Scalable Parallel Computational Geometry *

— *Summary* —

Frank Dehne †
School of Computer Science
Carleton University
Ottawa, Canada K1S 5B6
dehne@scs.carleton.ca

Parallel Computational Geometry is concerned with solving some given geometric problem of size n on a parallel computer with p processors (e.g., a PRAM, mesh, or hypercube multiprocessor) in time $T_{parallel}$. Let $T_{sequential}$ and $S_p = \frac{T_{sequential}}{T_{parallel}}$ denote the sequential time complexity of the problem and the *speedup* obtained by the parallel solution, respectively. If $S_p = p$, then the parallel algorithm is clearly *optimal*. Theoretical work for Parallel Computational Geometry has so far focussed on the case $\frac{n}{p} = O(1)$, also referred to as the *fine grained* case. However, for parallel geometric algorithms to be relevant in practice, such algorithms must be *scalable*, that is, they must be applicable and efficient for a wide range of ratios $\frac{n}{p}$.

Most existing multicomputers (e.g. the Intel Paragon, Intel iPSC/860, and CM-5) consist of a set of p *state-of-the-art* processors (e.g. SPARC proc.), each with considerable local memory, connected to some interconnection network (e.g. mesh, hypercube, fat tree). For these machines, $\frac{n}{p}$ is considerably larger than $O(1)$. Therefore the design of scalable algorithms is listed as one major goal in the 1992 "Grand Challenges" report [4].

Yet, only little theoretical work has been done for designing scalable parallel algorithms for Computational Geometry. Note that, if there exists an optimal fine grained algorithm, then, at least from a theoretical point of view, the problem is trivial. Standard simulation gives an optimal algorithm for any ratio $\frac{n}{p}$. Many multiprocessors provide system software tools, ususally referred to as *virtual processors*, for implementing such simulation.

However, for most interconnection networks used in practice, many problems do not as yet have fine grained algorithms with linear speedup, or such linear speedup parallel algorithms are impossible due to bandwidth or diameter limitations. Such a situation is depicted in Figure 1. It shows the speedup S_p as a function of p, for $1 \leq p \leq n$. The diagonal curve "A", represents an algorithm with linear speedup. The vertical line through $p = n$ represents the fine-grained case. Point "a" represents a fine-grained algorithm with linear speedup. As indicated above, linear speedup is impossible to obtain for some networks (e.g. meshes). Let point "b" represent an optimal fine-grained algorithm with less than linear speedup. For $p < n$, curve "B" (the straight line from "b" to the origin) shows the speedups obtained by standard simulation. However, if "b" represents the optimal fine-grained speedup, then the entire curve "B" does not necessarily represent the optimal speedups for all possible ratios $\frac{n}{p}$.

We can show that, for a variety of geometric problems, the actual curve of optimal speedups for all $1 \leq p \leq n$ is a convex curve similar to curve "C" depicted in Figure 1. That is, for $\frac{n}{p}$ larger than $O(1)$, we present algorithms with speedups considerably faster than what can be obtained through the standard *virtual processor* simulation of fine-grained algorithms. It is common knowledge that many theoretical Parallel Computational Geometry algorithms perform miserably in practice. The above observation

*This presentation reviews recent joint work with A. Fabri (INRIA, Sophia-Antipolis), C. Kenyon (ENS Lyon), and A. Rau-Chaplin (DIMACS).

†Research partially supported by the Natural Sciences and Engineering Research Council of Canada

F. Dehne, "Scalable parallel computational geometry," invited talk, in Proc. Canada-France Conference on Parallel and Distributed Computing, Montreal, 1994, Lecture Notes in Computer Science Vol. 805, pp.115-119.

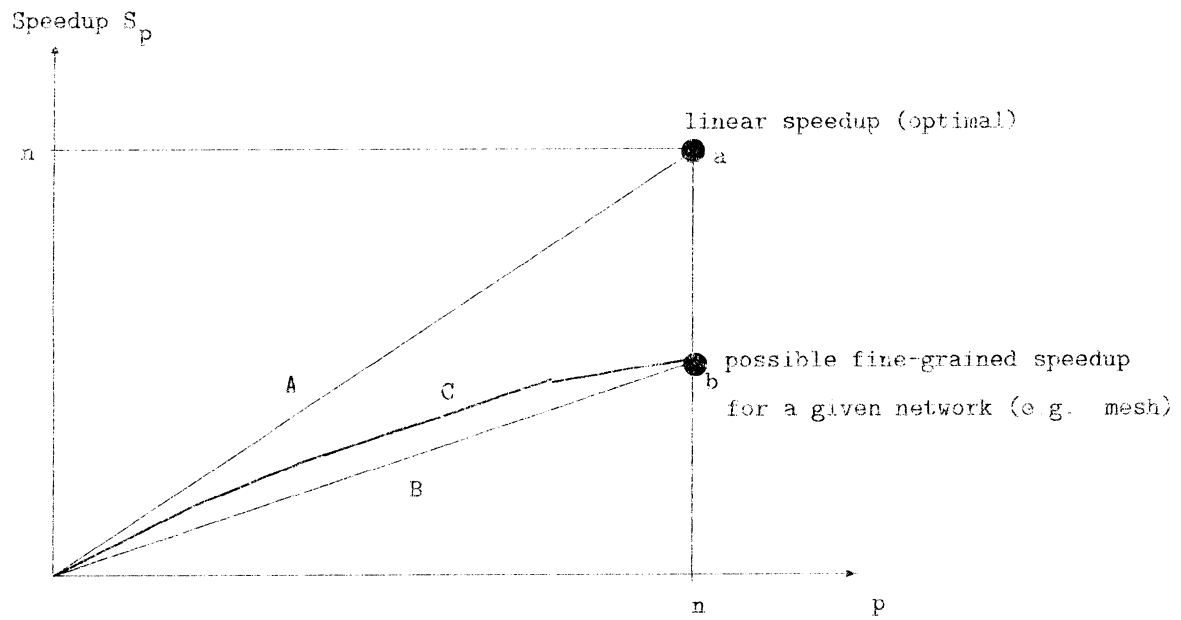

Figure 1: Speedups For Various Values Of p.

seems to be one of the reasons. In contrast, experiments with some of our methods show that our scalable algorithms designed for $\frac{n}{p}$ larger than $O(1)$ seem to have considerable practical relevance.

In a nutshell, the basic idea for our methods is as follows: We try to combine optimal sequential algorithms for a given problem with an efficient global routing and partitioning mechanism. We devise a constant number of partitioning schemes of the global problem (on the entire data set of n data items) into p subproblems of size $O(\frac{n}{p})$. Each processor will solve (sequentially) a constant number of such subproblems, and we use a constant number of global routing operations to permute the subproblems between the processors. Eventually, by combining the $O(1)$ solutions of it's $O(\frac{n}{p})$ size subproblems, each processor determines it's $O(\frac{n}{p})$ size portion of the *global* solution.

The above is necessarily an oversimplification. The actual algorithms will do more than just those permutations. The main challenge lies in devising the above mentioned partitioning schemes. Note that, each processor will solve only a constant number of $O(\frac{n}{p})$ size subproblems, but eventually will have to determine it's part of the entire $O(n)$ size problem. The most complicated part of the algorithm is to ensure that the algorithm will terminate after $O(1)$ global permutation rounds.

Deterministic Methods For Scalable Parallel Computational Geometry

In [2] we presented deterministic algorithms for the following well known geometric problems:

(1) Area of the union of rectangles in 2-space.

(2) $3D$-maxima.

(3) All nearest neighbors of a point set in 2-space.

(4) Lower envelope of non-intersecting line segments in 2-space (and with slightly more memory for possibly intersecting line segments).

(5) $2D$-weighted dominance counting.

(6) Multisearch on balanced search trees, segment tree construction, and multiple segment tree search.

We also studied the following applications of (6): the problem of determining for a set of simple polygons in 2-space all directions for which a uni-directional translation ordering exists, and determining for a set of simple polygons a multi-directional translation ordering.

Our scalable parallel algorithms for Problems 1-6 have a running time of

$$O(\frac{T_{sequential}}{p} + T_{sort}(n,p))$$

on a p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$ where $\frac{n}{p} \geq p$. $T_{sort}(n,p)$ refers to the time of a global sort operation.

Consider for example the *mesh* architecture. For the fine grained case, $\frac{n}{p} = O(1)$, a time complexity of $O(\sqrt{n})$ is optimal. Hence, simulating the existing results on a coarse grained machine gives $O(\frac{n}{p}\sqrt{n})$ time coarse grained methods. Our methods for the above problems run in time $O(\frac{n}{p}(\log n + \sqrt{p}))$, a considerable improvement over the existing methods. For the *hypercube*, our algorithms are optimal for $n \geq p^{\log p}$, in which case they also yield a considerable improvement over previous methods.

High Probability Methods For Scalable Parallel Computational Geometry

In [3] we present faster and more general *high probability* methods. They can be applied to any p-processor coarse grained multicomputer with arbitrary interconnection network and local memories of size $O(\frac{n}{p})$ where $\frac{n}{p} \geq p^{\alpha}$ and $\alpha > 0$ is a fixed constant. The following problems were studied:

(7) Lower envelope of non-intersecting line segments in $[0,1]^2$

(8) Visible portion of parallelepipeds in $[0,1]^d$, $d = O(1)$. For d=3, Problem 8 is the well known visibility problem for parallel rectangles in $[0,1]^3$.

(9) Convex hull of points in d-space, $d = O(1)$

(10) Maximal elements of points in d space, $d = O(1)$.

(11) Voronoi diagram of points in $[0,1]^d$, $d = O(1)$.

(12) All nearest neighbors for points in $[0,1]^d$, $d = O(1)$.

(13) Largest empty circle for points in $[0,1]^d$, $d = O(1)$.

(14) Largest empty hyperrectangle for points in $[0,1]^d$, $d = O(1)$

Our solutions for Problems 7-10 have, with high probability, a time complexity of

$$O(\frac{T_{sequential}}{p} + T_{pSUM}(p) + T_{comp}(n,p))$$

$T_{pSUM}(p)$ and $T_{comp}(n,p)$ denote the parallel time complexity to compute the partial sums of p integers (one stored at each processor) and compress a subset of data, respectively. Our solutions for Problems 11-14 have, with high probability, a time complexity of

$$O(\frac{T_{sequential}}{p} + T_{sort}(n,p))$$

Experimental Results

Our algorithms are simple and easy to implement. The constants in the time complexity analysis are small. Except for a small (fixed) number of communication rounds, all other computation is sequential and consists essentially of solving on each processor a small (fixed) number of subproblems of size n/p. Hence, it allows to use existing sequential code for the respective problem. For the communication rounds, we can use well studied existing code for data compression, partial sum, and sorting.

For coarse grained machines, our methods imply communication through few large messages rather than having many small messages. This is important for machines like the Intel iPSC where each message creates a considerable overhead. Note, however, that our analysis accounts for the length of messages.

We implemented some of our methods on an Intel iPSC/860 and a CM-5, and obtained very good timing results (even without much programming efforts). They indicate that our methods are of considerable practical relevance.

References

[1] S. G. Akl and K. A. Lyons, *Parallel Computational Geometry*, Prentice-Hall, 1993.

[2] F. Dehne, A. Fabri, and A. Rau-Chaplin, "Scalable parallel computational geometry for coarse grained multicomputers," in Proc. *ACM Symposium on Computational Geometry*, 1993, pp. 298-307.

[3] F. Dehne, C. Kenyon, and A. Fabri, "Scalable And Architecture Independent Parallel Geometric Algorithms With High Probability Optimal Time," Technical Report, School of Computer Science, Carleton University, Ottawa, Canada K1S 5B6, 1994.

[4] *Grand Challenges: High Performance Computing and Communications*. The FY 1992 U.S. Research and Development Program. A Report by the Committee on Physical, Mathematical, and Engineering Sciences. Federal Council for Science, Engineering, and Technology. To Supplement the U.S. President's Fiscal Year 1992 Budget.