Internationales Begegnungs- und Forschungszentrum f ür Informatik

Schloß Dagstuhl

Seminar Report 9537

Parallel and Distributed Algorithms

September 11 – 15, 1995

O v e r v i e w

The Dagstuhl Seminar on *Parallel and Distributed Algorithms* was organized by Cynthia Dwork (IBM Almaden Research Center, San José), Friedhelm Meyer auf der Heide (Universität Paderborn), and Ernst W. Mayr (Technische Universität München). It brought together 24 participants from 6 countries, 7 of them came from overseas.

The 22 talks presented cover a wide range of topics including parallel sorting algorithms, shared memory simulations, parallel data structures, scheduling algorithms, operating systems, analysis of computer architectures, and aspects of distributed computing like consistency, fault tolerance, and approximate agreement in distributed systems. The abstracts of all talks are documented in this seminar report.

The outstanding environment and organization of Schloß Dagstuhl greatly contributed to the success of the seminar.

Reported by Christian Scheideler

Participants

Artur Czumaj, Universität-GH Paderborn
Martin Dietzfelbinger, Universität Dortmund
Thomas Erlebach, TU München
Faith Fich, University of Toronto
Torben Hagerup, Max-Planck-Institut für Informatik, Saarbrücken
Zvi Kedem, New York University
Miroslaw Kutylowski, Universität-GH Paderborn
Yossi Matias, AT&T Bell Laboratories, Murray Hill
Ernst W. Mayr, TU München
Friedhelm Meyer auf der Heide, Universität-GH Paderborn
Burkhard Monien, Universität-GH Paderborn
Wolfgang J. Paul, Universität des Saarlandes
Andrea Pietracaprina, Università di Padova
Vijaya Ramachandran, University of Texas at Austin
Rüdiger Reischuk, Med. Universität Lübeck
Larry Rudolph, The Hebrew University of Jerusalem
Wojciech Rytter, University of Warsaw
Christian Scheideler, Universität-GH Paderborn
Uwe Schwiegelshohn, Universität Dortmund
Nir Shavit, MIT Cambridge
H. Raymond Strong, IBM Almaden Research Center, San José
Eli Upfal, IBM Almaden Research Center, San José
Rolf Wanka, Universität-GH Paderborn
Ralph Werchner, Universität Frankfurt

Contents

Eli Upfal
How Much Can Hardware Help Routing ?

Rolf Wanka
Simple but Fast Parallel Sorting on Multi-Dimensional Meshes

Abstracts

Contention Resolution in Hashing Based Shared Memory Simulations

by Artur Czumaj (joint work with F. Meyer auf der Heide and Volker Stemann)

We investigate the problem of simulating shared memory on the Distributed Memory Machine (DMM). Our approach uses multiple copies of shared memory cells distributed among the memory modules of the DMM via universal hashing. Thus the main problem is to design strategies that resolve contention in the memory modules. Developing ideas from random graphs and very fast randomized algorithms, we present new simulation techniques that enable us to improve the previously best results exponentially. Three randomized simulations are presented, all achieving better time-performance that previously known. The best of them simulates an n-processor CRCW PRAM ion an n-processor DMM with delay $\mathcal{O}(\log\log\log n \log^* n)$, with high probability.

We also present a general technique that can be used to turn these simulations to time-processor optimal ones, in the case of EREW PRAMs to be simulated. We obtain a time-processor optimal simulation of an $(n \log\log\log n \log^* n)$-processor EREW PRAM on an n-processor DMM with $\mathcal{O}(\log\log\log n \log^* n)$ delay.

We further demonstrate that the simulations obtained can not be significantly improved using our techniques. We show an $\Omega(\log\log\log n / \log\log\log\log n)$ lower bound on the expected delay for a class of PRAM simulations, called topological simulations, that cover all previously known simulations as well as the simulations presented in the paper.

Gossiping and Broadcasting vs. Computing Boolean Functions in Processor Networks

by Martin Dietzfelbinger

The gossip problem for an undirected graph G is the following: Assume each node initially has a piece of information, which is atomic, but replicable. In one round a node can send all the information it has gathered so far to one of its neighbors or can receive such information from one of its neighbors. How many rounds are necessary and sufficient so that all nodes get all the information? This problem has been studied intensively over some decades and has been completely or partially solved for many graph classes. The aim of this talk is to show that results from gossiping theory are closely related to the problem of computing Boolean functions in networks of processors under certain communication modes, in particular in the problem of synchronizing the processors, which corresponds to computing the OR function, as follows: initially, each processor has one input bit; finally, all processors know the output value. At the first glance, this task seems simpler than gossiping, since no atomic messages have to be preserved and since the algorithms that are allowed are more flexible in that they may use different communication patterns for different inputs. In the talk it is shown that for certain types of communication protocols ("telegraph mode": in one step a processor either can receive a message or send a message or do nothing, with prior knowledge whether a message will arrive or not), best algorithms for computing the OR and best gossip algorithms are the same. This result can be extended to different tasks and communication modes. For example, the problem of computing a Boolean function with the output appearing at one of the nodes is closely related to the problem of broadcasting one atomic message from one

source node to all nodes in the network.

Faster Approximate Agreement with Multi-writer Registers

by Faith Fich

In the approximate agreement problem, each of n processors has a real input x_i and all nonfaulty processors must compute a real output y_i in the range of the x_i's and at most ϵ apart from one another. An algorithm that solves this problem is presented.

Signature Sort

by Torben Hagerup (joint work with Rajeev Raman)

The *word-sorting problem* is to sort n integers of w bits each on a w-bit machine with a usual unit-time instruction repertoire including addition, multiplication, etc. n and w are independent parameters, except that $w \geq \log n$ (otherwise it is not possible to address all of the input). We describe a sequential and a parallel version of a new randomized sorting method called *signature sort*. Under the assumption $w \geq (\log n)^{2+\epsilon}$, for some fixed $\epsilon > 0$, sequential signature sort solves the word-sorting problem in $O(n)$ expected time, and parallel signature sort works in $O(\log n)$ expected time on an EREW PRAM with $O(n/\log n)$ processors. Signature sort is based on representing pieces of the input keys by shorter, unique integer signatures obtained via universal hashing.

Calypso: A Distributed Execution Platform for Parallel Computations

by Zvi Kedem (joint work with A.Baratloo and P. Dasgupta)

The importance of adapting distributed environments for use as parallel processing platforms is well established. However, current solutions do not always address important issues that exist in real networks. External factors like the sharing of resources, unpredictable behavior of the network and machines including slowdowns and failures, are present in multiuser networks and must be addressed. In using todays available toolkits for distributed programming, in general, the responsibility of handling these external factors is left to the programmer, a task that further complicates the development of an already difficult job of parallel programming.

Calypso is a prototype software system for writing and executing parallel programs on non-dedicated platforms, based on commercial off-the-shelf networked workstations, operating systems, and compilers. Among characteristics of the system are: (1) simple programming model incorporating shared memory constructs, (2) separation of the program and the execution parallelism to allow programs to scale as computers join an ongoing computation, (3) transparent utilization of unreliable shared resources by providing dynamic load balancing and fault tolerance, and (4) effective performance for large classes of medium- to coarse-grained computations.

A notable property of the system is that at its core lie theoretical algorithmic techniques seamlessly blending parallelism, load balancing, and fault tolerance. These techniques were originally developed in the context of abstract PRAM models and produced a general and flexible methodology for reliable parallel computing on asynchronous distributed platforms. The theoretical techniques were further developed so that they are now efficiently implemented using services provided by standard system software. The theoretical foundations were developed by (various subsets of): Y. Aumann, Z. Kedem, K. Palem, M. Rabin, A. Raghunathan,

and P. Spirakis. Some key relevant publications appeared in STOC'90, STOC'91, STOC'92, and FOCS'93. The current CALYPSO prototype was developed by A. Baratloo, P. Dasgupta, and Z. Kedem.

The Shearsort Algorithm on 3-Dimensional Meshes

by MIROSLAW KUTYLOWSKI (joint work with Rolf Wanka)

Shearsort is one of the classical sorting algorithms working on 2-dimensional meshes of processors performing compare-exchange operations. It consists of $\lceil \log l \rceil + 1$ rounds of $m + l$ parallel steps on a $l \times m$-mesh. If m is a power of 2, then $\log m + 1$ such rounds suffice.

There is a straightforward generalization of Shearsort to multi-dimensional meshes, but due to very complex behavior of this generalized algorithm no run time analysis has been known yet, except for very special cases. Experiments indicate that multi-dimensional Shearsort may satisfy similar time bounds as its 2-dimensional counterpart. We analyze the 3-dimensional case and show that on a $m_3 \times m_2 \times m_1$-mesh it suffices to perform $\log m_3 + \log m_2 + O(\frac{\log m_3}{\log m_2})$ rounds. If m_2 and m_1 are powers of 2, then $\log m_2 + \log m_1 + O(\log \log m_2 + \log \log m_1)$ rounds suffice.

Provably Efficient Scheduling for Languages with Fine-Grained Parallelism

by YOSSI MATIAS (joint work with G. Blelloch and P. Gibbons)

Many high-level parallel programming languages allow for fine-grained parallelism. As in the popular work-time framework for parallel algorithm design, programs written in such languages can express the full parallelism in the program without specifying the mapping of program tasks to processors. A common concern in executing such programs is to dynamically schedule tasks to processors so as to not only minimize the execution time, but also to minimize the amount of memory needed. Without careful scheduling, the parallel execution on p processors can use a factor of p or larger more memory than a sequential implementation of the same program.

We first identify a class of parallel schedules that are provably efficient in both time and space, even for programs whose task structure is revealed only during execution. For programs with sufficient parallelism, the schedule guarantees that the amount of memory used by the program is within a factor of $1 + o(1)$ of a sequential implementation. This space bound is obtained by proving a graph-theoretic result relating parallel and sequential traversals of directed acyclic graphs.

We then describe an efficient dynamic scheduling algorithm that generates schedules in this class, for languages with *nested* fine-grained parallelism. The algorithm is relatively simple, performing the necessary processor allocation and task synchronization while incurring at most a constant factor overhead in time and space. The correctness and performance guarantees of the algorithm rely on properties of depth-first-like traversals of series-parallel graphs. The algorithm is the first efficient solution to the scheduling problem discussed here, even if space considerations are ignored.

Algebraic Analysis of Parallel Process Models

by Ernst W. Mayr

We first consider binomial ideals over the rationals in the unknowns $x_1, \ldots, x_n$. It is known that Gröbner bases for such ideals are again binomial and obey a doubly exponential degree bound. We use this bound and another doubly exponential degree bound (due to Herrman/26) for the word problem for finitely presented commutative semigroups to derive an exponential space bound for the following problems:

1. the subword reachability problem in finitely presented commutative semigroups;
2. the problem of computing the minimal elements of an equivalence class in a finitely presented commutative semigroup;
3. the problem of computing the periods of an equivalence class in a finitely presented commutative semigroup;
4. the equivalence problem for finitely presented commutative semigroups;
5. the problem of computing the reduced Gröbner bases for a binomial ideal.

Worst-Case Analysis of Shared Memory Realizations on Parallel Machines

by Andrea Pietracaprina (joint work with G. Pucci and K.T. Herley)

We present deterministic lower and upper bounds on the slowdown required to simulate an (n, m)-PRAM on a parallel machine consisting of q processors and p memory banks, $q, p \leq n$, communicating through an interconnection network. Taking into account only memory contention, we prove that the simulation slowdown is at least $\Omega(n/q + ns/p)$ where s is a function of n, m and p which $\omega(1)$ unless $m = \Theta(n)$ or $p = O(1)$. Also, under the standard point-to-point assumption, we prove a bandwidth-based lower bound formulated in terms of the decomposition tree of the simulating network. The general result yields lower bounds of $\Omega((n/p)p^{1/d}(\log(m/2n^2)/\log\log(m/2n^2))^{1-1/d})$ and $\Omega(n/p\sqrt{p\log(m/2n^2)/\log\log(m/2n^2)})$ when specialized to p-node d-dimensional meshes and to the p-leaf pruned butterfly (a variant of Leiserson's fat-tree), where each node (leaf) hosts a processor and a memory bank.

As for the upper bounds, we introduce a novel scheme that exploits the splitting and combining of messages. Such a scheme can be implemented on an n-node d-dimensional mesh ($d = O(1)$) with $O(n^{1/d}(\log(m/n))^{1-1/d})$ slowdown, and on an n-leaf pruned butterfly with $O(\sqrt{n\log(m/n)})$ slowdown. Similar results can be obtained when the simulating machine consists of $p \leq n$ processor/memory pairs. These simulations attain the best worst-case slowdowns to date for such interconnections. Moreover, the one for the pruned butterfly is the first PRAM simulation scheme on an area-universal network, and employs novel sorting and routing primitives. Furthermore, the simulations are space-efficient and require a total amount of storage that is within a polylogarithmic factor of the size of the PRAM memory.

The QRQW PRAM: Accounting for Contention in Parallel Algorithms

by Vijaya Ramachandran (joint work with Phil Gibbons and Yossi Matias)

We introduce the queue-read, queue-write (QRQW) parallel random access machine (PRAM) model, which permits concurrent reading and writing to shared memory locations, but at a cost proportional to the number of readers/writers to any one memory location in a given step. The QRQW PRAM model reflects the contention properties of most commercially available parallel machines more accurately than either the well-studied CRCW PRAM or EREW PRAM models: the CRCW PRAM model does not adequately penalize algorithms with high contention to shared memory locations, while the EREW PRAM model is too strict in its insistence on zero contention at each step.

The QRQW PRAM is strictly more powerful than the EREW PRAM. We show a separation of sqrt(log n) between the two models, and presents faster and more efficient randomized QRQW PRAM algorithms for several basic problems such as leader election, linear compaction, multiple compaction, hashing, load balancing, and generating a random permutation. Furthermore, we show that the QRQW PRAM can be efficiently emulated with only logarithmic slowdown on Valiant's BSP model, and hence on hypercube-type non-combining networks, even when latency, synchronization, and memory granularity overheads are taken into account. This matches the best emulation result known for the EREW PRAM, and considerably improves upon the best emulation result known for the CRCW PRAM on such networks. Finally, we also present some lower bounds results for this model, including lower bounds on the time required for broadcasting and related problems and for the leader election problem.

Scattering in Processor Networks with Set-up Delays

by Rüdiger Reischuk (joint work with Andreas Jakoby)

We investigate the communication capacity and optimal data transmission schedules for processor networks connected by communication links, for example Transputer clusters. Each link allows the two processors at its endpoints to exchange data with a given fixed transmission rate τ_d. The communication itself is done in a blocking mode, that means the two processors have to synchronize before starting to exchange data and at any time each processor cannot communicate with more than one other processor (single-port model).

Our efficiency analysis will be more realistic by taking into account the setup time for a communication, which will be assumed to be a fixed constant $\tau_s > 0$. Thus, a large amount of data can be sent from one processor to a neighbour faster by a single long communication step than by a bunch of small data exchange steps: sending m data units in one step takes time $\tau_s + m\tau_d$. However, there is a tradeoff since the receiver has to wait until it has received the complete set of data before it can forward pieces to other processors.

The following prototype task called *scattering* will be considered: At the beginning one processor called the *source* possesses a set of unit size data packets, one for each processor in the network. The goal is to distribute the packets in minimal time to all recipients.

Our results concerning the complexity of this problem in arbitrary processor networks are as follows: for the general case, we give lower bounds on the minimal schedule length and show that to determine the length precisely is $\mathcal{NP}$–complete. Special classes of simple strategies are investigated in more detail. For certain networks they turn out to yield optimal schedules.

Finally, for specific regular networks like hypercubes and multidimensional grids we construct optimal schedules that can be computed efficiently, resp. good approximation algorithms.

Bit-Parallel, Free-Space, Optical Communication

by Larry Rudolph

In this paper, we examine several issues of bit-parallel free space optical communication such as arbitration operations, fault tolerance, and connector alignment.

This paper explores some of the advantages of bit-parallel, free-space, optical communication as opposed to the more common bit-serial communication techniques. While free-space optical communication promises to overcome many of the current technological limitations for truly massive, high-performance information interchange, we argue that it is important to send the bits in parallel unlike current optical communication technology.

This paper focuses on bit-parallel communication and shows how it can be used in several ways. In particular, a solution to the arbitration problem for controlling concurrent access is presented. We assume a newer, nonstandard architectural support to handle the unusual demands of arbitrarily large fan-in. Using a dual-rail transmission of binary data, we show how bit-parallel communication can tolerate faulty detectors. Finally, we also show how to handle the alignment requirements between transmitters and receivers.

Creating a Wider Bus Using Word Compaction

by Larry Rudolph

The effective bandwidth of a bus and external communication ports can be increased by using a variant of data compression techniques that compacts words instead of data streams. The compaction is performed by caching the high order bits into a table and sending the index into the table along with the low order bits. A coherent table at the receiving end expands the word into it original form. Compaction / expansion units can be placed between processor and memory, between processor and local bus, and between devices that access the system bus.

Simulations have shown that over 90 per cent of all information transferred can be sent in a single cycle when using a 32 bit processor connected by a 16 bit wide bus to a 32 bit memory module. This is for all forms of data, address, data, and instructions, and when a cache-based processor is used.

Sublinear Time Parallel Algorithms for Constructing Optimal Binary Search Trees

by Wojciech Rytter

The OBST problem consists in constructing an optimal binary sarch tree for a given sequence of weights p_i, q_i. The items are stored in the *inorder* in the internal nodes of the tree (their weights are p_i's, and the weights of external nodes are q_i's. The sum of weights equals 1).

The best sequential algorithm for the OBST problem is Knuth's algorithm having quadratic work. In this paper we are interested in parallel algorithms with subquadratic work for the OBST problem for *special* input sequences (when $p_i + q_i \geq \frac{const}{n}$ for eqach i) and for approximately optimal solutions with arbitrarily small error. Our algorithms work in $O(n^{0.6})$

time with n processors.

We also show that there are algorithms working in sublinear time with *expected* sublinear work, assuming the sequence of weights is randomly permuted. The algorithms use nonstandard decompositions of binary search trees into two different types of subtrees.

Space-Efficient Routing Strategies

by Christian Scheideler (joint work with F. Meyer auf der Heide)

We present a new approach towards space-efficient routing on arbitrary networks. This approach works in such a way that a network G with very space-efficient path system is simulated by some network H. Furthermore we present routing protocols that support this approach and obtain the following results.

1. For any vertex-symmetric network with n vertices, degree d, and diameter $D = \Omega(\log n)$ it holds for all $s \in [2, n]$:
   A randomly chosen function and any permutation can be routed in time $O(\log_s n \cdot D)$, w.h.p., if $O(s \cdot D \cdot d)$ space is available at each processor and $O(\log(s \cdot D))$ space is available in each packet for storing routing information. (The drawdack of this scheme is that it is randomized and requires unbounded buffer size.)

2. For any vertex-symmetric network with n vertices, degree d, and diameter $D = \Omega(\log n)$, there is a deterministic hot-potato routing protocol that can route any permutation in time $O(\log n \cdot D \log^{1+\epsilon} D)$ for any $\epsilon > 0$ if each vertex has a self-loop, and $O(D(\log\log D + \log d))$ space is available in each vertex and $O(\log D)$ space is available in each packet to store routing information.

   This protocol makes use of a new offline routing protocol for buffer size 1 which may be of independent interest. Furthermore, the protocol can be used to obtain the same bounds as above for arbitrary graphs with second largest eigenvalue λ if D is set to $\log_\lambda n$.

Besides these results we present routing protocols that take into account the bandwidth or buffer size of a network. These protocols yield work-optimal circuit switching in Butterflys with bandwidth $\log\log n$ and work-optimal hot potato wormhole routing in (m, d)-tori if the length of the worms is at least $d \log n$ ($n = m^d$) and $1 \leq d \leq \frac{\log n}{\log\log n}$.

Finally, we present a protocol for arbitrary shortest path systems with buffer size b that needs $O(\frac{C \cdot D^{1/b} + \log n}{b}(D + C + \log n))$ time to route all packets.

Parallel Scheduling Algorithms

by Uwe Schwiegelshohn

The talk started with introducing requirements for parallel job scheduling. Next, makespan scheduling was compared with flow time scheduling. We showed that while the flow time of the optimal makespan solution may be as much as $\sqrt{P}$ times the optimal flow time with P being the number of processors, the makespan of the optimal flow time solution is not more than $\Theta(1)$ times the optimal makespan. Then, a distributed scheduling approach based on a two level hierarchy was suggested. The bottom level addresses the scheduling problems of a single user and uses on-line makespan algorithms with a small competitive factor. For the flow

time algorithms of the top level we first demonstrated that the competitive factor may be a linear function of the number of tasks and that this approach is therefore not applicable for flow time scheduling. Instead it was suggested to use a scheduling approach where the ratio $\frac{execution\ time}{weight}$ is kept constant while the execution time itself is unknown. A new algorithm was introduced which uses a list of jobs ordered by $\frac{work}{weight}$ (modified Smith's ratio) for scheduling. When all jobs are known up front, the algorithm needs $O(m \log m)$ running time where m is the number of tasks and has a worst case approximation factor of 2.42 regardless of the execution times. While it requires preemption there are never more than two tasks active on one processor and no task migration is necessary during the execution of the schedule. Finally, it was discussed how to extend the method to include network topology, precedence constraints, deadlines and additional resource constraints.

Diffracting Data Structures

by Nir Shavit (joint work with Dan Touitou)

Shared *pools* and *stacks* are two widely used multiprocessor coordination structures that with applications ranging from simple producer/consumer buffers to job-schedulers and procedure stacks. The literature offers us a variety of possible pool implementations, from *queue-lock* based solutions to the wonderfully effective randomized *work-pile* techniques of Rudolph, Slivkin-Allaluf and Upfal and *stealing* techniques of Blumofe and Leiserson. Unfortunately, the former offer good performance under sparse access patterns, but scale poorly since they offer little or no potential for parallelism, while the latter offer good *expected* response time under high loads, but very poor performance as access patterns become sparse.

This talk introduces *elimination trees*, a novel form of the diffracting-tree data structures that provide pool and stack implementations with an average *constant* response time under high loads, and a logarithmic time termination guarantee under sparse patterns. Our empirical results show that unlike diffracting trees, and in spite of the fact that elimination trees offer a "deterministic" guarantee of coordination, they scale like the "probabilistic" *work-pile* and *stealing* methods, providing improved response time as the load on them increases.

Collective Consistency: Consistent Failure Detection for Parallel Computing

by H. Raymond Strong (joint work with C. T. Howard Ho and Cynthia Dwork)

This talk presents a new paradigm for the coordination of distributed processes that is useful in the context of parallel computing. Coordination is an abstraction of the many ways members of a group cooperate to reach collective goals. When the members of a group participate in a coordination protocol, each member receives some (possibly null) input value, and if everything works properly, deach member is expected to produce some output value (or action). We review the paradigms of distributed commit, agreement, and consensus, and discuss the various assumptions that must be made in order to guarantee that a protocol satisfies the paradigm. In general, these assumptions can only be assigned a "high probability".

Our new coordination paradigm is called collective consistency. The assumptions on which it depends are simpler than those of the coordination paradigms mentioned above. Like distributed commit, it does depend on the assumption that participants only deviate from their assigned programs by stopping. It also assumes eventual communication between participants who do not stop. Unlike anyu of the above mentioned coordination paradigms, collective con-

sistency does not require that any correctly functioning participant must eventually produce an output. In its weakest form collective consistency only requires that any two participants that do produce outputs and do not regard each other as having failed must produce the same output. This paradigm does not require any randomness or synchrony assumptions. A stronger variation requires that all outputs that are produced must be identical, independent of any failure detection. A nontrivial collective consistency protocol must have distinct scenarios in which all participants produce each of at least two different outputs. One can have a correct nontrivial protocol for consistency in an environment in which there can be no correct protocol for agreement or consensus.

We first motivate the new paradigm with a discussion of failure tolerance in parallel computing. Then we give a lower bound result that applies to all coordination protocols to show that we are on the right track with collective consistency. We discuss several variations of a protocol for collective consistency. Finally we enumerate a number of open questions including the question of whether collective consistency can be achieved in the context of Byzantine failures.

Simple but Fast Parallel Sorting on Multi-dimensional Meshes

by Rolf Wanka (joint work with Mirosław Kutyłowski)

We introduce a family of sorting algorithms for the d-dimensional m-sided mesh, with m a power of 2. These algorithms can be described and analyzed in a very simple way.

The first algorithm is a method that needs $\Theta(d^2 m \log m)$ steps. It does not need any form of routing and is a variant of the multi-dimensional Shearsort.

The second method allows additional routing phases. Here, $\Theta(d^2 m)$ parallel steps are sufficient to sort. If d is constant, this algorithm is asymptotically optimal.

The constant factors in the runtime for both algorithms are very small. Both methods are based on the fact that, if the contents of a mesh with snake-like indexing scheme is 2-ordered, a single iteration of the multi-dimensional Shearsort already sorts the entire sequence of keys.

How Much Can Hardware Help Routing ?

by Eli Upfal (joint work with A. Bodorin, B. Schieber, and P. Raghavan)

We study the extent to which complex hardware can speed up routing. Specifically, we consider the following questions. How much does adaptive routing improve over oblivious routing? How much does randomness help? How does it help if each node can have a large number of neighbors? What benefit is available if a node can send packets to several neighbors within a single time step? Some of these features require complex networking hardware, and thus it is important to investigate whether the performance justifies the investment.
By varying these hardware parameters, we obtain a hierarchy of time bounds for worst-case permutation routing. We develop a nearly complete taxonomy of the complexity of routing.

E-Mail Addresses

Artur Czumaj	artur@uni-paderborn.de
Martin Dietzfelbinger	dietzf@ls2.informatik.uni-dortmund.de
Thomas Erlebach	erlebach@informatik.tu-muenchen.de
Faith Fich	fich@cs.toronto.edu
Torben Hagerup	torben@mpi-sb.mpg.de
Zvi Kedem	kedem@cs.nyu.edu
Miroslaw Kutylowski	mirekk@uni-paderborn.de
Yossi Matias	matias@research.att.com
Ernst W. Mayr	mayr@informatik.tu-muenchen.de
Friedhelm Meyer auf der Heide	fmadh@uni-paderborn.de
Burkhard Monien	bm@uni-paderborn.de
Wolfgang J. Paul	paul@cs.uni-sb.de
Andrea Pietracaprina	andrea@artemide.dei.unipd.it
Vijaya Ramachandran	vir@cs.utexas.edu
Rüdiger Reischuk	reischuk@informatik.mu-luebeck.de
Larry Rudolph	Rudolph@cs.huji.ac.il
Wojciech Rytter	rytter@mimuw.edu.pl
Christian Scheideler	chrsch@uni-paderborn.de
Uwe Schwiegelshohn	uwe@carla.e-technik.uni-dortmund.de
Nir Shavit	shanir@theory.lcs.mit.edu
H. Raymond Strong	strong@almaden.ibm.com
Eli Upfal	ely@almaden.ibm.com
Rolf Wanka	wanka@uni-paderborn.de
Ralph Werchner	werchner@thi.informatik.uni-frankfurt.de

Addresses

Artur Czumaj
Universität-GH Paderborn
FB 17 - Mathematik/Informatik
Warburgerstr. 100
33098 Paderborn
Germany
Phone: +49-5251-60-6490

Martin Dietzfelbinger
Universität Dortmund
Fachbereich Informatik
Lehrstuhl II
44221 Dortmund
Germany
Phone: +49-231-755-4737 /-2777

Thomas Erlebach
TU München
Institut für Informatik
80290 München
Germany
Phone: +49-89-2105-84 95

Faith Fich
University of Toronto
Dept. of Computer Science
10 King's College Road
Toronto Ontario M5S 1A4
Canada
Phone: +1-416-978-6183

Torben Hagerup
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
Germany
Phone: +49-681-302-5358

Zvi Kedem
New York University
Courant Institute of Mathematical Sciences
251 Mercer Street
New York NY 10012-1185
USA
Phone: +1-212-998-3101

Miroslaw Kutylowski
Universität Paderborn
FB 17 - Mathematik/Informatik
D-33095 Paderborn
Germany
Phone: +49-5251-60-6461

Yossi Matias
AT&T Bell Labs - Murray Hill
Rm 2D-146
600 Mountain Avenue
Murray Hill NJ 07974-0636
USA
Phone: +1-908-582-70 68

Ernst W. Mayr
TU München
Institut für Informatik 14
80290 München
Germany
Phone: +49-89-2105-2680/2681 (Secr.)

Friedhelm Meyer auf der Heide
Universität-GH Paderborn
FB 17 - Mathematik/Informatik
D-33095 Paderborn
Germany
Phone: +49-5251-60-6480/6481 (Secr.)

Burkhard Monien
Universität-GH Paderborn
FB 17 - Mathematik/Informatik
D-33095 Paderborn
Germany
Phone: +49-5251-60-6707/6695 (Secr.)

Wolfgang J. Paul
Universität des Saarlandes
Fachbereich 14 - Informatik
Postfach 11 50
D-66041 Saarbrücken
Germany
Phone: +49-681-302 2436

Andrea Pietracaprina
Università di Padova
Dipartimento di Matematica Pura e Applicata
Via Belzoni 7
35131 Padova
Italy
Phone: +39-49-827 5949

Vijaya Ramachandran
University of Texas at Austin
Department of Computer Sciences
Taylor Hall 2.124
Austin TX 78712-1188
USA
Phone: +1-512-471-95 54

Rüdiger Reischuk
Med. Universität zu Lübeck
Technisch-Naturwissenschaftliche Fakultät
Institut für Theoretische Informatik
Wallstraße 40
D-23560 Lübeck
Germany
Phone: +49-451-7030-416

Larry Rudolph
The Hebrew University of Jerusalem
Department of Computer Science
91904 Jerusalem
Israel
Phone: +972-2-585261

Wojciech Rytter
University of Warsaw
Institute of Infomatics
Ul. Banacha 2
PL-02-097 Warszawa
Poland

Christian Scheideler
Universität-GH Paderborn
FB 17 - Mathematik/Informatik
33095 Paderborn
Germany
Phone: +49-5251-60-6433

Uwe Schwiegelshohn
Universität Dortmund
Lehrstuhl für Datenverarbeitungssysteme
44221 Dortmund
Germany
Phone: +49-231-755-26 34

Nir Shavit
MIT Cambridge
Lab of Computer Science
545 Technology Square
Cambrige MA 02139
USA
Phone: +1-617-253 5905

Eli Upfal
IBM Almaden Research
Dept. K53/802
650 Harry Road
San Jose CA 95120-6099
USA
Phone: +1-408-927-1788

Rolf Wanka
Universität-GH Paderborn
FB 17 - Mathematik/Informatik
D-33095 Paderborn
Germany
Phone: +49-5251-60-6433

Ralph Werchner
Universität Frankfurt
FB 20 Informatik
Robert-Mayer-Str. 11-15
60054 Frankfurt
Germany
Phone: +49-69-798-23424