Engineering Algorithms for Large Data Sets

Peter Sanders

Karlsruhe Institute of Technology
sanders@kit.edu

Abstract. For many applications, the data sets to be processed grow much faster than can be handled with the traditionally available algorithms. We therefore have to come up with new, dramatically more scalable approaches. In order to do that, we have to bring together know-how from the application, from traditional algorithm theory, and on low level aspects like parallelism, memory hierarchies, energy efficiency, and fault tolerance. The methodology of algorithm engineering with its emphasis on realistic models and its cycle of design, analysis, implementation, and experimental evaluation can serve as a glue between these requirements. This paper outlines the general challenges and gives examples from my work like sorting, full text indexing, graph algorithms, and database engines.

1 Algorithms for Large Data Sets

Application data sets from various sources have grown much faster than the available computational resources which are still governed my Moore's law but increasingly hit physical limitations like clock frequency, energy consumption, and reliability. To name just a few applications, one can mention sensor data from particle colliders like LHC at CERN, the world wide web, sequenced genome data – ultimately from most human individuals, or GPS traces from millions and millions of smart phone users that can yield valuable information, e.g., on the current traffic situation.

Large data sets are a fascinating topic for computer science in general and for algorithmics in particular. On the one hand, the applications can have enormous effects for our daily life, on the other hand they are a big challenge for research and engineering. The main difficulty is that a successful solution has to take into account issues from three quite different area of expertise: The particular application at hand, technological challenges, and the "traditional" areas of computer science know-how. I will focus on the algorithmic aspects here which will often be particularly interesting. Figure 1 illustrates this triangle of challenges. The problem is that, traditionally, individual persons and even teams are mostly proficient in only one of these three areas. The solution of this problem will have to bridge the gaps between the areas in several different ways. We probably have to stress interdisciplinary aspect of university education and we have to integrate technological aspects into the main stream of computer science teaching and research. For example, current research in algorithmics is still predominantly

P. van Emde Boas et al. (Eds.): SOFSEM 2013, LNCS 7741, pp. 29–32, 2013.

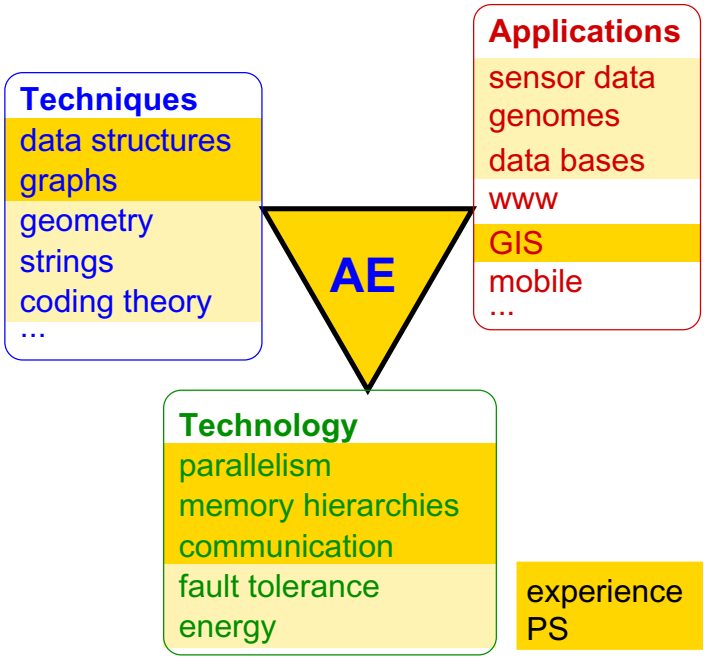

Fig. 1. Engineering algorithms for large data sets

using the von Neumann/RAM model with a sequential processor and homogeneous memory. In contrast, even non-large data applications need to take into account at least parallel processing and memory hierarchies if the application is in any way performance critical. Traditionalists might argue that the problems are already sufficiently challenging and difficult in the von Neumann model but obviously this is an argument from the ivory tower.

Algorithm engineering with its emphasis on realistic models and its cycle of design, analysis, implementation, and experimental evaluation can serve as a glue between these requirements. The talk will focus on examples from my work. This abstract mostly gives a few pointers to already existing papers. Much of the material presented in the talk is work in progress however including promising results in main-memory data bases, track reconstruction at CERN, genome sequencing, and phylogenetic tree reconstruction.

2 Examples from My Work

We have done a lot of work on sorting in some sense cumulating in a parallel external algorithm that won the sorting benchmark competition GraySort for large data sets in 2009 [9]. In contrast to previous codes, this algorithm also works for worst case inputs. This benchmark from the data base community

asks for the fastest machine/code sorting 100 Terabytes of 100 byte records on a file system. Interestingly, a similar code worked well for the category of the most energy efficient sorter JouleSort [2].

A more complex sorting problem asks for sorting all suffixes of a string. The resulting suffix array can be used for full text search, data compression and various applications in bioinformatics. Our simple linear time algorithm [6] has the additional advantage of being parallelizable [7] and externalizable [4].

An even more basic service is management of data on disk arrays. One interesting result is that parallel disks can emulate a single high capacity, high throughput logical disk allowing parallel access by using random redundant allocation of data: Any set of N requested data blocks can be retrieved from D disks in just $\lceil N/D \rceil + 1$ parallel I/O steps [13] (often, even the $+1$ can be dropped). Refinements for asynchronous access [11], variable block sizes [10], fault tolerance, heterogeneity [12] and many other issues are possible.

When processing large graphs, they have to be partitioned between many processors such that the interactions (e.g., number of cut edges) are small. We are intensively working on parallel and high quality graph partitioning algorithms for large graphs. Multilevel methods are the method of choice here since they combine near linear work with high quality [8].

With modern speedup techniques [3,5] for route planning in road networks, routing in continent sized networks can be done in the submillisecond range on a server and still without perceptible delay on a mobile device. However, for advanced applications we would like to integrate public transportation, historical congestion information, and real time information on the traffic situation. Currently we can handle historical information modelled as piece-wise linear functions [1] and a small number of traffic jams.

Acknowledgements. Partially supported by Helmholtz project Large-Scale Data Management and Analysis (LSDMA). We also would like to thank the DFG for several funding sources.

References

1. Batz, G.V., Geisberger, R., Neubauer, S., Sanders, P.: Time-Dependent Contraction Hierarchies and Approximation. In: Festa, P. (ed.) SEA 2010. LNCS, vol. 6049, pp. 166–177. Springer, Heidelberg (2010)
2. Beckmann, A., Meyer, Sanders, P., Singler, J.: Energy-efficient sorting using solid state disks. In: 1st International Green Computing Conference, pp. 191–202. IEEE (2010)
3. Delling, D., Sanders, P., Schultes, D., Wagner, D.: Engineering Route Planning Algorithms. In: Lerner, J., Wagner, D., Zweig, K.A. (eds.) Algorithmics. LNCS, vol. 5515, pp. 117–139. Springer, Heidelberg (2009)
4. Dementiev, R., Kärkkäinen, J., Mehnert, J., Sanders, P.: Better external memory suffix array construction. Special issue on Alenex 2005. ACM Journal of Experimental Algorithmics 12 (2008)

5. Geisberger, R., Sanders, P., Schultes, D., Vetter, C.: Exact routing in large road networks using contraction hierarchies. Transportation Science (2012)
6. Kärkkäinen, J., Sanders, P., Burkhardt, S.: Linear work suffix array construction. Journal of the ACM 53(6), 1–19 (2006)
7. Kulla, F., Sanders, P.: Scalable Parallel Suffix Array Construction. In: Mohr, B., Träff, J.L., Worringen, J., Dongarra, J. (eds.) PVM/MPI 2006. LNCS, vol. 4192, pp. 22–29. Springer, Heidelberg (2006); Parallel Computing 33, 605–612 (2007)
8. Osipov, V., Sanders, P., Schulz, C.: Engineering Graph Partitioning Algorithms. In: Klasing, R. (ed.) SEA 2012. LNCS, vol. 7276, pp. 18–26. Springer, Heidelberg (2012)
9. Rahn, M., Sanders, P., Singler, J.: Scalable distributed-memory external sorting. In: 26th IEEE International Conference on Data Engineering, pp. 685–688 (2010)
10. Sanders, P.: Reconciling simplicity and realism in parallel disk models. Special Issue on Parallel Data Intensive Algorithms and Applications. Parallel Computing 28(5), 705–723 (2002)
11. Sanders, P.: Asynchronous scheduling of redundant disk arrays. IEEE Transactions on Computers 52(9), 1170–1184 (2003); Short version in 12th ACM Symposium on Parallel Algorithms and Architectures, pp. 89–98 (2000)
12. Sanders, P.: Algorithms for Scalable Storage Servers. In: Van Emde Boas, P., Pokorný, J., Bieliková, M., Štuller, J. (eds.) SOFSEM 2004. LNCS, vol. 2932, pp. 82–101. Springer, Heidelberg (2004)
13. Sanders, P., Egner, S., Korst, J.: Fast concurrent access to parallel disks. Algorithmica 35(1), 21–55 (2003); Short version in 11th SODA, pp. 849–858 (2000)