Algorithmica (2006) 45: 263–267
DOI: 10.1007/s00453-006-1213-2

Algorithmica

Guest Editor's Introduction

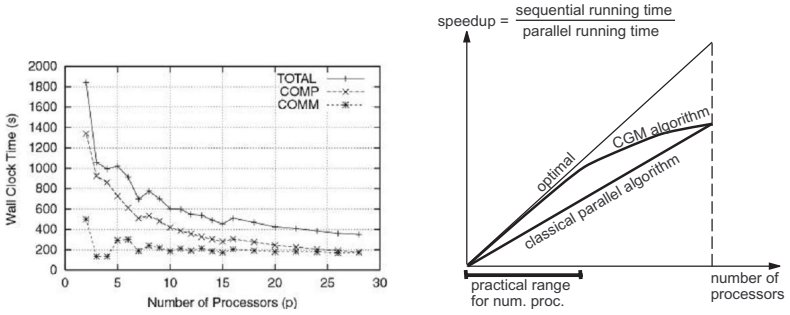
Frank Dehne[1]

Massive growth in data collections in diverse fields such as genomics, proteomics, particle physics and environmental monitoring has outstripped our present ability to analyze all this data. In addition to large data size, some data analysis processes, e.g. in genomics and proteomics, involve computationally very hard (e.g. NP-complete) problems. Research into *parallel computing* aims at creating enabling technology for solving such data intensive and computationally hard problems, thereby leading to new scientific discoveries (see, e.g. [1]).

The importance of parallel computing has been widely recognized. The history of parallel computers dates back to the ILLIAC IV used for the NASA space program in the seventies, followed by a long line of large and expensive supercomputers in the eighties and nineties built by companies like Cray, Thinking Machines, DEC, IBM, SGI and Sun. During recent years, a new type of very powerful but much less expensive parallel machines has emerged: the *processor cluster* built entirely from low-cost commodity hardware. It consists of p standard processors (typically rack mountable standard PCs) connected by a data switch (e.g. Gigabit Ethernet). The main advantage of processor clusters is that they have a vastly superior price performance ratio and that they are much more scalable. It is possible to build clusters with a very large number of processors, and the majority of the current *top* 500 supercomputers [12] are indeed processor clusters.

The better scalability of processor clusters is very important for *scientific applications* because it enables the solution of scientific problems of a scale that was previously unreachable, thereby leading to new scientific discoveries. The main disadvantage of processor clusters is that it is often much harder to design efficient *parallel algorithms* and *application software*. In processor clusters, computing and memory resources are distributed and much more loosely coupled than in traditional parallel machines. This leads to lower cost and better scalability but it also requires more sophisticated parallel algorithms and application software. An important rationale for many current installations of traditional supercomputers, despite the higher cost, is the fact that the targeted applications do have efficient parallel algorithms and software for traditional, shared memory, supercomputers but no efficient parallel solutions for processor clusters. This does not only lead to higher expenditures but it also prevents new science because the superior scalability of processor clusters cannot be exploited and important large-scale scientific computations remain impossible. In the popular literature, this lack of efficient parallel software for scalable systems is often called the "parallel software barrier".

[1] School of Computer Science, Carleton University, Ottawa, Ontario, Canada K1S 5B6. frank@dehne.net, http://www.dehne.net.



(a) (b)

Fig. 1. (a) Observed running time of a CGM algorithm for bipartite graph detection on an input graph with 10,000,000 nodes [2]. (b) Comparison of typical speedups: CGM algorithms and classical parallel algorithms.

The aim of the *Coarse-Grained Multicomputer* (CGM) model [9], [10] is to help break this barrier. The CGM model is a fully coarse-grained version of Valiant's BSP model [13] and addresses a fundamental and long open problem for parallel algorithm design: the need for a simple, practical and accurate model to analyze the performance of parallel algorithms. Accurate performance prediction is crucial for the development of efficient parallel software. In addition to analyzing parallel algorithm performance, the CGM model also gives guidance towards good parallel algorithm design. The CGM model is particularly well suited for cluster computing where message overheads are a big problem. The CGM model addresses this problem in a very effective way. Good CGM algorithms result in a fixed small number of large messages. This implies a fixed small communication overhead that shrinks proportionally (relative to total time) as data size increases. Figure 1(a) illustrates this effect for the problem of detecting whether a graph is bipartite [2]. The three curves shown represent the total time, local computation time (COMP) and communication time (COMM) of a CGM algorithm [11] executed on a Pentium-based LINUX cluster with Gigabit Ethernet interconnect. The important observation is that the curve for the communication time, as a function of the number of processors, is flat. For most algorithms based on previous models (e.g. the classical PRAM model or the more recent BSP [13]), the number of messages and resulting message overhead grow considerably with an increasing number of processors, with the consequence of considerable performance loss. CGM algorithms avoid such problems because they have only one compute thread per processor and a small fixed number of messages whose length grows with the increasing problem size, resulting in much improved performance and scalability. The typical effect on speedup is illustrated in Figure 1(b). In contrast to classical and BSP algorithms, the speedup for CGM algorithms is typically close to optimal when the number of processors is small relative to the number of data items in the input data set. For practical applications of high-performance computing, this is usually the case. The number of data items is typically many millions and the number of processors a few hundred or maybe thousands.

The CGM model has become popular with researchers who design and implement parallel algorithms for processor clusters. *Algorithmica* published a 1999 special issue on *Coarse-Grained Parallel Algorithms* [6]. The 2003 Conference on Computational

Science had an entire session dedicated to CGM algorithms. Most of the earlier work was on fundamental CGM algorithms and their implementation and testing on processor clusters. Our own group has been working, e.g. on CGM methods for solving large scale graph problems [11], [2] and Computational Geometry problems [9], [10] as well as parallel external memory methods [7]. More recently, the focus has shifted towards using the CGM paradigm for designing, implementing and analyzing large-scale parallel *applications* for processor clusters. Our group has been working, e.g. on parallel methods to build *data cubes* for *data warehousing* applications [8], resulting in the first parallel data cube prototype system that can create data cubes at a rate of more than 1TB per hour [4]. Another application includes a cluster-based parallel version of *CLUSTAL W* for large-scale gene and protein sequence analysis [3].

This special issue of Algorithmica is dedicated to CGM algorithms for large-scale *scientific applications*. As stated earlier, fields such as genomics, proteomics, particle physics and environmental monitoring are in need of scalable parallel solutions to overcome massive computational tasks that have created barriers to scientific progress. In particular, massive growth in scientific data collections has outstripped our present ability to analyze all this data and, in addition, fields such as computational biology require the solution of algorithmic problems with very high computational complexity (e.g. NP-complete) on these ever-increasing input data sets (see, e.g. [1]). Because of their superior scalability, processor clusters are important tools for solving such problems and the CGM paradigm provides a promising new approach for designing and implementing efficient parallel algorithms for processor clusters and help break the "parallel software barrier".

I now turn my attention to the papers published in this special issue. In response to the *Call for Papers*, 30 submissions were received of which 10 papers were selected for publication. These papers are grouped into three categories: *Bioinformatics*, *Large-Scale Data Analysis* and *Fundamental Methods*.

Bioinformatics: Abu-Khzam, Langston, Shanbhag and Symons study parallel problems that are fixed-parameter tractable (FPT). In particular, they study the NP-hard vertex cover problem and its applications in phylogeny and gene motif discovery. The paper presents outstanding performance results for their approach, using real-life data such as sh2 and sh3 domains. Due to the exponential nature of the problem, traditional methods (pre-FPT and parallel-FPT) were unable to handle input data with more than 100 data items, not even with months of computation. This paper reports on solving instances with more than 10,000 data items in a few hours. Keane, Page, Naughton, Travers and McInerney report on a fully cross platform coarse-grained distributed application for building large phylogenetic trees. Their new system overcomes many of the limitations imposed by the current set of parallel phylogenetic programs and it is now publicly available. The next two papers deal with gene and protein sequence comparison. Alves, Cáceres and Song present a new coarse-grained parallel algorithm for the all-substrings longest common subsequence operation which is used, e.g. to find approximate tandem repeats and the alignment of one sequence with several others that have a common subsequence. Driga, Lu, Schaeffer, Szafron, Charter and Parsons discuss parallel and sequential sequence alignment via a method called FastLSA. Experiments indicate that their method scales well and can be parameterized to take advantage of cache memory and main memory sizes.

Large-Scale Data Analysis: The second set of contributions deals with large-scale data analysis. The roots of this type of research are typically in the business domain, e.g. in data warehousing and online analytical processing. Here, the goal is to analyze very large databases of historical corporate data for business intelligence applications. The main research questions deal with the huge size and high dimensionality of the data and how to extract meaningful new information. With ever increasing scientific databases, similar problems arise in scientific data analysis. For example, very large and high-dimensional data sets need to be examined for environmental data analysis with the goal of determining past successes/failures in environmental management and predicting future trends. Typical operations include data summaries, data correlation, feature extraction and view extraction which are the topics of the next four papers. Chi, Koyutürk and Grama present a coarse-grained parallel algorithm for constructing summaries of distributed data sets. Chilson, Ng, Wagner and Zamar study the parallel computation of high-dimensional robust correlation and covariance matrices. Souza, Matwin and Japkowicz discuss how parallelism can be used to improve the performance of feature selection algorithms for data classification and present a coarse-grained parallel version of the feature selection algorithm FortalFS. Bhatt, Flahive, Wouters, Rahayu and Taniar study large-scale web ontologies and present a coarse-grained (data) parallel method for materialized ontology view extraction.

Fundamental Methods: The final two papers improve on fundamental methods that are part of many coarse-grained parallel algorithms for scientific applications: sorting and load partitioning/balancing. Chaudhry and Cormen introduce Slabpose Columnsort, a new oblivious algorithm for out-of-core sorting on distributed-memory clusters. Harvey, Das and Biswas present a new workload partitioner that dynamically balances processor workloads while minimizing data movement and runtime communication.

While the early years of research on coarse-grained parallel algorithms were mainly on fundamental methods, we have progressed to a state where sufficient basic methodology is available to support a shift of emphasis towards the study of large-scale applications based on coarse-grained parallel methods. Here, we focus on *scientific applications*. As shown in this special issue and other publications, coarse-grained parallel algorithms are a useful framework for developing new cluster-based high-performance solutions for scientific problems. As stated at the beginning, the goal is to overcome the parallel software barrier and enable new scientific discoveries.

References

[1] D.A. Bader. Computational biology and high-performance computing. *Communications of the ACM*, 47(11):35–41, 2004.

[2] A. Chan, F. Dehne, and R. Taylor. Implementing and testing cgm graph algorithms on pc clusters and shared memory machines. *International Journal of High Performance Computing Applications*, 19(1):81–97, 2005.

[3] J. Cheetham, F. Dehne, A. Rau-Chaplin, U. Stege, and P. J. Taillon. Solving large fpt problems on coarse-grained parallel machines. *Journal of Computer and System Sciences*, 67(4):691–706, 2003.

[4] Y. Chen, F. Dehne, T. Eavis, D. Green, A. Rau-Chaplin, and E. Sithirasenan. cgmOLAP: Efficient parallel generation and querying of terabyte size rolap data cubes. In *Proc*. 22*nd International Conference on Data Engineering* (*ICDE*), Atlanta, GA, 2006, to appear.

[5] Y. Chen, F. Dehne, T. Eavis, and A. Rau-Chaplin. Parallel rolap data cube construction on shared-nothing multiprocessors. *Distributed and Parallel Databases*, 15:219–236, 2005.

[6] F. Dehne (Guest Editor). Special issue on coarse-grained parallel algorithms. *Algorithmica*, 24(3/4), 1999.

[7] F. Dehne, W. Dittrich, D. Hutchinson, and A. Maheshwari. Bulk synchronous parallel algorithms for the external memory model. *Theory of Computing Systems*, 35(6):567–598, 2002.

[8] F. Dehne, T. Eavis, S. Hambrusch, and A. Rau-Chaplin. Parallelizing the data cube. *Distributed and Parallel Databases*, 11(2):181–201, 2002.

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

[10] F. Dehne, A. Fabri, and A. Rau-Chaplin. Scalable parallel computational geometry for coarse grained multicomputers. *International Journal on Computational Geometry*, 6(3):379–400, 1996.

[11] F. Dehne, A. Ferreira, E. Caceres, S.W. Song, and A. Roncato. Efficient parallel graph algorithms for coarse-grained multicomputers and BSP. *Algorithmica*, 33(2):183–200, 2002.

[12] H. Meuer, E.H Strohmaier, J. Dongarra, and H.D. Simon. *Top* 500 *Supercomputer Sites*. http://www.top500.org/.

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