Internationales Begegnungs- und Forschungszentrum für Informatik

Schloß Dagstuhl

Seminar Report 9737

Parallel and Distributed Algorithms

September 8 – 12, 1997

Overview

The fifth Dagstuhl Seminar on *Parallel and Distributed Algorithms* was organized by Ernst W. Mayr (TU München), Friedhelm Meyer auf der Heide (Universität Paderborn), and Larry Rudolph (The Hebrew University, Jerusalem). It brought together 31 participants from 8 countries, 12 of them came from overseas.

The 28 talks presented covered a wide range of topics including routing and sorting on interconnection networks, models of parallel computation, network-aware applications, instruction-level parallelism, etc. In addition, several open problems were posed during an evening session.

Enjoying the pleasant atmosphere provided by the Dagstuhl Center, the participants used the surroundings for lively discussions as well as recreational hiking. We would like to thank all those who contributed to the success of the seminar.

Reported by Ben Juurlink and Berthold Vöcking

Participants

Jean-Loup Baer, Seattle, USA
Yosi Ben-Asher, Haifa, Israel
Artur Czumaj, Paderborn, Germany
Ralf Diekmann, Paderborn, Germany
Martin Dietzfelbinger, Dortmund, Germany
Pierre Fraigniaud, Paris, France
Thomas Gross, Zürich, Switzerland
Torben Hagerup, Saarbrücken, Germany
Ben Juurlink, Paderborn, Germany
Clyde P. Kruskal, College Park, USA
Ludek Kucera, Prague, Czech Republic
Miroslaw Kutylowski, Paderborn, Germany
Klaus-Jörn Lange, Tübingen, Germany
Krzysztof Lorys, Wroclaw, Poland
Ernst W. Mayr, Munich, Germany
Friedhelm Meyer auf der Heide, Paderborn, Germany
Andrea Pietracaprina, Padova, Italy
Christine Rüb, Saarbrücken, Germany
Vijaya Ramachandran, Austin, USA
Larry Rudolph, Cambridge, USA
Wojciech Rytter, Warsaw, Poland
Uwe Schwiegelshohn, Dortmund, Germany
Jop Sibeyn, Saarbrücken, Germany
Peter Steenkiste, Pittsburgh, USA
H. Raymond Strong, San Jose, USA
Ted Szymanski, Montreal, Canada
Pilar de la Torre, Durham, USA
Eli Upfal, San Jose, USA
Berthold Vöcking, Paderborn, Germany
Uzi Vishkin, College Park, USA
Rolf Wanka, Paderborn, Germany

Contents

Christine Rüb
On Batcher's Merge Sorts as Parallel Sorting Algorithms

Wojciech Rytter
Parallel Tree-Contraction and Fibonacci Numbers

Uwe Schwiegelshohn
A Distributed Scheduling Algorithm for Parallel Discrete Event Simulation

Jop F. Sibeyn
Solving List Ranking Almost as Fast as Sorting

Peter Steenkiste
Application-specific Resource Management in Darwin

Ray Strong
Practical Membership Protocols

Ted H. Szymanski
Parallel Computing with Optical Interconnects

Torben Hagerup
Allocating Independent Tasks to Parallel Processors: An Experimental Study

Eli Upfal
Dynamic Analysis of Communication Processes

Uzi Vishkin
From Algorithm Parallelism to Instruction-Level Parallelism: An Encode-Decode Chain Using Prefix-Sum

Berthold Vöcking
Exploiting Locality for Data Management in Systems of Limited Bandwidth

Rolf Wanka
Sorting on a Massively Parallel System Using a Library of Basic Primitives

Open Problems

Posed by
Ernst W. Mayr
Friedhelm Meyer auf der Heide
Andrea Pietracaprina
Christine Rüb
Ted H. Szymanski

Abstracts

On the Performance of Cluster Architectures

by Jean-Loup Baer (joint work with Xiaohan Qin)

We consider the performance of shared-memory cluster-based architectures where each cluster is a shared-bus multiprocessor augmented with a protocol processor maintaining cache coherence across clusters.

We first evaluate the performance of various cluster configurations including the impact of adding a remote shared cache in each cluster. We use Mean Value Analysis to estimate intra and inter-cluster cache miss latencies as well as overall execution time. Architectural parameters are carefully selected and application parameters, such as frequency and types of cache misses, are obtained through trace-driven simulation.

The model exercised –and validated – on three applications shows that without remote caches the performance of cluster-based architectures is mixed. With remote caches, clusters consistently outperform the single bus system.

We then show, via simulation, that with the introduction of user-based communication primitives, the overhead of a software implementation on the protocol processor can be overcome and the performance is competitive with that of a hardware based cache coherence solution.

The Partitioned PRAM Model, Re-Doing Basic PRAM Algorithms with Limited Resources

by Yosi Ben-Asher

The attempt is to research phenomena of practical parallel execution in a theoretical setting. We argue that practice and theory do not have a common model which both sides can use (due to their diffrent nature). Hence, the best one can do, is to propose a theoretical setting where the gap of practical parallel programming can be studied as a theoretical problem. Programmers can then use the model to write parallel programs, however, since there is no real meeting between theory and practice, then the implementation of the model is a puerly practical thing and theory can not handel this problem. The theoretical setting should use the following rules:

The theoretical analog for practical parallel programming will be the concept of tradeoff (TD) relating the execution time to the available resources $T = f(x_1, \ldots, x_k)$. The complexity of a problem is obtained when the lower bound TD is equal the upper bound TD achieved by an actual algorithm $A(x_1, \ldots x_k)$ for every possible value of $x_1, \ldots, x_k$. Thus the realization of an algorithm as a program require $x_1, \ldots x_k$ to be actual parameters of the program. TD should be classified according to scalability properties of the form $f(x_1, \ldots, c \cdot x_i, \ldots x_k) = c \cdot f(x_1, \ldots, x_i, \ldots x_k)$ (replacing the notion of speedup and efficiency). Experiments should be made to prove the following Meta claim:

1- Any parallel execution follows the a curve similar to that of the TD. 2- Optimal experimental values of $x_1, \dots, x_k$ (when $A()$ achieves the complexity of the problem) implies optimal performances against any possible program for the same problem.

$x_1, \dots, x_n$ reflect theoretical resources whose value is determined experimentally, thus elevating the model from the messy details of the underlying architecture. The model should be conductive for actual programming stiles (either message passing or shared memory).

In this talk I have presented such a model described a simple lower bound for computing the sum of n numbers and discussed experimental results with Matrix multiplication. The proposed PPRAM model is a modification of the m-PRAM (proposed by Vishkin and Wigderson 1985) with the requirment that the imput should be partitined between the processors.

Adaptitive Allocation Processes

by Artur Czumaj (joint work with Volker Stemann)

We investigate various randomized processes allocating balls into bins that arise in applications in dynamic resource allocation and on-line load balancing. We consider the scenario when m balls arriving sequentially are to be allocated into n bins on-line and without using a global controller.

Traditionally, the main aim of allocation processes is to place the balls into bins to minimize the maximum load in bins. However, in many applications it is equally important to minimize the number of trails performed by the balls (the allocation time). We study adaptive allocation schemes that achieve optimal tradeoffs between the maximum load, the maximum allocation time, and the average allocation time.

We investigate allocation processes that may reallocate the balls. We provide a tight analysis of the maximum load of processes that during placing a new ball may reassign the balls in up to d randomly chosen bins.

We study infinite processes, in which in each step a random ball is removed and a new ball is placed according to some scheduling rule. We present a novel approach that establishes a tight estimation of the time needed for the infinite process to be in the state near to its equilibrium.

Load Balancing Strategies for Scientific Computing Applications

by Ralf Diekmann

In this talk, I will focus on applications from the field of Scientific Computing, a class of problems which classically demand for very large amounts of computations, but which are also able to exploit the potentials of large massively parallel systems. Examples are Finite Element or Finite Difference simulations of fluid dynamics or structural mechanics problems. Such applications typically iterate quite simple algorithms on large data-sets which are structured as a two- or three-dimensional graphs (called meshes).

If the simulation is static, i.e. the mesh is known in advance and does not change during run-time, the load balancing problem reduces to the task of mapping the application graph onto the processor network. For modern parallel architectures, this problem can further be relaxed to graph partitioning. I will give a short survey of existing graph partitioning algorithms and present a new one which, if applied to the 2-way partitioning problem, is able to meet bounds on the bisection width of regular graphs.

Modern techniques in numerical simulation are dynamic in the sense that they adapt the mesh based on error estimates of the current solution. In this case, a dynamic load balancing problem arises: The distribution of the mesh over the processors has to be adjusted in order to keep the workload balanced. A number of different dynamic load balancing strategies can be used to handle this problem. For this talk, I will focus on a two-step approach: The first step determines how much load to move where and the second actually identifies the data-items to move. For the first step, local iterative methods can be used. I will shortly introduce existing methods and discuss some performance and usability issues of diffusive algorithms.

The second step is crucial for certain numerical algorithms, especially for preconditioners which are based on domain decomposition. Here the *shape* of subdomains heavily influences the overall run-time. I will present some first attempts to consider this shape while migrating load items.

On Analyzing the Cost of Communication in Networks

by Martin Dietzfelbinger

We study the question of how many bits must be exchanged in an asynchronous network to compute a function f, if the input components are distributed among the processors. Tiwari (1987) showed how this problem can be reduced (for many networks, in particular those with a large diameter) to the following very simple scenario: $k+1$ processors $P_0, \ldots, P_k$, connected by k links to form a linear array, are to compute a function $f(x, y)$, $x \in X$, $y \in Y$, on a finite domain $X \times Y$, where x is only known to P_0, y is only known to P_k; the intermediate processors $P_1, \ldots, P_{k-1}$ do not have any information. The processors compute $f(x, y)$ by exchanging binary messages across the links, according to some protocol Φ. Let $D_k(f)$ denote the minimal complexity of such a protocol Φ, i. e., the total number of bits sent across all links for the worst case input, and let $D(f) = D_1(f)$ denote the (standard) 2-party communication complexity of f. Tiwari proved that $D_k(f) \geq k \cdot (D(f) - O(1))$ for almost all functions f and conjectured this inequality to be true for all f. His conjecture was falsified by Kushilevitz, Linial, and Ostrovsky (1996): they exhibited a function f for which $D_k(f)$ is essentially bounded above by $\frac{3}{4}k \cdot D(f)$. The best general lower bound known is $D_k(f) \geq k \cdot (\sqrt{D(f)} - \log k - 3)$. We prove

$$D_k(f) \geq 0.146 \cdot k \cdot D(f), \text{ for arbitrary } f \text{ and } k.$$

Applying the general framework provided by Tiwari, we may derive lower bounds for the communication complexity of computing functions in general asynchronous networks on the basis of their two-party communication complexity.

A General Theory for Deadlock Avoidance

by PIERRE FRAIGNIAUD (joint work with E. Fleury)

Deadlock occurs while routing in a network when several messages $M_1, \ldots, M_k$ block each other: M_i is holding some resources (namely buffers) that are required by M_{i-1} to proceed further, this for all i. Several characterizations of deadlock-free routing function have been derived in the literature, each of them for a particular type of routing function: static *vs.* adaptive, vertex-dependent *vs.* input-dependent, unicast *vs.* multicast, etc. In this talk, we will present a framework that allows to unify all these characterizations in a single theorem. This framework is based on a simple though general description of what is a routing function.

An abstract version of this paper will appear in the proceedings of the 10th International Conference on Parallel and Distributed Computing (PDCS '97), New Orleans, Oct. 1997.

The Kitchawan Scalable Shared Memory Multiprocessor Operating System

by HUBERTUS FRANKE (joint work with Orran Krieger, Marc Auslander, Bryan Rosenburg, Bob Wisniewski, and Marc Snir)

We are developing a new general-purpose operating system for cache-coherent multiprocessors. Target systems range from the small-scale multiprocessors that we expect will become ubiquitous, to the very large-scale non-symmetric multiprocessors that are becoming increasingly important in both commercial and technical environments. By designing the system from the start for cache-coherent multiprocessors, we expect to achieve a high degree of spatial and temporal locality in code and data. This locality will result in substantial performance advantages, even on small-scale servers. To support large servers, Kitchawan is designed to scale to hundreds of processors and to address the reliability, fault-containment, and fault-tolerance requirements of large commercial applications. We exploit the 64-bit addressing capability of modern processors to improve performance on the full range of systems and to facilitate scalability on large systems. We employ microkernel and object-oriented technology to make the system easier to maintain, easier to extend to support new platforms and new applications, and easier to customize to support the needs of important subsystems (e.g., databases, web servers). Due to the in-cache locking facilities provided by future architectures, we are targeting very fine grain locking schemes with very little memory and runtime overhead for the uncontended case. One of the key technologies we employ are building block compositions, which are abstract topologies of system level facilities, e.g. memory management. Trusted servers can replace parts of these topologies by downloading refined implementation into the topology for particular applications. We treat IPCs as method invocations on objects located in different address spaces. This allows us to have the IPC facility select an available thread thus avoiding potential bottlenecks in the system. Further it enables us to pass arguments via registers or use handoff scheduling. We are developing the OS concurrently on x86 and on PowerPC architectures to keep us honest with respect to portability (e.g. endian issues, memory layout, etc.).

Network-aware Applications: New Challenges for Parallel and Distributed Computing

by Thomas Gross

Parallel and distributed applications execute on systems that are based on general-purpose, shared networks. The performance (bandwidth, latency, etc.) of such networks changes during the course of a computation. A user may experience unpredictable response-time (if the application requires more resources than the network can provide) or may not take full advantage of available resources (if the application conservatively limits its demands). Network-aware applications adjust their resource demands in response to resource availablility (and system-aware applications consider all aspects of the execution environment, in addition tothe network). This talk discusses the construction of such adaptive applications. We report on a framework that was developed to simplify application development for a variety of network architectures.

Further information on this research can be found via the homepage of the CMCL Laboratory at Carnegie Mellon: `http://www.cs.cmu.edu/~cmcl`

Experimental Validation of Parallel Computation Models on the Intel Paragon

by Ben Juurlink

New experimental data validating some of the proposed parallel computation models on the Intel Paragon is presented. This architecture is characterized by a large bandwidth and a relatively large startup cost of a message transmission, which makes it very important to send a few large messages instead of many small ones. For this reason, we implemented a BSP library on top of the native NX message passing library that postpones communication until the end of the superstep and combines all packets destined for the same processor into a single message. Our results show that on the Paragon this technique cannot reduce g to a value which is comparible to the reciprocal of the bisection bandwidth of the communication network, because the link speed is so high that memory access time and buffer management account for a significant part of the communication overhead. Furthermore, it is shown that the execution time of an algorithm implementation can be significantly reduced if the number of startups is taken into consideration, even if reducing the number of startups increases the communication volume and the number of supersteps.

Edge-disjoint Paths and Network Routing

by Ludek Kucera

Let us start by an example motivationg the talk: 16-64 workstations connected to an interconnection network through PCI or SBus cards (1Gbit/s) and using packets of 64 bytes or more. In such a distributed multiprocessor system, considered by many as the most important distributed parallel architecture of the near future, is it possible to achieve (hardware) packet latencies of order of 50 ns? Since a packet length, expressed in time units, is at least 512 ns, i.e. 10 times more that the required latency, there is practically no difference between standard packet switching and circuit switching. Moreover, if one packet is blocked by another one in a network, the average waiting time is 256 ns (assuming random traffic), and therefore it has to occure only very infrequently, which is difficult to guarantee is nonadaptive' or minimal adaptive routing is used. This is why we study deflection (fully adaptive) circuit switching, which essentially means to respond to a communication request by connecting a packet source with its destination by a path which is edge-disjoint with other paths that have already been established, and to keep the path for some time. The problem addressed in the talk is to estimate the probability that such a path does not exist, assuming random network traffic. Using theory of random graphs, we show that, in a typical network, there are 3 phases: 1. low load (e.g. in 8x8 torus, less than about 14 nodes is connected by a free path (not necessarily of minimal length). Even naive routing algorithms can find such a path. 2. medium load (e.g. in 8x8 torus, 14-35 be connected, but some exceptions (usually isolated nodes) are likely. Some naive' algorithms fail, more sophisticated ones still give good results (low latency). 3. high load - only a minority of node pairs are connected by free paths, no efficient circuit switching is possible. It is shown that the first two load ranges cover practically all sustainable bandwidth of packet switching algorithms with limited or no adaptability, but exhibit much smaller latencies for medium load - 50 ns seens to be feasible is good deflection algorithms are used.

Generating Random Permutation on Parallel and Distributed Systems

by Mirosław Kutyłowski (joint work with Artur Czumaj, Przemka Kanarek, and Krzysztof Loryś)

We consider the problem of permuting items at random according to uniform probability distribution in parallel and distributed systems. Till now, there have been three different approaches to construct algorithms for solving this problem:

- the approach based on integer sorting (the items to be permuted receive random keys and are sorted according to these keys);
- the approach based on parallelizing shuffle algorithm (which is a simple sequential strategy to shuffle card decks at random);
- the approach based on dart-throwing (each processor acting on behalf of an item throws a dart into an array, the order of darts in the array implicitly determines a permutation).

We propose two new paradigms for solving this problem. The first one is to generate a certain random network and to deliver packets according to the routes given by the network. This results in a CREW PRAM algorithm with runtime $O(\log \log n)$. The drawback of this method is that it uses concurrent read operations and hypergeometric random generators. On the other hand, the permutations are generated *exactly* uniformly at random.

Another approach is based on a simple randomized protocol, called *distributed mixing*. The Markov chain defined by the protocol has property of rapid mixing in polylogarithmic time. A fast simulation of this process on parallel machines leads to novel algorithms for generating random permutations. For example, this gives an $O(\sqrt{\log n})$-time algorithm on the QRQW PRAM, or an $O(\log \log n)$-time algorithm on the OCPC-computer. Nevertheless, the distributed mixing protocol itself and the proof methods used for its analysis (a special way of utilizing coupling techniques on Markov chains) seem to have numerous applications in many remote areas.

New Criteria for Parallelization

by Klaus-Jörn Lange

This talk presents some results concerning the classification of problems and algorithms wrt. their parallelization. Parallel complexity theory offers the notions of *P*–completeness vs *NC*–membership to distinguish between inherent sequential and well parallelizable problems. But using reducibilities allowing polynomial growth this approach only characterizes the possibility of decreasing polynomial running times down to polylogarithmic ones. In addition, this approach doesn't preserve the order of processor-time-products and hence cannot treat matters of efficiency and real speed-up.

In order to treat the question of decreasing a polynomial running time down to a smaller polynomial one, Klaus Reinhardt introduced the notion of *strict sequential P–completeness*. He exhibited a hardest inherent sequential problem in the following sense: the problem is solvable in linear time on a RAM, but if there should exist a parallel algorithm with runing time $O(n^{1-\epsilon})$ using a polynomial number of processors then every problem in *P* would enjoy a considerable speed-up.

Concerning the lack of reasonable reducibility notions which would preserve time-processor-products and would be based on moderate parallel models it seems to be reasonable to look for other criteria which are related to the existence of efficient parallel solutions. Recently, Rolf Niedermeier introduced the notion of *recursive divisibility*. An algorithm is recursively divisible if it consists in a fixed number of layers where each layer consists in an asy and regular data movement followed by the parallel execution of n^ϵ totally independent identical tasks. On the hand, this notion seems to favour well structured parallel algorithms as they are typically designed for grids. On the other hand, it seems to be closely related to the classes NC and NC^1.

These results have been obtained within the project KOMET (DFG: La618/3-2).

Efficient Embeddings of Treelike Graphs into Hypercubes

by Ernst W. Mayr (joint work with Volker Heun)

We consider the problem of embedding binary trees and treelike graphs (i.e., graphs which have small extended edge bisection width, a concept which gets defined in the talk and is related to standard bisection width with respect to vector weights) into the boolean hypercube and its many variants. Basic parameters to measure the quality of such embeddings are the load, the dilation, the expansion and the (node) congestion. We present an efficient algorithm to embed any graph with small extended edge bisection width, requiring only an efficient subroutine for computing the extended edge bisectors. We give applications of the general theorem for the following families of graphs: graphs with small treewidth or pathwidth, circular arc and interval graphs, k-outerplanar graphs.

In the second part of the talk we present a deterministic algorithm for embedding dynamically growing binary trees into the hypercube. The algorithm migrates vertices of the binary tree to different hypercube nodes if the embedding in a subrange of the tree becomes congested. If the binary tree grows by at most one leaf per step, our algorithm exhibits amortized overhead bounded by a constant factor.

Deterministic Routing of h-Relations on the Multibutterfly

by Andrea Pietracaprina

In this paper we devise an optimal on-line deterministic algorithm for routing h-relations on an N-input/output multibutterfly. The algorithm, which is obtained by generalizing the circuit-switching techniques of Arora, Leighton, and Maggs (1996), routes any h-relation with messages of X bits, in $O(h(X+\log N))$ steps in the bit model, and in $O(h\lceil X/\log N\rceil+\log N)$ communication steps (which account only for link transfers) in the word model. Unlike other recently developed algorithms, our algorithm does not need extra levels of expanders added to the multibutterfly, thus reducing the overall network layout area. Moreover, the network topology does not depend on h.

QSM: A General Purpose Shared-Memory Model for Parallel Computation

by Vijaya Ramachandran

A fundamental challenge in parallel processing is to develop effective models for parallel computation that balance simplicity, accuracy, and broad applicability. In particular, a simple "bridging" model, i.e., a model that spans the range from algorithm design to architecture to hardware, is an especially desirable one.

We propose the Queuing Shared Memory (QSM) model as a bridging shared-memory model for parallel computation. The QSM provides a high-level shared-memory abstraction for parallel algorithm design, as well as the capability to model bandwidth limitations and other features of current parallel machines, as evidenced by a randomized work-preserving emulation of the QSM on the BSP, which is a lower-level, distributed-memory model. In addition to the QSM model and its emulation on the BSP, we present algorithmic results for several basic problems on the QSM.

How Parallel Programs Execute on Next Generation Parallel Machines

by Larry Rudolph

The next generation parallel machines will consist of a collection of commodity symmetric multiprocessors (SMPs), a high speed network, and an intellegent processor to network interface unit (NIU). The naive way to program such a machine is to exploit shared memory within the SMP and message-passing between SMP sites. However, parallel programs should not know about such a structure. Instead, they should scale naturally across SMP boundries.

We show that message-passing is really just an efficient implementation of a shared queue. Such a shared queue, as well as many other shared memory operations are a special case of "memory-with-strange-semantics." In fact, because of memory consistency problems arising from compiler and microprocessor optimizations, all of shared memory has strange semantics. It is thus better that the programmer is aware of these semantics.

The parallel job scheduler will also affect how programs execute. A good scheduler will dynamically identify those jobs that should be gang-scheduled, those that may, and those that may-not be.

Finally, we describe the MIT StarT-Voyager machine that is currently under construction. We show how it implements memory-with-strange-semantics, for message passing and for distributed shared memory. We also describe its parallel job scheduler.

On Batcher's Merge Sorts as Parallel Sorting Algorithms

by Christine Rüb

We examine the average running times of Batcher's bitonic merge sort and Batcher's odd-even merge sort when they are used as parallel merging algorithms. It can be shown that the average running time of these two algorithms can be improved considerably by keeping the amount of communication at a minimum and by always sending the smaller elements to the processor with the smaller index. Here we average over all possible permutations of the input.

To derive upper bounds for the average running times of the two sorting algorithms we first examine the two underlying merging algorithms. To make sure that the second condition from above is fulfilled, the two sorted input sequences have to be stored alternatively at the processors – in the case of bitonic merge this means that the underlying network is the balanced merge network. We show that for both merging algorithms the running time for a specific input is closely related to the maximum rank distance for elements in the two input sequences. Then we prove upper bounds on the average rank distance for two randomly generated sorted sequences. This leads to a bound of $\Theta((n/p)(1+\log(1+p^2/n)))$ for the average running time of odd-even merge and of $O((n/p)(1+\log(1+p^2/n)))$ for the average running time of bitonic merg (here n is the size of the input and p is the number of processors used). Additionally we show that the probability that j more steps than the average number of steps is executed is bounded by $(1/2)^{(2^j)^2}$ for both algorithms. From this follows that the average running time of the two sorting algorithms is not much larger than the sum of the average running times of the recursive calls to the merging algorithm. Namely, we show that these average running times are bounded by $O((n/p)(\log n+(\log(1+p^2/n))^2))$.

We also present experimental results, obtained by a simulation program, for various sizes of input and numbers of processors.

Parallel Tree-Contraction and Fibonacci Numbers

by Wojciech Rytter (joint work with W.Plandowski and T.Szymacha)

We show a new property of **Fibonacci numbers** which is related to the analysis of a very simple and natural parallel tree contraction algorithm called the *Simultaneous Substitutions* method. We show that the size of the smallest tree which requires t contractions equals exactly the t-th *Fibonacci number*. This implies the **sharp bound** on the number of iterations of the tree contraction algorithm to be $\log_\Phi n+const$, where Φ is the golden ratio and Φ is the smallest constant with this property. We contribute also to combinatorics of trees.

A Distributed Scheduling Algorithm for Parallel Discrete Event Simulation

by Uwe Schwiegelshohn (joint work with Edwin Naroska)

Assume a parallel processor with distributed memory and a discrete event simulation problem. The simulation problem is partitioned and distributed onto the nodes of the parallel processor. The processes assigned to a node are to be scheduled such that causality is preserved (conservative simulation) and blocking of processes on other nodes is avoided as much as possible. At the same time the amount of communication between the nodes must be limited to prevent network congestion. We propose to use a non blocking Chandy–Misra–Bryant algorithm with null messages. The number of null messages is limited by restricting them to edges (signal links) between processes of different partitions. These null messages provide so called lookahead information, i.e. the earliest time instant any process on a partition may cause an event at the target process of a second partition. Now, all available events on a node are processed in order of increasing lookahead instead of using the time stamps of the events directly. The lookahead computation is done by using precomputed shortest path data as all time delays between the processes are assumed to be constant. In case of a VLSI simulation problem (e.g. VHDL) the amount of computation can further be reduced by taking into account activation edges which represent clock signals. Finally, some simulation and speed–up results are presented.

Solving List Ranking Almost as Fast as Sorting

by Jop F. Sibeyn

Novel algorithms are presented for parallel and external memory list-ranking. The same algorithms can be used for computing basic tree functions, such as the depth of a node.

The parallel algorithm stands-out by its low memory use, its simplicity and its performance. For a large range of problem sizes, it is as fast as the fastest previous algorithms. On a Paragon with 100 PUs, each holding 10^6 nodes, we obtain speed-up 30.

For external-memory list-ranking, the best algorithm so far is an optimized version of independent-set-removal. Actually, this algorithm is not good at all: for a list of length N, the paging volume is about $72 \cdot N$. Our new algorithm reduces this to $18 \cdot N$. The algorithm has been implemented, and the theoretical results are confirmed. Programs are available from `http://www.mpi-sb.mpg.de/~jopsi/dprog/prog.html`.

Application-specific Resource Management in Darwin

by Peter Steenkiste

We envision the deployment of an electronic services market that will deliver a wide range of electronic services over the network. This market will allow applications to combine resources at endpoints with resources inside the network to deliver high-quality products to end-users. Electronic services will range from simple data delivery services to sophisticated value-added services such as video conferencing and data mining. Complex services will often be implemented in terms of lower-level services, so the market will have a hierarchical structure. The Darwin project is developing a comprehensive set of resource management mechanisms supporting such a market based on three basic mechanisms. The resources allocated to an application will be integrated in a virtual application mesh that forms the basis of runtime resource management and quality of service optimization. Each resource is managed by a hierarchical resource manager that satisfies the combined priorities and constraints of the services and applications sharing the resource. Finally, since both network conditions and application requirements can change, application-specific adaptation is needed to optimize quality of service at runtime.

In this talk, we review the goals of the Darwin project, describe the three resource management mechanisms used in Darwin, and outline some of the algorithmic problems raised by the project.

Practical Membership Protocols

by Ray Strong (joint work with John Palmer and Eli Upfal)

The context for this work is the management of regular parallel computations consisting of alternating phases of computation and communication among a group of member processes. In this context a membership protocol is used, together with failure detection protocol in order to allow a parallel computation to reorganize in response to slow or crashed members. A participant in a membership protocol takes a membership view (set of names of members) as input, exchanges views with other members, and is intended to output a coordinated view of the membership as quickly and consistently as possible. The ideal membership protocol would be fast, safe, andnontrivial. Here fast means guaranteed termination within a small number of timed view exchanges. The number may depend on the number of members but not on their behavior. The safety property is symmetric: if Alice and Bob are members with differing views, then Alice is not in Bob's view and Bob is not in Alice's view. In a WDAG96 paper, Dwork, Ho, and Strong showed that there is no ideal membership protocol. In this paper, we provide a fast membership protocol that satisfies the following asymmetric safety property: if Alice and Bob are members with differing output views, then either Alice is not in Bob's view or Bob is not in Alice's view. Moreover, if no failures are causally concurrent with its execution, then the protocol terminates within at worst 3 rounds of view exchange with symmetric safety and preserves a maximal clique of members that communicate unhampered by earlier failures. If failures are neither causally concurrent nor causally prior to its execution, then the protocol terminates in one round, preserving the entire membership.

Parallel Computing with Optical Interconnects

by Ted H. Szymanski

Continuing increases in microprocessor technology will place heavy demands on the interconnection networks of the future. There is a growing consensus that bit-parallel optical interconnects will appear in commercial systems wihtin a few product generations. In this talk, we consider how bit-parallel optical interconnects can impact shared memory multiprocessors in the near future. A one dimensional reconfigurable optical ring-like network, which is under development in Canada, is described. The network can support 10 .. 100's of bit-parallel channels. Each channel supports full broadcasting, and can also be partitioned into multiple independent smaller segments. The one dimensional optical network can be generalized to two or more dimensions. Such a network can in principle support 10's of terabits of low latency bandwidth, and can potentially act as an enabling technology for a new generation of bandwidth intensive computing machines.

Allocating Independent Tasks to Parallel Processors: An Experimental Study

by Torben Hagerup

We study a scheduling or allocation problem with the following characteristics: The goal is to execute a number of unspecified tasks on a parallel machine in any order and as quickly as possible. The tasks are maintained by a central monitor that will hand out batches of a variable number of tasks to requesting processors. A processor works on the batch assigned to it until it has completed all tasks in the batch, at which point it returns to the monitor for another batch. The time needed to execute a task is assumed to be a random variable with known mean and variance, and the execution times of distinct tasks are assumed to be independent. Moreover, each time a processor obtains a new batch from the monitor, it suffers a fixed delay. The challenge is to allocate batches to processors in such a way as to achieve a small expected overall finishing time. We introduce a new allocation strategy, the Bold strategy, and show that it outperforms other strategies suggested in the literature in a number of simulations.

Dynamic Analysis of Communication Processes

by Eli Upfal

Most theoretical work on communication networks has focused on batch, or static routing: A set of packets is injected into the system at time 0, and the routing algorithm is measured by the time it takes to deliver all these packets to their destinations, assuming that no more packets are injected into the system in the meantime. This communication paradigm leads to a reach and interesting theory but rarely reflects the practical reality of communication networks. Most real-life networks operate in a *dynamic* mode whereby new packets are continuously injected into the system. Each processor usually controls only the rate at which it injects its own packets and has only a limited knowledge of the global state. This situation is better modeled by a stochastic paradigm whereby packets are injected to the system according to some distribution, and the routing algorithm is evaluated according to its long term behavior. In particular, quantities of interest are the maximum arrival rate for which the system is stable (that is, arrival rate that ensures that the expected number of packets waiting in queues does not grow with time), and the expected time a packet spends in the system in the steady state.

In this talk we survey several recent results on dynamic analysis of communication algorithms for packet routing, deflection routing, and virtual circuit switching routing.

From Algorithm Parallelism to Instruction-Level Parallelism: An Encode-Decode Chain Using Prefix-Sum

by Uzi Vishkin

A novel comprehensive and coherent approach for the purpose of increasing instruction-level parallelism (ILP) is devised. The key new tool in our envisioned system update is the addition of a parallel prefix-sum (PS) instruction, which will have efficient implementation in hardware, to the instruction-set architecture. This addition gives for the first time a concrete way for recruiting the whole knowledge base of parallel algorithms for that purpose. The potential increase in ILP is demonstrated by experimental results for a test application.
The main technical contribution is in the form of a "completeness theorem". Perhaps surprisingly, the current abstract proves that in an envisioned system which employs parallel PS functional units, a proper use of a serial programming language suffices for the following. With a moderate effort, one can program a parallel algorithm (in a serial language), so that a parallelizing compiler (even without run-time methods!) will be able to extract the same (i.e., "complete") ILP from such serial code as from code written in a parallel language. Alternatively, rather than have the programmer produce the serial code, a pre-compiler could derive it from a parallel language. The most interesting idea in the proof is the reliance on the new parallel PS for circumventing collision-ambiguity in references to memory. Other new ideas in the paper include hardware-design of a prefix-sum unit and an on-line algorithm for high-bandwidth register-files.
An informal upshot of this paper is the following general insight: to accommodate paral-

lelism in uniprocessor systems (from algorithms to ILP), it is sufficient to *only* add (and, of course, incorporate) parallel prefix-sum functional units to standard serial system designs.

Exploiting Locality for Data Management in Systems of Limited Bandwidth

by Berthold Vöcking (joint work with B. Maggs, F. Meyer auf der Heide, and M. Westermann)

We introduce strategies for data management in computer systems in which the computing nodes are connected by a relatively sparse network. In particular, we consider the problem of placing and accessing a set of shared objects that are read and written from the nodes in the network. These objects are, e.g., global variables in a parallel program, pages or cache lines in a virtual shared memory system, shared files in a distributed file system, or pages in the World Wide Web. A data management strategy consists of a placement strategy that maps the objects (possibly dynamically and with redundancy) to the nodes, and an access strategy that describes how reads and writes are handled by the system (including the routing). We investigate static and dynamic data management strategies.

In the static model, we assume that we are given an application for which the rates of read and write accesses for all node–object pairs are known. The goal is to calculate a static placement of the objects to the nodes in the network and to specify the routing such that the network congestion is minimized. We introduce efficient algorithms that calculate optimal or close–to–optimal solutions for tree–connected networks, meshes of arbitrary dimension, and internet–like clustered networks. These algorithms take time only linear in the input size.

In the dynamic model, we assume no knowledge about the access pattern. An adversary specifies accesses at runtime. Here we develop dynamic caching strategies that also aim to minimize the congestion on trees, meshes, and clustered networks. These strategies are investigated in a competitive model. For example, we achieve competitive ratio 3 for tree–connected networks and competitive ratio $O(d \cdot \log n)$ for d–dimensional meshes of size n. Further, we present an $\Omega(\log n/d)$ lower bound for the competitive ratio for on–line routing in meshes, which implies that the achieved upper bound on the competitive ratio for meshes of constant dimension is optimal.

Sorting on a Massively Parallel System Using a Library of Basic Primitives: Modeling and Experimental Results

by Rolf Wanka (joint work with Alf Wachsmann)

A comparative study of implementations of the following sorting algorithms on the Parsytec SC320 reconfigurable, asynchronous, massively parallel MIMD machine was presented: Bitonic Sort, Odd-Even Merge Sort, Odd-Even Merge Sort with guarded split&merge, and two variants of Samplesort. The experiments are performed on 2- up to 5-dimensional wrapped butterfly networks with 8 up to 160 processors. We make use of library functions that provide primitives for global variables and synchronization, and we show that it is possible to implement efficient and portable programs easily. In order to predict the performance, we model the runtime of an access to a global variable by a certain trilinear function and the runtime of a synchronization by a certain bilinear function. Our experiments show that, in the context of parallel sorting, this model that can be applied easily is sufficiently detailed to give good runtime predictions. The experiments confirming the predictions point out that Odd-Even Merge Sort with guarded split&merge is the fastest method if the processors hold few keys. If there are many keys per processor, a combination of Samplesort and Odd-Even Merge Sort is the fastest method.

Open Problems

(posed during an open problems session on wednesday evening).

Ernst Mayr

We consider the problem of embedding 1-1 binary trees into hypercubes. The minimal hypercube whose size is at least that of the tree is called the *optimal* hypercube for the tree. The dilation of an embedding is the maximal length of a path to which a tree edge gets mapped.

1. (Havel's conjecture) Can all binary trees be embedded into their optimal hypercube with dilation 2?
2. Try to find an efficient embedding algorithm for embedding binary trees into their optimal hypercube with dilation ≤ 4.
3. Is it true that all balanced binary trees are subgraphs of their optimal hypercube?

Friedhelm Meyer auf der Heide

A pyramid graph P_m consists of nodes $V_m = \{\langle l, i\rangle, l = 1, \ldots, m, i = 1, \ldots, l\}$ and directed edges $\{(\langle l+1, i\rangle, \langle l, j\rangle), l = 1, \ldots, m-1, j \in \{i-1, i\}\}$.

Consider the following game, for a fixed number p of tokens:
The game starts on an empty graph on nodes V_m. In each round:

- Player A marks p not yet marked nodes of V_m.
- Player B inserts one of the two edges of P_m going into node x, for each of the p nodes x marked by player A.

Goal of player A is to construct a path from a node on level m to the root $(1, 1)$ of P_m using as few rounds as possible. Player B acts as an adversary.

Question: How many rounds are necessary, how many are sufficient in an optimal strategy of player A?

It is easily seen that, for $p = \frac{q(q+1)}{2}$, $\lceil \frac{l}{q} \rceil$ rounds are sufficient. Is this best possible?

The game can be used to prove a lower bound for the speedup attainable when simulating a 2-dimensional Turing machine on a p-processor parallel machine. Optimality of the strategy mentioned above would show that the speedup is $O(\sqrt{p})$ only.

Andrea Pietracaprina

In the last decade considerable effort has been spent in defining "good" models for parallel computation. As a result, several proposals have been made, although there is not yet a general agreement that clearly identifies one of the proposed models as "The Model". While models are often evaluated based on qualitative considerations, rigorous criteria for their quantitative evaluation are needed. In fact, some effort has been spent in showing experimentally that the cost functions associated with certain models provide exact performance predictions. This is typically accomplished by comparing predicted and actual running times on sets of benchmark applications.

However, in order to comply with simplicity and portability requirements, a model of computation is likely to provide only asymptotic estimates of performance, giving up the ability of predicting exact running times. In this case, it becomes hard to carry out an experimental validation of the model. What is needed is some reasonable, widely accepted, general mechanism of "proving" that a model accurately predicts asymptotic behaviour. So the question is whether there exist benchmark applications or test suites, and general evaluation criteria, that can be employed to assess the validity of a computational model, or to compare different models, with respect to asymptotic analysis.

The above problem stems from an undergoing joint research with N. Amato and G. Pucci.

Christine Rüb

Suppose you are given a permutation on a hypercube with the following property: If an element has to be sent from processor P_i to processor P_j, then the hamming distance between i and j is bounded by r. Is it possible to realize such a permutation in time $O(r)$?

Ted Szymanski

The Impact of Optics in the Next Millenium

The following problem is somewhat philosophical and discusses the potential impact of optics in future computing machines.

Prior to the days of the transistor, computing machines were constructed with vacuum tubes and discrete wires and were based on a serial stored-program paradigm. Formal models for such machines evolved. Several decades after the introduction of the transistor, we observe that the same models of computing machines are highly relevant. It seems that the introduction of the transistor, followed by the integrated circuit, has had a tremendous influence on the packaging and commercial viability of computing machines, and has facilitated the introduction of large parallel machines and their associated formal models, but has had relatively little influence on the formal models of individual computing machines.

An interesting question deals with the potential impact of optics in future computing machines. With reasonable likelyhood, optics will over time evolve to span optical links between electronic boards, followed by optical links between integrated circuits, perhaps followed by optical links between logic gates within an integrated circuit. All of these topics have been addressed in the literature. It is thus interesting to contemplate how optics will impact formal models of computing systems in the future. Optics is expected to have a tremendous impact in the area of interconnections. For example, it is technologically feasible to replace the conventional bandwidth-limited electronic mesh by an optical mesh-like network with hundreds or thousands of high-bandwidth optical channels in each row or column. Optics may also have a tremendous impact in the area of bit-parallel memory systems. For example, massively bit-parallel optical memory systems based on holographic techniques are expected to evolve over time. Optics may thus facilite new classes of parallel interconnects or parallel machines and their formal models. The interesting question is whether optics and integrated optics will result in new formal models for individual computing machines, or will optics follow the path of the transistor, and have a tremendous impact on packaging of computing machines, and facilite new classes of large parallel systems of computing machines and their formal models, but have relatively little impact on formal models of individual computing machines. Perhaps time will tell.

E-Mail Addresses

Jean-Loup Baer	baer@cs.washington.edu
Yosi Ben-Asher	yosi@mathcs2.haifa.ac.il
Artur Czumaj	artur@uni-paderborn.de
Ralf Diekmann	diek@uni-paderborn.de
Martin Dietzfelbinger	dietzf@ls2.informatik.uni-dortmund.de
Pierre Fraigniaud	pierre@lri.fr
Hubertus Franke	frankeh@watson.ibm.com
Thomas Gross	Thomas.Gross@cs.cmu.edu
Torben Hagerup	torben@mpi-sb.mpg.de
Ben Juurlink	benj@uni-paderborn.de
Clyde P. Kruskal	kruskal@cs.umd.edu
Ludek Kucera	ludek@kam.ms.mff.cuni.cz
Miroslaw Kutylowski	mirekk@uni-paderborn.de
Klaus-Jörn Lange	lange@informatik.uni-tuebingen.de
Krzysztof Lorys	lorys@ii.uni.wroc.pl
Ernst W. Mayr	mayr@informatik.tu-muenchen.de
Friedhelm Meyer auf der Heide	fmadh@uni-paderborn.de
Andrea Pietracaprina	andrea@artemide.dei.unipd.it
Christine Rüb	rueb@mpi-sb.mpg.de
Vijaya Ramachandran	vlr@cs.utexas.edu
Larry Rudolph	rudolph@lcs.mit.edu
Wojciech Rytter	rytter@mimuw.edu.pl
Uwe Schwiegelshohn	uwe@ds.e-technik.uni-dortmund.de
Jop Sibeyn	jopsi@mpi-sb.mpg.de
Peter Steenkiste	prs@cs.cmu.edu
H. Raymond Strong	strong@almaden.ibm.com
Ted Szymanski	teds@macs.ee.mcgill.ca
Pilar de la Torre	dltrr@cs.unh.edu
Eli Upfal	ely@almaden.ibm.com
Berthold Vöcking	voecking@uni-paderborn.de
Uzi Vishkin	viskin@umiacs.umd.edu
Rolf Wanka	wanka@uni-paderborn.de

Addresses

Prof. Jean-Loup Baer
University of Washington
Department of Computer Science and Engineering
P.O. Box 352350
Seattle WA 98195-2350
USA
☎ +1-206-685-1376
Fax: +1-206-543-2969

Dr. Yosi Ben-Asher
University of Haifa
Dept.of Mathematics and Computer Science
31905 Haifa
Israel
Fax: +972-4-240-024

Dr. Artur Czumaj
Universität-GH Paderborn
Heinz Nixdorf Institut
FB 17 - Mathematik/Informatik
Fürstenallee 11
33102 Paderborn
Germany
☎ +49-5251-60-6491
Fax: +49-5251-60-6482

Ralf Diekmann
Universität-GH Paderborn
FB 17 - Mathematik/Informatik
Fürstenallee 11
33102 Paderborn
Germany
☎ +49-5251/60-67 30
Fax: +49-5251/60-66 97

Prof. Dr. Martin Dietzfelbinger
Universität Dortmund
FB Informatik
Lehrstuhl II
D-44221 Dortmund
Germany
☎ +49-231-755-47 37
Fax: +49-231-755-20 47

Pierre Fraigniaud
Laboratoire de Recherche en Informatique
Universitè Paris XI
F-91405 Orsay Cedex
France
☎ +33-1-6915-6906

Thomas Gross
Carnegie Mellon University
School of Computer Science
Schenley Park
Pittsburgh PA 15213-3980
USA
☎ +1-412-268-7661
Fax: +1-412-268-5576

Dr. Torben Hagerup
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
Germany
☎ +49-681-9325-108
Fax: +49-681-9325-199

Dr. Ben Juurlink
Universität-GH Paderborn
Heinz Nixdorf Institut
FB 17 - Mathematik/Informatik
Fürstenallee 11
33102 Paderborn
Germany
☎ +49 5251 60 64 57
Fax: +49 5251 60 64 82

Prof. Clyde P. Kruskal
Univ. of Maryland at College Park
Department of Computer Science
A.W. Williams Building
College Park MD 20742
USA
☎ +1-301-405-2683
Fax: +1-301-405-6707

Dr. Ludek Kucera
Charles University
Department of Applied Mathematics
Malostranske nam. 25
118 00 Praha 1
Czech Republic
☎ +420-2-2191-4233
Fax: +420-2-451-0995

Dr. habil Miroslaw Kutylowski
Universität-GH Paderborn
Heinz Nixdorf Institut
FB 17 - Mathematik/Informatik
Fürstenallee 11
33102 Paderborn
Germany
☎ +49-5251-60-6461
Fax: +49-5251-60-64 82

Prof. Dr. Klaus-Jörn Lange
Universität Tübingen
Wilhelm-Schickard-Institut für Informatik
Sand 13
D-72076 Tübingen
Germany
☎ +49-7071-297-75 67
Fax: +49-7071-6 81 42

Krzysztof Lorys
Universytet Wroclawski
Instytut Informatyki
Przesmyckiego 20
PL-51-151 Wroclaw
Poland

Prof. Dr. Ernst W. Mayr
TU München
Institut für Informatik
80290 München
Germany
☎ +49-89-289-2 26 80 /-2 26 81 (Secr.)
Fax: +49-89-289-2 52 97

Prof. Dr. Friedhelm Meyer auf der Heide
Universität-GH Paderborn
Heinz Nixdorf Institut
FB 17 - Mathematik/Informatik
Fürstenallee 11
33102 Paderborn
Germany
☎ +49-5251-60-6480
Fax: +49-5251-60-6482

Dr. Andrea Pietracaprina
Università di Padova
Dipartimento di Matematica Pura
e Applicata
Via Belzoni 7
35131 Padova
Italy
☎ +39-49-827-5987
Fax: +39-49-875-8596

Dr. Christine Rüb
Max-Planck-Institut für Informatik
Im Stadtwald
66123 Saarbrücken
Germany
☎ +49-681-9325-125
Fax: +49-681-9325-999

Prof. Dr. Vijaya Ramachandran
University of Texas at Austin
Department of Computer Sciences
Taylor Hall 2-124
Austin TX 78712-1188
USA
☎ +1-512-471-95 54
Fax: +1-512-471-88 85

Prof. Dr. Larry Rudolph
Massachusetts Institute of Technology
Lab for CS
545 Technology Square
Cambridge MA 02139 USA
☎ +1-617-253-6562
Fax: http://www.csg.lcs.mit.edu/ rudolph

Prof. Dr. Wojciech Rytter
University of Warsaw
Dept. of Mathematics & Informatics
Institute of Informatics
Ul. Banacha 2
02-097 Warsaw 59
Poland
☎ +48-22-6583165
Fax: +48-22-6583164

Prof. Dr.-Ing. Uwe Schwiegelshohn
Universität Dortmund
Lehrstuhl für Datenverarbeitungssysteme
44221 Dortmund
Germany
☎ +49-231-755-26 34
Fax: +49-231-755-32 51

Dr. Jop Sibeyn
Max-Planck-Institut für Informatik
Algorithms and Complexity Group (AG1)
Postfach 15 11 50
D-66041 Saarbrücken
Germany
☎ +49-681-9325 118
Fax: +49-681-9325 199

Peter Steenkiste
Carnegie Mellon University
School of Computer Science
5000 Forbes Avenue
Pittsburgh PA 15213-3891
USA
☎ +1-412-268-3261
Fax: +1-412-268-6787

H. Raymond Strong
IBM Almaden Research
Dept. K53/802
650 Harry Road
San Jose CA 95120-6099
USA
☎ +1-408-927-1758
Fax: +1-408-927-3030

Prof. Pilar de la Torre
University of New Hampshire
Department of Computer Science
M206 Kingsbury Hall
Durham NH 03824
USA
☎ +1-603-862-2682
Fax: +1-603-862-3493

Berthold Vöcking
Universität-GH Paderborn
Heinz Nixdorf Institut
FB 17 - Mathematik/Informatik
Fürstenallee 11
33102 Paderborn
Germany
☎ +49-5251-60-6433
Fax: +49-5251-60-6482

Dr. Rolf Wanka
Universität-GH Paderborn
FB 17 - Mathematik/Informatik
Fürstenallee 11
33102 Paderborn
Germany
☎ +49-5251-60-6434
Fax: +49-5251-60-6482

Prof. Ted Szymanski
McGill University
Department of Electrical Engineering
3480 University Street
Montreal PQ H3A 2A7
Canada
☎ +1-514-398-5934
Fax: +1-514-398-4470

Dr. Eli Upfal
IBM Almaden Research
Dept. K53/802
650 Harry Road
San Jose CA 95120-6099
USA
☎ +1-408-927-1788
Fax: +1-408-927-3030

Prof. Dr. Uzi Vishkin
Univ. of Maryland at College Park
Inst. for Advanced Computer Studies
A.V. Williams Bldg.
College Park MD 20742-3251
USA
☎ +1-301-405-6763
Fax: +1-301-314-9658