04301 Abstracts Collection
Cache-Oblivious and Cache-Aware Algorithms
— Dagstuhl Seminar —

Lars Arge[1], Michael A. Bender[2], Erik Demaine[3], Charles Leiserson[3] and Kurt Mehlhorn[4]

[1] Duke University, Durham, USA
large@daimi.av.dk
[2] SUNY at Stony Brook, USA
bender@cs.sunysb.edu
[3] MIT Cambridge, USA
edemaine|cel@mit.edu
[4] MPI Saarbrücken, DE
mehlhorn@mpi-sb.mpg.de

Abstract. The Dagstuhl Seminar 04301 "Cache-Oblivious and Cache-Aware Algorithms" was held in the International Conference and Research Center (IBFI), Schloss Dagstuhl, from 18.07.2004 to 23.07.2004. During the seminar, several participants presented their current research, and ongoing work and open problems were discussed. Abstracts of the presentations given during the seminar as well as abstracts of seminar results and ideas are put together in this paper. The first section describes the seminar topics and goals in general. Links to extended abstracts or full papers are provided, if available.

Keywords. Cache oblivious, cache aware, external memory, I/O-efficient algorithms, data structures

Efficient Tree Layout in a Multilevel Memory Hierarchy

Michael A. Bender (SUNY at Stony Brook)

We consider the problem of laying out a tree with fixed parent/child structure in hierarchical memory. The goal is to minimize the expected number of block transfers performed during a search along a root-to-leaf path, subject to a given probability distribution on the leaves. This problem was previously considered by Gil and Itai, who developed optimal but slow algorithms when the block-transfer size B is known. We present faster but approximate algorithms for the same problem; the fastest such algorithm runs in linear time and produces a solution that is within an additive constant of optimal.

In addition, we show how to extend any approximately optimal algorithm to the cache-oblivious setting in which the block-transfer size is unknown to

the algorithm. The query performance of the cache-oblivious layout is within a constant factor of the query performance of the optimal known-block-size layout. Computing the cache-oblivious layout requires only logarithmically many calls to the layout algorithm for known block size; in particular, the cache-oblivious layout can be computed in $O(N \log N)$ time, where N is the number of nodes.

Finally, we analyze two greedy strategies, and show that they have a performance ratio between $\Omega(\log B / \log \log B)$ and $O(\log B)$ when compared to the optimal layout.

Keywords: Tree layout; unbalanced trees; cache-oblivious; data structure; approximation algorithm

Joint work of: Alstrup, Stephen; Bender, Michael A.; Demaine, Erik D.; Farach-Colton, Martin; Rauhe, Theis; Thorup, Mikkel

Pipelining the Memory Hierarchy

Gianfranco Bilardi (Universitá di Padova)

Pipelining the memory hierarchy is a natural avenue to explore in order to reduce the impact of memory latencies on running time. In this talk we consider the following issues:

(a) the physical feasibility of memory structures that afford high pipelinability without sacrifying latency of individual accesses; (b) the impact of pipelining on algorithmic running time; (c) the issues that an agressively pipelined memory raises for processor design; (d) open problems and research directions.

Keywords: Memory pipeline, memory models, processor models, physical scalability, algorithmic models

Joint work of: Bilardi, Gianfranco; Kattamuri Ekanadham; Pratap Pattnaik

External-Memory Exact and Approximate All-Pairs Shortest-Paths

Rezaul Chowdhury Alam (Univ. of Texas at Austin)

We present several new results for finding all pairs shortest paths in a V-node, E-edge undirected graph. We present a cache-oblivious algorithm for computing AP-BFS on undirected graphs in $O(V.sort(E))$ I/Os matching the I/O complexity of its cache-aware counterpart. This algorithm can also be used to compute the unweighted diameter of an undirected graph in the same I/O bound and $O(V + E)$ space. We also present an efficient cache-aware algorithm to compute approximate APSP on unweighted undirected graphs with small additive error. The algorithm produces estimated distances with an additive error of at most $2(k-1)$, where $2 <= k <= logV$ is an integer, and $E >= VlogV$. All of

our results improve earlier results. For approximate APSP we provide the first nontrivial results.

These results are included in a paper that will be presented at the 2005 ACM-SIAM Symposium on Discrete Algorithms (SODA). That paper also includes results on weighted APSP that are described in the abstract of the Dagstuhl talk by the second author.

Keywords: All pairs shortest paths; external memory; cache oblivious; cache aware; approximation algorithm; I/O-efficient algorithms

Joint work of: Vijaya Ramachandran

Cache-Oblivious Data Structures for Orthogonal Range Searching

Andrew Danner (Duke University)

We develop cache-oblivious data structures for orthogonal range searching, the problem of finding all T points in a set of N points in R^d lying in a query hyper-rectangle. Cache-oblivious data structures are designed to be efficient in arbitrary memory hierarchies.

We describe a dynamic linear-size data structure that answers d-dimensional queries in $O((N/B)^{1-1/d} + T/B)$ memory transfers, where B is the block size of any two levels of a multilevel memory hierarchy. A point can be inserted into or deleted from this data structure in $O(log_B^2 N)$ memory transfers. We also develop a static structure for the two-dimensional case that answers queries in $O(log_B N + T/B)$ memory transfers using $O(N log_2^2 N)$ space. The analysis of the latter structure requires that $B = 2^{2^c}$ for some non-negative integer constant c.

Keywords: cache oblivious; data structure; range query

Joint work of: Agarwal, Pankaj; Arge, Lars; Danner, Andrew; Holland-Minkley, Bryan

Asynchronous Parallel Disk Sorting

Roman Dementiev (MPI für Informatik)

We develop an algorithm for parallel disk sorting, whose I/O cost approaches the lower bound and that guarantees almost perfect overlap between I/O and computation.

Previous algorithms have either suboptimal I/O volume or cannot guarantee that I/O and computations can always be overlapped.

We give an efficient implementation that can (at least) compete with the best practical implementations but gives additional performance guarantees.

For the experiments we have configured a state of the art machine that can sustain full bandwidth I/O with eight disks and is very cost effective.

Keywords: Algorithm engineering; algorithm library; external memory sorting; large data sets, overlapping I/O and computation; parallel disks; prefetching, randomized algorithm; secondary memory; I/O-efficient algorithms

Cache-Oblivious Data Structures and Algorithms for Undirected BFS and SSSP

Rolf Fagerberg (Univ. of Southern Denmark - Odense)

We present improved cache-oblivious data structures and algorithms for breadth-first search and the single-source shortest path problem on undirected graphs with non-negative edge weights. Our results remove the performance gap between the currently best cache-aware algorithms for these problems and their cache-oblivious counterparts. Our shortest-path algorithm relies on a new data structure, called bucket heap, which is the first cache-oblivious priority queue to efficiently support a weak DecreaseKey operation.

Keywords: cache oblivious; data structure; single source-shortest path; breadth-first search; bucket heap

Joint work of: Brodal, Gerth Stølting; Fagerberg, Rolf; Meyer, Ulrich; Zeh, Norbert

Cache-Oblivious Searching and Sorting in Multisets

Arash Farzan (University of Waterloo)

We study three problems related to searching and sorting in multisets: determining the most frequent element (the mode), duplicate elimination and finally multi-sorting. We give deterministic and randomized cache-oblivious algorithms for these problems. We are interested in the cache complexity (i.e. the number of cache misses) of the algorithms. Our randomized algorithms match the proven asymptotic lower bounds. The deterministic algorithms are close to the lower bounds. Moreover, the cache-aware versions of the deterministic algorithms are asymptotically optimal and simpler than the previously known algorithms. All of our algorithms make the asymptotically optimal number of comparisons.

Keywords: cache oblivious; deterministic algorithm; randomized algorithm; multiset; multiset searching; multiset sorting

Joint work of: Farzan, Arash; Munro, Ian

Concurrent Cache-Oblivious Search Trees

Jeremy Fineman and Seth Gilbert (MIT - Cambridge)

The B-tree is the classic data structure for maintaining searchable data in external memory. Recent experiments have shown, however, that cache-oblivious search trees can outperform traditional B-trees. Before cache-oblivious search trees can replace traditional B-trees in industrial applications, they must support concurrent operations.

This talk presents the first study of cache-oblivious search trees. We develop two classes of concurrent cache-oblivious search trees based on the two main approaches for serial cache-oblivious search trees. The first data structure is based on an exponential search tree. This data structure supports insertions and searches/successor queries with a nearly optimal number of block transfers per operation. The data structure uses write locks but supports nonblocking reads. The second data structure is based on the packed-memory-based search tree. This data structure supports insertions and deletions, searches/successor queries, and range queries. We present both lock-based (with nonblocking reads) and lock-free variants of this data structure. The serial performance of both concurrent data structures matches the serial performance of both serial data structures they replace.

Keywords: Cache-oblivious B-tree; exponential tree; packed memory; lock-based; lock-free; nonblocking; concurrent; parallel; data structure

Joint work of: Bender, Michael A., Fineman, Jeremy T., Gilbert, Seth G., Kuszmaul, Bradley C.

The Priority R-Tree: A Practically Efficient and Worst-Case-Optimal R-Tree

Herman J. Haverkort (Universität Karlsruhe)

The query efficiency of a data structure that stores a set of objects, can normally be assessed by analysing the number of objects, pointers etc. looked at when answering a query. However, if the data structure is too big to fit in main memory, data may need to be fetched from disk. In that case, the query efficiency is easily dominated by moving the disk head to the correct locations, rather than by reading the data itself.

To reduce the number of disk accesses, once can group the data into blocks, and strive to bound the number of different blocks accessed rather than the number of individual data objects read. An R-tree is a general-purpose data structur that stores a hierarchical grouping of geometric objects into blocks. Many heuristics have been designed to determine which objects should be grouped together, but none of these heuristics could give a guarantee on the resulting worst-case query time.

We present the Priority R-tree, or PR-tree, which is the first R-tree variant that always answers a window query by accessing $O((N/B)^{1-1/d}+T/B)$ blocks, where N is the number of d-dimensional objects stored, B is the number of objects per block, and T is the number of objects whose bounding boxes intersect the query window. This is provably asymptotically optimal. Experiments show that the PR-tree performs similar to the best known heuristics on real-life and relatively nicely distributed data, but outperforms them significantly on more extreme data.

Keywords: R-Trees; data structure

Joint work of: Arge, Lars; de Berg, Mark; Haverkort, Herman J.; Yi, Ke

Extended Abstract: http://drops.dagstuhl.de/opus/volltexte/2005/155

External A*

Shahid Jabbar (Universität Dortmund)

We study External A*, a variant of the conventional (internal)A* algorithm that makes use of external memory, e.g., a hard disk. The approach applies to implicit, undirected, unweighted state space problem graphs with consistent estimates. It combines all three aspects of best-first search, frontier search and delayed duplicate detection and can still operate on very small internal memory. The complexity of the external algorithm is almost linear in external sorting time and accumulates to $O(sort(|E|)+scan(|V|))$ I/O operations, where V and E are the set of nodes and edges in the explored portion of the state space graph. Given that delayed duplicate elimination has to be performed, the established bound is I/O optimal.

In contrast to the internal algorithm, we exploit memory locality to allow blockwise rather than random access. The algorithmic design refers to external shortest path search in explicit graphs and extends the strategy of delayed duplicate detection recently suggested for breadth-first search to best-first search. The approach has been successfully implemented. We conducted experiments with sliding-tile puzzle instances and have been able to solve some of the hardest instances of the 15-puzzle that were previously unsolvable by internal A* due to enormous memory requirement.

Keywords: A*; best-first search; frontier search; duplicate detection; shortest path; I/O-efficient algorithms

Practical Cache Oblivious B-Trees on Disk

Bradley C. Kuszmaul (MIT - Cambridge)

One frequent argument in favor of cache-oblivious algorithms is that we can get rid of the "voodoo" parameters that characterize the cache.

We found, surprisingly, that cache-oblivious B-trees ($OB-trees) often outperform traditional disk-access-model B-trees (DAM B-trees).

We implemented and measured static and dynamic cache oblivious B-trees using memory mapping. We compared the performance to Berkeley DB DAM B-trees and several DAM B-trees that we implemented both with file I/O and memory mapping.

For static tree, which only support lookup, the $OB-tree achieved about twice the performance of a DAM B-tree with 4K blocks, and about 25% better performance than a DAM B-tree with 128K blocks. Thus in the static case, DAM B-trees with big blocks (about one track) are nearly as good as $OB-trees.

For dyanmic trees, which can support inserts and queries, we measured random inserts and inserts all in the same place. For random inserts and queries, the $OB-tree beats the traditional btree, usually by more than a factor of two. For inserts all in the same place, which is the worst case for $OB-trees, the $OB-trees often do as well as the DAM B-tree, but are sometimes half as fast.

It's not just about the voodoo. "It's the performance, stupid."

Keywords: B-tree; cache oblivious; search tree; disk access model; static tree; dynamic tree

Joint work of: Bender, Michael; Farach-Colton, Martin; Kasheff, Zardosht; Kuszmaul, Bradley C.

Cache-Oblivious Algorithms

Charles Leiserson (MIT - Cambridge)

Computers with multiple levels of caching have traditionally required techniques such as data blocking in order for algorithms to exploit the cache hierarchy effectively. These "cache-aware" algorithms must be properly tuned to achieve good performance using so-called "voodoo" parameters which depend on hardware properties, such as cache size and cache-line length.

Surprisingly, however, for a variety of problems including matrix multiplication, FFT, and sorting asymptotically optimal cache-oblivious algorithms do exist that contain no voodoo parameters. They perform an optimal amount of work and move data optimally among multiple levels of cache. Since they need not be tuned, cache-oblivious algorithms are more portable than traditional cache-aware algorithms.

We employ an ideal-cache model to analyze these algorithms. We prove that an optimal cache-oblivious algorithm designed for two levels of memory is also

optimal across a multilevel cache hierarchy. We also show that the assumption of optimal replacement made by the ideal-cache model can be simulated efficiently by LRU replacement. We also provide some empirical results on the effectiveness of cache-oblivious algorithms in practice.

Keywords: cache aware; cache oblivious; cache hierarchy; ideal cache model

Joint work of: Leiserson, Charles; Prokop, Harald

The Cost of Cache-Oblivious Searching

Alejandro López-Ortiz (University of Waterloo)

Tight bounds on the cost of cache-oblivious searching are proved. It is shown that no cache-oblivious search structure can guarantee that a search performs fewer than $\lg e \log_B N$ block transfers between any two levels of the memory hierarchy. This lower bound holds even if all of the block sizes are limited to be powers of 2. A modified version of the van Emde Boas layout is proposed, whose expected block transfers between any two levels of the memory hierarchy arbitrarily close to $[\lg e + O(\lg \lg B / \lg B)] \log_B N + O(1)$. This quantity approaches $\lg e + o(1) \approx 1.443$ as B increases. The expectation is relative to the initial placement of the first element of the structure in memory.

As searching in the Disk Access Model (DAM) can be performed in $\log_B N + 1$ block transfers, this result shows a separation between the 2-level DAM and cache-oblivious memory-hierarchy models. By extending the DAM model to k levels, multilevel memory hierarchies can be modelled. It is shown that as k grows, the search costs of the optimal k-level DAM search structure and of the optimal cache-oblivious search structure rapidly converge. This demonstrates that for a multilevel memory hierarchy, a simple cache-oblivious structure almost replicates the performance of an optimal parameterized k-level DAM structure.

Keywords: disk access model; cache oblivious; lower bound; search

Joint work of: López-Ortiz, Alejandro; Bender, Michael A.; Brodal, Gerth Stølting; Fagerberg, Rolf; Ge, Dongdong; He, Simai; Hu, Haodong; Iacono, John

External Memory Algorithms for Diameter and All-Pairs Shortest Paths on Sparse Graphs

Ulrich Carsten Meyer (MPI für Informatik)

We develop I/O-efficient algorithms for diameter and all-pairs shortest-paths (APSP). For general undirected graphs $G(V, E)$ with non-negative edge weights and $E/V = o(B/\log V)$ our approaches are the first to achieve $o(V^2)$ I/Os. We also show that for unweighted undirected graphs, APSP can be solved with just $O(V \cdot \mathrm{sort}(E))$ I/Os.

Both our weighted and unweighted approaches require $O(V^2)$ space. For diameter computations we provide I/O-space tradeoffs. In the paper, we also provide improved results for both diameter and APSP computation on directed planar graphs.

Keywords: external memory; diameter; all-pairs shortest paths; sparse graphs; planar graph; I/O-efficient algorithms

Joint work of: Arge, Lars; Meyer, Ulrich Carsten; Toma, Laura

On the Adaptiveness of Quicksort

Gabriel Moruz (Aarhus University)

Hoare in 1961 introduced Quicksort as a simple randomized sorting algorithm. Hoare proved that the expected number of comparisons performed by the algorithm is $O(n \log n)$. In practice the running time is strongly dependent on the number of element swaps performed, affecting the running time up to a factor of two. In this paper it is proved that Quicksort performs expected $O(n \log(Inv/n))$ element swaps, where Inv denotes the number of inversions in the input sequence. Experimental results are presented confirming the influence of the number of inversions on the running time.

Keywords: quicksort; randomized algorithm; sort; inversions

Joint work of: Moruz, Gabriel; Brodal, Gerth S.; Fagerberg, Rolf

Will the I/O model survive till the 22nd century?

Rasmus Pagh (The IT University of Copenhagen)

This rather philosophical talk considers the possibility of a memory model that allows many pieces of data to be fetched at the same time even if not spatially close. The main argument is that there is no inherent physical reason why memory devices should perform better on algorithms with spatially local access to data. The performance of the model on sorting and BFS is considered.

Keywords: Models of computation; memory model; sorting; breadth-first search

An Experimental Comparison of Empirical and Model-driven Evaluation

Keshav Pingali (Cornell University)

A key step in program optimization is the estimation of optimal values for parameters such as tile sizes and loop unrolling factors.

Traditional compilers use simple analytical models to compute these values. In contrast, library generators like ATLAS use global search over the space of parameter values by generating programs with many different combinations of parameter values, and running them on the actual hardware to determine which values give the best performance.

It is widely believed that traditional model-driven optimization cannot compete with search-based empirical optimization because tractable analytical models cannot capture all the complexities of modern high-performance architectures, but few quantitative comparisons have been done to date.

To make such a comparison, we replaced the global search engine in ATLAS with a model-driven optimization engine, and measured the relative performance of the code produced by the two systems on a variety of modern high-performance architectures. Since both systems use the same code generator, any differences in the performance of the code produced by the two systems can come only from differences in optimization parameter values used by the two systems. Our experiments on ten different platforms show that model-driven optimization can be surprisingly effective, and can generate code with performance comparable to that of code generated by ATLAS using global search.

Keywords: compiler; ATLAS; optimization; code generation

Joint work of: Pingali, Keshav; Yotov, Kamen; Li, Xiaoming; Ren, Gang; Garzaran, Maria; Padua, David; Stodghill, Paul

Cache-Oblivious Shortest Paths in Graphs Using Buffer Heap

Vijaya Ramachandran (Univ. of Texas at Austin)

We present the Buffer Heap (BH), a cache-oblivious priority queue that supports Delete-Min, Delete, and Decrease-Key operations in $O((1/B)\log(N/M))$ amortized block transfers from external memory, where N is the number of elements.

Using the Buffer Heap we present cache-oblivious algorithms for undirected and directed single-source shortest path (SSSP) problems on graphs with non-negative edge-weights. On a graph with V vertices and E edges, our algorithm for the undirected case performs $O(V + (E/B)\log(V/M))$ block transfers and for the directed case performs $O((V + E/B)\log(V/M))$ block transfers.

For both priority queue with Decrease-Key operation, and for single-source shortest path problem on general graphs, our results give the first non-trivial cache-oblivious bounds.

Finally, we briefly describe a very recent result on a variant of the Buffer Heap called the Slim Buffer Heap, and its application to the all-pairs shortest path problem on weighted undirected graphs.

The results on Buffer Heap were presented at the 2004 ACM Symposium on Parallelism in Algorithms and Architectures (SPAA). The extension to the Slim Buffer Heap and weighted APSP are included in a paper that will be presented at the 2005 ACM-SIAM Symposium on Discrete Algorithms (SODA). That paper also includes the results described in the abstract of the Dagstuhl talk by the first author.

Keywords: buffer heap; cache oblivious; priority queue; external memory; single-source shortest path; all-pairs shortest paths; data structure

Joint work of: Chowdhury, Rezaul; Ramachandran, Vijaya

Cache-Optimizations for Numerical Algorithms

Ulrich Rüde (Universität Erlangen-Nürnberg)

We will present strategies and techniques for tuning the performance of numerical algorithms for cache-based systems. One example will be the multigrid method which is an asymptotically optimal method for solving sparse linear systems arising from the discretization of partial differential equations. The second example will be the Lattice Boltzmann method, an algorithm based on cellular automata for simulating fluid flow.

Both classes of applications are fundamentally limited by bandwidth and latency restrictions of modern computer architectures. One of the difficulties is the interference between the CPU micro architecture and the memory system design that makes it very difficult to predict the observed performance and consequently to design truly cache-oblivious algorithms in practice.

Keywords: cache optimization; numerical algorithms; multigrid; sparse linear systems; Lattice Boltzmann method; fluid flow

Lower bounds for I/O complexity of range search

Vasilis Samoladas (TU Crete - Chania)

There is increased interest in the I/O compexity of range search problems. We review lower bounds techniques and results for a number of planar range search problems. We demonstrate that in many cases the I/O complexity is dictated by data placement constraints rather than search costs. We present the model of Indexability and explore space-time trade-offs.

Keywords: external; data structure; lower bound; range search; space; I/O-complexity

Joint work of: Sanders, Peter; Samoladas, Vasilis

Cache-aware List Ranking

Jop Frederik Sibeyn (Universität Halle-Wittenberg)

List-ranking is a central problem in many applications. Once solved, a list can be turned into an array on which operations can be performed in a cache-friendly way. For a list of length n, the simplest algorithm performs poorly due to its $4*n$ cache misses. It is easy to reduce this number to $2*n$. For further reductions, one might adapt external-memory algorithms, but due to their considerably increased complexity in other respects, this is not profitable at the current cost of cache misses. However, there is an algorithm performing $1*n$ cache misses, which offers a good compromise. In practice, this algorithm is almost 50% faster than the $2*n$ algorithm. The analysis in the talk is based on a simple three-parameter cost-model, which has proven to be sufficiently accurate to predict which of several possible algorithms will perform best on the available hardware.

Keywords: list ranking; cache aware; algorithm; external memory

New Cache-Aware Algorithms for Sparse-Matrix Factorizations

Sivan Toledo (Tel Aviv University)

In the talk I will describe two issues concerning cache-efficient factorizations of sparse symmetric matrices. The first part of the talk will describe a new cache-aware, but not cache-oblivious schedules for these factorizations. These schedules were designed for out-of-core implementions. They present interesting challenges both in terms of finding equivalent cache-oblivious algorithms, and in terms of designing a cache-efficient but parallel schedules.

In the second part of the talk I will focus on the cache efficiency of individual operations on nodes of the tree within the same tree model. I will describe two common algorithms, the multifrontal algorithm and the left-looking algorithm. I will show that on some matrices one algorithm achieves a high level of data reuse but the other does not, while on other matrices the situation is reversed. The problem of finding an algorithm that always performs at least as well as the best of the two is open, as well as characterizing optimal cache efficiency in these algorithms.

Keywords: cache aware; algorithm; sparse matrix; factorization; out of core

I/O-Efficient Planar Topological Sort

Laura I. Toma (Bowdoin College - Brunswick)

We present I/O-efficient algorithms for topologically sorting a planar directed acyclic graph (DAG) in $O(sort(N))$ I/Os, where $sort(N)$ is the number of I/Os needed to sort N elements. First we describe an $O(sort(N))$ algorithm that exploits properties of the dual graph using ideas from the PRAM topological sort algorithm. We also describe a simplified algorithm that runs in $O(scan(N))$ I/Os if a B^2-partition of the graph is given. The algorithm exploits the acyclicity of the input graph and reduces the problem to the same problem on a substitute graph defined on the separator vertices. These results are the first progress on the long-standing open problem of topological sorting.

Keywords: planar graph; topological sort; I/O-efficient algorithms; dag; PRAM

Joint work of: Toma, Laura I.; Arge, Lars; Zeh, Norbert

A Simple Algorithm for I/O-efficiently Pruning Dense Spanners

Jan Vahrenhold (Universität Münster)

Given a geometric graph $G = (S, E)$ in R^d with constant dilation t, and a positive constant ϵ, we show how to construct a $(1 + \epsilon)$-spanner of G with $O(|S|)$ edges using $O(sort(|E|))$ I/O operations.

Keywords: dense spanners; spanner; pruning; I/O-efficient algorithms

Joint work of: Gudmundsson, Joachim; Vahrenhold, Jan

Full Paper: http://drops.dagstuhl.de/opus/volltexte/2005/156

Paradigms for Programming Matrix Algorithms: Processor Obliviousness

David Wise (Indiana University)

In my tangential approach to cache-oblivious matrix algorithms, the quadtree decomposition of common factorizations, I uncovered a new perspective on processor scheduling that, for this group, might be called "processor oblivious" coding.

The recursive decomposition of Cholesky factorization naturally yields three (maybe four) mutually recursive functions, each side-effecting a block. One of these—the rank-k update— is performed $\binom{n}{3}$ times at the base case, where there are $n \times n$ base blocks. This, alone accounts for the cubic flop rate. Remarkably,

however, its results can be held in memory visible only to a single processor until they are "finalized" there by another function (triangularSolve), executed only $\binom{n}{2}$ times. That is, even in scheduling distributed multiprocessors, we can *much* relax the choreography of data communication simply by following the block-recursive paradigm for programming.

Keywords: cache oblivious; algorithm; matrix; processor scheduling; processor oblivious; Cholesky factorization

Bounded-Weight I/O-Efficient Shortest Paths

Norbert Zeh (Dalhousie University)

We present I/O-efficient algorithms for shortest paths in undirected graphs. These algorithms are based on a refinement of Mehlhorn/Meyer's clustering idea used to speed-up undirected breadth-first search. The first, very simple, algorithm achieves I/O-complexity $O(\sqrt{(VE \lg W)/B} + sort(V+E))$ for random edge weights between 1 and W. A refinement of the algorithm achieves the same complexity in the worst case for edge weights in the same range. For random edge weights between 0 and 1, the complexity of the algorithm becomes $O(\sqrt{VE/B} + ((V+E)\lg B)/B + sort(V+E))$.

Keywords: Shortest paths, graph algorithms, I/O-efficient algorithms; shortest paths

Joint work of: Meyer, Ulrich; Zeh, Norbert