Algorithmica (1999) 24: 173–176

Guest Editor's Introduction

F. Dehne[1]

During the late eighties, the *parallel algorithms* field was in a serious crisis. Many problems had been studied, and various upper and lower bounds had been established for them under different computational models ranging from shared memory models to network models like hypercubes, two-dimensional meshs, or three-dimensional cubes. Most commercial parallel machines did indeed, and do still, use one of these four interconnection methods. Yet, when it came to implementing the theoretical results on those machines, the speedups obtained were often very disappointing. Some practitioners even questioned the usefulness of parallel algorithms altogether.

The nineties have brought a dramatic change to the field with the emergence of BSP style, coarse-grained, parallel computing models. In his ground breaking paper [4], Valiant proposed the *Bulk Synchronous Parallel* (*BSP*) bridging model for parallel computation based on a generic model of a parallel machine; see Figure 1. He introduced the concept that parallel computation should be modeled as a series of supersteps rather than individual message passing steps or shared memory accesses. Every BSP algorithm consists of a sequence of such supersteps, each consisting of various message passing (or shared memory access) operations. Supersteps are separated by barrier synchronization. Every message sent in one superstep is available for the recipient only in the subsequent superstep. The BSP model made parallel computation *coarse grained* and led to a substantial reduction in synchronization overhead. The *Coarse-Grained Multicomputer* (*CGM*) model [1] added strictly coarse-grained communication. A CGM algorithm is a special case of a BSP algorithm where all communication operations of one superstep are performed in one, single, *h-relation* with $h \leq n/p$. For this paper, n refers to the problem size and p to the number of processors.

A comparison of the proceedings of the eminent conference in the field, the ACM Symposium on Parallel Algorithms and Architectures (SPAA), between the late eighties and the time from the mid nineties to today reveals a startling change in research focus. Today, the majority of research in parallel algorithms is within the coarse-grained, BSP style, domain. What is the problem with fine-grained computing? After all, any fine-grained algorithm with speedup S for n virtual processors yields, through simple *virtual processor* simulation, speedup $S/(n/p)$ for $p < n$ real processors. Hence, any parallel algorithm with linear speedup $S = n$ for n virtual processors yields linear speedup p for any given p processors. The virtual processor concept is even built into several parallel operating systems.

[1] School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6. dehne@scs.carleton.ca. http://www.scs.carleton.ca/~dehne.

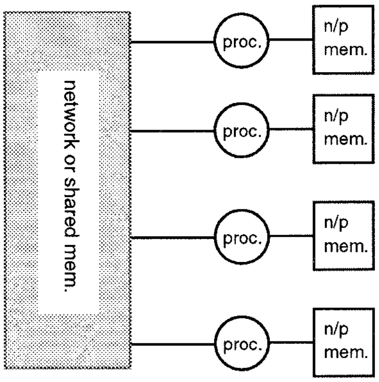

Fig. 1. Generic model of a parallel machine.

The major problem is, as usual, in the assumption. In many cases, fine-grained parallel algorithms do not achieve linear speedup. In particular, their actual implementations do not usually result in linear speedup. In fact, we often cannot even verify the latter, since we usually do not have n processors available. Furthermore, many situations exist where it is even theoretically impossible to obtain linear speedup for $p = n$. A simple example is the sorting problem for the mesh. For the following discussion, we refer to Figure 2 which shows the speedup S of a parallel algorithm as a function of the number p of processors available. The diagonal ($S = p$) represents linear speedup. The vertical line through $p = n$ represents fine-grained algorithms. Fine-grained algorithm design attempts to "push" the speedup as high as possible on this vertical line. Assume we obtain an optimal, but less than linear, speedup represented by point A. The straight line, B, from A to the origin represents the speedups obtained through virtual processor simulation. The main observation is that, even if A is optimal for $p = n$, the other points on B do not necessarily represent optimal speedups for those values of p. The curves labeled C and D represent, in many cases, the speedups obtained through BSP and CGM algorithms, respectively. The main idea is to study in detail the implications of $p < n$. Valiant calls it "slack." CGM algorithms are, in general, more efficient than BSP because they impose additional constraints on the communication pattern, eliminating small single messages. For $p = n$, curves C and D reach the same point A, but for $p < n$ they are usually much closer to the diagonal. For several problems, the shape of the entire *optimal curve* has by now been determined. In practice, n/p is usually rather large and corresponds to the size of the local memory at each processor. Hence, only the leftmost portions of curves C and D are often relevant in practice, where the difference to curve B can be very large.

As indicated above, recent interest in coarse-grained, BSP style, parallel algorithms has risen sharply, and many other arguments have been brought forward in favor of them. The most important one is probably that these algorithms, when implemented on currently available multiprocessors, do actually perform well and exhibit speedups similar to what was predicted in their analysis. It is important to note, though, that the

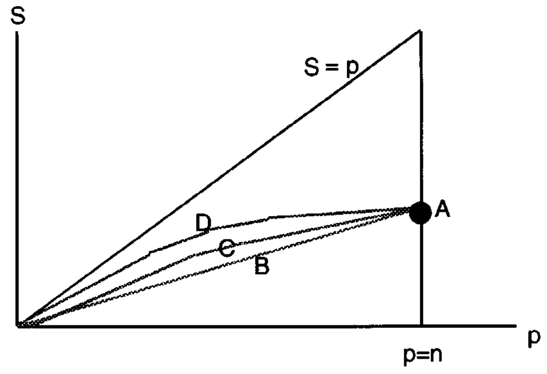

Fig. 2. Comparison of speedups for fine-grained, BSP, and CGM algorithms.

current state-of-the-art is clearly not yet the ultimate solution. It is only a first step in, what we hope is, the right direction. Many systems designers will observe immediately that some important aspects of today's commercial multiprocessors are still not accounted for. There are new results on including memory hierarchies [2]. Network caching is another important aspect that needs to be studied for this model. The main challenge is to model the impact of these effects without introducing too many parameters and too much detail that makes the model impossible to use.

I would now like to turn my attention to this special issue's contribution to the field of coarse-grained parallel algorithms. During recent years, a variety of coarse-grained, BSP style, parallel algorithms has been presented for various problems (see, e.g., [3] for a survey). The majority of the papers in this special issue continue this work by presenting algorithms for problems that had previously no coarse-grained parallel solutions or by improving on previous results. The main goal is to minimize the number of supersteps, the total amount of data communicated, and the amount of local computation. Another major goal for many of these papers is to obtain experimental "verification" of the theoretical results. As we have seen, translating theoretical results into "real" speedups is a nontrivial issue and needs to be constantly verified. In addition, there is the hope that we will gain insights that will help improve our model. The papers in this special issue deal with a variety of topics including string search (Ferragina and Luccio), computational geometry (Ferreira et al., Bäumker et al., Belloch et al., Deng and Zhu), matrix operations (McColl and Tiskin, Lim et al., Kaltofen and Lobo), permutation routing (Cormen and Clippinger), and parallel selection (Saukas and Song). Two papers study the modeling of bandwidth (Adler et al.) and the relationship between the BSP and LogP models (Bilardi et al.), respectively.

Coarse-grained parallel models have brought considerable progress to the parallel algorithms field but the current state-of-the-art is clearly in need of further research. This special issue intends to support that process. If it also convinces some new, bright, graduate students that this is a good field for their thesis research, then we have more than achieved our goal.

References

[1] F. Dehne, A. Fabri, and A. Rau-Chaplin, Scalable parallel geometric algorithms for coarse grained multicomputers, in *Proc*. *ACM* 9*th Annual Symposium on Computational Geometry*, pages 298–307, 1993

[2] F. Dehne, W. Dittrich, and D. Hutchinson, ‘Efficient external memory algorithms by simulating coarse grained parallel algorithms, in *Proc*. 9*th ACM Symposium on Parallel Algorithms and Architectures* (*SPAA* ’97), 1997, pages 106–115.

[3] S. Goetz, Coarse grained parallel algorithms, a survey, http://www.scs.carleton.ca/˜bsp.

[4] L. Valiant, A bridging model for parallel computation, *Communications of the ACM*, Vol. 33, No. 8, August 1994.