MATHEMATICAL AND COMPUTATIONAL SCIENCES

Publications

Titles and Abstracts

April 1989

THINKING MACHINES CORPORATION

MATHEMATICAL AND
COMPUTATIONAL SCIENCES

Publications

Titles and Abstracts

April 1989

THINKING MACHINES
CORPORATION

Four Vector-Matrix Primitives

Thinking Machines Corporation
Technical Report
Ajit Agrawal, Guy E. Blelloch, Robert L. Krawitz, Cynthia A. Phillips
April 1989

Abstract

This paper describes four APL-like primitives for manipulating dense matrices and vectors and describes their implementation on the Connection Machine[1] hypercube multiprocessor. These primitives provide a natural way of specifying parallel matrix algorithms independently of machine size or architecture. We illustrate their use in three numerical algorithms: a vector-matrix multiply, a Gaussian-elimination routine and a simplex algorithm. We describe implementations of the primitives assuming load-balanced embeddings of matrices and vectors on a hypercube multiprocessor architecture. The primitives may indicate a change from one embedding to another. The implementations are efficient in the frequently occurring case where there are fewer processors than matrix elements. In particular if there are $m > p \lg p$ matrix elements, where p is the number of processors, then the implementations of some of the primitives are asymptotically optimal in that the processor-time product is no more than a constant factor higher than the running time of the best serial algorithm. Furthermore, the parallel time required is optimal to within a constant factor.

We have implemented the primitives on the Connection Machine System, and this implementation improved the running time of some of our applications by almost an order of magnitude over a naive implementation. We give Connection Machine timings for the primitives and the algorithms.

To appear in the proceedings of *First Annual ACM Symposium on Parallel Algorithms and Architectures*

[1]Connection Machine System is a registered trademark of Thinking Machines Corporation.

Node Orderings and Concurrency in Structurally-symmetric Sparse Problems

Yale University
Department of Computer Science
Technical report
Iain S Duff and S. Lennart Johnsson
March 1989

Abstract

In the solution of structurally-symmetric sparse linear systems by direct methods it is possible to exploit concurrency not only within the elimination operations at a single pivot step but also over the eliminations using different pivots. The pivot ordering that minimizes the number of arithmetic operations does not, in general, minimize the time for the concurrent solution of a system of equations. We investigate minimum degree and nested dissection orderings, as well as a few other ordering schemes, with respect to potential solution time and total arithmetic complexity for a few benchmark problems and an idealized parallel computer. The benchmark problems include 2-dimensional grid problems and some other standard sparse matrix test problems.

Presented at the *First International Conference on Vector and Parallel Processing*, Loen, Norway, June 1986

To appear in *Algorithms for Vector and Parallel Supercomputers*, Wiley, 1989.

Solving the Wide Angle Wave Equation on a Data Parallel Computer

Thinking Machines Corporation
Technical Report
S. Lennart Johnsson, Anne Trefethen, and David Browning
In preparation

Abstract

Three-dimensional acoustics problems are computationally very demanding for an accurate computation of the field over the frequency range and domain of interest in many applications. Even with todays supercomputers simulations require very long execution times. It is important to use current and future supercomputers effectively for an accurate modeling of the problem. In this paper we focus on the implementation of the method for solving the three-dimensional wide-angle wave equation proposed by Lee and others on the Connection Machine system. The method makes use of an Alternating Direction Method for solving the parabolic approximation to the wave equation. On the Connection Machine system we have implemented the algorithm using substructuring and odd-even cyclic reduction for the solution of the reduced set of equations. The address space of the Connection Machine is configured as a two-dimensional array. The computed solution for a model problem is compared with the exact solution. We present some highlights of the implementation, including performance measurements for different grid sizes and mappings thereof to the Connection Machine.

To appear in the Proceedings of the *Second IMACS Symposium on Computational Acoustics*, 1989.

A Dataparallel Implementation of an Explicit Method for the Three-Dimensional Compressible Navier-Stokes Equations.

Thinking Machines Corporation
Technical Report CS-89/4
Pelle Olsson and S Lennart Johnsson
March 1989

Abstract

The fluid flow in a three dimensional, twisted channel, is modeled by both the full, Navier-Stokes equations and the Euler equations. For both models we use an explicit finite difference method with a three stage Runge-Kutta method for integrating the system of equations in time. A second-order accurate, centered difference scheme is used for spatial derivatives of the flux variables. For the Euler equations as well as the Navier-Stokes equations artificial viscosity is introduced in order to stabilize the numeric scheme. The artificial viscosity introduces fourth order centered differences into the discrete, numeric scheme.

The explicit method has been implemented on the Connection Machine model CM–2, a data parallel computer configurable with up to 65,536 processors and 512 Mbytes of primary storage. A performance of 1.05 Gflops/s, single-precision, was demonstrated for a fully configured Connection Machine system. The performance scales in proportion to the number of processors. The performance on 8k processor configurations was 135 Mflops/s, on 16k processors 265 Mflops/s, and 525 Mflops/s on 32k processors. The efficiency is independent of the machine size, as expected. A considerable performance improvement is expected with efficient implementation of functional kernels such as convolution, and matrix-vector multiplication.

A Study of Dissipation Operators for the Euler Equations and a Three-dimensional Channel Flow.

Thinking Machines Corporation
Technical Report CS-89/3
Pelle Olsson and S. Lennart Johnsson
March 1989

Abstract

Explicit methods for the solution of fluid flow problems are of considerable interest in supercomputing. These methods parallelize well. The treatment of the boundaries is of particular interest both with respect to the numeric behavior of the solution, and the computational efficiency. We have solved the three-dimensional Euler equations for a twisted channel using second-order, centered difference operators, and a three stage Runge-Kutta method for the integration. Three different fourth-order dissipation operators were studied for numeric stabilization: one positive definite, one positive semidefinite, and one indefinite. The operators only differ in the treatment of the boundary. For computational efficiency all dissipation operators were designed with a constant bandwidth in matrix representation, with the bandwidth determined by the operator in the interior. The positive definite dissipation operator results in a significant growth in entropy close to the channel walls. The other operators maintain constant entropy.

Several different implementations of the semidefinite operator obtained through factoring of the operator were also studied. We show the difference both in convergence rate and robustness for the different dissipation operators, and the factorizations of the operator due to Eriksson. For the simulations in this study one of the factorizations of the semidefinite operator required 70 - 90% of the number of iterations required by the positive definite operator. The indefinite operator was sensitive to perturbations in the inflow boundary conditions. The simulations were performed on a 8,192 processor Connection Machine system model CM–2. Full processor utilization was achieved, and a performance of 135 Mflops/s in single precision was obtained. A performance of 1.1 Gflops/s for a fully configured system with 65,536 processors was demonstrated.

Embedding Meshes in Boolean Cubes by Graph Decomposition

Yale University
Department of Computer Science
Technical Report
Ching-Tien Ho and S. Lennart Johnsson
March 1989

Abstract

This paper explores the embeddings of multidimensional meshes into minimal Boolean cubes by graph decomposition. The dilation and the congestion of the product graph $(G_1 \times G_2) \rightarrow (H_1 \times H_2)$ is maximum of the two embeddings $G_1 \rightarrow H_1$ and $G_2 \rightarrow H_2$. The graph decomposition technique can be used to improve the average dilation and average congestion for existing mappings. One property used frequently in mesh embedding by graph decomposition is that a $\ell_1 \times \ell_2 \times \cdots \times \ell_k$ mesh is a subgraph of the product graph of the two meshes $\ell'_1 \times \ell'_2 \times \cdots \times \ell'_k$ and $\ell''_1 \times \ell''_2 \times \cdots \times \ell''_k$, if $\ell_i \leq \ell'_i \ell''_i$ for all $1 \leq i \leq k$. The graph decomposition technique combined with some particular two-dimensional embeddings allows for minimal expansion, dilation two, congestion two embeddings of about 87% of all two-dimensional meshes, asymptotically. With the graph decomposition technique, and some three-dimensional mappings presented in this paper, more than 96% of all three-dimensional meshes contained in a $512 \times 512 \times 512$ mesh can be embedded in a minimal Boolean cube with dilation two. The graph decomposition technique is also used to generalize the mesh embeddings to include wrap-around with an increase in the dilation by at most 1, compared to a mesh without wrap-around. The expansion is preserved for the majority of meshes, if a wrap-around feature is added to the mesh.

The Finite Element Method on a Data Parallel Computing System

Thinking Machines Corporation
Technical Report CS-89/2
Kapil K. Mathur and S. Lennart Johnsson
January 1989

Abstract

A data parallel implementation of the finite element method on the Connection Machine system CM–2 is presented. This implementation assumes that the elementary unit of data is an unassembled nodal point. In the context of the CM–2, each virtual processor represents an unassembled nodal point and nodal points shared between elements are replicated on different virtual processors. An algorithm for computing each elemental stiffness matrix concurrently, as well as different elemental stiffness matrices concurrently, without inter–processor communication is presented. The performance of the elemental stiffness matrix computation is in the range $1.6 - 1.9$ GFlops s^{-1}. The sparse system of linear equations that results from the finite element discretization has been solved by a conjugate gradient method with a diagonal preconditioner. The rate of convergence of the conjugate gradient iterations for boundary conditions which correspond to uniaxial deformations depends nonlinearly on the order of interpolation of the elements and linearly on the mesh discretization. Sample code segments are provided to illustrate the programming environment on a data parallel architecture.

To appear in the *Journal of High-Speed Computing*

A Numerical Formulation for Anisotropy in Metal Forming Analysis

Thinking Machines Corporation
Technical Report
Paul R. Dawson and Kapil K. Mathur
In preparation

Abstract

Changes in the mechanical state of metals typically accompany the deformations associated with forming operations. Often these changes substantially affect the properties of the product. A mathematical model for the large strain anisotropic deformation behavior of polycrystalline metals and its implementation in a finite element formulation for viscoplastic flow are summarized. The novel feature of this formulation is use of constitutive models that are rich in microstructural detail. Several issues related to use of such micromechanical models in deformation process simulation are discussed. Two primary forming operations are considered with an interest in the degree to which properties, especially the crystallographic texture, are modified by deformation. The processes considered are slab rolling and wire drawing. Part of the simulation results are orientation distributions of the grains in aggregates of material points of the workpiece. Such results show how the combined ideal and redundant deformations of a process affect the final product properties, an important advance over computations based on the ideal deformations alone.

Histogram Computation on Distributed Memory Architectures

Yale University
Department of Computer Science
Technical Report 682
Dimitris C. Gerogiannis, Stelios C. Orphanoudakis and S. Lennart Johnsson
January 1989

Abstract

One data-independent and one data-dependent algorithm for the computation of image histograms on parallel computers are presented, analyzed, and implemented on the Connection Machine system CM–2. The data-dependent algorithm has a lower requirement on communication bandwidth by only transferring bins with a non-zero count. Both algorithms perform *all-to-all reduction*, which is implemented through a sequence of exchanges as defined by a butterfly network. The two algorithms are compared based on predicted and actual performance on the Connection Machine CM–2. With few pixels per processor the data-dependent algorithm requires on the order of $\sqrt{B}$ data transfers for B bins compared to B data transfers for the data-independent algorithm. As the number of pixels per processor grows the advantage of the data-dependent algorithm decreases. The advantage of the data-dependent algorithm increases with the number of bins of the histogram.

Data Structures and Algorithms for the Finite Element Method on a Data Parallel Supercomputer

Thinking Machines Corporation
Technical Report CS-89/1
S. Lennart Johnsson and Kapil Mathur
January 1989

Abstract

This article describes the formulation and implementation of the finite element method on a data parallel computing system, such as the Connection Machine system. Data structures, storage requirements, communication and parallel arithmetic complexity are analyzed in detail for the cases when a processor is assigned to a finite element, and when a processor is assigned to a nodal point per element. Data parallel algorithms for grid generation, evaluation of the elemental stiffness matrices, and for the iterative solution of the linear system are presented. An algorithm for computing the elemental stiffness matrices concurrently, as well as computing the matrix elements of a single elemental stiffness matrix concurrently without communication is presented. A conjugate gradient solver with diagonal pre–conditioner is used for the solution of the linear system. Results from an implementation of the finite element method in three dimensions based on iso–parametric brick elements are also presented. For single–precision floating–point operations the measured peak performance is in the range 1.1 - 1.8 Gflops s/s for evaluating the elemental stiffness matrices and 0.5 - 0.7 Gflops s/s for the conjugate gradient solver. The time per conjugate gradient iteration for an application with $\sim 400,000$ degrees of freedom is approximately 1.25 s for double–precision (software) floating–point operations. With hardware support for double–precision floating–point operations, the time per conjugate gradient iteration for a finite element with with $\sim 400,000$ degrees of freedom is projected to be ~ 0.15 s.

Optimizing Tridiagonal Solvers for the Alternating Direction Method

Yale University
Department of Computer Science
Technical Report 679
S. Lennart Johnsson and Ching-Tien Ho
January 1989

Abstract

Sets of tridiagonal systems occur in many applications. Fast Poisson solvers and Alternate Direction Methods make use of tridiagonal system solvers. Network based multiprocessors provide a cost effective alternative to traditional supercomputer architectures. We investigate the complexity of concurrent algorithms for the solution of multiple tridiagonal systems on Boolean cube configured multiprocessors with distributed memory. Variations of odd-even cyclic reduction, parallel cyclic reduction, and algorithms making use of data transposition with or without substructuring and local elimination, or pipelined elimination are considered. A simple performance model is used for algorithm comparison, and the validity of the model is verified on an Intel iPSC/1. For many combinations of machine and system parameters, pipelined elimination, or equation transposition with or without substructuring is optimum. We present hybrid algorithms that at any stage choose the best algorithm among the considered ones for the remainder of the problem.

It is shown that the optimum partitioning of a set of independent tridiagonal systems among a set of processors yields the embarrassingly parallel case. If the systems originate from a lattice and solutions are computed in alternating directions, then to first order the aspect ratio of a computational lattice shall be the same as that of the lattice forming the base for the equations.

Our experiments demonstrate the importance of combining in the communication system for architectures with a relatively high communications start-up time.

To appear in the *SIAM Journal of Scientific and Statistical Computing.*

Protein Structure Prediction by Memory-base Reasoning

Thinking Machines Corporation
Technical Report
Xiru Zhang, David Waltz, Jill P. Mesirov
December 1988

Abstract

Memory-based reasoning (MBR) is a technique that makes intensive use of memory to recall some specific episodes from the past for problem solving. It is used in this research to predict protein structures based on 112 known structures selected from the Brookhaven Protein Databank. The ϕ and ψ angles of each amino acid in a protein are used to represent its 3-D structure. For this particular problem, we extend MBR to include a recursive procedure to refine its initial prediction and a varying "window" size to take into account the interaction between amino acids apart from different distances along the amino acid sequence. The system implemented, *PHI-PSI*, has been tested with all the available data. It does better than distribution-based guesses for most of the ϕ and ψ angle values.

Submitted to *IJCAI-89*

Protein Sequence Comparison on the Connection Machine CM-2

Thinking Machines Corporation
Technical Report
Robert Jones, Washington Taylor, Xiru Zhang, Eric Lander, Jill P. Mesirov

December 1988

Abstract

Dynamic programming algorithms provide a very sensitive method for comparing protein sequences but are computationally expensive when applied to sequence databases. Parallel computers offer one way in which to combine sensitivity and speed. Here we describe our implementation of one of these algorithms on a massively-parallel computer, the Connection Machine CM2, its performance and its use in studying relationships between proteins.

To appear in the proceedings of *The Interface between Computation Science and Nucleic Acid Sequencing*, 1988.

Embedding Hyper-Pyramids into Hypercubes

Yale University
Department of Computer Science
Technical Report 667
Ching-Tien Ho and S. Lennart Johnsson
December 1988

Abstract

A $\hat{P}(k,d)$ hyper-pyramid is a level structure of k Boolean cubes where the cube at level i is of dimension id, and a node at level $i-1$ connects to every node in a d dimensional Boolean subcube at level i, except for the leaf level k. Hyper-pyramids contain pyramids as proper subgraphs. We show that a $\hat{P}(k,d)$ hyper-pyramid can be embedded in a Boolean cube with minimal expansion and dilation 2. The congestion is bounded from above by $\frac{2^{d+1}}{d+1}$ and from below by $1+\lceil\frac{2^d-d}{kd+1}\rceil$. For $\hat{P}(k,2)$ hyper-pyramids we present a dilation 2 and congestion 2 embedding. In addition to expansion, dilation, and congestion we also characterize the embedding with the active-degree, and the node-load. The former property gives the maximum number of cube edges being used at any node, and the latter property measures the maximum number of messages a cube node needs to handle. The active degree for the embeddings is equal to the number of cube edges per node, i.e., $kd+1$, and the node-load is bounded from above by $O(2^d)+O(kd)$ with a congestion of $O(\frac{2^d}{d})$. For the $\hat{P}(k,2)$ hyper-pyramid embedding we present, the node-load is $2k+5$.

We also present embeddings of a $\hat{P}(k,d)$ hyper-pyramid together with 2^d-2 $\hat{P}(k,d)$ hyper-pyramids such that only one cube node is unused. The dilation of the embedding is $d+1$ with a congestion of $O(2^d)$. An alternate embedding with dilation $2d$ and congestion $O(\frac{2^d}{d})$ is also presented. The active-degree is $kd+1$ for both embeddings. The node-load is $O(d2^d)+O(kd)$ for the former and $O(2^d)+O(kd)$ for the latter embedding. Specialized to hyper-pyramids $\hat{P}(k,2)$ we present two embeddings: one with dilation 3, congestion 3 and a node-load of $3k+5$; the other with dilation 4, congestion 5 and a node-load of $2k+9$. As a corollary a complete n-ary tree can be embedded in a Boolean cube with dilation $\max(2,\lceil\log_2 n\rceil)$ and expansion $2^{k\lceil\log_2 n\rceil+1}/\frac{n^{k+1}-1}{n-1}$.

Experiences with Large Scale Network Optimization on the Connection Machine

The Wharton School
University of Pennsylvania
Cindy Phillips, Stavros A. Zenios
November 1988

Abstract

Network optimization problems appear in several areas of application from operations research, transportation, engineering design, financial planning and others. Such problems are characterized, quite often, by their very large size. Massively parallel computers like the Connection Machine (CM) appear to be well-suited for both sparse and dense implementations of dual relaxation algorithms for network optimization. In this report we summarize recent experiences with the solution of large scale network optimization problems using the CM. We discuss key features of the implementation of parallel algorithms for assignment and strictly convex nonlinear network problems and present results with numerical experiments.

To appear in *Impact of Recent Computer Advances on Operations Research*, Elsevier Science Publishing, Co.

Shuffle Permutations on Boolean Cubes

Yale University
Department of Computer Science
Technical Report 653
S Lennart Johnsson and Ching-Tien Ho
October 1988

Abstract

In this paper we prove lower bounds and present algorithms optimal within a small constant factor for *generalized shuffle permutations* on Boolean cubes. A *generalized shuffle permutation* is a permutation where a global address $(a_{q-1}a_{q-2}\ldots a_0)$ receives its new content from a global address $(a_{\delta(q-1)}a_{\delta(q-2)}\ldots a_{\delta(0)})$, with $\delta(\alpha_0)=\alpha_1, \delta(\alpha_1)=\alpha_2, \ldots, \delta(\alpha_{\sigma-1})=\alpha_0$ for $\alpha_i \in \{0,1,\ldots,q-1\}, \sigma \leq q$. For packet switched communication restricted to one port at a time per processor, the minimum number of communications in sequence is equal to the number of address bits to which the permutation is applied. The data transfer time of the permutation is proportional to the size of the data set per processor *and* the number of address bits being part of the permutation. With concurrent communication on all ports of every processor the data transfer time is proportional to the size of the data set per processor. Depending on communication capability, message size, cube size, data transfer rate, and communication start-up time, different algorithms must be chosen for a communication time optimal within a small constant factor. The analysis is verified by experimental results on the Intel iPSC/1.

A Deterministic Cellular Automaton with Diffusive Behavior

Thinking Machines Corporation
Technical Report
Bruce M. Boghosian, C. David Levermore
October 1988

Abstract

It is a classical result that an ensemble of independent unbiased random walks on the one dimensional lattice, Z, and moving at discrete times, Z^+, has a continuum limit given by a diffusion equation. More recently, systems of randomly walking particles interacting via an exclusion principle have been studied. Another interesting problem is that of using deterministic dynamical systems for the same purpose. Of course, to the extent that the underlying microscopic dynamics of atoms in real diffusing media are deterministic, we know that this should be possible.

In this work we describe a completely deterministic cellular automaton that exhibits diffusive behavior in one dimension, possibly with spatial inhomogeneity. We analyze this automaton both theoretically and experimentally to investigate its continuum limit. We experimentally find significant deviations from the Chapman-Enskog theory; these deviations are due to a buildup of correlations that invalidates the molecular chaos assumption.

Appeared in *Discrete Kinetic Theory, Lattice Gas Dynamics and Foundations of Hydrodynamics*, Torino, Italy, 19-23 September 1988

Stable Dimension Permutations on Boolean Cubes

Yale University
Department of Computer Science
Technical Report 617
Ching-Tien Ho and S. Lennart Johnsson
October 1988

Abstract

In this paper we present lower bounds and algorithms optimal within a small constant factor for *stable dimension permutations* on Boolean cubes. A *stable dimension permutation* is a permutation where a global address $(a_{q-1}a_{q-2}\ldots a_0)$ receives its new content from a global address $(a_{\delta(q-1)}a_{\delta(q-2)}\cdots a_{\delta(0)})$, where δ is a permutation function on $\{0,1,\ldots,q-1\}$. With communication restricted to one port at a time for each processor, the lower bound has a term proportional to the number of processor dimensions being part of the dimension permutation for the number of communications in sequence, and a term for the data transfer time that is proportional to the same number of dimensions *and* the size of the data set per processor. With concurrent communication on all ports of every processor, the bound for the data transfer time is reduced to become proportional only to the size of the data set per processor. We also show that for an optimal algorithm the time for a dimension permutation cannot be reduced by using the cube dimensions not being part of the dimension permutation, if data is allocated to the entire cube. However, if data is only allocated to a subcube, then the dimensions not being part of the subcube can be used to reduce the time complexity of the dimension permutation. The bandwidth of the Boolean cube is fully explored by dividing the data set to be communicated between a pair of processors into subsets, one for each path between the pair of processors. The *k-shuffle* permutation, the *bit-reversal* permutation, and matrix transposition, are special cases of *stable dimension permutations*. Depending on communication capability, message size, cube size, data transfer rate, and communication start-up time, different algorithms must be chosen for a communication time optimal within a small constant factor.

Expressing Boolean Cube Matrix Algorithms in Shared Memory Primitives

Yale University
Department of Computer Science
Technical Report 636
Ching-Tien Ho and S. Lennart Johnsson
July 1988

Abstract

Generic communication primitives can be used for many algorithms on Boolean cubes. Here we focus on expressing such primitives and algorithms for matrix multiplication in terms of shared memory type programming primitives. All processors share the same global address space. The communication primitives realize nearest-neighbor communication and global operations such as broadcasting from one processor to a set of processors, the reverse operation of plus-reduction, and matrix transposition (dimension permutation). We consider both the case where communication is restricted to one processor port at a time, and the case of concurrent communication on all processor ports. The communication algorithms are provably optimal within a factor of two. We describe both constant storage algorithms, and algorithms with reduced communication time, but a storage need proportional to the number of processors and the matrix sizes (for a one-dimensional partitioning of the matrices). The choice of the described matrix multiplication algorithms depends on machine size relative to the matrix sizes, the matrix shapes, and the architectural parameters of the machine.

Interactive Scientific Visualization and Parallel Display Techniques

Thinking Machines Corporation
Technical Report VZ88-1
J.A. Sethian, James B. Salem, A.F. Ghoniem
March 1988

Abstract

In this paper, we describe a new graphics environment for real-time visualization of the results of numerical simulations of computational fluid mechanics. Within this environment, the researcher may interactively perform real-time flow visualization experiments on numerical data which parallel those performed in the laboratory on physical apparatus. This provides an effective and interactive way to analyze the underlying physical mechanisms of the flow, and to compare results with laboratory experiment. The system is implemented on a data parallel supercomputer directly connected to a frame buffer. Since most fluid visualization techniques are highly parallel in nature, this allows us to obtain real-time display of fluid motion. The fluid diagnostic tools include display of moving color contours for scaler fields, and smoke/dye injection of passive tracer particles for velocity fields.

We demonstrate our interactive graphics fluid flow system by analyzing data generated from a numerical simulation of laminar and turbulent flow over a backwards-facing step. Input parameters are menu-driven, and we typically achieve approximately 15 screen updates per second, producing essentially real-time motion.

Appeared in *Proceedings of the Supercomputing Conference '88*, pp. 132-139

Systolic FFT Algorithms on Boolean Cube Networks

Yale University
Department of Computer Science
Technical Report 619
S. Lennart Johnsson, Ching-Tien Ho, Michel Jacquemin and Alan Ruttenberg
March 1988

Abstract

Systolic algorithms are typically devised for one or two dimensional arrays of processing elements. In this paper we describe a systolic Cooley-Tukey Fast Fourier Transform algorithm for Boolean n-cubes with a substantial amount of storage per cube node. In mapping a Cooley-Tukey type FFT to such a network the main concerns are effective use of the high connectivity/bandwidth of the Boolean n-cube, the effective use of the computational resources, the effective use of the storage bandwidth, if there is a storage hierarchy, and pipelines should the arithmetic units have such a feature. Another important consideration in a multiprocessor, distributed storage architecture is the allocation and access to coefficients, if they are precomputed. We describe FFT algorithms that use both the storage bandwidth and the communication system optimally, and which require storage of $P + nN$ coefficients for a transform on $P \geq N$ data elements. A complex to complex FFT on 16 million points is predicted to require about 1.5 seconds on a Connection Machine model CM-2.

In the Proceedings of the *International Conference on Systolic Arrays*, May, 1988

Finding $95800^4 + 217519^4 + 414560^4 = 422481^4$ on the Connection Machine

Thinking Machines Corporation
Technical Report NT88-1
Roger E. Frye
March 1988

Abstract

The smallest counterexample to Euler's generalization of Fermat's Last Theorem is $95800^4 + 217519^4 + 414560^4 = 422481^4$. I explain how this solution was found by exhaustive data parallel search on several Connection Machines.

To appear in *Proceedings of the Supercomputing Conference '88*

Real-Time Interactive Visualization of Computational Fluid Mechanics

Thinking Machines Corporation
Technical Report
J.A. Sethian, James B. Salem
March 1988

Abstract

We describe and demonstrate a new graphics environment for real-time visualization of the data from numerical simulations of fluid mechanics. Since such simulations often generate massive amounts of data, our environment provides an effective way to interact in real-time with the flow. We take advantage of the highly parallel machine. The fluid diagnostic tools allow real-time display of the motion of smoke dye and bubble wire tracers injected into the moving flow, as well as display of moving color contours for scalar fields such as temperature and pressure.

We demonstrate our interactive graphics fluid flow system by analyzing data generated from a numerical simulation of laminar and turbulent flow over a backwards-facing step. Input parameters are menu-driven, and we typically achieve approximately 3-7 screen updates per second, producing essentially real-time motion.

To appear in the *International Journal of Supercomputer Applications*

Study of Protein Sequence Comparison Metrics on the Connection Machine CM-2

Thinking Machines Corporation
Technical Report CB88-2
Eric Lander, Jill P. Mesirov, Washington Taylor IV
March 1988

Abstract

Software tools have been developed to do rapid large-scale protein sequence comparisons on databases of amino acid sequences, using a data parallel computer architecture. This software enables one to compare a protein against a database of several thousand proteins in the same time required by a conventional computer to do a single protein-protein comparison, thus enabling biologists to find relevant similarities much more quickly, and to evaluate many different comparison metrics in a reasonable period of time. We have used this software to analyze the effectiveness of various scoring metrics in determining sequence similarity, and to generate statistical information about the behavior of these scoring systems under the variation of certain parameters.

To appear in *Proceedings of the Supercomputing Conference '88*

A Cellular Automata Simulation of Two-Phase Flow on the CM-2 Connection Machine Computer

Thinking Machines Corporation
Technical Report CA88-1
Bruce M. Boghosian, Washington Taylor IV, Daniel H. Rothman
March 1988

Abstract

A cellular automaton (CA), recently developed by Rothman and Keller [1], simulates the flow of two incompressible, immiscible, viscous fluids in two dimensions. We have simulated this automaton on the CM-2 Connection Machine using a sequence of logical operations and table lookups to determine the new state of a CA site from its old state and those of its neighbors. The logical operations are performed in parallel by each of the Connection Machine processors, while the table lookups use the indirect addressing capabilities among groups of 32 processors.

This paper begins with a general description of CA fluids, including the issue of isotropy, the choice of a rule set, and the averaging procedure used to obtain hydrodynamical quantities. This is followed by a brief comparison with more conventional methods of simulating fluids. The CM-2 Connection Machine is then described, with emphasis on the indirect addressing capabilities of the machine. Section 2 gives a complete description of the Rothman-Keller model for two-phase flow. Section 3 then describes how the indirect addressing is used in the simulation algorithm, and how a symmetry in the dynamics is used to reduce the size of the lookup tables by a factor of six. Finally, in Section 4, a time sequence of results showing the separation of two immiscible phases from an initially homogenized state is presented.

To appear in *Proceedings of the Supercomputing Conference '88*

Spanning Balanced Trees in Boolean Cubes

Yale University
Department of Computer Science
Technical Report 611
Ching-Tien Ho and S. Lennart Johnsson
February 1988

Abstract

A *Spanning Balanced n-tree* (SBnT) in a Boolean n-cube is a spanning tree in which the root has fanout n, and all the subtrees of the root have $O(\frac{2^n}{n})$ nodes. The number of tree edges in each dimension of the n-cube is of order $O(\frac{2^n}{n})$. The spanning balanced n-tree allows for scheduling disciplines that realize lower bound (within a factor of two) one-to-all personalized communication, all-to-all broadcasting, and all-to-all personalized communication on a Boolean n-cube. The improvement in data transfer time over the familiar binomial tree routing is a factor of $\frac{n}{2}$ for concurrent communication on all ports and one-to-all personalized communication and all-to-all broadcasting. For all-to-all personalized communication on all ports concurrently the improvement is of order $O(\sqrt{n})$. We give distributed routing algorithms defining the spanning balanced n-tree. The balanced n-tree is not unique, and we provide a few definitions of n-trees that are effectively edge-disjoint. Some implementation issues are also discussed.

Binary numbers obtained from each other through rotation forms necklaces that are full if the period is equal to the length of the number, otherwise they are degenerate. As an intermediary result we show that the ratio between the number of degenerate necklaces and the total number of necklaces with l bits equal to one is at most $\frac{4}{4+n}$ for $1 \leq l < n$.

To appear in *SIAM Journal on Scientific and Statistical Computing*, July, 1989.

QED on the Connection Machine

Thinking Machines Corporation
Technical Report CS-88/1
Clive Baillie, S. Lennart Johnsson, Luis Ortiz, and G. Stuart Pawley
January 1988

Abstract

Physicists believe that the world is described in terms of gauge theories. A popular technique for investigating these theories is to discretize them onto a lattice and simulate numerically by a computer, yielding so-called lattice gauge theory. Such computations require at least 10^{14} floating-point operations, necessitating the use of advanced architecture supercomputers such as the Connection Machine made by Thinking Machines Corporation. Currently the most important gauge theory to be solved is that describing the sub-nuclear world of high energy physics: Quantum Chromodynamics (QCD). The simplest example of a gauge theory is Quantum Electro-dynamics (QED), the theory which describes the interaction of electrons and photons. Simulation of QCD requires computer software very similar to that for the simpler QED problem. Our current QED code achieves a computational rate of 1.6 million lattice site updates per second for a Monte Carlo algorithm, and 7.4 million site updates per second for a microcanonical algorithm. The estimated performance for a Monte Carlo QCD code is 200,000 site updates per second (or 5.6 Gflops/sec).

In the Proceedings in the *Third Conference on Hypercube Concurrent Computers and Applications*, January, 1988.

Protein Sequence Comparison on a Data Parallel Computer

Thinking Machines Corporation
Technical Report CB88-1
Eric Lander, Jill P. Mesirov, Washington Taylor IV
January 1988

Abstract

In this paper we discuss the issues involved in implementing a general dynamic program on a data parallel computer to compare proteins for good subsequence matches, based on a variety of scoring metrics. A standard serial algorithm can be optimally parallelized. Careful allocation of machine resources has enabled us to compare an entire database of 2000 proteins against itself in about the same time that it would take to run one protein against the database using conventional computers. The results gleaned from this program provide information about scoring metrics and allow clustering of groups of related proteins. This information can be of assistance in determining the biochemical function of some proteins.

Appeared in *Proceedings of the 1988 International Conference o Parallel Processing*, Penn State Press, pp. 257-263.

Exploiting Symmetry in High-Dimensional Finite Difference Calculations

Thinking Machines Corporation
Technical Report NA87-2
W. Daniel Hillis and Washington Taylor

Optimum Broadcasting and Personalized Communication in Hypercubes

Yale University
Department of Computer Science
Technical Report 610
S. Lennart Johnsson and Ching-Tien Ho
December 1987

Abstract

Effective utilization of communication resources is crucial for good overall performance in highly concurrent systems. In this paper we address four different communication problems in Boolean n-cube configured multiprocessors: (1) one-to-all broadcasting: distribution of common data from a single source to all other nodes; (2) one-to-all personalized communication: a single node sending unique data to all other nodes; (3) all-to-all broadcasting: distribution of common data from each node to all other nodes; and (4) all-to-all personalized communication: each node sending a unique piece of information to every other node. Three new communication graphs for the Boolean n-cube are proposed for the routing, and scheduling disciplines provably optimum within a small constant factor proposed. One of the new communication graphs consists of n edge-disjoint spanning binomial trees, and offers optimal communication for case 1; a speed-up with a factor of n over the spanning binomial tree for large data volumes. The other two new communication graphs are a balanced spanning tree, and a graph composed of n rotated spanning binomial trees. With appropriate scheduling and concurrent communication on all ports of every processor, routings based on these two communication graphs offer a vspeed-up of up to $\frac{n}{2}$, $\frac{n}{2}$ and $O(\sqrt{n})$ over the routings based on the spanning binomial tree for cases 2, 3 and 4 respectively. All three new spanning graphs offer optimal communication times for cases 2, 3 and 4 and concurrent communication on all ports of every processor. The graph consisting of n edge-disjoint spanning trees offers graceful degradation of performance under faulty conditions. Timing models and complexity analysis have been verified by experiments on a Boolean cube configured multiprocessor.

To appear in *IEEE Transactions on Computers*, September, 1989.

Ensemble Architectures and Their Algorithms: An Overview

Yale University
Department of Computer Science
Technical Report 580
S. Lennart Johnsson
November 1987

Abstract

During recent years, the number of commercially available parallel computer architectures has increased dramatically. The number of processors in these systems varies, from a few processors up to a many as 64k processors for the Connection Machine. In this paper, we discuss some of the technology issues that are the underlying driving force and focus on a particular class of parallel computer architectures. This class is often called Ensemble Architectures, and they are interesting candidates for future high performance computing systems. The ensemble configurations discussed here are linear arrays, 2-dimensional arrays, binary trees, shuffle-exchange networks, Boolean cubes, and cube connected cycles. We discuss a few algorithms for arbitrary data permutations, and some particular data permutation and distribution algorithms used in standard matrix computations. Special attention is given to data routing. Distributed routing algorithms in which elements with distinct origin and distinct destinations do not traverse the same communication link make possible a maximum degree of pipelined communications. The linear algebra computations discussed are: matrix multiplication, dense and general banded systems solvers, linear recurrence solvers, tridiagonal system solvers, fast Poisson solvers, and very briefly, iterative methods.

In *Numerical Algorithms for Modern Parallel Computer Architectures*, Vol. 13, *IMA Series in Mathematics and its Applications*, Springer Verlag, 1988.

The Communication Efficiency of Meshes, Boolean Cubes and Cube Connected Cycles for Wafer Scale Integration

Yale University
Department of Computer Science
Technical Report 579
Abhiram Ranade and S. Lennart Johnsson
November 1987

Abstract

In this paper we analyze the emulation of two-dimensional meshes, butterfly networks, and spanning trees on meshes, Boolean cubes, and Cube Connected Cycles (CCC) networks. We consider three models for signal propagation along a wire: constant delay, capacitive delay, and resistive delay. We also present novel layouts for hypercubes and CCC's that offer better performance for some problems, while essentially maintaining the performance for other problems. The mesh interconnection performs better on all emulations for all delay models, if the communication throughout determines the performance. With resistive delay model, meshes also offer the best latency for all emulations. The hypercube and CCC layouts yield lower latency for emulating butterfly networks and spanning trees for the constant delay and capacitive delay models.

In *The 1987 International Conference on Parallel Processing*, pp. 479-482.

A Microcode Compiler for Cellular Automata Research on the CM-2 Connection Machine Computer

Thinking Machines Corporation
Technical Report
Bruce M. Boghosian, Luis F. Ortiz, Washington Taylor IV
November 1987

Abstract

This document describes a microcode compiler for cellular automata research on the CM-2 Connection Machine Computer. It allows for the specification of any update rule on a multidimensional Cartesian lattice, using either logical operations or lookup tables, or some combination thereof.

Computing Fast Fourier Transforms on Boolean Cubes and Related Networks

Yale University
Department of Computer Science
Technical Report 598
S. Lennart Johnsson, Ching-Tien Ho, Michel Jacquemin and Alan Ruttenberg
October 1987

Abstract

High performance architectures are using an ever increasing number of processors. The Boolean cube network has many independent paths between any pair of processors. It provides both a high communications bandwidth as well as the ability to emulate many other networks without contention for communication channels. Of particular interest for the Fast Fourier Transform (FFT) is the ability to emulate butterfly networks, which defines the communication pattern of the FFT. Each node of a Boolean cube network of N nodes has a degree of $\log_2 N$. For a large number of nodes the number of channels required at the chip boundary may be unfeasibly large with several nodes toa chip, and a network with slightly lower connectivity, such as Cube Connected Cycles networks, may be preferable. The communication system is the most critical resource in many high performance architectures, and its effective use is imperative. We describe FFT algorithms that use both the storage bandwidth and the communication system optimally for an architecture such as the Connection Machine that has 65536 processors interconnected in a Boolean cube related network. We also describe the necessary data allocation, and the allocation and use of the twiddle factors.

In *Advanced Algorithms and Architectures for Signal Processing II", SPIE 1987*, Vol 826, pp. 223-231, 1987.

Algorithms for Multiplying Matrices of Arbitrary Shapes Using Shared Memory Primitives on Boolean Cubes

Yale University
Department of Computer Science
Technical Report 569
S. Lennart Johnsson and Ching-Tien Ho
October 1987

Abstract

We investigate the multiplication of two arbitrarily shaped matrices on Boolean cube configured multiprocessors. We present algorithms in terms of generic communication primitives which effectively allows the programmer to express the algorithms as shared memory algorithms. Some of the communication primitives we present yield communication within a factor of two of the lower bound, and have the best known routing times. For the multiplication of a PxQ matrix by a QxR matrix three loops are required in a language like Fortran 77. One, two or all three loops may be parallelized. We show that with the communication primitives we use parallelizing all three loops yield a complexity that at most is equal to that of parallelizing two loops, which in turn is at most equal to that of parallelizing only one loop, for all matrix shapes. In parallelizing only one loop the processors shall be aligned with the axis (P,Q, or R) with the maximum number of elements to yield the lowest arithmetic and communication complexity. In parallelizing two loops the processors shall be assigned to the plane with the maximum number of elements. In parallelizing two or all three loops the aspects ratio of the processor array shall be the same as that of the matrix element domain. We also derive expressions for the optimal number of processors and show that for large start-up times for communication the optimum number of processors may be significantly smaller than the number of matrix elements in the dimensions parallelized. Experimental results for the Intel iPSC are presented.

In Proceedings of *ARO Workshop on Parallel Processing and Medium Scale Multiprocessors*, SIAM, 1986.

Data Parallel Programming and Basic Linear Algebra Subroutines

Yale University
Department of Computer Science
Technical Report 584
S. Lennart Johnsson
September 1987

Abstract

Data Parallel programming is conceptually simple and provides powerful programming primitives as in shared memory models of computation. With an appropriate underlying architecture, primitives requiring global memory access do not require significantly longer execution times than primitives only requiring local access. In the data parallel programming model primitives are also available for expressing simple forms of local data interaction in relative coordinates, as for instance required in relaxation on multidimensional lattices.

One particular data parallel computer is the Connection Machine. In this paper, we describe some of the data parallel aspects of the programming languages provided on the Connection Machine. We comment on the implementation of level-1 and level-2 BLAS, and describe the implementation of one level-3 BLAS function. Matrix multiplication is discussed in detail. For matrices of the size of the machine or larger, only the kernel function is required. The matrix multiplication kernel yield a performance of up to 5.2 Gflops in single precision on the CM-2 with the floating point option. For matrices considerably smaller than the machine, all three nested loops in a Fortran 77 program can be made parallel, and expressed with few instructions without any loop constructs.

In *Scientific Software*, Vol. 14., IMA Series in Mathematics and its Applications, Springer Verlag, 1988, pp. 183-196.

Algorithms for Matrix Transposition on Boolean n-cube Configured Ensemble Architectures

Yale University
Department of Computer Science
Technical Report 572
S. Lennart Johnsson and Ching-Tien Ho
September 1987

Abstract

In a multiprocessor with distributed storage the data structures have a significant impact on the communication complexity. In this paper we present a few algorithms for performing matrix transposition on a Boolean n-cube. One algorithm performs the transpose in a time proportional to the lower bound both with respect to communication start-ups and element transfer times. We present algorithms for transposing a matrix embedded in the cube by a binary encoding, a *binary-reflected* Gray code encoding of rows and columns, or combinations of these two encodings. The transposition of a matrix when several matrix elements are identified to a node by *consecutive* or *cyclic* partitioning is also considered and lower bound algorithms given. Experimental data are provided for the Intel iPSC and the Connection Machine.

In *SIAM J. Matrix Analysis*, July 1988, vol. 9, no. 3, pp 419-454.

Highly Parallel Banded Systems Solvers

Yale University
Department of Computer Science
Technical Report 581
S. Lennart Johnsson
August 1987

Abstract

We present algorithms for the solution of banded systems of equations on parallel architectures, in particular ensemble architectures, ie., architectures that have a large number of processing elements. Each processor has its own local storage. The band is considered dense. Concurrent elimination of a single variable yields a linear speed-up for ensembles configured as tori, or Boolean cubes, if $N \gg m$, with a maximum ensemble size of $m\ (m+R)$ (or $2m(m+R)$) processors for a banded system of N equations, bandwidth $2m+1$ and R right hand sides. The minimum attainable computational complexity is of order $O(N)$. Concurrent elimination of multiple variables as well as concurrent elimination of each such variable yields a minimum complexity of $O(m+m\log_2\frac{N}{m})$ for a total of $(2m+R)\ N$ ensemble nodes. To attain this complexity the ensemble should be configured as clusters, each in the form of a torus of dimension m by $2m+R$, or a Boolean cube of appropriate dimension. Furthermore, corresponding processors in different clusters are assumed to be interconnected to form a binary tree, shuffle-exchange, perfect shuffle, or Boolean cube network. The number of clusters should be of order $O(\frac{N}{m})$ for minimum computational complexity.

In *Parallel Computations and Their Impact on Mechanics*, AMD-Vol. 86, pp. 187-208, ASME, December 1987.

Directions in High Performance Computation

Yale University
Department of Computer Science
Technical Report 574
S. Lennart Johnsson
June 1987

Abstract

Evolving technology is driving high performance computer architecture towards highly concurrent systems. We review some of the technology influencing this direction, and discuss some of the architectural, algorithmic, and programming system consequences of this change. Finally, we briefly describe some of the essential features of the Connection Machine, a commercially available computer with an architecture and programming system that includes several of the features we expect to find in many high performances architectures in the future.

In *Proceedings of the American Statistical Association 19th Symposium on Computer Science and Statistics*, March 8 -11, 1987.

Multiple Tridiagonal Systems, the Alternating Direction Methods and Boolean Cube Configured Multiprocessors

Yale University
Department of Computer Science
Technical Report 532
S. Lennart Johnsson and Ching-Tien Ho
June 1987

Abstract

In this paper we investigate the complexity of concurrent algorithms for the solution of multiple tridiagonal systems as they appear, for instance in fast Poisson solvers, or in Alternating Direction Methods. We consider divide-and-conquer algorithms in the form of several variations of odd-even cyclic reduction, and algorithms making use of data transposition and local elimination. The processors are assumed to be configured as a Boolean cube. A simple performance model is established, and the validity of the model verified on an Intel iPSC. Depending on machine characteristics, a divide-and-conquer algorithm such as balanced cyclic reduction, or a transpose based algorithm, may be optimum. In general, the former is preferable for sufficiently many processors. Indeed, a hybrid scheme in which the last several steps of the divide-and-conquer algorithm, is lower complexity than either. The transition point depends on the architectural parameters such as communication start-up time, data channel transfer rate, maximum packet size, and the time for an arithmetic operation, but also the number of independent systems to be solved. For the Intel iPSC with its moderate number of processors the transpose algorithms are in general preferable.

It is shown that the optimum partitioning of a set of independent tridiagonal systems among a set of processors yields the embarrassingly parallel case. If the systems originates from a lattice and solutions are computed in alternating directions, then to first order the aspect ratio of a computational lattice shall be the same as that of the lattice forming the base for the equations.

Some of the transpose based algorithms are novel variations of Gaussian elimination. Our experiments demonstrate the importance of combining in the communication system for architectures with a relatively high communications start-up time.

On the Embedding of Arbitrary Meshes in Boolean Cubes with Expansion Two Dilation Two

Yale University
Department of Computer Science
Technical Report 576
Ching-Tien Ho and S. Lennart Johnsson
April 1987

Abstract

An embedding based on a binary-reflected Gray code encoding of the mesh points in each dimension attains the minimum dilation, but the expansion of a d-dimensional mesh may be close to 2^d. A direct mapping method that yields dilation 2 and minimum expansion embeddings for most aspect ratios of two-dimensional meshes is presented. The average edge dilation asymptotically approaches one. The direct mapping method is also applied to higher dimensional meshes with preserved dilation.

In *The 1987 International Conference on Parallel Processing*, pp. 188-191.

A Fast Parallel Algortihm for Labeling Connected Components in Image Arrays

Thinking Machines Corporation

Technical Report NA86-2

Willie Lim, Ajit Agrawal, Lena Nekludova

December 1986
Revised April 1987

Abstract

A fast parallel algorithm for labeling connected components in a 2-D array of pixels is discussed. The algorithm has a time complexity of O(log N) for the exclusive read exclusive write model of parallel computation and where the time to access to any memory location in any processor is constant. The algorithm is implemented on a Connection Machine by mapping the 2-D array of values into a 2-D array of processors. On this machine time complexity is measured in terms of router cycles i.e. message transmissions through the routing network. Initially each processor is assigned a unique label or id. The label of a region is the largest processor label of the processors in the region. Processors on region boundaries are connected as rings by using pointers. The boundary label i.e. the largest processor label of the processors, in the boundary of each boundary is computed. The processors on all the boundaries in the region are linked together into a long one and its label is computed. The label for this long boundary is the label of the region. This label is then propagated to all the interior processors in the region.

Appeared in *Proceedings of the 1987 International Conference on Parallel Processing*, Penn State Press, pp. 783-786

Fast PDE Solvers on Fine and Medium Grain Architectures

Yale University
Department of Computer Science
Technical Report 583
S. Lennart Johnsson
April 1987

Abstract

Fast solvers for partial differential equations are often based on tridiagonal system solvers, the Fast Fourier Transform (FFT), and/or a combination thereof. We describe in some detail the implementation of tridiagonal system solvers and the FFT on a massively parallel architecture. The computational complexity of these two methods is compared, and some of the optimization issues that arise in medium scale architectures are discussed. Both odd-even cyclic reduction and the FFT for P equations requires $O(\log_2 P)$ steps and arithmetic time on a massively parallel architecture. The difference in arithmetic time is only a small constant factor. We derive this factor for real and complex systems. Odd-even cyclic reduction and the FFT have similar, but not identical communication topology. For the odd-even cyclic reduction algorithm it is equivalent to a reduction data manipulator network, whereas, that of the FFT is the familiar butterfly network. Communication time is dependent upon the ability of the communication system to support these two communication requirements with the data mapping being used. We derive the number of communications required by the two algorithms both for unlimited and limited buffer sizes, and the total number of element transfers. Measured times for implementations on the Connection Machine are presented.

In *Advances in Computer Methods for Partial Differential Equations*, IMACS (International Association for Mathematics and Computers in Simulation), vol. 6, pp. 405-410, 1987.

The FFT and Fast Poisson Solvers on Parallel Architectures

Yale University
Department of Computer Science
Technical Report 582
S. Lennart Johnsson
March 1987

Abstract

In this paper we highlight some of the issues that need to be addressed in finding efficient implementations of the conventional radix-2 and radix-4 Fast Fourier Transform algorithm on parallel architectures. We consider the mapping problem for Ensemble Architectures, and the solution of Poisson's equation using FFT's. We also point out some properties of matrix transposition, which are particularly relevant for Boolean cube configured architectures and related networks. Finally, we comment on FFT in the context of submicron technology and wafer-scale integration.

In Proceedings of the Mathematical Sciences Institute *Workshop on Fast Fourier Transforms for Vector and Parallel Computers*, Cornell University, March 22-25, 1987.

Solving Schrödinger's Equation on the Intel iPSC by the Alternating Direction Method

Yale University
Department of Computer Science
Technical Report 502
Faisal Saied, Ching-Tien Ho, S. Lennart Johnsson, and Martin H. Schultz
January 1987

Abstract

We consider the numerical solution of the Schrödinger's equation and investigate several different algorithms for implementing the Alternating Direction Method on hypercubes. We indicate the relative merits of the algorithms depending on cube parameters such as arithmetic speed, communication latency, transfer rate, the packet size, and the cost of reordering data locally. We present timings for the Intel iPSC that show that Alternating Direction Methods can be implemented efficiently on hypercubes.

In *Hypercube Multiprocessors 1987*, pp. 680-691, SIAM 1987.

A Parallel $O(logN)$ Algorithm for Finding Connected Components in Planar Images

Thinking Machines Corporation

Technical Report NA87-1

Ajit Agrawal, Lena Nekludova, Willie Lim

December 1986

Abstract

For an EREW model of parallel machine, we present an O(log N) algorithm, with small constant, for finding connected components in a 2-dimensional multicolored image, consisting of N pixels. The algorithm requires O(N) processors. The best known bounds for the problem is O(logN) for finding connected components of a graph in a weaker CRCW model.

Solving Banded Systems on a Parallel Processor

Yale University
Department of Computer Science
Technical Report 519
Jack Dongarra and S. Lennart Johnsson
November 1986

Abstract

In this paper we examine ways of solving dense, banded systems on different parallel processors. We start with some considerations for processors with vector instructions, then discuss various algorithms for the solution of large, dense, banded systems on a parallel processor. We analyze the behavior of the parallel algorithms on distributed-storage architectures configured as rings, two-dimensional meshes with end-around connections (tori), Boolean n-cube configured architectures, and bus-based and switch-based machines with shared storage. We also present measurements for two-bus based architectures with shared storage, namely, the Alliant FX/8 and the Sequent Balance 21000.

In *Journal of Parallel Computing*, vol. 5, no. 2, pp. 219-246,1987.

Alternating Direction Methods on Multiprocessors

Yale University
Department of Computer Science
Technical Report 382
S. Lennart Johnsson, Youcef Saad and Martin H. Schultz
September 1986

Abstract

We propose several implementations of the Alternating Direction Method for solving parabolic partial differential equations on multiprocessors. A complexity analysis of these implementations shows that the method can be made highly efficient on parallel architectures by using pipelining and variations of the classical Gaussian elimination algorithm for solving tridiagonal systems. Previously, we showed that we could obtain linear speedups for moderate numbers of processors in a ring architecture. In this paper we discuss extensions to a large number of processors in a 2–D grid architecture and a hypercube.

In *SIAM J. Sci. Stat. Comp.*, Vol. 8, No. 5, September, 1987.

Solving Sparse Systems of Linear Equations on the Connection Machine

Thinking Machines Corporation
Technical Report
Charles E. Leiserson, Jill P. Mesirov, Lena Nekludova, Stephen M. Omohundro, John Reif, Washington Taylor IV
1986

Abstract

The Connection Machine is a 65,536-processor computer which was designed for artificial intelligence applications. This paper shows that the machine is suitable for numerical computations as well. We describe a program for solving sparse systems of linear equations based on parallel nested dissection which has been implemented on the Connection Machine.

Solving Tridiagonal Systems on Ensemble Architectures

Yale University
Department of Computer Science
Technical Report 436
S. Lennart Johnsson
November 1985

Abstract

The concurrent solution of tridiagonal systems on linear and 2-dimensional arrays, complete binary trees, shuffle-exchange and perfect shuffle networks, and boolean cubes by elimination methods are devised and analyzed. The methods can be obtained by symmetric permutations of some rows and columns, and amounts to cyclic reduction or a combination of Gaussian elimination and cyclic reduction, (*GECR*). The ensembles have only local storage and no global control. Synchronization is accomplished via message passing to neighboring processors.

The parallel arithmetic complexity of *GECR* for N equations on a K processor ensemble is $O(N/K + log_2 K)$, and the communication complexity is $O(K)$ for the linear array, $O(\sqrt{K})$ for the 2-dimensional mesh, and $O(log_2 K)$ for the networks of diameter $O(log_2 K)$. The maximum speed-up for the linear array is attained at $K \approx (N/\alpha)^{1/2}$ and for the 2-d mesh at $K \approx (N/2\alpha^{2/3}$, where $\alpha =$ (the time to communicate one floating-point number)/(the time for a floating-point arithmetic operation). For the binary tree the maximum speed-up is attained at $K = N$, and for the perfect shuffle and boolean k-cube networks, $K = N/(1+\alpha)$ yields the maximum speed-up. The minimum time complexity is of order $O(N^{1/2})$ for the linear array, of order $O(N^{1/3})$ for the mesh, and of order $O(log_2 N)$ for the binary tree, the shuffle-exchange, the perfect shuffle and the boolean k-cube.

The relative decrease in computational complexity due to a truncation of the reduction process in a highly concurrent system is much greater than on a uniprocessor. The reduction in the arithmetic complexity is proportional to the number of steps avoided, if the number of processing elements equals the number of equations. So is also the reduction in the communication complexity for ensembles configured as binary trees, shuffle-exchange and perfect shuffle networks, and boolean cubes.

Partitioning the ensemble into subsets of processors is shown to be more efficient for the solution of multiple independent problems than pipelining the solutions over the entire ensemble. A *balanced* cyclic reduction algorithm is presented for the case where each system is spread uniformly over the processing elements, and its complexity is compared with Gaussian elimination.

In *SIAM J. Sci. Stat. Comp*, Vol. 8, no. 3, pp. 354-392, May 1987.

Communication Efficient Basic Linear Algebra Computations on Hypercube Architectures

Yale University
Department of Computer Science
Technical Report 361
S. Lennart Johnsson
September 1985

Abstract

This paper presents a few algorithms for embedding loops and multi-dimensional arrays in hypercubes with the emphasis on proximity preserving embeddings. A proximity preserving embedding minimizes the need for communication bandwidth in computations requiring nearest neighbor communication. Two storage schemes for "large" problems on "small" machines are suggested and analyzed, and algorithms for matrix transpose, multiplying matrices, factoring matrices, and for solving triangular linear systems are presented. A few complete binary tree embeddings are described and analyzed. The data movement in the matrix algorithms is analyzed and it is shown that in the majority of cases the directed routing paths only intersect at nodes of the hypercube allowing for a maximum degree of pipelining.

In *Journal of Parallel and Distributed Computing*, Vol. 4, No. 2, pp. 133-172, April, 1987.

Solving Narrow Banded Systems on Ensemble Architectures

Yale University
Department of Computer Science
Technical Report 418
S. Lennart Johnsson
August 1985

Abstract

We present concurrent algorithms for the solution of narrow banded systems on ensemble architectures, and analyze the communication and arithmetic complexities of the algorithms. The algorithms consist of three phases. In phase 1 a block tridiagonal system of reduced size is produced through largely local operations. Diagonal dominance is preserved. If the original system is positive definite and symmetric, so is the reduced system. It is solved in a second phase, and the remaining variables are obtained through local backsubstitution in a third phase. With a sufficient number of processing elements, there is no first and third phase. We investigate the arithmetic and communication complexity of Gaussian elimination and block cyclic reduction for the solution of the reduced system on boolean cubes, perfect shuffle and shuffle-exchange networks, binary trees and linear arrays.

With an optimum number of processors the minimum solution time on a linear array is of an order that ranges from $O(m^2\sqrt{Nm})$ to $O(m^3 + m^3 log_2(N/m))$ depending on the bandwidth, the dimension of the problem, and the times for communication and arithmetic. For boolean cubes, cube-connected cycles, perfect shuffle and shuffle-exchange networks, and binary trees the minimum time is $O(m^3 + m^3 log_2(N/m))$ including the communication complexity.

In *ACM TOMS,* Vol. 11, No. 3, 1985.

Combining Parallel and Sequential Sorting on a Boolean n-cube

California Institute of Technology
Department of Computer Science
Technical Report
S. Lennart Johnsson
April 1984

Abstract

Three parallel algorithms for sorting M uniformly distributed elements on a Boolean n-cube of $N = 2^n$ $N \leq M$ processors are presented. Two of the algorithms combine sequential sort with bitonic sort to accomplish a time complexity of $O(MlogM/N)$ for $N \ll M$,and $O(log^2 M)$ for $M \simeq N$. One algorithm sorts the elements cyclically, such that a processor holds sorted elements that are congruent $modN$. The other algorithm sorts subsequences of M/N consecutive elements into each processor. The third algorithm is a parallel bucket sort that sorts the elements into L buckets in time $O(M/N + L)$ if $N \leq L$ and time $O(M/N + L + logN - logL)$ for $N > L$.

In the Proceedings of the *1984 International Conference on Parallel Processing*, August 21-24, 1984.

An Algebraic Description of Array Implementations of FFT Algorithms

California Institute of Technology
Department of Computer Science
Technical Report
S. Lennart Johnsson and Danny Cohen
1982

Abstract

Fast Fourier Transform, FFT, algorithms are interesting for direct hardware implementation in VLSI. The description of FFT algorithms is typically made either in terms of graphs illustrating the dependency between different data elements or in terms of mathematical expressions without any notion of how the computations are implemented in space or time. In this paper a notation that explicitly models the distribution of computations in space and time is used to describe a decimation-in-frequency type FFT algorithm. Expressions in the notation used in this paper can be given an interpretation in the implementation domain.

In *The 20th Annual Allerton Conference on Communication, Control and Computing*, Monticello, Illinois, October 6 - 8, 1982, pp. 126 - 134.

VLSI Algorithms for Doolittle's, Crout's and Cholesky's Methods

California Institute of Technology
Department of Computer Science
Technical Report
S. Lennart Johnsson
1982

Abstract

In order to take full advantage of the emerging VLSI technology it is required to recognize its limited communication capability and structure algorithms accordingly. In this paper concurrent algorithms for the methods of Crout, Doolittle and Cholesky are described and compared with concurrent algorithms for Gauss', Given's and Householder's method. The effect of pipelining the computations in two dimensional arrays is given special attention.

In the Proceedings of the *International Conference on Circuits and Computers*, September 29 - October 1, 1982, pp. 372-377.

Pipelined Linear Equation Solvers and VLSI Technology

California Institute of Technology
Department of Computer Science
Technical Report
S. Lennart Johnsson
1982

Abstract

Many of the commonly used methods for solution of linear systems of equations on sequential machines can be given a concurrent formulation. The concurrent algorithms take advantage of independence of operations in order to reduce the time complexity of the methods. During the course of computations specified by the algorithm data has to be routed to the various places of computation. Pipelining can be used to avoid broadcasting in VLSI arrays for computation. Pipelining will in general allow for a reduced cycle time but may force data to be spread out in time, as is the case for Gaussian elimination. What the required spacing is depends on the pipelining and the data flow.

In the paper concurrent algorithms and their pipelining for Gaussian elimination, Householder transformations and Given's rotations are discussed. Gaussian elimination and Given's rotations can use two dimensional arrays while Householder transformation uses a one dimensional array. If partial pivoting is necessary in Gaussian elimination, then one dimension of the array is essentially lost and a linear array is almost as efficient as a two-dimensional array. Householder transformations that are numerically stable may perform the triangulation in shorter time, if partial pivoting is necessary in Gaussian elimination. The amount of arithmetic that a node in the arrays perform is somewhat different for the different methods. The difference is largest for the boundary cells. However, it should be feasible to design a common node of very low complexity that very efficiently supports a range of methods for the solution of linear systems of equations.

In the Proceedings of *Microelectronics 1982*, Adelaide, Australia, May 12 - 14, 1982, pp. 42 - 47, The Institution of Engineers, Australia, National Conference Publication No. 82/4.

A Computational Array for the QR-method

California Institute of Technology
Department of Computer Science
Technical Report
S. Lennart Johnsson
1982

Abstract

The QR method is a method for the solution of linear systems of equations. The matrix R is upper triangular and Q is a unitary matrix. In equation solving Q is not always computed explicitly. The matrix R can be obtained by applying a sequence of unitary transformations to the matrix defining the systems of equations. Householder's method or Given's method can be used to determine unitary transformation matrices. This paper describes a concurrent algorithm and corresponding array for computing the triangular matrix R by Householder transformations. Particular attention is given to issues such as broadcasting and pipelining.

In *Proceedings, Conference on Advanced Research in VLSI*, Ed. P. Penfield, Artech House, 1982, pp. 123 - 129.

A Mathematical Approach to Modeling the Flow of Data and Control in Computational Networks

California Institute of Technology
Department of Computer Science
Technical Report
S. Lennart Johnsson and Danny Cohen
1981

Abstract

This paper proposes a mathematical formalism for the synthesis and qualitative analysis of computational networks that treats data and control in the same manner. Expressions in this notation are given a direct interpretation in the implementation domain. Topology, broadcasting, pipelining, and similar properties of implementations can be determined directly from the expressions.

This treatment of computational networks emphasizes the space/time tradeoff of implementations. A full instantiation in space of most computational problems is unrealistic, even in VLSI. Therefore, computations also have to be at least partially instantiated in the time domain, requiring the use of explicit control mechanisms, which typically cause the data flow to be nonstationary and sometimes turbulent.

In *VLSI Systems and Computations*, Eds. Kung, Sproull, Steele, Computer Sciences Press, Rockville, 1981, pp. 213 - 225.

A VLSI Approach to Real-Time Computation Problems

California Institute of Technology
Department of Computer Science
Technical Report
S. Lennart Johnsson and Danny Cohen
1981

Abstract

This paper presents a formalism for describing the behavior of computational networks at the algorithmic level. It establishes a direct correspondence between mathematical expressions defining a function and the networks which compute that function. By formally manipulating the symbolic expressions that define a function, it is possible to obtain different networks that compute the function. Certain important characteristics of computational networks, such as computational rate, performance and communication requirements can directly be determined from this mathematical description.

The use of this formalism for design and verification is demonstrated on a few computational networks for functions typical in signal processing.

In *25th Annual International Technical Symposium and Exhibit of the Society for Photo-Optical Instrumentation Engineers*, San Diego, August 24 - 28, 1981, Vol. 298, pp. 48 - 59, SPIE - The Society for Optical Engineering.

Towards a Formal Treatment of VLSI Arrays

California Institute of Technology
Department of Computer Science
Technical Report
S. Lennart Johnsson, Uri Weiser, Danny Cohen and Alan L. Davis
1981

Abstract

A formalism to describe the behavior of computational networks at the algorithmic level is introduced. A direct correspondence between mathematical expressions defining a function and computational networks computing that function can be established by the notation. Different networks guaranteed to compute a given function can be obtained by formal manipulation of expressions. Important characteristics of computational networks, such as computational rate, performance and communication requirements can be determined directly from that mathematical description of a network. The use of the notation for design and verification will be demonstrated on computational networks for Finite Impulse Response filters, matrix operations and the Discrete Fourier Transform.

The progression of computations can often be modelled by wave fronts in an illuminating way. Our notation supports such a modelling as is shown. A computational network can be abstracted to a graph. The duality between this form of representation and mathematical expressions is discussed.

In *Proceedings, Second Caltech Conference on VLSI*, Pasadena, January 19 - 21, 1981.

Computational Arrays for Band Matrix Equations

California Institute of Technology
Department of Computer Science
Technical Report 4287
May 1981
S. Lennart Johnsson

Abstract

This report presents systolic algorithms for matrix equations of bandwidth m and size N on arrays of $P \times P$ processors for $P \leq m$. Blocks of matrix elements are stored in each processor. A collection of blocks form a processing window. Blocks are allocated cyclicly. We also provide systolic block algorithms for dense matrices, and give an algorithm for partial pivoting. The array is perfectly load balanced, and the speed-up is of order $O(P^2)$ without pivoting. Without tree-like interconnections for the pivot selection the speed-up with partial pivoting is of order $O(P)$.

A Note on Householder's Method, Sparse Matrices and Concurrency

California Institute of Technology
Department of Computer Science
Technical Report 4089
S. Lennart Johnsson
December 1980

Abstract

Householder transformations result in more fill-in than Gaussian elimination. In this report we show that the concurrency is also less than for Gaussian elimination. In an parallel random access model of computation the order of the complexity is the same as that of Gaussian elimination with partial pivoting.

Gaussian Elimination on Sparse Matrices and Concurrency

California Institute of Technology
Department of Computer Science
Technical Report 4087
S. Lennart Johnsson
December 1980

Abstract

Sparse matrices offer a higher degree of concurrency than dense matrices for LU-decomposition. All vertices that are not adjacent can be eliminated concurrently. We show that greedy elimination schemes are not optimal, in general. There is also a trade-off between concurrency and fill-in. A perfect elimination graph such as that of a tridiagonal matrix yields a very limited concurrency if no fill-in is the objective for the elimination order. Two vertices can be eliminated concurrently. However, nested dissection yields an elimination tree that allows elimination in $log_2 N$ steps for a system of N equations. The number of arithmetic operations is approximately twice that of the perfect elimination scheme.

Gaussian Elimination on Sparse Matrices and Concurrency

California Institute of Technology
Department of Computer Science
Technical Report 4027
S. Lennart Johnson
December 1980

Abstract

Sparse matrices offer a higher degree of concurrency than dense matrices for LU-decomposition. All vertices that are not adjacent can be eliminated concurrently. We show that greedy elimination schemes are not optimal, in general. There is also a trade-off between concurrency and fill-in. A perfect elimination graph such as that of a tridiagonal matrix yields a very limited concurrency if no fill-in is the objective for the elimination order. Two vertices can be eliminated concurrently. However, nested dissection yields an elimination tree that allows elimination in $\log_2 N$ steps for a system of N equations. The number of arithmetic operations is approximately twice that of the perfect elimination scheme.