Parallel Virtual Memory

F. Dehne* W. Dittrich† D. Hutchinson* A. Maheshwari*‡

1 Motivation

Parallel algorithms for the *Bulk Synchronous Parallel* (BSP) and closely related *Coarse Gained Multicomputer* (CGM) programming model assume that all data can be distributed over the main memories of the processors involved. In practice, this may not be the case. For large scale applications where parallel processing is helpful, the total amount of data often exceeds the total main memory available and parallel disk I/O becomes a necessity. A common scenario for distributed memory parallel machines assumes that each processor has multiple local hard disks available, where each disk is only accessible by the respective local processor. Parallel disk I/O has been identified as a critical component of a suitable high performance computer for a number of the Grand Challenge problems (see for instance the Scalable I/O Initiative project [5]).

For sequential computing, the *virtual memory* concept has long been established as a standard method for managing external memory. Its main advantage is that it allows the application programmer to access a large virtual memory without having to deal with the intricacies of blocked secondary memory accesses. We present a similar, transparent, *parallel virtual memory* programming environment for BSP-style parallel computing.

2 Definitions

We give a brief introduction to the PDM (Parallel Disk Model), BSP, CGM, and EM-CGM (external memory CGM). For more details and references consult [2,3].

The *Parallel Disk Model* (PDM) introduced by Vitter and Shriver is used to model a two level memory hierarchy consisting of D parallel disks connected to $v \geq 1$ processors which communicate via a shared internal memory or a network. The PDM uses the following additional parameters: N = problem size, M = internal memory size, B = block transfer size, where $M < N$, and $1 \leq DB \leq M/2$. The PDM cost measure is the number of I/O operations required by an algorithm, where DB items can be transferred between internal memory and disk in a single I/O operation.

The Bulk-Synchronous Parallel (BSP) parallel computing model was introduced by Valiant [6]. It is intended as a bridging model; a standard model on which both hardware and software designs can agree. It is neither a hardware nor a programming model, but somewhere in between. A dominant feature of BSP is the superstep, which is an interval of time in which the processors compute independently, and then communicate results. Before the next superstep begins, the processors synchronize to ensure that they have all completed the current superstep.

A *BSP algorithm* $\mathcal{A}$ on a problem $\mathcal{P}$ and computer configuration $\mathcal{C}$ can be characterized by the parameters $(N, v, g, L, \lambda, \beta, \alpha)$, where N is the problem size (in problem items), v is the number of processors, g is the time required for a communication operation, L is the minimum time required for the processors to synchronize, λ is the number of BSP supersteps required, β is the computation time of (the busiest) processor, and α is the number of communication operations performed by a processor. In all cases, "time" is measured in "number of processor cycles". The time required by a BSP algorithm under the model is $\beta + g \cdot \alpha + \lambda \cdot L$.

A stringent goodness criterion used with BSP is the c-optimality criterion. In order for $\mathcal{A}$ to be c-optimal, **(1)** $\frac{\beta}{\beta^*/v}$ is in $c + o(1)$, where β^* is the time for an optimal sequential algorithm which solves $\mathcal{P}$, and **(2)** the computation time of $\mathcal{A}$ asymptotically dominates its communication time.

An h-relation is a BSP communication operation in which the amount of data sent or received by any processor is at most h items. A *CGM algorithm* is a special case of a BSP algorithm where the communication part of each superstep consists of exactly one h-relation with $h = \Theta(\frac{N}{v})$. An algorithm for a CGM with multiple disks attached to each processor is referred to as an *EM-CGM algorithm*. The cost of an EM-CGM algorithm is $\beta + g \cdot \alpha + G \cdot \kappa + \lambda \cdot L$, where G is the time required for a parallel I/O operation to a processor's local disks, and κ is the number of such operations required. Goodness criteria proposed [2,4] for the EM-CGM model include **(1)** an extended version of c-optimality, where the computation time is also required to dominate the I/O time, **(2)** relaxed criteria *I/O-efficiency* and *communication-efficiency*, where the I/O and communication times may be asymptotically the same as the computation time, and **(3)** I/O-optimality, where the I/O time may be necessarily larger than the computation time, but the number of I/O operations meets the lower bound for the problem.

*School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6, {dehne, hutchins, maheshwa}@scs.carleton.ca. Research partially supported by the Natural Sciences and Engineering Research Council of Canada.

†Bosch Telecom, Backnang, Germany, wolfgang.dittrich@bk.bosch.de.

‡Part of this work was done while visiting the MPI Informatik, Saarbrücken, Germany

F. Dehne, W. Dittrich, D. Hutchinson, and A. Maheshwari, "Parallel virtual memory," in Proc. 10th Annual ACM-SIAM Symposium on Discrete Algorithms, Baltimore, MD, 1999, pp. 889-890.

3 Theoretical Results

The ACM Workshop on Strategic Directions in Computing Research recently issued a challenge [1] to combine the PDM model with the BSP and related models. In [2] we proposed instances of such a combined model, which we call EM-BSP, EM-BSP*, and EM-CGM. The new models measure the time for computation, communication and I/O, not just the number of I/O operations used.

In [2] we also presented *randomized* simulation methods for mapping BSP like algorithms written for large parallel virtual memory to external memory algorithms for a multi disk, multi processor machine. The results showed that a v-processor CGM algorithm $\mathcal{A}$ with communication time $g\alpha+\lambda L$, computation time $\beta+\lambda L$, and local memory μ can be simulated as a p-processor EM-CGM algorithm with computation time $(1+o(1))\frac{v}{p}\beta+O(L\lambda\frac{v}{pk})$, communication time $O(gl(\frac{v}{p}\alpha+\lambda\frac{\log(M/B)}{k})+L\lambda\frac{v}{pk})$, and I/O time $O(Gl\frac{v}{p}(\frac{\mu\lambda}{BD}))$ with probability $1-\exp(-\Omega(l\log l\cdot\log\frac{M}{B}))$ for suitable $l\geq 1$, $\beta=\omega(\lambda\mu)$, $M=\Theta(k\mu)$, $v=\Omega(pkD\cdot\log\frac{M}{B})$, $M/B\geq p^{\epsilon}$ and arbitrary constants k, $\epsilon>0$. Conditions for meeting the extended c-optimality criterion were also identified.

Our newest results include *deterministic* simulation methods for mapping a CGM algorithm with parallel virtual memory to an external memory environment (see [3] for details).

(1) A v-processor CGM algorithm $\mathcal{A}$ with λ supersteps, local memory size μ, computation time $\beta+\lambda L$ and communication time $g\alpha+\lambda L$ can be simulated, deterministically, as a p-processor EM-CGM algorithm $\mathcal{A}'$ with computation time $\frac{v}{p}(\beta+O(\lambda\mu))+\frac{v}{p}\lambda L$, communication time $\frac{v}{p}g\alpha+\frac{v}{p}\lambda L$, and I/O time $\frac{v}{p}G\cdot O(\lambda\frac{\mu}{DB})+\frac{v}{p}\lambda L$ for $v\geq p$, $M=\Theta(\mu)$, $N=\Omega(vDB)$, $B=O(\frac{N}{v^2})$.

(2) The above deterministic simulation, applied to known BSP and CGM algorithms, yields EM-CGM algorithms with **(a)** I/O complexity $O(\frac{N}{pDB})$ for *sorting, permutation, matrix transpose*, lower envelope of line segments (N denotes the size of the input plus output), area of union of rectangles, $3D$-maxima, weighted dominance counting for planar point set, **(b)** I/O complexity $\tilde{O}(\frac{N}{pDB})$ for 3-dimensional convex hull and planar Voronoi diagram (the underlying CGM algorithms are probabilistic), nearest neighbor problem for planar point set, **(c)** I/O complexity $O(\frac{N\log v}{pDB})$ for trapezoidal decomposition and triangulation, and **(d)** I/O complexity $O(\frac{N\log v}{pDB})$ for list ranking, Euler tour of a tree, connected components, spanning forest, lowest common ancestor in a tree, tree contraction, expression tree evaluation, open ear decomposition, and biconnected components.

Superficially, some of these results appear to contradict known lower bounds, but this is not the case, and can be explained by our constraints on the parameters. In [3] we show that such constraints are not restrictive in practice.

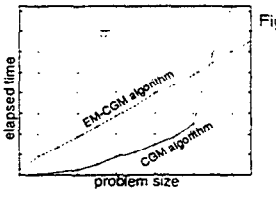

Fig.1

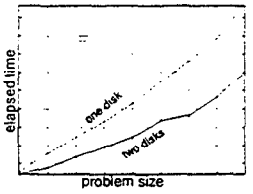

Fig.2

4 Experimental Results

We are implementing a prototype system on a network of Pentium processors connected via a 2 GB Ethernet switch and with multiple disks per processor. Preliminary experiments confirm that our approach outperforms the standard virtual memory techniques used by the Linux operating system. Figure 1 shows a performance comparison between an original CGM sorting algorithm and the corresponding EM-CGM code. For small data size, the overhead for the external memory mapping makes the CGM algorithm outperform the EM-CGM algorithm. However, once the main memory is all utilized, the performance of the CGM algorithm degenerates. The CGM timings for larger problem size were obtained by using the standard UNIX paging mechanism. The EM-CGM algorithm obtained through our parallel virtual memory method continues "smoothly" and clearly outperform the CGM code. Figure 2 shows how our EM-CGM simulation reacts to multiple disks. It shows that the method converts additional, parallel, disks into a significant performance increase.

5 References

[1] T. H. Cormen and M. T. Goodrich, "Position Statement, ACM Workshop on Strategic Directions in Computing Research: Working Group on Storage I/O for Large-Scale Computing," http://www.cs.dartmouth.edu/ thc/SDCR96/ CormenModel/index.html, 1996.

[2] F. Dehne, W. Dittrich and D. Hutchinson, "Efficient External Memory Algorithms by Simulating Coarse-Grained Parallel Algorithms," Proc. ACM Symp. on Parallel Algorithms and Architectures, 1997, 106-115

[3] F. Dehne, W. Dittrich, D. Hutchinson, A. Maheshwari, "Reducing I/O complexity by simulating coarse grained parallel algorithms", Technical Report, http://www.scs.carleton.ca/~dehne/papers

[4] W. Dittrich, D. Hutchinson, and A. Maheshwari, "Blocking in Parallel Multisearch Problems," Proc. ACM Symp. on Parallel Algorithms and Architectures, 1998.

[5] J. Poole et al., "Survey of I/O Intensive Applications", Scalable I/O Initiative, Applications WG, SIO WP-1, http://www.cacr.caltech.edu/SIO/

[6] L. G. Valiant, "A Bridging Model for Parallel Computation", Communications of the ACM, 33(8), 1990, 103-111