

New Paradigms in Data Structure Design: Word-Level Parallelism and Self-Adjustment

EPSRC Grant GR/L 92150: Final Report

Rajeev Raman
Department of Mathematics and Computer Science
University of Leicester

Personnel

PI: Rajeev Raman (RR); co-investigator: Tomasz Radzik (TR).
PDRA: Maureen Korda (MK), Naila Rahman (NR) and S. Srinivasa Rao (SSR).
PGRA: Yoan Jose Pinzon (YP).

Background

Ways of representing and efficiently manipulating data are central to many computer programs. Over the years, computer scientists have studied many different data structures for representing and manipulating data within computer main and secondary memory. These investigations have shown, both in practice and in theory, that the choice of data structures often has a considerable effect on algorithmic performance.

Perhaps the single most fundamental class of data structuring problems broadly involve maintaining or manipulating a (possibly changing) set S of keys (which are linearly ordered). These include basic problems such as *sorting*, *searching* and *priority queues*. Most BSc courses in Computer Science teach classical solutions to these problems such as quicksort, radix sort, balanced search trees and binary heaps. However, there is now a compelling reason to re-assess the performance of these nearly 40-year-old solutions—the divergence between the computational models on which these solutions were developed, and real-life CPU architectures. This divergence can cause asymptotic analysis—the foundation of modern algorithm theory—to give incorrect performance predictions, even for very large problem instances.

Two major causes of divergence are *Word-Level Parallelism* and *Hierarchical Memory*. The former refers to the increasing word-size of modern CPUs, and the increasingly sophisticated operations that can be performed on a word in ‘one’ time step, which are reminiscent of those available in vector parallel (super)computers of the late 20th century. This renders the ‘comparison’ model obsolete for many common applications that deal with integer or floating-point keys. The latter refers to the large and increasing gulf between CPU and memory speeds, which requires one to adapt external-memory techniques even for problem instances that fit into main memory. Another strand of the project dealt with the use of *self-adjusting* data structures in improving the practical performance of graph algorithms.

In the late 90s there were a number of significant advances in the theory of algorithms that used WLP to improve upon the time complexity of classical comparison-based algorithms for problems

such as sorting, searching and priority queues [1, 21]. A few researchers, particularly Yijie Han (Kansas) and Mikkel Thorup (AT&T), have continued to develop (increasingly intricate) algorithms with better time complexity bounds (see e.g. [7]). From a practical perspective, the advent of CPUs with large word-sizes and powerful instructions to operate upon these words opened up the tantalising prospect of extremely fast practical solutions to these problems based on WLP.

The WLP-based algorithms are based on the RAM (random-access machine) model. A key assumption of the RAM model is that accessing a memory location takes as much time as an arithmetic operation. LaMarca and Ladner [10] noticed that the growing difference between memory and CPU speeds meant that a lot of highly-optimised code for sorting and searching (developed by e.g. Knuth and Sedgewick) now performed relatively poorly. There has now been a much more thorough mathematical analysis of this phenomenon—to which this project has contributed substantially—and how it relates to the theory of external-memory algorithms [23].

From the practical perspective, we have succeeded in developing very high-performance practical sorting and searching code, some based on WLP and some on our improved understanding of the memory hierarchy in modern CPUs. Other high-performance software (e.g. numerical libraries) uses *both*. An important future direction is the development of algorithms that use WLP to minimise operation counts and external-memory techniques to reduce overheads in the memory hierarchy.

There were a number of other contributions of this project, which we briefly mention, but whose context we omit for lack of space. WLP is extremely useful in *succinct* data structures (see e.g. [11]), which are very frugal in their space consumption, but permit operations to be executed rapidly (roughly, WLP is used to operate on data that has been tightly packed into computer words). WLP is also very useful for string pattern-matching provided that the size of the pattern (in characters) is roughly of the order of the number of bits in a computer word (see [12] for example). This project has contributed to the growing body of knowledge in this area. Finally, the work on self-adjustment has led to the development, implementation and experimental evaluation of new dynamic tree data structures, extending Sleator and Tarjan’s seminal work [22].

Key Advances and Supporting Methodology

In roughly chronological order the main themes of the project were as follows:

1. (Empirical Evaluation of WLP for Searching Algorithms) At the start of the project MK and RR began an investigation of Willard’s trie [9]. Excellent (sometimes over 2x) speedups over red-black trees were demonstrated in a comprehensive test. The issue of how to create generic template code (using C++) when the data structure may inspect the representation of the keys was addressed. This is a key step in enabling the reuse of the same code for integer or floating-point keys, for example.
2. (Empirical Evaluation of WLP for Sorting Algorithms) Concurrently, NR (self-funded at the time) and RR developed a simulator for large word-lengths and a highly-optimised implementation of a sorting algorithm by Kirkpatrick and Reisch [15]. Although the implementation was very efficient in terms of number of instructions, it failed to out-perform radix sort in real life, and radix sort in turn was out-performed by Quicksort. This was contrary to predictions made by asymptotic analysis.
3. (Design and Empirical Evaluation of Memory Hierarchy-Aware Sorting Algorithms) The unexpected results from (2) led NR and RR to investigate further the cache behaviour of radix

sort and other distribution sorting algorithms. Building on the recent work of [10], a number of papers [16, 17, 18] were published on modelling and analysing the cache behaviour of sorting algorithms. Key contributions include: A detailed approximate analysis of conflict misses in distribution sorting [16], extending work of [10]; an analysis of cache misses in non-uniform distribution sorting algorithms [17]; and, demonstrating performance gains by adding the Translation-Lookaside Buffer (TLB) to the cache model of [10].

4. (Design and Empirical Evaluation of Dynamic Tree Data Structures) Concurrently with (1)-(3) TR gave an implementation of a dynamic tree data structure based on self-adjusting data structures [13]. This work extended the functionality of the original dynamic tree data structure of Sleator and Tarjan [22]. The code was later incorporated into the LEDA software package.
5. (Design and Empirical Evaluation of Cache-Oblivious Data Structures) NR and RR, together with Richard Cole (New York University) and Michael Bender (SUNY Stony Brook) investigated *cache-oblivious* data structures for predecessor searching and other problems. The practical studies showed that cache-oblivious data structures performed very well [14], and it was noted that this was likely due to an improved TLB performance. In addition [3] gave cache-oblivious algorithms for persistent data structures and planar point location.
6. (Design of Succinct Data Structures) Concurrently with (3), RR investigated *succinct* data structures in collaboration with SSR and Venkatesh Raman (IMSc, Chennai), as well as Erik Demaine (Waterloo/MIT) and Ian Munro (Waterloo), leading to a number of publications [2, 19, 20]. Several other papers are in the pipeline. Succinct data structures use essentially the information-theoretic minimum number of bits to represent their data, but also support operations as rapidly as their non-succinct counterparts. WLP is used extensively in succinct data structures, to enable rapid ‘unpacking’ of tightly packed data. Key contributions include the use of WLP techniques developed previously by RR to solving an open problem of Benoit et al. [2]; and the development of a succinct indexable dictionary, together with succinct representations of prefix sums [20]. The latter has already been used for succinct compressed indices [5].
7. (Design and Empirical Evaluation of WLP-based String Algorithms) YP investigated WLP-based algorithms in the context of string matching. Together with Maxime Crochemore (Marne-la-Vallée) and Lauren Mouchard (Rouen) and Costas Iliopoulos (King’s College London), YP gave improved algorithms for approximate string matching and the LCS problem [8, 4].
8. (Miscellaneous) Together with Torben Hagerup (Frankfurt), WLP was also applied to solving the *quasidictionary* problem [6]. This problem arises when an application needs to maintain pointers to ‘items’ stored within a data structure.

Project Plan Review

The project plan was followed closely for the first year or so, when it was discovered that poor data cache performance can outweigh the gains made by WLP. This effect is slightly less evident for searching than for sorting, so MK continued to work on WLP implementations of searching data structures. After the departure of MK (for a private industry R&D position), RR continued exploring data cache issues with NY.

We did not advertise for MK's replacement as we planned to use the funds to extend the appointment of Dr Yang, who was employed on GR L/81618 (PI: TR). Dr Yang's experience with network flows would have been invaluable for the project. Unfortunately Dr Yang left to take up a lectureship. At this point it was decided to use the remaining funds to support a number of shorter-term appointments, as we knew of people who were capable of and interested in contributing to the project. These included NY, YP and SSR. As NY had already contributed to the project, YP had some prior experience of WLP-based algorithms for string matching and RR and SSR had collaborated previously on succinct data structures, no time was wasted in "learning the ropes."

Research Impact and Benefits to Society

Research Impact

The results of the project were disseminated in a number of high-profile venues. For example:

- RR gave a plenary talk on Exploiting Word-Level Parallelism at the 9th Australasian Workshop on Combinatorial Algorithms (AWOCA '99), Perth, Australia.
- RR co-organised *Advances in Data Structures*, a pre-conference workshop associated with the 19th FST&TCS Conference in Chennai, India, in December 1999, with Ramesh Hariharan. Speakers at the workshop included Sunil Arya (Hong Kong), Gerth Brodal (BRICS, Aarhus), Paolo Ferragina (Pisa), Peter Bro Miltersen (BRICS, Aarhus), Ian Munro (Waterloo), Torben Hagerup (Frankfurt) and Giuseppe Italiano (Rome); see <http://www.imsc.ernet.in/fsttcs99/data.html> for details. RR spoke on [16] at this workshop.
- RR spoke on [16, 17] at a meeting on Efficient Algorithms at MFI, Oberwolfach, in August 2000. See http://www.mfo.de/Meetings/Meeting_Program_2000.html#T0032 for details.
- RR spoke on [16, 17, 18] at a workshop ("Festsymposium") to commemorate the 10th anniversary of the founding of the Max-Planck-Institute for Computer Science at Saarbrücken in December 2000.
- RR spoke on [20] at a workshop on Data Structures at IBFI, Schloß Dagstuhl in February 2002. See <http://www.dagstuhl.de/02091/Report/> for details.
- NR gave a presentation on hardware caches at a workshop on External Memory Algorithms at IBFI, Schloß Dagstuhl, in March 2002. See <http://www.mpi-sb.mpg.de/~sanders/gisem> for details.

Seminars on this work were also given at Goldsmiths, Liverpool, Warwick and Swansea; at London Algorithms Day (1998), UKCRC Algorithms and Complexity Day at Warwick and at BCTCS (2000, 2001). Thus, the work was widely publicised.

Some publications were selected to special issues of journals:

- [13] appeared in a special issue the *ACM J. Experimental Algorithmics*, with Giuseppe Italiano and Andrew Goldberg as guest editors, devoted to some of the best papers from the First Workshop on Algorithm Engineering (WAE '97).
- [16] appeared in a special issue of the *ACM J. Experimental Algorithmics*, with Jeff Vitter and Christos Zaroliagis as guest editors, devoted to some of the best papers from the third Workshop on Algorithm Engineering (WAE '99).

- [18] is to appear in a special issue of *ACM J. Experimental Algorithmics*, with Bernard Moret and Andrew Goldberg as guest editors, devoted to some of the best papers from the 2nd Workshop on Algorithm Engineering and Experiments (ALENEX '00).
- [20] was submitted by invitation to a special issue of *J. Algorithms*, with David Eppstein as the guest editor, devoted to some of the best papers from the 13th ACM-SIAM Symposium on Discrete Algorithms (SODA '02).

One of the aims of the project was to raise the profile of Experimental Algorithmics in the UK. Towards this end, TR and RR organised the *3rd Workshop on Algorithm Engineering (WAE '99)*, in London from July 19-21, 1999. See <http://www.dcs.kcl.ac.uk/events/wae99>.

Benefits to Society

Our work has laid the theoretical and empirical basis for the construction of a set of standard libraries for C++, such as STL, that would be significantly more efficient than existing ones. Indeed, our research (along with those of others in the same direction) has led to changes being implemented in libraries such as LEDA. For example, the standard 'search tree' in LEDA is based on (cache-efficient) B-trees rather than skip lists, as a result of the increased awareness of the importance of the cache.

The recent International Review of Computer Science Research in the UK mentions that algorithms is a "key enabling technology" that promises substantial benefits to the competitiveness of the UK in key fields "including information retrieval, medical applications of computing, scientific computing, and so on". Realising these benefits is to a greater or lesser degree predicated on a strong culture of empirical work in algorithms, as this is now an accepted prerequisite for the successful transfer of algorithmic ideas to technology. This project has helped to substantially raise the UK's profile and strength in empirical algorithmics.

Explanation of Expenditure

The following research staff were employed by the contract as Post-Doctoral RAs¹: MK (18 months), NR (12 months 50% P/T plus 5 months F/T) and SSR (6 months).

In addition, YP was employed for 6 months (50% P/T) as a Post-Graduate RA, bringing the total amount of research staff time to 38 months FTE. This was not quite affordable under the original budget, but was covered by viring just under 20% of the Travel and Subsistence budget to Staff costs.

Further Research

It is planned to utilise NR's and RR's expertise in data cache optimisations to speed up several particle simulation algorithms used by the Algorithmic Dynamics group at Leicester. RR's expertise with succinct data structures is to be applied (along with a PhD student) to the problem of compact in-memory XML representations, with particular reference to manipulating astronomical VOTable data. As already mentioned, a number of theoretical papers on succinct data structures are in the pipeline.

¹NR and SSR submitted their PhD theses in mid-2002.

REFERENCES

- [1] Arne Andersson, Torben Hagerup, Stefan Nilsson, Rajeev Raman: Sorting in Linear Time? *JCSS* 57(1): 74-93 (1998). *STOC '95* special issue.
- [2] David Benoit, Erik D. Demaine, J. Ian Munro, Rajeev Raman, Venkatesh Raman and S. Srinivasa Rao, Representing Trees of Higher Degree, TR 2001/46, University of Leicester (2001).
- [3] Michael A. Bender, Richard Cole, Rajeev Raman: Exponential Structures for Efficient Cache-Oblivious Algorithms. *Proc. ICALP '02*, LNCS 2380, pp. 195-207 (2002).
- [4] Maxime Crochemore, Costas S. Iliopoulos, Yoan J. Pinzon: Speeding-up Hirschberg and Hunt-Szymanski LCS Algorithms. *Proc. SPIRE 2001*, 59-67.
- [5] Ankur Gupta, Roberto Grossi and Jeff Vitter: High-Order Entropy-Compressed Text Indices. To appear in *SODA 2003*.
- [6] Torben Hagerup, Rajeev Raman: An Efficient Quasidictionary. *Proc 8th SWAT*, LNCS 2368, 1-18 (2002).
- [7] Yijie Han: Deterministic sorting in $O(n \log \log n)$ time and linear space. *Proc 34th ACM STOC*, 602-608 (2002).
- [8] Costas S. Iliopoulos, Laurent Mouchard, Yoan J. Pinzon: The Max-Shift Algorithm for Approximate String Matching. *Proc 4th WAE*, LNCS 2141, pp. 13-25 (2001).
- [9] Maureen Korda, Rajeev Raman: An Experimental Evaluation of Hybrid Data Structures for Searching. *Proc 2nd WAE*, LNCS 1668, pp. 213-227 (1999).
- [10] Anthony LaMarca, Richard E. Ladner: The Influence of Caches on the Performance of Sorting. *J. Algorithms* 31(1): 66-104 (1999).
- [11] J. Ian Munro, Venkatesh Raman: Succinct Representation of Balanced Parentheses, Static Trees and Planar Graphs. *Proc. 38th IEEE FOCS*, 118-126 (1997). To appear in *SIAM J. Computing*.
- [12] Gene Myers: A Fast Bit-Vector Algorithm for Approximate String Matching Based on Dynamic Programming. *JACM*, 46(3): 395-415 (1999).
- [13] Tomasz Radzik: Implementation of Dynamic Trees with In-Subtree Operations. *ACM Journal of Experimental Algorithmics* 3: Article 9 (1998).
- [14] Naila Rahman, Richard Cole, Rajeev Raman: Optimised Predecessor Data Structures for Internal Memory. *Proc. 4th WAE*, LNCS 2141, pp. 67-78 (2001).
- [15] Naila Rahman, Rajeev Raman: An Experimental Study of Word-level Parallelism in Some Sorting Algorithms. *Proc. 2nd Workshop on Algorithm Engineering*, 193-203 (1998).
- [16] Naila Rahman, Rajeev Raman: Analysing Cache Effects in Distribution Sorting, *ACM J. Experimental Algorithmics* 5: Article 14 (2000).
- [17] Naila Rahman, Rajeev Raman: Analysing the Cache Behaviour of Non-uniform Distribution Sorting Algorithms. *Proc ESA 2000*, LNCS 1879, pp. 380-391 (2000).
- [18] Naila Rahman, Rajeev Raman: Adapting Radix Sort to the Memory Hierarchy. TR 2000/02, King's College London (2000). Preliminary version in *Proc. 2nd Workshop on Algorithms and Experiments (ALENEX 2000)*.
- [19] Rajeev Raman, Venkatesh Raman, S. Srinivasa Rao: Succinct Dynamic Data Structures. *Proc WADS 2001*, LNCS 2125: 426-437.
- [20] Rajeev Raman, Venkatesh Raman, S. Srinivasa Rao: Succinct indexable dictionaries with applications to encoding k-ary trees and multisets. *Proc 13th SODA*: 233-242 (2002).
- [21] Mikkel Thorup, Undirected Single-Source Shortest Paths with Positive Integer Weights in Linear Time. *JACM* 46(3): 362-394 (1999).
- [22] Daniel Dominic Sleator, Robert Endre Tarjan: A Data Structure for Dynamic Trees. *JCSS* 26(3): 362-391 (1983).
- [23] Jeff Vitter: External memory algorithms and data structures: dealing with MASSIVE DATA. *ACM Computing Surveys* 33(2): 209-271 (2001).
- [24] Dan E. Willard: New Trie Data Structures Which Support Very Fast Search Operations. *JCSS* 28(3): 379-394 (1984).