L/g on the number of messages that can be in transit from any processor or to any processor at any time.

In a recent paper Bilardi *et al* [12] provide a critical assessment of the relative power of the BSP and LogP models. They present efficient cross simulations between the two models, and show that the two models can be implemented with similar performance on most point-to-point networks. Bilardi *et al* conclude that at least within the limits of asymptotic analysis, the two models can be viewed as closely related variants within the bandwidth-latency framework for modeling parallel computation.

Several authors have pointed out that the BSP model may not accurately reflect the cost of sending unbalanced communication patterns, and experimental evidence measuring this inaccuracy is provided in [68]. Modifications to the BSP model to more accurately reflect communication costs have been proposed, for example the E-BSP [44] as well as the Y-BSP [26]. These models, as well as an earlier model, the DRAM of Leiserson and Maggs [48], incorporate cost measures which vary greatly depending on the underlying network architecture; both the Y-BSP and the E-BSP incorporate cost measures designed to capture network proximity. Blelloch *et al* [16] propose the $(d, \mathbf{x})$-BSP model as a refinement for the BSP that provides more detailed modeling of memory bank contention and delay. This model has two additional parameters: the delay d, which is the gap parameter at the memory banks, and the expansion $\mathbf{x}$ which is the ratio of memory banks to processors. Blelloch *et al* argue that the $(d, \mathbf{x})$-BSP more accurately models shared-memory machines with a high bandwidth communication network and more memory banks than processors.

In a recent work, Adler *et al* [1] consider a variant of the BSP model, the BSP(m), that replaces the per-processor bandwidth parameter g by an aggregate bandwidth parameter m. They show that in general the BSP(m) has a possible advantage over the BSP in situations of imbalanced communication. Some variants of the BSP and LogP, including the BDM, LogGP, E-BSP and BSP*, account for spatial locality and allow for "block transfer" [43, 5, 44, 10, 11]. Another variant [49] extends the LogP with hierarchical memory model characterizing each processor, building on [7, 6]. Several models were proposed (e.g. [55, 67]) for the design and analysis of parallel algorithms that account for parallel disk I/O (see [20] and the references therein).

Most of the models discussed above are message-passing and distributed memory models. Advocates such as Vishkin [65], Kennedy [46], Smith [60], and Blelloch [15] have long presented arguments in support of the shared-memory abstraction. In a recent work, Gibbons *et al* [31] proposed the *Queuing Shared Memory (*QSM*)* model as a candidate for a shared memory bridging model. They argue that this model may be competitive in effectiveness to the BSP and LogP models, by providing efficient emulations on the BSP and on the $(d, \mathbf{x})$-BSP. The QSM consists of p processors, each with its own private memory, communicating by reading and writing locations in a shared memory. The interconnection network supporting this communication is characterized by a per-processor throughput parameter g. Processors execute a sequence of bulk-synchronous phases (supersteps), each consisting of an arbitrary interleaving of the operations shared-memory reads, shared-memory writes, and local computation. Consider a QSM phase with maximum contention κ; i.e., κ is the maximum, over all locations x, of the number of processors reading x or the number

Comparison of Some Models of Parallel Computation			
model	**synchrony**	**communication**	**parameters**
PRAM [27]	lock-step	shared memory	p
Module Parallel Computer (MPC) [53]	lock-step	distributed memory	p
LPRAM [3]	lock-step	shared memory	p, ℓ
Phase LPRAM [30]	bulk-synchrony	shared memory	p, ℓ, s
Bulk-Synchronous Parallel (BSP) [63]	bulk-synchrony	message-passing	p, g, L
Postal model [9]	asynchronous	message-passing	p, ℓ
LogP model [21]	asynchronous	message-passing	p, g, ℓ, o
QRQW Asynchronous PRAM [34]	asynchronous	shared memory	p
QRQW PRAM [32]	bulk-synchrony	shared memory	p
Block PRAM (BPRAM) [2]	lock-step	shared memory	p, ℓ, B
Block Distributed Memory (BDM) [43]	bulk-synchrony	distributed memory	p, g, L, B
PRAM(m) model [52]	lock-step	shared memory	p, m
Interval model [51]	bulk-synchrony	message-passing	p, I
Queuing Shared Memory (QSM) [31]	bulk-synchrony	shared memory	p, g

Table 1: (from [33]) *A comparison of several models of parallel computation.* The fourth column indicates the parameters of the model, where p is the number of processors, ℓ is the communication latency (i.e. the time to deliver a message point-to-point or to access the shared memory), s is the cost for a barrier synchronization among all the processors, L is a single parameter that accounts for the sum of ℓ and s, g is the bandwidth gap (i.e. the rate at which processors can perform local operations divided by the rate at which the processors can sustain interprocessor or processor-memory communication), o is the overhead at the processor to send or receive a message, B is the block size (i.e. the number of consecutive cells sent on a write or retrieved on a read), m is the number of shared memory cells available for both reading and writing, and I is the maximum of ℓ, g, and s.

of processors writing x. Let w be the maximum number of local operations done by any processor during the phase; and let h be the maximum number of share-memory read or shared-memory write operations done by any processor during the phase. Then the time cost, t, of the superstep is defined to be $t = \max(w,\ g \cdot h,\ \kappa)$. A comparison of some models of parallel computation is given in Table 1. Let us also mention that a few models incorporate powerful aggregate communication primitives [14, 17], for providing an easier programming model.

To conclude we refer the reader to some recent survey and position papers [49, 51, 38, 19, 33, 42], as well as to the collection of position papers in [66]. We finally mention that several groups have been recently involved in implementations and experimentations of parallel algorithms, using bandwidth-limited general purpose models (see, e.g., [8, 13, 18, 21, 22, 28, 36, 39, 54]).

Finding models that can provide a satisfying balance between ease of use and accuracy remains a challenge of primary importance. We have pointed out several recent attempts in this direction, and we are likely to witness more efforts in the near future, both in the theoretical and in the experimental levels. Are we likely to see convergence into a single accepted model, or are we going to see a plurality of models, depending on application domain, platform or taste? This remains to be seen.

References

[1] M. Adler, P. B. Gibbons, Y. Matias, and V. Ramachandran. Modeling parallel bandwidth: Local vs. global restrictions. In *Proc. 9th ACM Symp. on Parallel Algorithms and Architectures*, June 1997. To appear.

[2] A. Aggarwal, A. K. Chandra, and M. Snir. On communication latency in PRAM computations. In *Proc. 1st ACM Symp. on Parallel Algorithms and Architectures*, pages 11–21, June 1989.

[3] A. Aggarwal, A. K. Chandra, and M. Snir. Communication complexity of PRAMs. *Theoretical Computer Science*, 71(1):3–28, 1990.

[4] A.V. Aho, J.E. Hopcroft, and J.D. Ullman. *The Design and Analysis of Computer Algorithms.* Addison-Wesley Publishing Company, Inc., Reading, Massachusetts, 1974.

[5] A. Alexandrov, M. F. Ionescu, K. E. Schauser, and C. Sheiman. LogGP: Incorporating long messages into the LogP model — one step closer towards a realistic model for parallel computation. In *Proc. 7th ACM Symp. on Parallel Algorithms and Architectures*, pages 95–105, July 1995.

[6] B. Alpern and L. Carter. Towards a model for portable parallel performance: exposing the memory hierarchy. In *Portability and Performance for Parallel Processing*, pages 21–41. John Wiley & Sons, 1994.

[7] B. Alpern, L. Carter, and E. Feig. Uniform memory hierarchies. In *Proc. 31st IEEE Symp. on Foundations of Computer Science*, pages 600–608, October 1990.

[8] D. A. Bader and J. JàJà. Practical parallel algorithms for dynamic data redistribution, median finding, and selection. In *Proc. 10th International Parallel Processing Symposium*, pages 292–301, April 1996.

[9] A. Bar-Noy and S. Kipnis. Designing broadcasting algorithms in the postal model for message-passing systems. In *Proc. 4th ACM Symp. on Parallel Algorithms and Architectures*, pages 13–22, June-July 1992.

[10] A. Baumker and W. Dittrich. Fully dynamic search trees for an extension of the BSP model. In *Proc. 8th ACM Symp. on Parallel Algorithms and Architectures*, pages 233–242, June 1996.

[11] A. Baumker, W. Dittrich, and F. Meyer auf der Heide. Truly efficient parallel algorithms: 1-optimal multisearch for an extension of the BSP model. Technical report, University of Paderborn, 1996.

[12] G. Bilardi, K.T. Herley, A. Pietracaprina, G. Pucci, and P. Spirakis. BSP vs LogP. In *8th ACM Symp. on Parallel Algorithms and Architectures*, pages 25–32, 1996.

[13] R.H. Bisseling and W.F. McColl. Scientific computing on bulk synchronous parallel architectures. In *Proc. 133th IFIP World Computer Congress*, pages 509–514, 1994.

[14] G. E. Blelloch. *Vector Models for Data-Parallel Computing.* The MIT Press, Cambridge, MA, 1990.

[15] G. E. Blelloch. Programming parallel algorithms. *Communications of the ACM*, 39(3):85–97, 1996.

[16] G. E. Blelloch, P. B. Gibbons, Y. Matias, and M. Zagha. Accounting for memory bank contention and delay in high-bandwidth multiprocessors. In *Proc. 7th ACM Symp. on Parallel Algorithms and Architectures*, pages 84–94, July 1995.

[17] G. E. Blelloch, C. E. Leiserson, B. M. Maggs, C. G. Plaxton, S. J. Smith, and M. Zagha. A comparison of sorting algorithms for the Connection Machine CM-2. In *Proc. 3rd ACM Symp. on Parallel Algorithms and Architectures*, pages 3–16, July 1991.

[18] T. Cheatham, A. Fahmy, D.C. Stefanescu, and L.G. Valiant. Bulk synchronous parallel computing – a paradigm for transportable software. In *Proc. IEEE 28th Hawaii Int. Conf. on System Science*, January 1995.

[19] T. Cheatman. Linguistic constructs for BSP style programming. In *Proc. 1996 ICPP Workshop on Challenges for parallel processing*, pages 96–102, August 1996.

[20] Y.-J. Chiang, M. T. Goodrich, E. F. Grove, R. Tamassia, D. E. Vengroff, and J. S. Vitter. External-memory graph algorithms. In *Proc. 6th ACM-SIAM Symp. on Discrete Algorithms*, pages 139–149, January 1995.

[21] D. Culler, R. Karp, D. Patterson, A. Sahay, K.E. Schauser, E. Santos, R. Subramonian, and T. von Eicken. LogP: Towards a realistic model of parallel computation. In *Proc. 4th ACM SIGPLAN Symp. on Principles and Practices of Parallel Programming*, pages 1–12, May 1993.

[22] D. E. Culler, A. Dusseau, R. Martin, and K. E. Schauser. Fast parallel sorting under LogP: from theory to practice. In *Proc. Workshop on Portability and Performance for Parallel Processing*, Southhampton, England, July 1993.

[23] D.E. Culler, R.M. Karp, D. Patterson, A. Sahay, E.E. Santos, K.E. Schauser, R. Subramonian, and T. von Eicken. LogP: A practical model of parallel computation. *Communications of the ACM*, pages 78–85, November 1996.

[24] U. Dayal, P.M.D. Gray, and S. Nishio, editors. *Proc. 21st International Conference on Very Large Data Bases*, Vendor sessions 1–5, pages 677–701 1995.

[25] P. de la Torre and C. P. Kruskal. Towards a single model of efficient computation in real parallel machines. *Future Generation Computer Systems*, 8:395–408, 1992.

[26] P. de la Torre and C. P. Kruskal. Submachine locality in the bulk synchronous setting. In *Proc. Euro-Par'96*, pages 352–358, August 1996.

[27] S. Fortune and J. Wyllie. Parallelism in random access machines. In *Proc. 10th ACM Symp. on Theory of Computing*, pages 114–118, May 1978.

[28] A. V. Gerbessiotis and C. J. Siniolakis. Deterministic sorting and randomized median finding on the BSP model. In *Proc. Eighth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 223–232, June 1996.

[29] A. V. Gerbessiotis and L. G. Valiant. Direct bulk-synchronous parallel algorithms. *Jour. Parallel and Distributed Computing*, 22:251–267, 1994.

[30] P. B. Gibbons. A more practical PRAM model. In *Proc. 1st ACM Symp. on Parallel Algorithms and Architectures*, pages 158–168, June 1989. Full version in *The Asynchronous PRAM: A semi-synchronous model for shared memory MIMD machines*, PhD thesis, U.C. Berkeley 1989.

[31] P. B. Gibbons, Y. Matias, and V. Ramachandran. Can a shared-memory model serve as a bridging model for parallel computation? In *Proc. 9th ACM Symp. on Parallel Algorithms and Architectures*, June 1997. To appear.

[32] P. B. Gibbons, Y. Matias, and V. Ramachandran. The Queue-Read Queue-Write PRAM model: Accounting for contention in parallel algorithms. *SIAM Journal on Computing*, 1997. To appear. Preliminary version appears in *Proc. 5th ACM-SIAM Symp. on Discrete Algorithms*, pages 638-648, January 1994.

[33] P.B. Gibbons. What good are shared-memory models? In *Proc. 1996 ICPP Workshop on Challenges for parallel processing*, pages 103–114, August 1996.

[34] P.B. Gibbons, Y. Matias, and V. Ramachandran. The Queue-Read Queue-Write Asynchronous PRAM model. In *Proc. Euro-Par'96, Workshop on Theory and Models of Parallel Computation, Springer LNCS 1124*, pages 279–292, August 1996.

[35] M. Goodrich. Parallel Algorithms Column 1: Models of computation. *Sigact News*, 24:16–21, December 1993.

[36] M. Goudreau, K. Lang, S. Rao, T. Suel, and T. Tsantilas. Towards efficiency and portability: Programming with the BSP model. In *Proc. Eighth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 1–12, June 1996.

[37] S. Hambrusch and A. Khokhar. C^3: An architecture-independent model for coarse-grained parallel machines. In *Proc. 6th IEEE Symp. on Parallel and Distributed Processing*, pages 544–551, 1994.

[38] S.E. Hambrusch. Models for parallel computation. In *Proc. 1996 ICPP Workshop on Challenges for parallel processing*, pages 92–95, August 1996.

[39] D. R. Helman, D. A. Bader, and J. JàJà. Parallel algorithms for personalized communication and sorting with an experimental study. In *Proc. Eighth Annual ACM Symposium on Parallel Algorithms and Architectures*, pages 211–222, June 1996.

[40] T. Heywood and S. Ranka. A practical hierarchical model of parallel computation: I. The model. *Journal of Parallel and Distributed Computing*, 16:212–232, 1992.

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

[42] J. JáJá. On combining technology and theory in search of a parallel computation model. In *Proc. 1996 ICPP Workshop on Challenges for parallel processing*, August 1996.

[43] J. JáJá and K. W. Ryu. The Block Distributed Memory model. Technical Report UMIACS-TR-94-5, University of Maryland Institute for Advanced Computer Studies, College Park, MD, January 1994.

[44] B. H. H. Juurlink and H. A. G. Wijshoff. The E-BSP Model: Incorporating general locality and unbalanced communication into the BSP Model. In *Proc. Euro-Par'96*, pages 339–347, August 1996.

[45] R. M. Karp and V. Ramachandran. Parallel algorithms for shared-memory machines. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A*, pages 869–941. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1990.

[46] K. Kennedy. A research agenda for high performance computing software. In *Developing a Computer Science Agenda for High-Performance Computing*, pages 106–109. ACM Press, 1994.

[47] F. T. Leighton. *Introduction to Parallel Algorithms and Architectures: Arrays · Trees · Hypercubes*. Morgan Kaufmann, San Mateo, CA, 1992.

[48] C. E. Leiserson and B. M. Maggs. Communication-efficient parallel algorithms for distributed random-access machines. *Algorithmica*, 3(1):53–77, 1988.

[49] Z. Li, R.H. Mills, and J.H. Reif. Models and resource metrics for parallel and distributed computation. In *Proc. 28th Hawaii International Conference on System Sciences*. IEEE Press, January 1995.

[50] P. Liu, W. Aiello, and S. Bhatt. An atomic model for message-passing. In *Proc. 5th ACM Symp. on Parallel Algorithms and Architectures*, pages 154–163, June-July 1993.

[51] B. M. Maggs, L. R. Matheson, and R. E. Tarjan. Models of parallel computation: A survey and synthesis. In *Proc. 28th Hawaii International Conf. on System Sciences*, pages II: 61–70, January 1995.

[52] Y. Mansour, N. Nisan, and U. Vishkin. Trade-offs between communication throughput and parallel time. In *Proc. 26th ACM Symp. on Theory of Computing*, pages 372–381, 1994.

[53] K. Mehlhorn and U. Vishkin. Randomized and deterministic simulations of PRAMs by parallel machines with restricted granularity of parallel memories. *Acta Informatica*, 21:339–374, 1984.

[54] R. Miller. A library for bulk-synchronous parallel programming. In *Proc. of the British Computer Society Parallel Processsing, Specialist Group Workshop on General Purpose Parallel Computing*, December 1993.

[55] M. H. Nodine and J. S. Vitter. Large-scale sorting in parallel memories. In *Proc. 3rd ACM Symp. on Parallel Algorithms and Architectures*, pages 29–39, July 1991.

[56] D. Pountain. Parallel goes populist. *Byte ("the magazine of technology integration")*, pages 88NA3–88NA8, May 1997.

[57] A. Radding. Multiprocessing: Scaling up. *Information Week ("for business and technology managers")*, pages 62–71, March 18 1996.

[58] J. H. Reif, editor. *A Synthesis of Parallel Algorithms*. Morgan-Kaufmann, San Mateo, CA, 1993.

[59] *Sigact News*, 26(2):14, June 1995.

[60] B. Smith. Invited lecture, *7th ACM Symp. on Parallel Algorithms and Architectures*, July 1995.

[61] L. Snyder. Type architectures, shared memory and the corollary of modest potential. *Annual Review of Computer Science*, pages 289–318, 1986.

[62] T. Thompson. The world's fastest computers (cover story). *Byte ("the magazine of technology integration")*, pages 45–64, January 1996. (http://www.byte.com/art/9601/sec6/sec6.htm).

[63] L. G. Valiant. A bridging model for parallel computation. *Communications of the ACM*, 33(8):103–111, 1990.

[64] L. G. Valiant. General purpose parallel architectures. In J. van Leeuwen, editor, *Handbook of Theoretical Computer Science, Volume A*, pages 943–972. Elsevier Science Publishers B.V., Amsterdam, The Netherlands, 1990.

[65] U. Vishkin. A parallel-design distributed-implementation (PDDI) general purpose computer. *Theoretical Computer Science*, 32:157–172, 1984.

[66] U. Vishkin, editor. *Developing a Computer Science Agenda for High-Performance Computing.* ACM Press, 1994.

[67] J. S. Vitter and E. A. M. Shriver. Optimal disk I/O with parallel block transfer. In *Proc. 22nd ACM Symp. on Theory of Computing*, pages 159–169, May 1990.

[68] H. A. G. Wijshoff and B. H. H. Juurlink. A quantitative comparison of parallel computation models. In *Proc. 8th ACM Symp. on Parallel Algorithms and Architectures*, pages 13–24, June 1996.