DOE/ER/25009--769

DE89 016373

C^{3}P-769

1989 — The First Year of the Parallel Supercomputer

Geoffrey C. Fox*

Caltech Concurrent Computation Program
1201 E. California Blvd., MS 206-49
Pasadena, California 91125

gcf@procyon.caltech.edu

Plenary Presentation at the Fourth Hypercube Conference
Monterey 6 March 1989

* *Supported by Department of Energy: Applied Mathematical Sciences - Grant: DE-FG03-85ER25009; Program Manager of the Joint Tactical Fusion Program Office; National Science Foundation: Center for Research in Parallel Computation - CCR-8809615, Benchmarking Grant - ACS-8719502; and a joint study with IBM*

1989 — The First Year of the Parallel Supercomputer *

Geoffrey C. Fox
Caltech Concurrent Computation Program
Mail Code 206-49
Pasadena, California 91125

June 5, 1989

Abstract

Parallel Computing has come of age with several commercial and in-house systems available which not only promise, but realize supercomputer or better performance. We survey several major computations underway on hypercubes, transputer arrays and the SIMD Connection Machine CM-2 and AMT DAP. Where possible, we compare parallel implementations with those on CRAY and other high performance conventional computers. We summarize these experiences as a set of lessons for applications, decomposition, performance, hardware and software for parallel machines.

1 Introduction

In my banquet talk [Fox:88b] in last year's hypercube conference [Fox:88c], I reviewed several applications and algorithms that had been implemented on "real" parallel computers — mainly hypercubes. The results were encouraging; as shown in Table 1, 90% of the applications parallelize well in a manner that scales to many nodes. The main requirement is that the problem be large and have some sort of algorithmic synchronization to ensure that the nodes can be naturally coordinated. The source of parallelism is essentially always domain decomposition or data parallelism; a simple universal technique to produce high performance scaling parallel algorithms. We introduced the concepts of *synchronous*, *loosely synchronous*, and *asynchronous* to describe the time or synchronization structure of the problems. In this terminology, SIMD computers are appropriate for synchronous and MIMD for loosely synchronous problems. With this classification, we found that about one half of all the problems surveyed were directly suitable for SIMD machines; the other half could make some use of the additional flexibility of the MIMD architecture. It was left for further research to quantify the advantage of MIMD machines for problems which only "slightly" violated the *synchronous* condition. The promise of hypercubes have been particularly well illustrated by the nice results from SANDIA on the performance of their 1024 node NCUBE hypercube on six prototypical applications [Gustafson:88a].

Until recently, there were several parallel computers that were "interesting" or "cost-effective". However, in the past they were not effective competition for the conventional supercomputers; they lacked both CPU power and the necessary hardware or software infrastructure to support major computations. The situation is now changed, several computations are now underway on parallel computers that are comparable to or better than the state of the art supercomputer (usually CRAY) calculations. In Section 2, we survey several of these at Caltech describing some of the results that bear on general issues in the parallel computer field. We will also present some performance comparisons from a recent project led by Paul Messina [Messina:89a], [Pfeiffer:88a]. This evaluation considered the machines listed in Table 2 where the more conventional high performance computers were also considered by Kuck's PERFECT club [Berry:88a]. In Table 3, we list the explicit machines used in the parallel supercomputer applications; only the AMT DAP was not considered by Messina's group. We also will consider, but not present in detail, results from the transputer based systems which have similar architecture and performance characteristics to the NCUBE hypercube. We expect transputer systems to be productive high performance machines in the near future as larger configurations come into service. We will not discuss potentially interesting machines, such as the iPSC2-VX hypercube as we only have INTEL hypercubes without vector boards at Caltech, and in this bare form, the iPSC-2 has modest performance, and cannot be considered a supercomputer.

In the final Section 3 of this report, we conclude with a summary of the lessons learned.

2 Applications

Here we outline a dozen separate applications. Some are single calculations; others, such as Sec. 2.1, represent several distinct computations.

2.1 Lattice Monte Carlo Simulations

General Remarks

Lattice theories represent one of the most computationally intense class of problems [Baillie:89b]. They arise from

*Supported by Department of Energy: Applied Mathematical Sciences - Grant: DE-FG03-85ER25009; Program Manager of the Joint Tactical Fusion Program Office; National Science Foundation: Center for Research in Parallel Computation - CCR-8809615, Benchmarking Grant - ASC-8719502; and a joint study with IBM.

Table 1: Summary of 84 Separate Applications on Parallel Computers

No.	Application Field
9	Biology
4	Chemistry & Chemical Engineering
14	Engineering
10	Geology & Earth/Space Science
13	Physics
5	Astronomy & Astrophysics
11	Computer Science
18	Numerical Algorithm

Application Classification	No.	Fraction	
Synchronous (S)	34	0.40	Total S+LS
Loosely Synchronous (LS) (not synchronous)	30	0.36	0.76
Embarrassingly Parallel (EP)	6	0.07	Total S+LS+EP
– runs on SIMD			0.90
– needs MIMD	6	0.07	Clear Scaling
Truly Asynchronous	8	0.10	Unclear Scaling

Table 2: Advanced architecture computers studied in the Caltech Performance Evaluation Project [Messina:89a].

Machine	Description
NCUBE	Hypercube with custom scalar processors
Mark III	Hypercube with MC68020/68882 processors
Mark IIIfp	Mark III hypercube with XL Weitek chip set
INTEL iPSC/1	Intel 80286/80287-based hypercube
BBN Butterfly	MIMD network of MC68020/68881-based processors
Alliant FX/8	Shared Memory vector multiprocessor
Sequent Balance	NS32032/32081-based shared memory multiprocessor
Sequent Symmetry	Intel 80386-based shared memory multiprocessor with optional scalar Weitek chips
Encore Multimax	NS32332-based shared memory multiprocessor
Cydrome Cydra 5	Very Long Instruction Word machine
CRAY X-MP/48	4-node vector supercomputer
CRAY-2	4-node vector supercomputer with large memory
SCS-40	Vector mini-supercomputer, CRAY X-MP compatible
ETA-10 E	4 vector processors with shared memory
Connection Machine 2	Massively parallel SIMD machine with 16K nodes and Weitek chips

Table 3: Parallel Supercomputers

Machine	Configuration	Key Characteristics
NCUBE hypercube	1024 nodes — SANDIA 576 nodes — Caltech	Scalar nodes with about 0.1 megaflop per node
Transputer Array (MEIKO...)	32 nodes — Caltech Large system coming into use at Edinburgh	Scalar nodes, each with about 4 times performance of NCUBE node
Mark IIIfp hypercube	128 nodes — Caltech (only used in 32 node chunks so far)	Each node is a (short vector) pipelined FPU. 1–2 megaflops with rather disappointing compiler. 5–8 megaflops/node in assembly language (easier than microcode used in previous WEITEK chip sets).
Connection CM-2	64K — Los Alamos 16K — ANL/Caltech 64 K 1 bit processors is really 2048 32 bit processor for floating point work	64K single bit node system peaks out at about 1 gigaflop. Probably will improve.
AMT DAP 510	1024 nodes — ANL	Mesh of single bit processors — faster than those of CM-2

numerical approaches to statistical physics or, in the path integral approach, to quantum field theories. One has an effective partition function

$$Z = \int d\varphi_i \exp(-S[\varphi_i]) \tag{1}$$

where even in today's modest problems, the integral running over the fields $\varphi_i(x)$ can have over one million dimensions (degrees of freedom). Measurements or observables are then found from

$$\langle O \rangle \;= \int d\varphi_i O(\varphi_i) \exp(-S[\varphi_i])/Z \tag{2}$$

In the Monte Carlo method, one replaces the integral (2) by the sum

$$\langle O \rangle = \lim_{N\to\infty} \sum_i O(\varphi_i)/N_c \tag{3}$$

over N_c configurations. These configurations are generated successively by making a series of small changes — usually for single sites [Metropolis:53a]. This ensures that one keeps configurations φ_i distributed according to the function $\exp[-S(\varphi_i])$ which emphasizes the minute region of phase space which is not exponentially suppressed. For the currently hardest calculations — dynamical QCD with a 16^4 lattice — it takes 50–100 hours of time on CM-2 running at one gigaflop to produce a statistically distinct (uncorrelated) configuration. Very many such configurations are needed in the averages eq.(3). The slow evolution produced by the successive application of a single site update, has encouraged development of cluster update methods, but these are only known at the present for the simpler theories. This is a critical issue; not only could clustering speed up the computation, it could alter the necessary architecture as we will illustrate later.

In general, parallelism is straightforward for these problems — one uses domain decomposition of the underlying regular space. There have been several algorithmic improvements recently — which can usually be viewed as better importance sampling in the Monte Carlo integral. The need to continually update algorithms favors relatively general purpose machines with flexible high level software. This suggests to me, in the long run, that commercial parallel machines will be more successful than the many special purpose computers constructed within the high energy physics community.

Discrete Spin: Ising and Potts Models (C. Baillie, P. Coddington)

These have actions S given by

$$S = \sum_{\langle ij \rangle} \sigma_i \sigma_j \tag{4}$$

over nearest neighbors ij in a space of dimension d. The spins σ_i are one bit (Ising) or several bits each (Potts). These problems are ideal for bit serial machines like the AMT DAP or Connection Machine CM-2 — although one cannot use the WEITEK floating point units on the latter for these bit oriented problems.

We are currently using the AMT DAP 510 to investigate a 256^3 three-dimensional Ising model. It is conventional to rate machines by the speed at which they generate new configurations — measured in spin updates/second. The DAP 510 compares well with the fastest supercomputers coming in at 0.6×10^9 spin updates/sec compared with the Hitachi S-820/80, which is only 40% faster [Ito:88a]. However, this performance rating is misleading as our physics calculation is dominated by the calculation by eq.(3) of observables. In fact, we have only crudely coded the update stage on the DAP 510 so that is a factor of 100 slower than the value quoted above. Even this slow update only consumes 1% of the total time. We do expect to optimize (using machine language APAL) the dominant measurement phase and speed up the simulations by a factor of at least ten.

Continuous Spin: Two Dimensional XY and $O(3)$ Models (J. Apostolakis, C. Baillie, R. Gupta)

These systems are given by actions with the form of eq.(4) with spins σ_i that are $N \times N$ matrices of the group $O(N)$. We finished, last summer, a major $O(2)$ (or XY) calculation using the novel "over-relaxation" single site update algorithm [Gupta:88a]. This used the 128 node FPS T Series hypercube at Los Alamos and realized two megaflops per node. These calculations are being continued for the $O(3)$ and similar groups by my graduate student, John Apostolakis. The high statistics of the XY study allowed the refutation of the conclusions of a recent paper [Seiler:88a] and a confirmation of the theoretical predictions of Kosterlitz and Thouless [Kosterlitz:73a]. The T Series hypercube is a poorly designed machine and is only suitable for a small class of regular calculations; it has poor scalar compared to vector performance and slow communication. We used a 256×256 lattice allowing the other dimension to be viewed as a large vector (multiples of 128) achieving good performance from the inflexible WEITEK chip based node. Some of our calculations have been marred by hardware glitches, requiring that one, for instance, avoid the hardware vector divide and long vector (> 256) instructions.

Niedermayer and Wolff [Niedermayer:88a], [Wolff:89a] have introduced effective clustering methods which currently we believe are unsuitable for the FPS architecture. We expect to use the MEIKO transputer array to continue these calculations. Note that the SIMD CM-2 would perform well on the algorithm we used for the T Series, but we currently believe that a true MIMD architecture may be needed for the clustering calculation. The regular vectors needed by the T Series, make this old hypercube essentially SIMD in character; more precisely in the language of [Fox:88b], it requires synchronous problems for good performance. The currently known clustering algorithms are properly loosely synchronous. We are currently experimenting with SIMD implementations which, although inefficient, may still have sufficiently good performance [Baillie:89c].

Pure Gauge QCD (S. Otto, J. Flower, H. Ding, C. Baillie)

In this case, the action S takes the form

$$S = \beta \sum_{\substack{\text{plaquettes} \\ p}} (1 - ReTrUp) \qquad (5)$$

where Up is the product of (SU(3)) link matrices around the elementary plaquettes — these are the eight 1×1 loops including a given link which joins two sites. This calculation can be both vectorized and parallelized even for the so called random block lattice [Chiu:86a], [Chiu:88e]. It can achieve good performance on essentially all architectures with MIMD, SIMD, or vector characteristics. The large number of floating calculations in eq.(5) (explicitly 3347 for each link) dwarfs overheads such as communication.

Even a modest calculation on a 16^4 lattice requires 262,144 degrees of freedom, 19 megabytes of memory and of order 10^{15} flops of CPU power. This problem was originally tackled [Otto:84a] on the first 64 node Cosmic Cube Hypercube [Seitz:85a] with a $12^3 \times 16$ lattice and repeated on the 128 node Mark II hypercube with a larger 20^4 lattice. These machines had peak performance below 5 mflop. We have implemented this application on several more powerful machines with the performance given below.

CRAY XMP (1 processor)	60 mflops
NCUBE (1024 nodes)	80 mflops
JPL Mark IIIfp hypercube (128 nodes)	500 mflops
Connection Machine CM-2 (64K nodes)	900 mflops

In Figure 1, we show results from Ding on the Mark IIIfp hypercube on $24^3 \times 10$ lattice [Ding:89a]. This implementation involved 8,000 lines of C and WEITEK XL assembly code — the commercial compiler for this pipelined chip set is poor. The CM-2 results correspond to 3,000 lines of *LISP written by Brickner [Brickner:89a]. One needs eight virtual processors (corresponding to eight separate calculations) to get good performance.

Figure 3 shows the performance evaluation for this algorithm; this used FORTRAN and C codes based on Otto's original hypercube program. Amusingly, this version of the code vectorizes poorly even though we know that with an optimized implementation, the algorithm performs well on the vector supercomputers.

The flexibility of parallel machines is illustrated by Chiu's calculations [Chiu:88e] illustrated by Figure 2. These use a random block lattice where a different 40^4 lattice is calculated on each node of the 1024 node NCUBE hypercube at SANDIA; an *embarrassingly parallel* application in the language of [Fox:88b].

Dynamical Fermion QCD (C. Baillie, R. Brickner, R. Gupta, G. Kilcup, A. Patel, S. Sharpe)

This "ultimate" QCD calculation includes the quark (fermion) degrees of freedom φ at the lattice sites with action

$$S = -\varphi^\dagger (D + m)^{-1} \varphi \qquad (6)$$

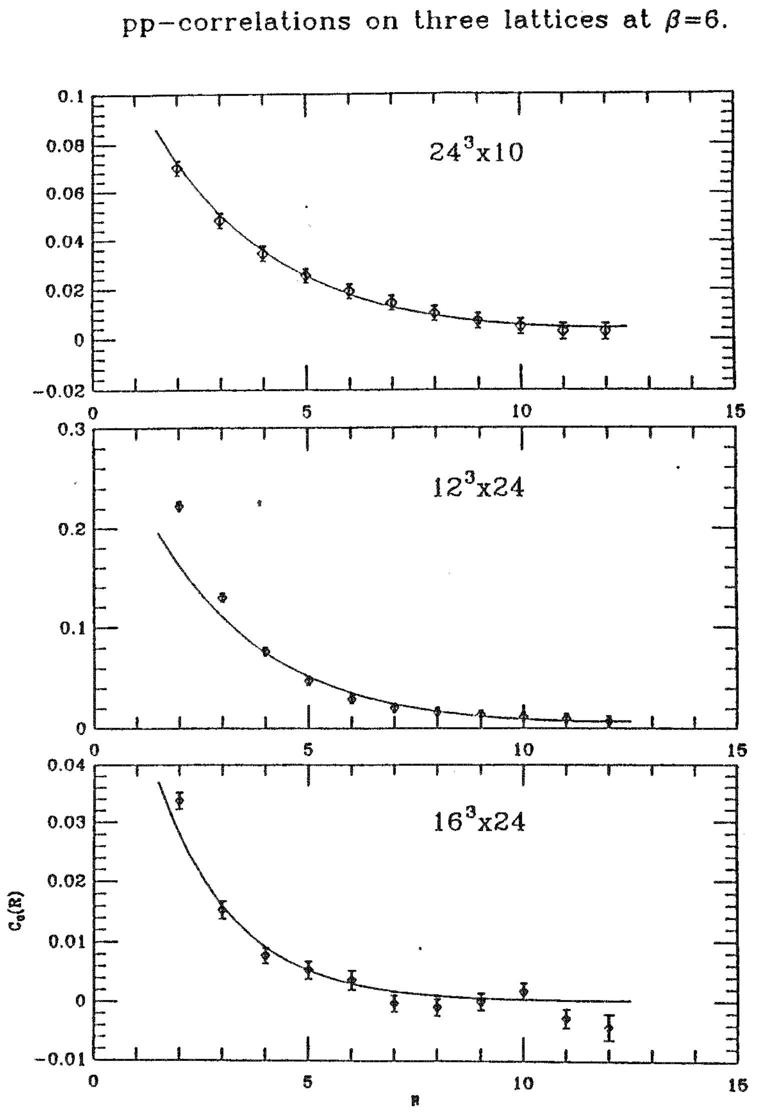


Figure 1: The zero momentum correlation function used to extract the $\bar{q}q$ potential from three lattices at coupling $\beta = 6$ [Ding:89a]. Each graph summarizes about two weeks worth of computation on a 32 node Mark IIIfp hypercube.

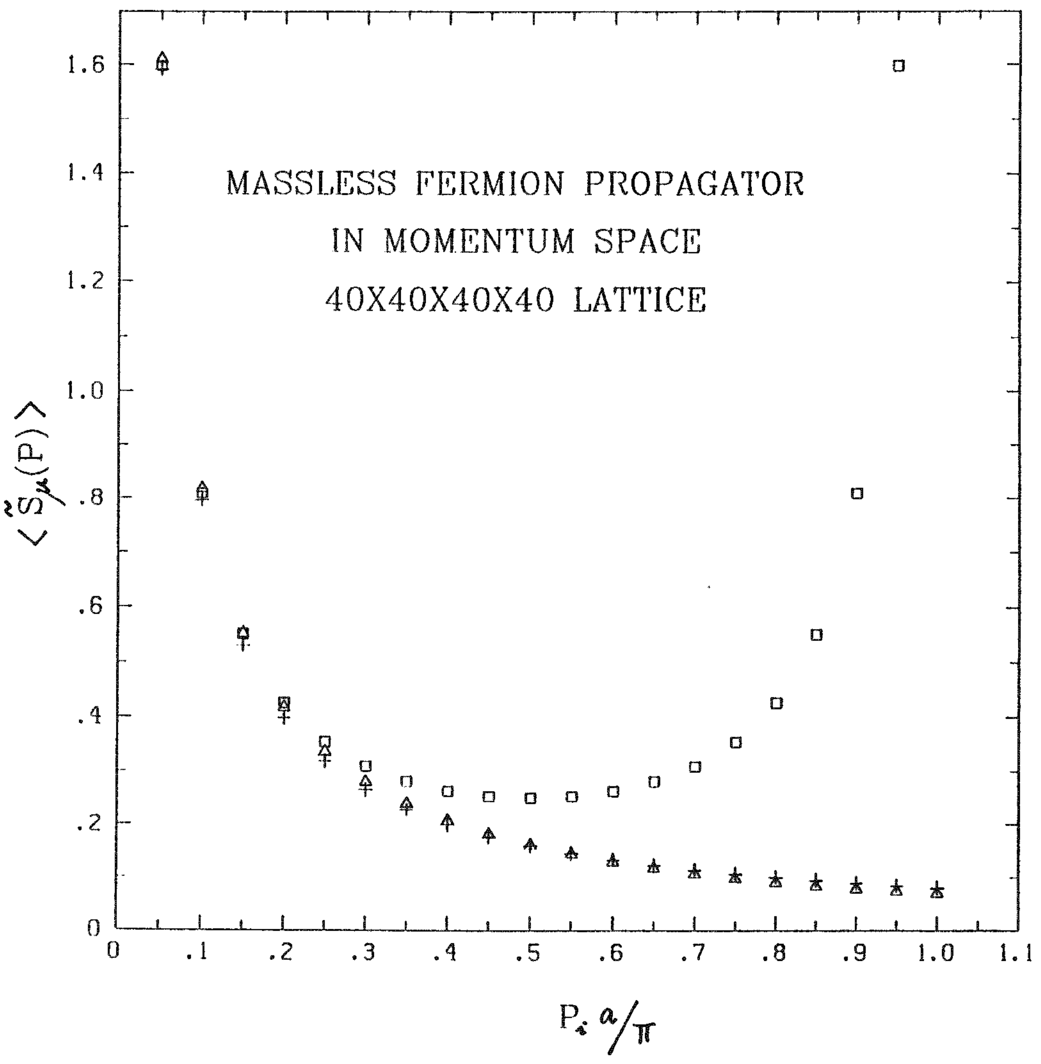


Figure 2: Massless Fermion Propagator in Momentum Space verifying that the new random block lattice method extrapolates properly to the continuum limit [Chiu:88e]. The 40^4 lattice was simulated on the 1024 node SANDIA NCUBE.

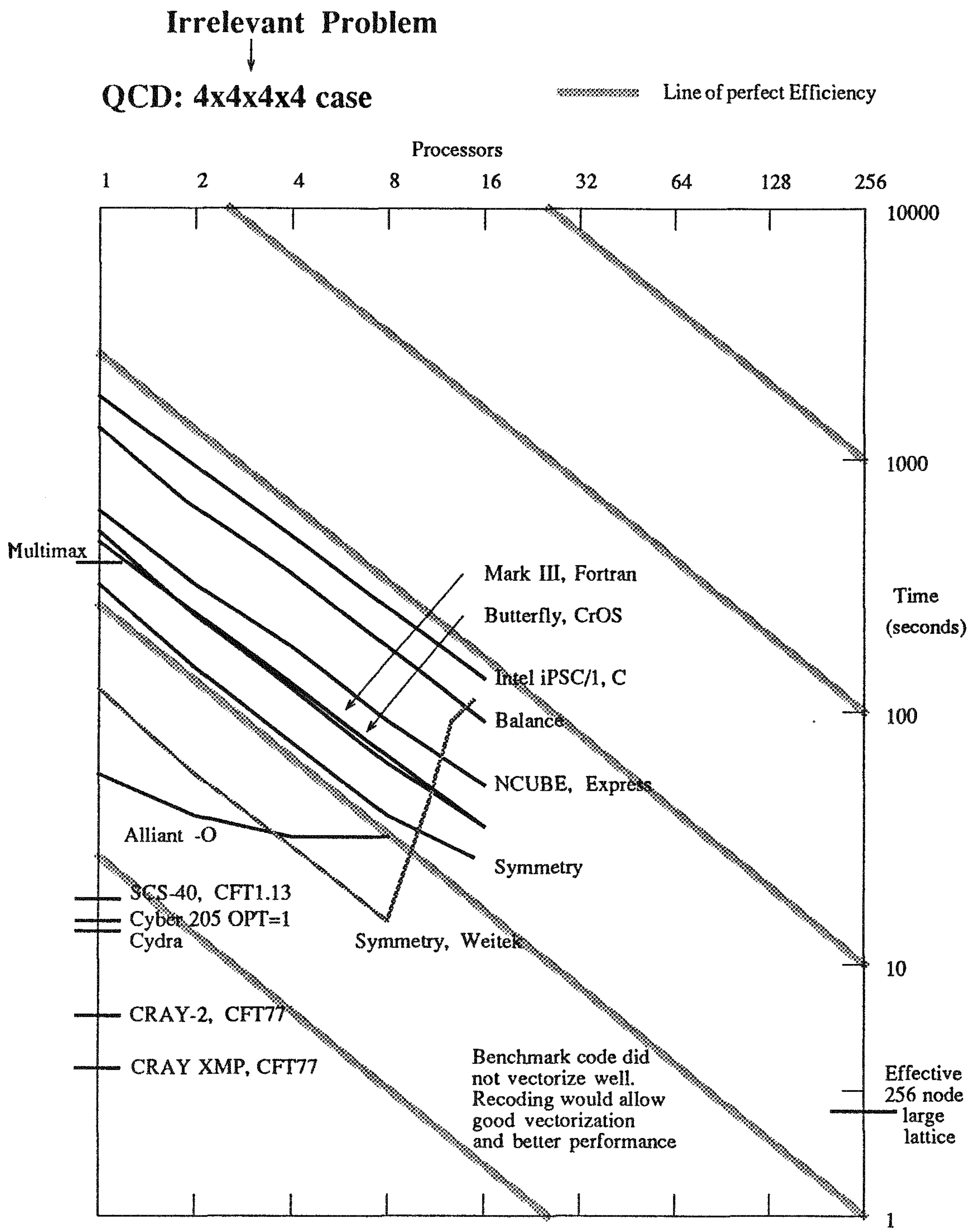


Figure 3: Performance of a set of computers on the 4^4 lattice QCD problem studied in Figure 1 [Messina:89a]. The results should be interpreted carefully as realistic calculations need much larger lattices.

Technically, one uses a Wilson fermion formulation and this is dominated by inverting the 65536×65536 (on a 16^4 lattice) sparse fermion matrix $D + m$. This is currently done with a conjugate gradient or minimal residue method. The CRAY-2 uniprocessor code runs at 100 megaflops and the full size Connection Machine CM-2 has about one gigaflop performance [Baillie:89e]. We expect this to improve as the CM-2 support software for floating point calculations is developed. We anticipate this problem to need 10^{16} flops or $\sim 3,000$ CM-2 hours for even initial studies.

Currently, there are no good clustering algorithms for either the pure gauge or fermion QCD problem. The regular "dumb" algorithm used so far does not require a sophisticated architecture. This could change when an effective clustering method is introduced.

The Competition

DOE has awarded two collaborations large ($\sim 6,000$ hours) blocks of time for "grand challenge" QCD calculations on the CRAY-2 and ETA-10 for QCD calculations. NSF has certainly already devoted much more time than this to this problem at the supercomputer centers. Conventional and parallel supercomputers are compared in Table 4.

With several months running on the CM-2 or Mark IIIfp, parallel supercomputers are very competitive with the ETA-10 and CRAY-2. The scalar node MIMD architecture of the NCUBE is not well optimized for this highly regular vectorizable problem.

2.2 High T_c Superconductivity (Barnes, Kotchan)

A Toronto group has been using the Caltech NCUBE to study the Quantum Anisotropic Heisenberg Model [Barnes:88a], [Barnes:88b], and [Barnes:89a]. This problem is related to the systems of Sec. 2.1, except one is studying the dynamical and not the statistical properties of a spin system. The Hamiltonian H is given by, for a nearest neighbor sum i, j over a two dimensional grid:

$$H = \sum_{(i,j)} [S_i^z S_j^z + g(S_i^x S_j^x + S_i^y S_j^y)] \qquad (7)$$

($g = 0$ is Ising, $g = \infty$ is XY model)

This can be studied as a three dimensional lattice theory, but Barnes has developed an equivalent random walk approach for solving Schrödinger's equation with an imaginary time T:

$$-\frac{\partial \psi}{\partial T} = H\psi(T) \qquad (8)$$

Typically, an 8×8 lattice is evolved separately on each node by the NCUBE with approximately 10^6 independent evolutions needed. This algorithm is reminiscent of the (far more complex) neutron transport calculations studied at DOE laboratories. This embarrassingly parallel algorithm was easily implemented in C on the NCUBE. The 256 node NCUBE hypercube achieved three times the performance of the original FORTRAN implementation for the CRAY XMP.

Figure 4 shows the NCUBE calculation revealing structure at the transition point $g = 1$. Current calculations correspond to several hundred hours of CRAY XMP time.

2.3 Exchange Energies in He^3 at a Temperature of 0.1 mK° (S. Callahan, M. Cross)

Callahan's Caltech condensed matter Ph.D. involved a Monte Carlo method to calculate exchange energies in solid He^3 [Callahan:88a], [Callahan:88b]. Rather modest systems were used with 54–128 particles arranged in a three-dimensional spatial mesh which is further extended in time. Use of the 512 node NCUBE required parallelism in several aspects of the problem. The forces are not nearest neighbor and decomposition of their calculations over space leads to a factor of four in parallelism. Decomposing time (direction of path) leads to another factor of 16. This 64 fold data parallelism is combined with $2 \rightarrow 8$ independent runs (*i.e.*, decompose space of random configurations in the integral of eq.(2)).

In an unfair comparison, the 64 node NCUBE has an efficiency of only 64%, but outperforms the CRAY XMP by over a factor of two. However, this used the C language for the CRAY which only realized a few megaflops. Note more positively that our implementation used Salmon's CUBIX environment [Fox:88a] allowing the identical code of about 2500 lines of C to run on either the NCUBE hypercube or CRAY!

Callahan's thesis involved a total of about 250 hours computation on the 512 node NCUBE — another supercomputer level calculation.

2.4 Computational Fluid Dynamics (CFD)

Caltech has so far not solved any large CFD applications on parallel machines although the parallelization methodology is clear and for instance, the geophysics group has made extensive hypercube calculations of related finite element problems [Nour-Omid:87b], [Raefsky:88a], [Raefsky:88b], [Gurnis:88a], [Lyzenga:85a], and [Lyzenga:88a]. We mention here the DIME project of Williams which generates a general irregular finite element mesh and solves the resultant equations [Williams:88a], [Williams:88d]. Currently, DIME can only tackle two dimensional nonlinear or three dimensional linear problems with triangular elements. Current applications include Navier Stokes simulations in two dimensions [Williams:89a], calculation of fields due to an electric fish and high energy physics string dynamics. Interesting model problems, but not supercomputer level [Fox:88v].

In Figure 5, we show a demonstration project with the mesh generated for a Mach 3 flow over a step [Williams:88e]. DIME both generates the mesh and automatically dynamically load balances the mesh points to optimize machine performance. At one time, we thought that decomposition of irregular problems would be a stumbling block; however, it is now clear that it is straightforward. Williams uses an orthogonal recursive bisection

Table 4: Approximate Dynamical Fermion QCD Performance for the "Grand Challenge"

Machine	Performance (megaflops)	Time Allocated in 1989 and comments
ETA-10 (1 processor)	350	1 year
CRAY-XMP (1 processor)	100	1 year
Mark IIIfp (128 nodes)	750	
NCUBE (1024 nodes)	100	Not competitive
CM-2 (64K nodes)	>1,000	
Several SIMD Coarse	1,000→	limited by
Grain Special Purpose Computers	10,000	software

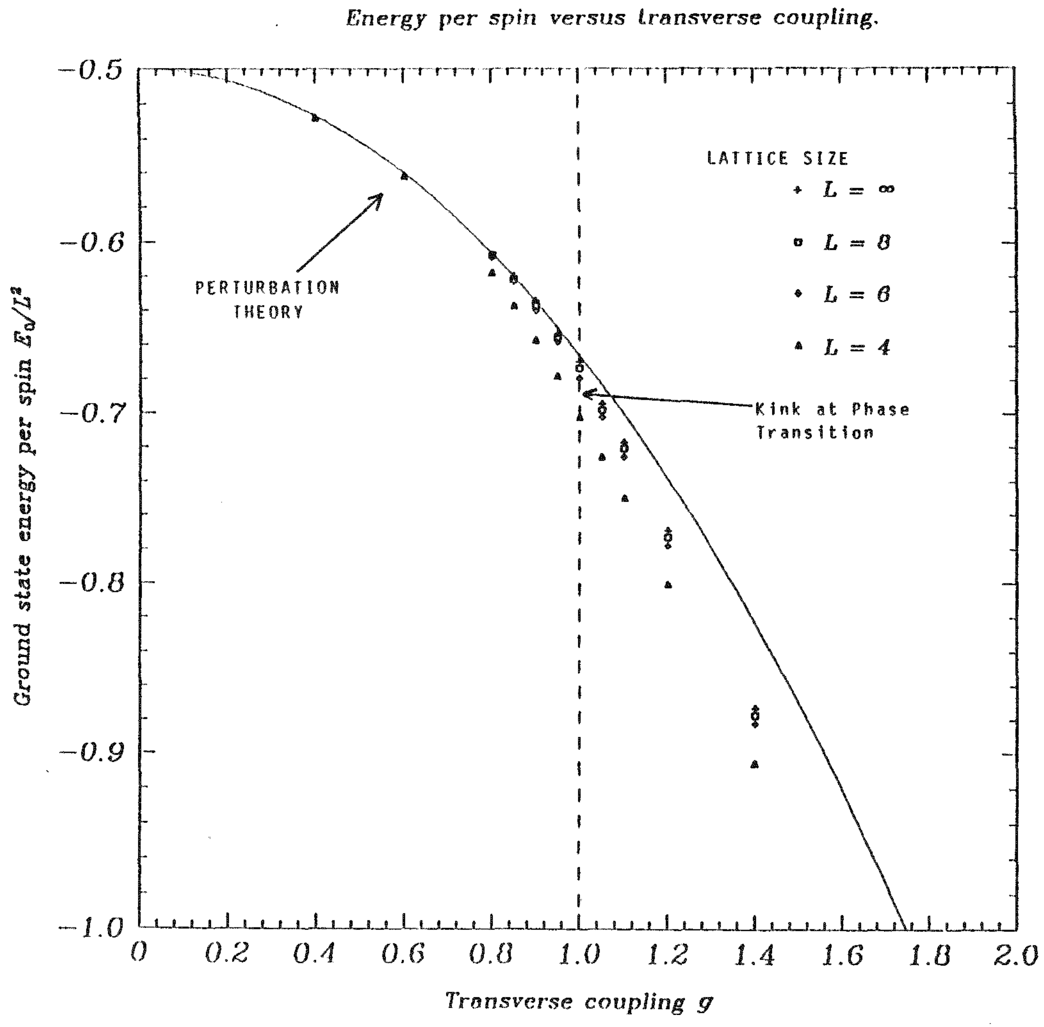


Figure 4: Energy per spin versus transverse coupling for the anistropic Heisenberg model simulated on the NCUBE [Barnes:89a]. Significant is the transition at coupling $g = 1$.

method [Fox:88nn], [Baden:87a] which is sufficient for this problem. More powerful methods based on neural networks and other heuristic optimization methods apply very generally [Barhen:88a], [Chen:88a], [Ercal:88a], [Fox:88e], [Koller:88a], [Fox:88f], [Livingston:88a].

The irregular adaptive mesh is naturally implemented as a linked list data structure. This is hard to vectorize on conventional supercomputers, but parallelizing well at least in MIMD machines, such as the NCUBE. It would be interesting to study the SIMD implementation. Seemingly, SIMD machines would cope with data access irregularity, but the irregularities in the computational graph (varying multiplication of nodal points) cause inefficiencies in the SIMD case, which are absent for MIMD machines.

2.5 Plasma Physics (Liewer, Zimmerman (JPL), Decyk, Dawson (UCLA))

Plasma Physics computations represent interesting challenges for distributed memory machines because the PIC or particle in the cell algorithm used involves two distinct decompositions. Our example problem involves calculation of the orbits of plasma electrons in their own electromagnetic field as considered by a JPL-UCLA group on the JPL Mark IIIfp hypercube [Liewer:89a], [Liewer:88e], [Liewer:88b]. In the first stage of the calculation, one finds the field using an FFT; this involves a decomposition with an equal number of mesh points on each node of the hypercube. Then, one transforms to a separate decomposition shown in Figure 6(a) with equal number of particles in each node; this latter is the particle update or "push" part of the computation where the particle positions are evolved in the field. Each stage can be efficiently implemented on the hypercube, but transforming between the two distinct decompositions must be done at each time step. A general strategy for this has been discussed by Walker [Walker:89a] at this conference using the *crystaLaccumulator* algorithm [Fox:88a]. This does not attempt to localize the calculation for each particle to a single node, but rather distributes it with calculations done "on the fly" as information is routed through the hypercube. This method was originally developed by Furmanski for neural network simulations [Furmanski:87a], [Fox:88g]. It can only be implemented well on machines like the NCUBE and transputer arrays; the Mark IIIfp and Ametek S2010, where calculation and communication subsystems are separated, do not support the *crystaLaccumulator* well. If important, this algorithm requires communication subsystems that support a combination of messages in a similar fashion to fetch and add, as proposed, for combining networks in shared memory parallel computers [Gottlieb:86a].

Currently, we only have measurements for the more straightforward strategy where information is routed to the destination node and then combined; rather than being combined en route. This is almost certainly the best algorithm for the Mark III hypercube which has separate communication and calculation subsystems. The combining overhead would be severe for Walker's approach on this and similar machines, where the interface between communication and calculation on the node introduces a significant latency. In Table 5, we compare the performance of (a) the push stage and (b) the total code for a variety of machines. The 64 node Mark IIIfp hypercube is about twice the performance of the CRAY XMP on the push stage, but only comparable for the total calculation. This indicates that either the FFT or movement between decompositions is inefficient on the hypercube. Note that both the CRAY-XMP and Mark IIIfp are running far from their peak vector performance; each is about a factor of eight below peak. In Figure 6(b), we show the performance comparisons from [Messina:89a] for the full calculation. This implementation involves 4,000 lines of FORTRAN.

2.6 Astronomical Data Analysis (Anderson, Gorham, Kulkarni, Prince)

This group has pioneered the use of the NCUBE for astronomical data analysis [Fox:88v]. Our Caltech NCUBE system has a small (4 disks) parallel disk farm connected to the main hypercube; these disks are controlled by a SUN-4 which also has additional peripherals, including the necessary tape drives.

Radio Astronomy

In the most exciting work, radio data from the Arecibo radio telescope was taken on December 26, 1988 in particularly advantageous circumstances in that the holiday spirit reduced the ambient interference — especially from a nearby naval base. Data is taken with a 0.5 millisecond time interval and Fourier transformed (a large 2^{24} one-dimensional FFT) to look for peaks corresponding to radio pulses from the rapidly rotating neutron star. Two new pulsars have been discovered, using the NCUBE. Both are located in the globular cluster M 15, making a total of three known pulsars in this globular cluster. The discovery of these pulsars has prompted a reanalysis of current ideas concerning the origin of neutron stars in globular clusters.

The computation involves both the FFT which is efficiently implemented on the hypercube, and an I/O intensive stage taking a total time comparable to the FFT. This first I/O dominated stage is overlapped with a calculation which corrects for frequency dispersion in the interstellar medium. The measured I/O performance of the system is modest at 40 Kbytes/sec/drive including all overheads. Our ESMD disk drives on the NCUBE are rated at a factor of 25 higher performance and this suggests the need for better I/O software on the NCUBE.

The striking peak corresponding to the neutron star rotation period of 30.5 milliseconds is shown in Figure 8 for the second pulsar discovered by the NCUBE. This pulsar is part of a binary system and required a further compute intensive "acceleration correction" to remove the orbital effects of the binary system. This additional computation has negligible I/O to disk but substantial internode communication. Processing 90 minutes of data taken at Arecibo takes about two hours for the dispersion correction and FFT stages (which discovered the first pulsar), while about 40 hours of 512 node NCUBE time were needed for the pulsar shown in figure 8. The major I/O and large

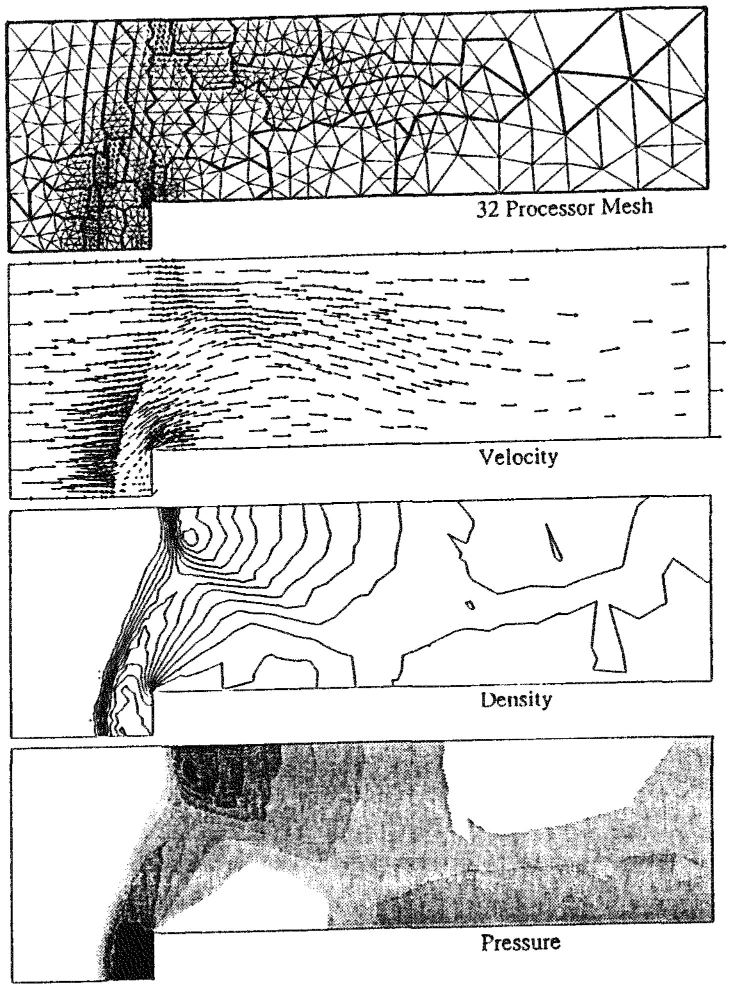


Figure 5: Simulation of Mach 3 flow over a step [Williams:88e] showing fluid velocity, density and pressure. This was solved with an adaptive mesh (thin solid lines) on a 32 processor NCUBE hypercube with dynamical load balancing shown by thick solid lines in top diagram.

Table 5: Performance of One-Dimensional BEPS1 Plasma Physics Code [Liewer:89a]

(a) **Comparison of Push Times per Particle on Various Computers**

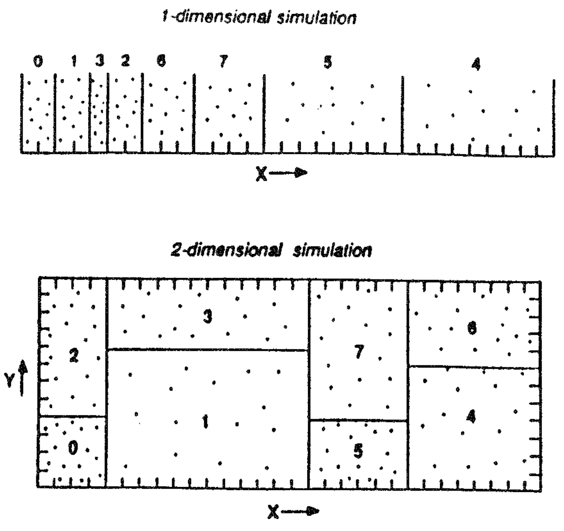
	Computer	Push Time μsecs	
(Particle Update Only)	Mark IIIfp (64 processor)	0.8	($\sim$50 megaflops)
	CRAY XMP/48 (1 processor)		
	Vectorized	1.5	($\sim$25 megaflops)
	Scalar	4.1	
	CRAY 2 (1 processor)		
	Vectorized	2.1	
	Scalar	10.1	
	IBM 3090 VF		
	Vectorized	2.9	
	Scalar	6.0	
	Mark III (64 processor)	3.9	
	Alliant FX/8	12.6	
	VAX 11/750, F.P.A.	200.9	
	Convex C-1 (vector)	19.5	

(b) **Total Run Time Comparison**

(Particle and Grid Updates) (720,896 particles, 1024 grid points, 1000 time steps)

Mark IIIfp (64 nodes)	1062 secs
Cray 2 (1 processor)	1714 secs

(a)



(b)

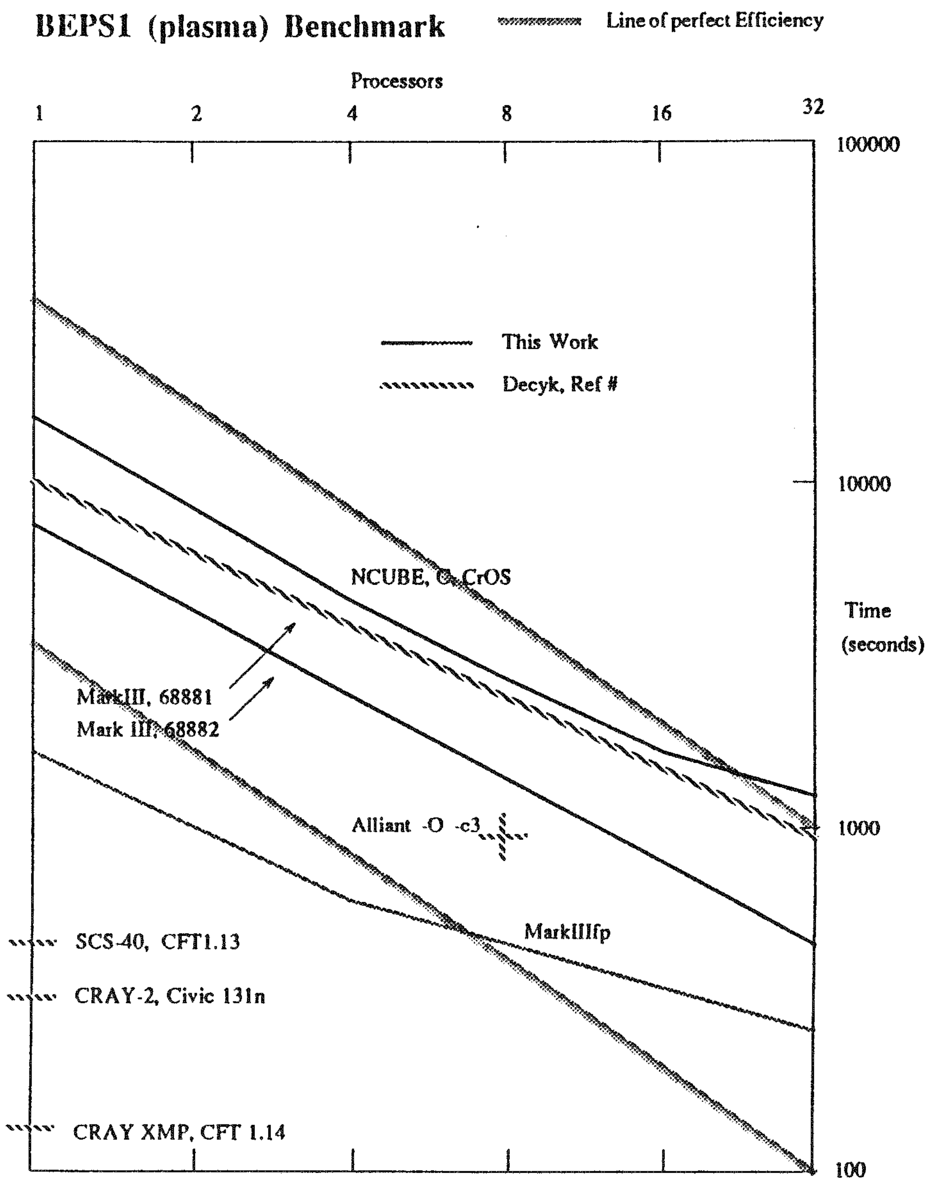


Figure 6: (a) The particle distribution in the load balanced "push" stage of the PIC code of [Liewer:89a] in one and two dimensions; (b) The benchmark results of [Messina:89a] for the one dimensional particle in the cell UCLA code [Decyk:88a] called BEPS1.

memory requirements of this calculation make it hard to compare the NCUBE with a CRAY or IBM 3090 performance.

Optical Astronomy

Traditionally the resolution of ground based optical telescopes is limited by atmospheric turbulence to abaout one second of arc (1/3600th of a degree). However, it is possible to eliminate the effects of turbulence using interferometry techniques familiar from radio astronomy, and achieve resolution of 30 milliarcseconds. Using the 200" Mount Palomar telescope, one divides the total aperture into approximately one thousand 15 cm disks. The correlations between these disks are summarized in 10^6 Fourier coefficients — the bispectral function. These are averaged over many samples or frames lasting from 10–100 ms over which the turbulence is essentially constant. This technique has been implemented on the NCUBE where a 20 minute Palomar exposure on an asteroid was analyzed for 10 hours on a 256 node subcube and obtained the best image resolution ever seen for an asteroid. Similar results are shown in Figure 7 where the new technique resolves a binary star; the improvement in resolution is about a factor of 30 over traditional methods in each linear dimension.

This novel method is still in its infancy; it illustrates graphically how powerful computers can open up new approaches to scientific problems.

2.7 Quantum Chemistry Reaction Dynamics (Wu, Hipes, Kuppermann, Cuccaro)

Kuppermann's group has been developing a fundamental approach to the understanding of chemical reactions [Cuccaro:88a], [Kuppermann:86a]. A goal is the description of reactions like $F+H_2 \rightarrow FH+H$ which are the bases of an important chemical laser. This is a difficult computation involving a factor of a thousand more computation than the prototypical initial example.

$$X + H_2(v,j,m) \rightarrow \begin{cases} X + H_2(v',j',m') & \text{nonreactive} \\ H + XH(v'',j'',m'') & \text{reactive} \end{cases} \qquad (9)$$

where

$v = 0,1,2,\ldots$; labels vibrational energy content
$j = 0,1,2,\ldots$; labels rotational energy content
$m = 0,\pm1,\pm2,\ldots,\pm j$; labels spatial orientation of molecule

Operationally, this problem involves solving Schrödinger's equation $H\psi = E\psi$ for the wave function ψ in the novel hyperspherical basis set. Here H, the Hamiltonian, is a second order linear differential operation in six variables and E is the total collision energy. This computation breaks down into two phases which we describe separately below [Hipes:88a], [Hipes:88b].

Phase I: Calculate Basis Functions and Matrices

1. Construct a primitive basis set composed of product of analytic and numerical functions — each processor solves an independent tridiagonal eigenvalue problem using bisection.
2. Evaluate 2D integrals using the primitive basis functions which requires spline interpolation onto the quadrature grid. Each processor calculates a subset of the matrix of such integrals.
3. Assemble the integrals into a real symmetric matrix. Parallel reduction by Householder transformations. Redundant tridiagonal QR algorithm in each processor to get eigenvalues/vectors.
4. Calculate more matrices of integrals using primitives and coefficients from step 3. These matrix elements are distributed among the processors.
5. I/O to store matrices
6. Repeat 51 times

Steps 1, 2, and 4 are trivially parallel and involve independent computation. Step 3 needs significant parallel algorithms adapted from work by Patterson at JPL [Patterson:88a].

This has been fully implemented for $H + H_2$ reactive scattering on the 32 node Mark IIIfp hypercube with 3,000 lines of coding. The total runtime time was 9.8 hours of which 2.2 hours were I/O (stage 5). This will be improved soon with the high performance CIO (Concurrent I/O) hardware on the Mark IIIfp. The same calculation on the SCS-40 (which has about 25% the performance of the CRAY XMP) took 71 hours.

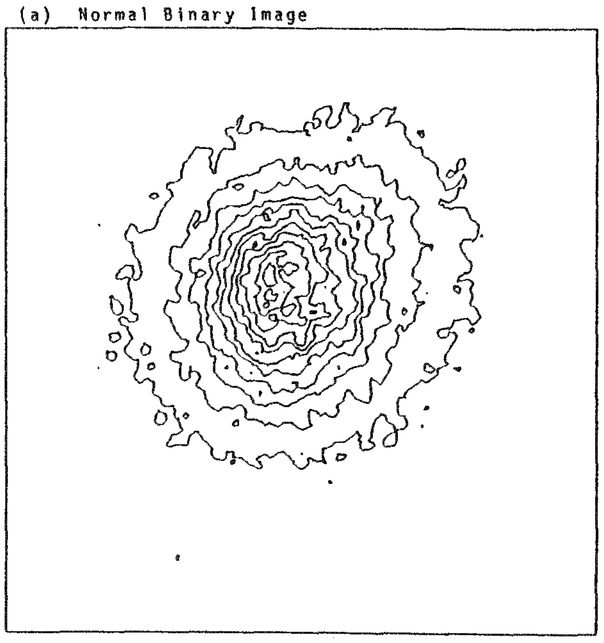
This initial phase is followed by:

Phase II: Integration of Coupled Linear Systems of Ordinary Differential Equations

This phase uses Johnson's algorithm [Johnson:73a] — a fourth order special purpose integrator for chemistry simulations. The resultant algorithm is dominated by matrix inversion (and not LU decomposition!) with some matrix multiplication. I/O is also needed to initialize with the results of Phase I which determined the matrix elements. This involved 2,000 lines of code and 74 coupled ordinary differential equations integrated for 250 steps. 31 energies were calculated simultaneously to reduce I/O overheads and the resultant calculation took two hours on the Mark IIIfp with an additional I/O overhead of 10%. A typical transition probabiality curve is compared with the corresponding SCS-40 calculations in Figure 9.

In Figure 10(a), Messina's group has accumulated the results of the Phase II (or LOGD for logarithmic derivative) benchmark which is essentially 65×65 matrix inversion [Messina:89a]. The vector machines do well on this calculation and these systematics are summarized in Table 6.

The good performance of the CRAY on matrix algorithms is further illustrated in Figure 10(b) for matrix multiplication. The NCUBE looks good compared to CRAY's



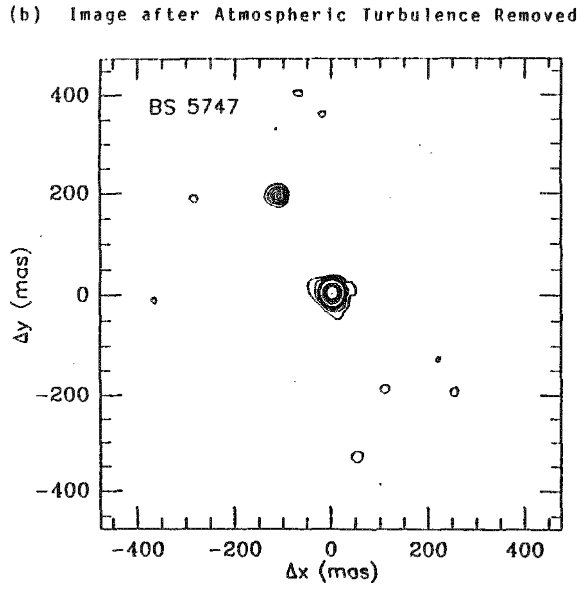


Figure 7: (a) The raw data which is a six second exposure of BS 5747 (β Corona Borealis) with a diameter of $\sim$ 1 arcsec. The magnification of (a) and (b) are $\sim \times 1000$ in linear scale over how they appear at the 200" Palomar telescope focus; (b) Reconstructed image after NCUBE analysis, with same scale as (a). This reconstruction required about 6,000 frames of 100 ms duration. The central star is approximately six times brighter than the companion.

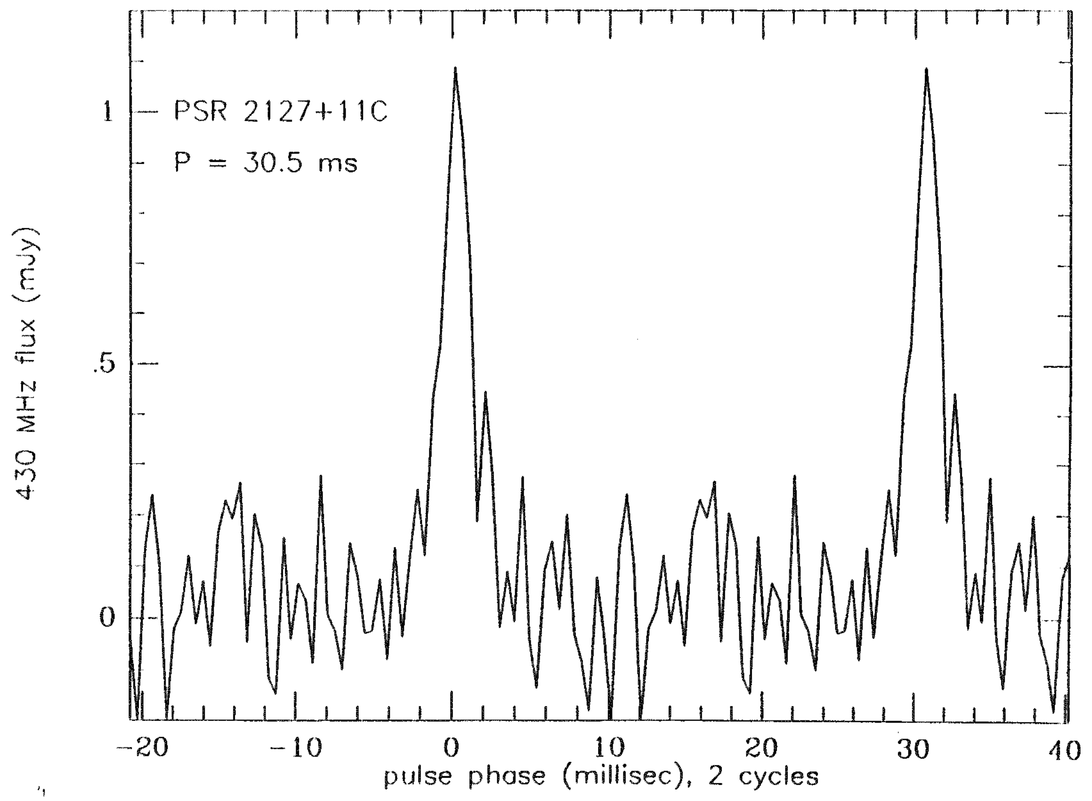


Figure 8: The peak, after NCUBE analysis, corresponding to a new binary pulsar PSR 2127+11C discovered in the globular cluster M 15 with a pulse period of 30.5 millisecs. The data was taken at the Arecibo Radio Telescope at 430 MHz on December 26, 1988.

Table 6: Performance of High Performance Computers. The listed numbers are approximate megaflops.

	CM-2 64K "huge" vector	CRAY XMP 1 processor "long" vector	Mark IIIfp Hypercube 128 nodes "short" vector	NCUBE Hypercube 1024 nodes scalar
Super Regular (e.g., large full matrix)	3000	200	750	100
Typical Regular (e.g., QCD)	1000	75	500	100
Irregular (e.g., chess, clustering)	Fails	10	100	50

$$H + H_2\,(v=0, j=2, m=1) \xrightarrow[E]{} (v=0, j=2, m=1)\,H_2 + H$$

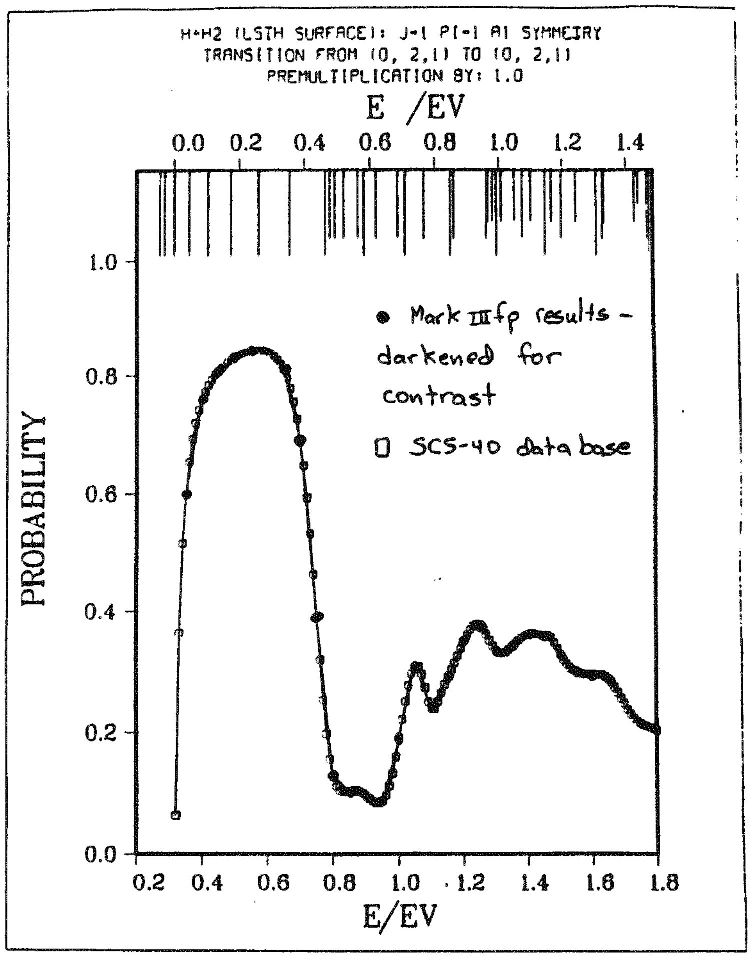


Figure 9: Results for the chemical reaction $H + H_2(v = 0, j = 2, m = 1) \rightarrow (v = 0, j = 2, m = 1)H_2 + H$ and a comparison of the SCS-40 and Mark IIIfp hypercube calculations [Hipes:88b].

(a)

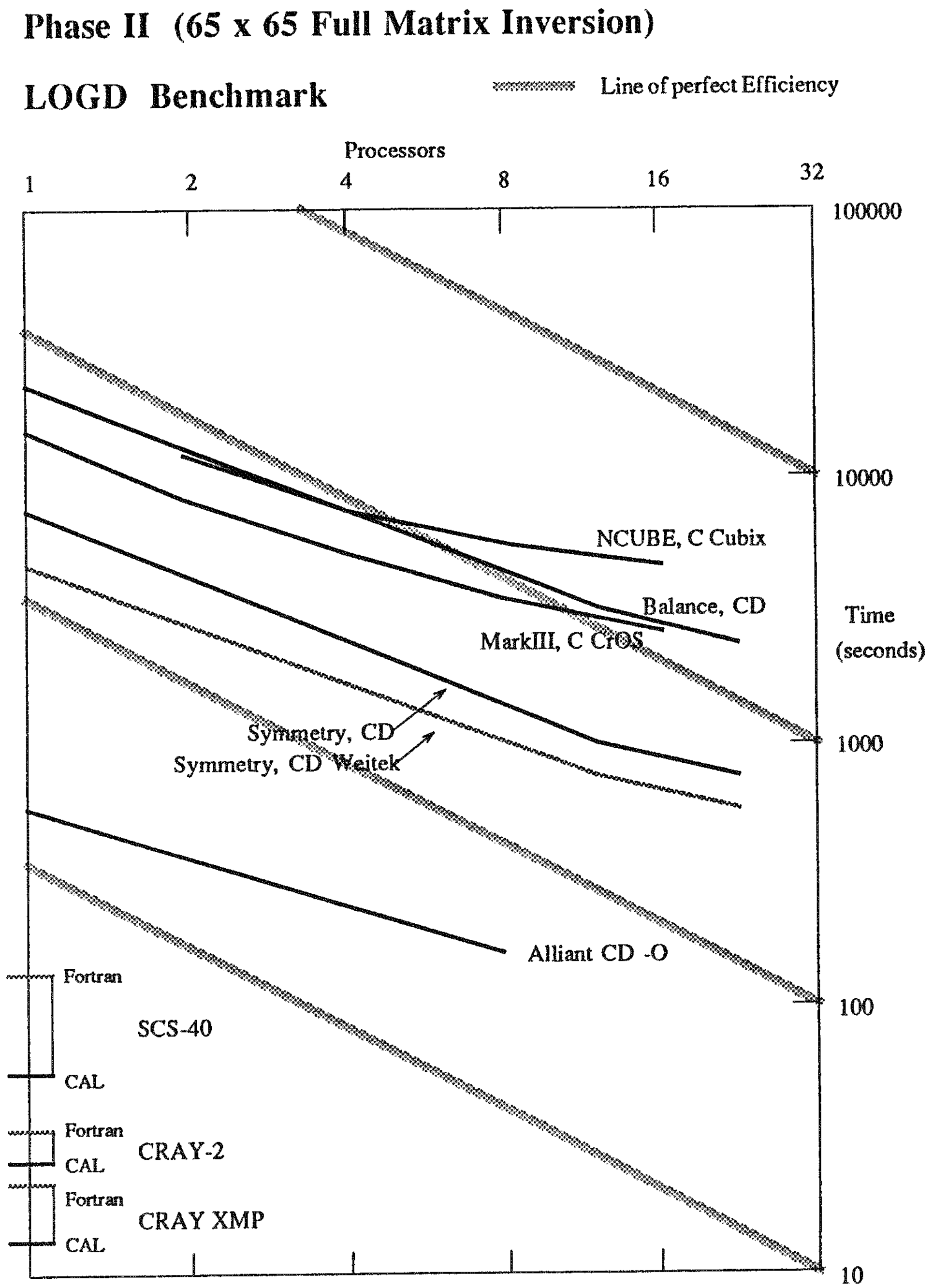
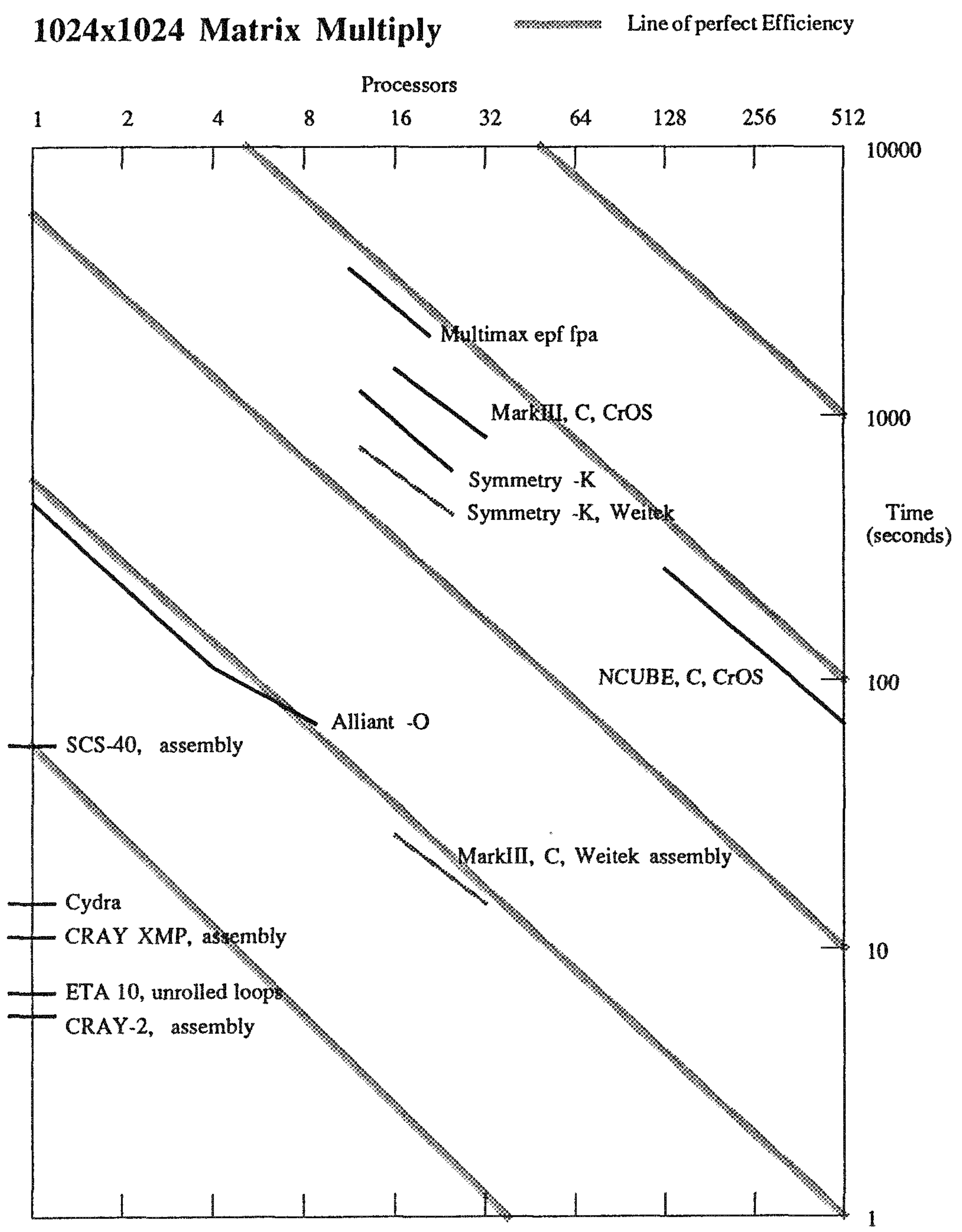


Figure 10: (a) Results from [Messina:89a] for the LOGD code which is essentially the 65 × 65 matrix inversion kernel of the logarithmic derivate chemical reaction code. (b) A further linear algebra benchmark from [Messina:89a]. This graph is for 1024 × 1024 matrix multiplication.

(b) **1024x1024 Matrix Multiply**



on problems like those in Secs. 2.2, 2.3, 2.4, 2.6, 2.10, 2.11, which are irregular and vectorize poorly. Matrix algebra and, to a lesser extent, algorithms like QCD (Sec 2.1) show the vector machines at their best. This is not surprising — they were designed for this problem class! Table 6 attempts to show a progression from SIMD to long vector supercomputers to hypercubes with short vector nodes to scalar node MIMD machines. This corresponds to increasingly general purpose machines. As commented in Sec. 2.1, machines like the NCUBE are not competitive in peak performance on regular problems, but they offer good performance on range of problems and their speed degrades slowly as one increases the irregularity of the problem.

2.8 Grain Dynamics by Lattice Gas (Gutt)

A very interesting use of the NCUBE was recently reported in Gary Gutt's Ph.D. thesis at Caltech [Gutt:89a]. Earlier, Werner's Ph.D. research had studied grain dynamics using the first hypercubes [Werner:87a], [Werner:88a]. This work put in detailed Newtonian dynamics to study the motion of sand and other granular material. This is an interesting alternative to conventional continuum approximations to material dynamics. Gutt proposes an intermediate model for such systems using cellular automata or lattice gas techniques that have already been applied to fluids [Frisch:86a]. Gutt's automata are quite dense (of order one for every two lattice sites) and one must store the relative displacements of each automata from the lattice site positions. Thus, this automata method does not use binary arithmetic, but rather 32 bit arithmetic. To improve performance, integer and not floating point arithmetic is used. Gutt's thesis used about 200 hours of NCUBE 512 node time with the largest simulation involving 0.5×10^6 grains on a 8064×128 lattice. This involves Poiseuille flow down a pipe driven by gravity. We do not have a CRAY implementation of this code, but it is possible that irregularities in lattice site occupancy would make vectorization difficult. The parallelization on the NCUBE is straight forward and efficient.

2.9 Ocean General Circulation Model

This is a salutary lesson in parallelizing dusty decks. Our original plan was that this 20,000 line FORTRAN CRAY code would typify issues involved in converting similar but larger and more sophisticated meteorological codes. The program solves a three dimensional ocean model with the Navier Stokes equations and driving terms from wind, temperature and salinity. It uses a time stepped evolution with successive over relaxation to solve Poisson's equation for the fluid.

We kept a careful record of the time spent on this project recorded in Table 7. Parallelization involved a simple domain decomposition implemented by changing DO loop indices in the original code and adding communication calls. There was an amusing (frustrating) difficulty with decomposing in the north-south dimension which we now believe was an unphysical approximation introduced to improve an original small memory CYBER 205 version. As it stands, the code could only be parallelized (decomposed) in the other two directions (east-west, depth). The parallel code used all 256 megabytes of memory on the 512 node NCUBE and did not need memory management necessary in CRAY version. This 512 node hypercube performance was comparable to that of CRAY.

A success, you might think, but there is a tragic end as the NCUBE version does not currently agree with that for the CRAY. Maybe this is a bug introduced by the parallelization, but we doubt it. We have studied the CRAY code and believe it is incorrect; maybe vectorization introduced an error in handling the boundary conditions? We tried to obtain help from the originators of the code, but to no avail. We could find no one who would take responsibility for the version we were dealing with.

We deduce from this experience that parallelizing existing code can be quite simple and quick — see the "three day" entry in Table 7 for the essential parallelization step. However, such endeavors should only be undertaken with the help of someone really knowledgeable in and responsible for the sequential code.

2.10 Astrophysical Particle Dynamics (P. Quinn, J. Salmon, M. Warren)

N-body calculations have been revolutionized by a clustering technique introduced by Appel [Appel:85a] and developed significantly by Barnes, Hut and Greengard [Barnes:86a], [Greengard:87a]. The basic idea is simple; consider a cluster of M stars for which we need to calculate the interaction with a single star (far) outside the cluster. This straightforwardly requires $O(M)$ steps, but can gain a factor of M by ignoring the details of the cluster and just computing with its center of mass. As implemented by Barnes and Hut, one can apply this idea recursively generating a tree (quad tree in two dimensions) as illustrated in Figure 11 with, at most, one particle in the cluster at the lowest level of tree [Warren:88b], [Warren:88c]. The naive calculation takes a time for each simulation (time) step for a system of N particles

$$T_{\text{step}}^{\text{naive}} = \frac{1}{2} N^2 t_{\text{2particle}} \tag{10}$$

while explicit implementation, shows that the clustering method takes time

$$T_{\text{step}}^{\text{cluster}} = (20 - 50) N \log_2 N t_{\text{2particle}} \tag{11}$$

The cluster method has superior performance for $N \gtrsim 1000$ particles. The current limit of O(10,000) particles for the $O(N^2)$ algorithm is increased by an order of magnitude for the Barnes Hut method. The possibility of large N of $O(10^6)$ particles opens up several important astrophysical calculations including

- Study of the growth of fluctuations in the early universe
- Dynamics of globular clusters where one finds in nature of $O(10^6)$ stars and a difficult calculation as very short range interactions (binary stars) are critical

Table 7: Steps in Parallelizing OGCM

	Time in Days
Get Original FORTRAN Running on CRAY and Understand use of Program	10
Generate Working Sequential Code for SUN	10
Construct Test Dataset	10
Find that our FORTRAN environment on NCUBE hypercube needed upgrade as up to now we had used C language	10
Parallelize Code	3
Total	43

- Galaxy structure and the collision of galaxies.

We have just finished a calculation of the last type, illustrated in Figures 11 and 12, which used about 200 hours on the 512 node NCUBE [Salmon:89a].

This computation has several interesting features. The cluster tree is rebuilt each time step; a stage which is negligible in the sequential version, but which appears to take of order 30% of the concurrent execution time in the $N = 180,000$ particle simulation of [Salmon:87a]. Initially, we found load imbalance, but this was solved by dynamically redistributing particles at each time step. The information for this was found from the "workload" at the previously calculated time step. As illustrated in Figure 11, orthogonal recursive bisection is used to distribute the particles. Communication is required to fetch those parts of the tree that are stored outside the node and will be needed for updating particles within the node. This ensures maximum re-use of the communicated data and low communication overhead — about 10%. This approach does, however, use 75% of the available NCUBE memory and limit the simulation size. In spite of this, we are able to consider, on the current NCUBE, large problems that are difficult to implement on the limited memory CRAY-XMP and future such hypercubes with several megabytes of memory per node will allow much larger values of N.

The program was implemented with 3,400 lines of C code for the NCUBE, and some timing information is given in Table 8 for a single time step. This has a comparison with FORTRAN CRAY code.

We see that the 256 node NCUBE outperforms the CRAY-XMP as reported in [Hernquist:87a] for 10^5 particles, even though the hypercube efficiency is quite low. The CRAY efficiency is even lower!

In Figure 13, we show results from [Messina:89a] for what is essentially the $O(N^2)$ particle dynamics algorithm. This vectorizes well on CRAY and runs with $> 95\%$ efficiency on the NCUBE; this time the CRAY-XMP is four times the performance of the 256 node NCUBE.

Considering SIMD architectures, we are not certain how to implement the clustering algorithm to get good performance on machines like the CM-2 [Hillis:87b]. This is related to the difficulties we saw in Sec. 2.1 with Monte Carlo clustering and in Sec. 2.4 with adaptive grids on SIMD machines. Note that the "regular" clustering/multiscale algorithms such as multigrid or the FFT run quite well on SIMD machines; the difficulties in implementations for synchronous machines occurs for geometrically irregular cluster algorithms.

Table 8: Performance on Astrophysical Particle Dynamics

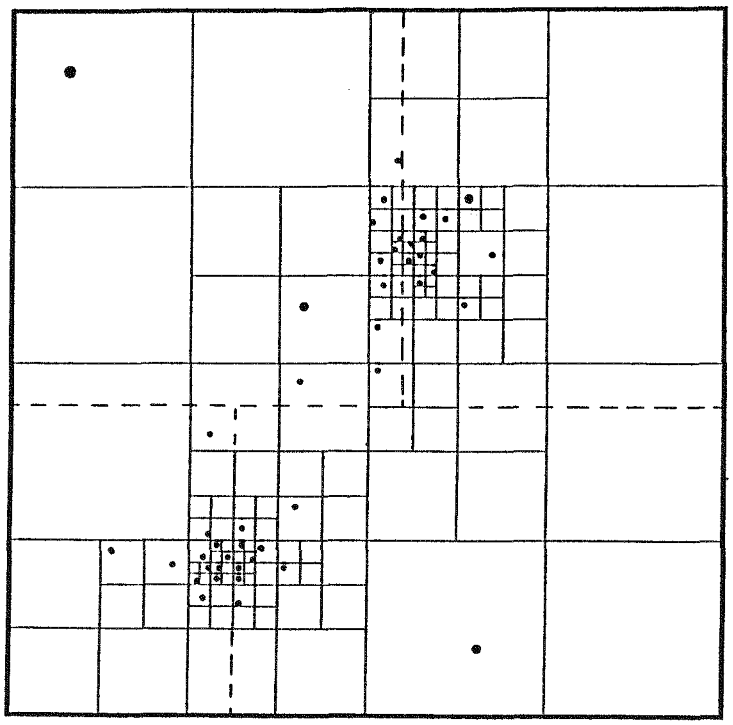
	10^4 Particles	10^5 Particles
CRAY-XMP Some Optimization	10 secs	130 secs
256 node NCUBE		
— time	21 secs	118 secs
— efficiency	24%	56%

2.11 Computer Chess (Felten, Otto, Morison)

Computer chess involves constructing a tree of possible moves and dynamically pruning it with the $\alpha - \beta$ technique as illustrated in Figure 14 [Felten:88g], [Felten:88h], [Felten:88i]. On a parallel machine, the tree ("data domain") is decomposed over the nodes. We found that a real time graphics display (using the NCUBE parallel graphics subsystem) was critical in achieving a factor of five better performance. This allowed us to change the algorithm for processor assignment and improve the load balance. The measured speed up is shown in Figure 15 with a speed up of 101(170) seen on 256(512) nodes for trees of depth appropriate for a middle game. Note that the speed up increases as the problem gets bigger, *i.e.,* as one spends a longer time on each move. This illustrates that for problems with real time constraints, increasing processor performances increases parallelization efficiency by increasing the size of the problem that can be solved in a given fixed time.

This was a very difficult code to design and develop as

Hierarchical Decomposition
and Domain Decomposition



Solid Lines - Hierarchical boxing, presented for simplicity in two dimensions. A system of particle is shown, and the recursive subdivision of space induced by these particles

Dashed Lines - A typical decomposition onto a 4 node hypercube by orthogonal recursive bisection

Figure 11: A collection of interacting particles in two dimensions with the hierarchical quadtree and its load balanced decomposition onto four nodes [Warren:88b], [Warren:88c].

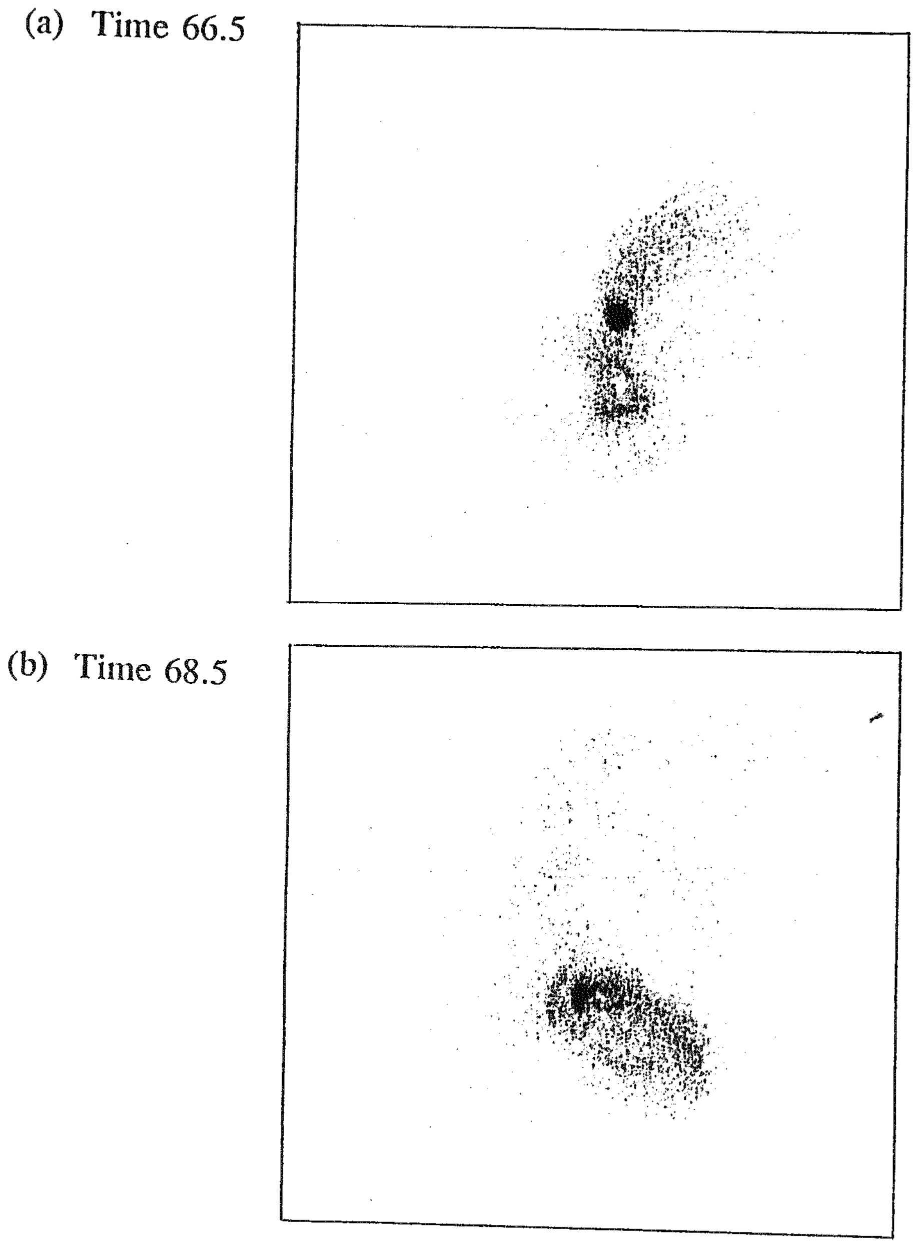


Figure 12: Typical results of the simulation on the NCUBE of the collision between two galaxies with a total of 180,000 objects [Salmon:89a]. Each "object" should be viewed as a collection of stars.

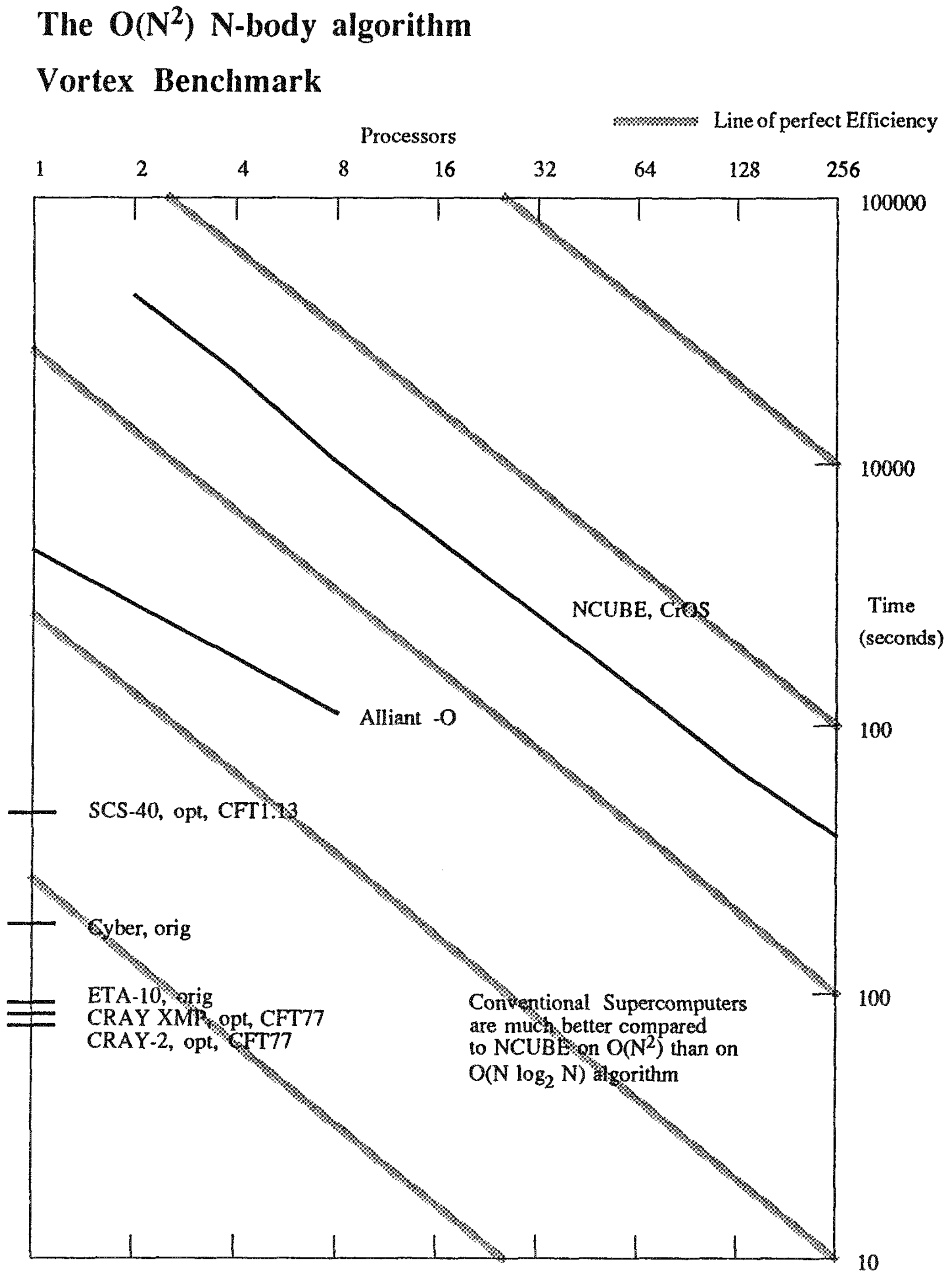


Figure 13: Performance of an $O(N^2)$ vortex dynamics algorithm on a variety of high performance computers [Messina:89a].

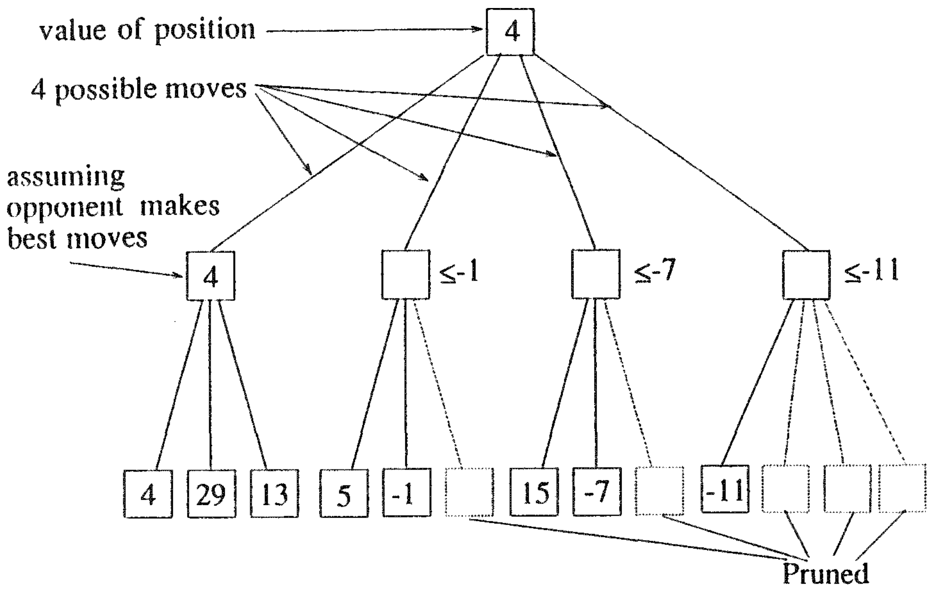


Figure 14: A schematic for a chess tree showing branches pruned away [Felten:88g].

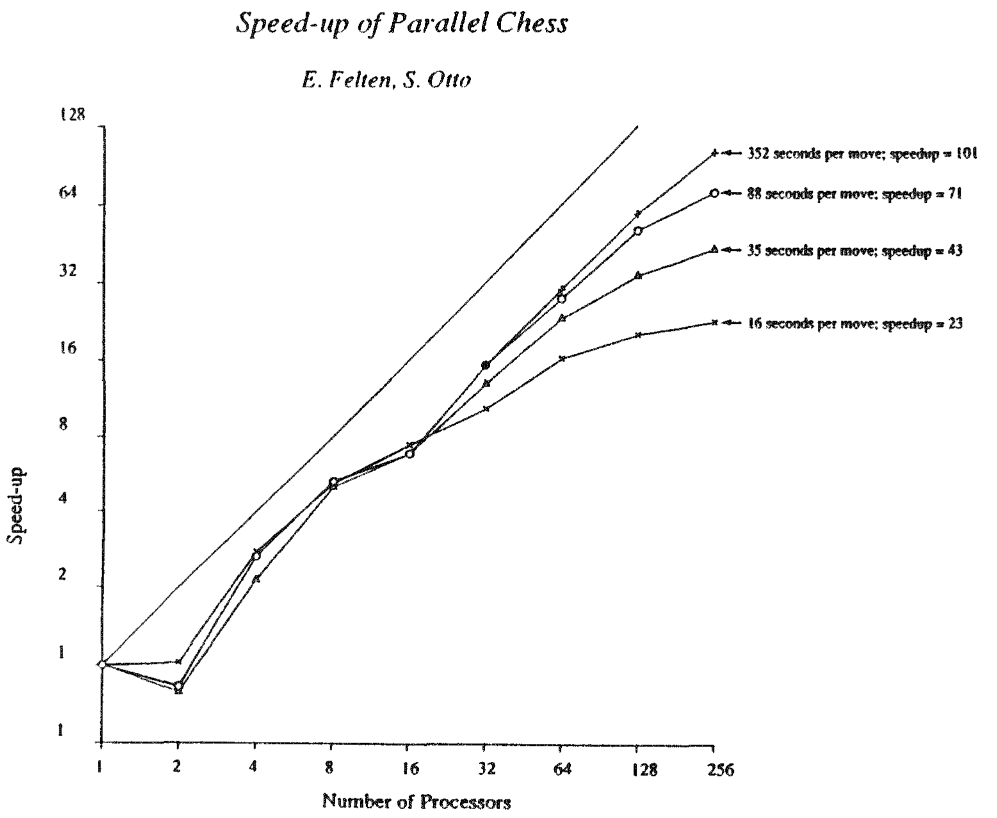


Figure 15: Measured speed up for the parallel chess program as a function of number of NCUBE nodes. The four curves correspond to four tree depths with larger trees showing larger speed ups [Felten:88g], [Felten:88i].

the algorithm is *asynchronous*. Probably, the technical implementation and in particular the debugging were harder than the algorithmic issues. The result is 8,000 lines of C code using the commercial NCUBE operating system VERTEX with a "special shared memory" enhancement to allow concurrent access and update of a distributed database — the so-called transposition table of currently evaluated positions. In contrast, all the other Caltech hypercube scientific calculations use our internal *loosely synchronous* communications system CrOS which is faster than VERTEX [Fox:88a].

Let us consider the future of computer chess. We estimate that the 256 node NCUBE has a U. S. chess rating of 2,100 at present, or in a more familiar unit of megamoves searched per second, we have the results in Table 9a.

The NCUBE does quite well compared to a CRAY (a 512 node NCUBE is approximately one head of a CRAY XMP), but neither is competitive with special purpose machines. However, we can consider using the same parallelization technique developed for the NCUBE, but for an array of special purpose chess chips — not the general purpose microprocessor used on the NCUBE. We conjecture that a system of some 8,000 special purpose chips like those in Deep Thought, would achieve a speed up of 1,000 and be very competitive with Kasparov as shown in Table 9b. Each node processor of our world chess champion would have for chess about 100 times the power of a NCUBE node. The communication overhead for the NCUBE case is about 10%; the "Deep Thought" chips would need an internode bandwidth about ten times that of NCUBE to keep a manageable (50%) communication overhead. Such a system seems quite practical, but outside my group's resources. Rather, we are concentrating on a different approach; can one improve chess programs by the use of neural network based position evaluators?

2.12 Kalman Filters (Gottschalk)

A JPL team headed by D. Curkendall [Meier:89a] has developed a sophisticated battle management simulation. This includes threat generation (launch missiles), tracking, engagement planning (launch anti-missiles), and graphics. The total of 200,000 lines of code is one of the largest single parallel computer projects — much larger than the sum of codes described so far in the previous eleven subsections! The current, so called, SIM88 project completely simulates up to 250 objects launched from six sites. There is an interesting hybrid approach to the simulation with each component (tracking satellite, planning platform) functionally decomposed with a very coarse grained object oriented model. Traditional data parallelism is used within functions (objects) assigned to a subcube of the hypercube. Correspondingly, a hybrid software model CENTAUR for the Mark IIIfp hypercube supported general but slow inter object communication and fast loosely synchronous CrOS communication within objects [Burns:88a]. Formally, the simulation is an *asynchronous* event driven simulation at the object level, but the predictable and coarse grain nature of the object to object communication allowed efficient implementation within a simple conservative framework [Chandy:81a].

Table 10: Components (Time Complexity in Arbitrary Units) of Old and New Trackers

Tracker Version	Calculation	Overhead	Lines of C Code
SIM 87	1	1	4,000
SIM 88	1	3	20,000

Let us focus on the tracking component which used a parallel implementation of a traditional multi-target Kalman filter tracker [Gottschalk:87f], [Gottschalk:88a]. Much of the calculation can be done independently in each node when one distributes the tracks. There are some significant overheads when tracks share measurements and load balance is an issue addressed by dynamically redistributing the data at each measurement cycle.

In Figure 16, we show the performance comparison where shared memory machines do very well — these avoid the data shuffling overheads for overlapping tracks. This algorithm was designed for the so called boost phase when there is modest parallelism coming from the decomposition of a total of a few hundred objects. In Figure 16, with a total of 480 targets, the coarse grain machines with smaller number of nodes are clearly preferable to the NCUBE with many slower nodes.

In Table 10, we indicate that the communication overhead is perhaps overestimated in Figure 16 as it depends critically on the sophistication of the track model. Improving this increases the fully parallel calculation without changing the absolute values of overheads. Thus, the newer SIM 88 algorithm has higher efficiency than the original SIM 87 implementation of Figure 16.

We expect the situation to change in the harder mid-course phase where up to 10^5 real or decoy objects can be anticipated. This will need new algorithms — perhaps neural networks [Fox:89h] — and the parallelism issues will be quite different. In the boost phase, it appears that existing parallel machines with a few megaflops of performance will allow real time tracking; mid course will require true supercomputers.

3 Lessons

Here we collect together some lessons we have drawn from "using real parallel computers to solve real problems with real software".

This information is obtained from both the supercomputer applications of Sec. 2 and for instance the broader surveys of [Fox:88b], [Fox:88l].

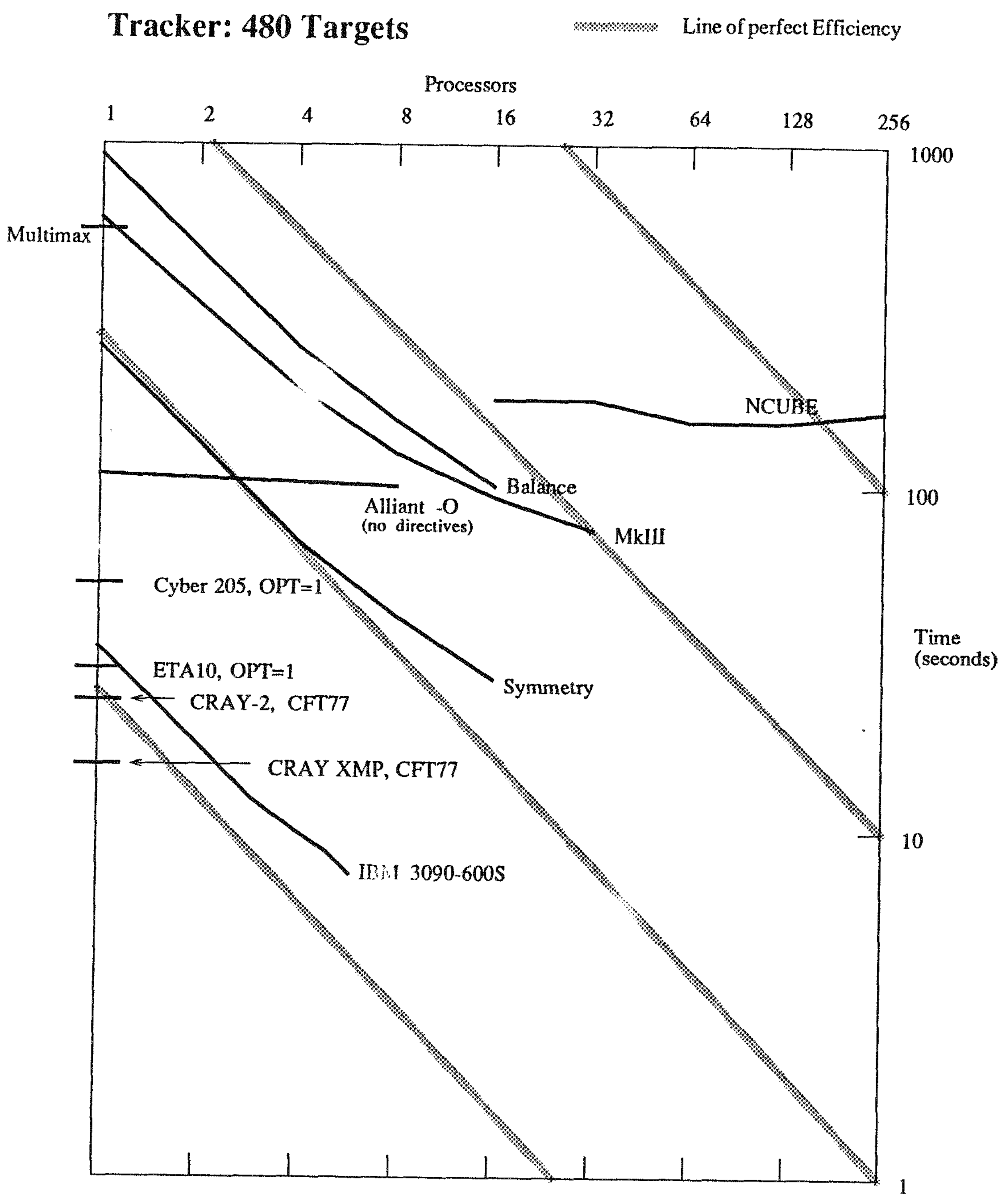


Figure 16: Performance evaluation results for the so called SIM87 tracker on 480 objects and 10 sites from [Messina:89a]. We show two versions of the NCUBE timings using either VERTEX (marked "slow CrOS") on an optimized high performance crystalline communication environment marked "fast CrOS" [Baillie:88f], [Baillie:87c]. This also uses a faster algorithm with better load balancing.

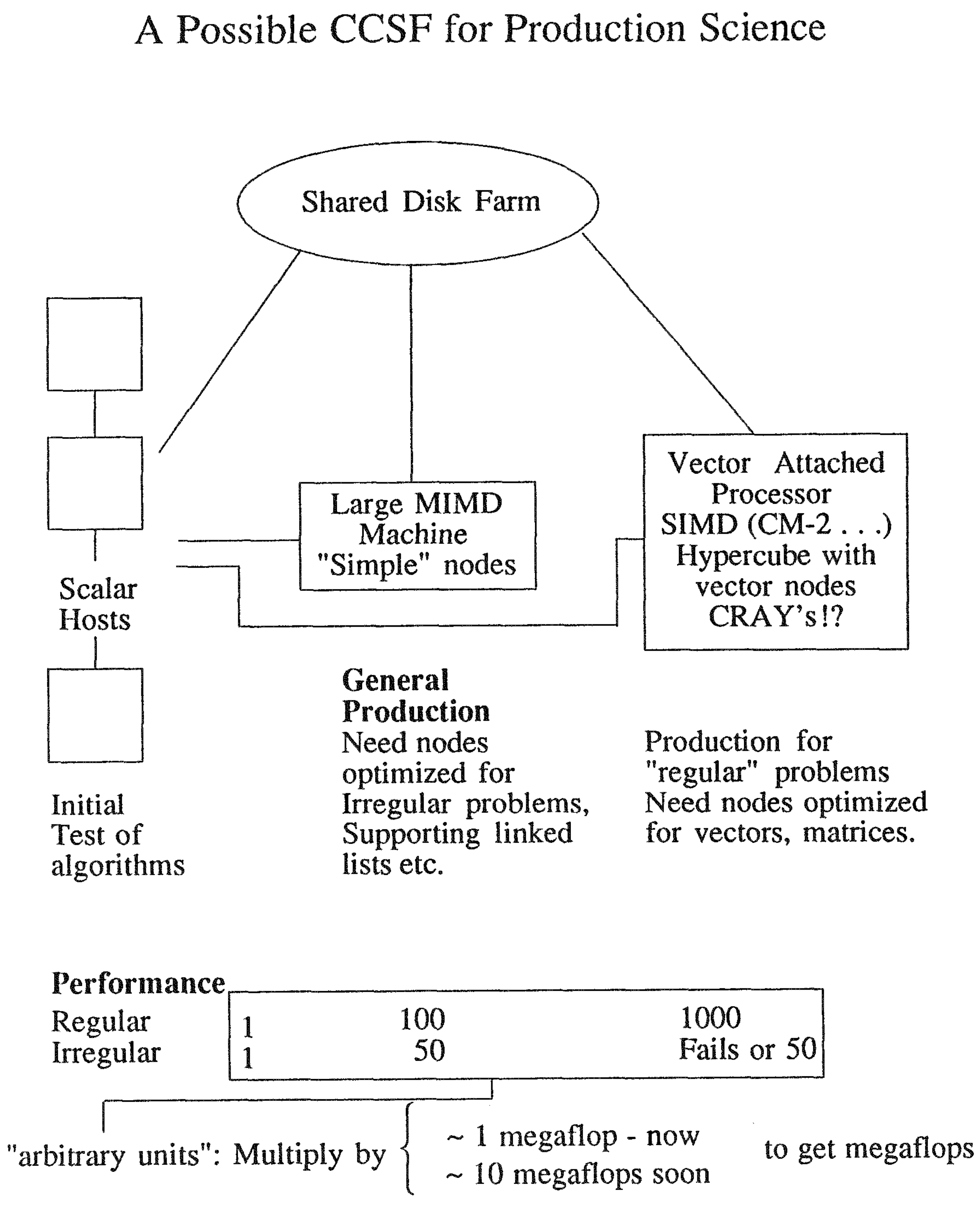

A Possible CCSF for Production Science
Shared Disk Farm
Scalar Hosts
Large MIMD Machine "Simple" nodes
Vector Attached Processor SIMD (CM-2 . . .) Hypercube with vector nodes CRAY's!?
Initial Test of algorithms
General Production
Need nodes optimized for Irregular problems, Supporting linked lists etc.
Production for "regular" problems Need nodes optimized for vectors, matrices.
Performance
Regular 1 100 1000
Irregular 1 50 Fails or 50
"arbitrary units": Multiply by
~ 1 megaflop - now
~ 10 megaflops soon
to get megaflops

Figure 17: A possible design for a computer "center" built around advanced architecture high performance systems.

Table 9: (a) Chess Positions Searched per Second

Machine	Performance (10^6 moves/sec)
Special Purpose	
BELLE (1980)	0.075
HITECH (1985)	0.25
Deep Thought (1988)	2.5
New AT and T Machine (1989?)	5
General Purpose	
256 node NCUBE	0.025
4 Processor CRAY XMP	0.15

(b) Ratings of the Best Chess Players

Player	Rating
HITECH	2450
Deep Thought	2600
Kasparov	2850
1000×Deep Thought	2850?

3.1 Application Lessons

3.1.1 One can achieve high performance on essentially all scientific computations which are

- Large (necessary condition)
- *Loosely synchronous* — MIMD (sufficient condition)
- or *synchronous* — SIMD

More research is needed to clarify Table 1 and see how far one can violate synchronization and get good results for irregular problems on SIMD machines. Note that even for loosely synchronous problems, synchronous communication is usually sufficient; communication is typically only necessary at the macroscopic synchronization points of the algorithm [Fox:88a], [Fox:88b], [Fox:88ll].

3.1.2 Domain Decomposition or data parallelism is a universal source of parallelism that scales to large numbers of nodes [Fox:88a], [Hillis:86a].

3.1.3 These results are true on a broad range of computer architectures (SIMD, MIMD, shared, distributed memory, hypercubes, (transputer) meshes ...).

3.1.4 University successes on parallel computers have come with 1,000–10,000 line codes written from scratch for a particular machine.

3.1.5 It is not clear how to extrapolate these successes to up to 10^6 line commercial codes where you have perhaps less knowledge as to the inner workings of program *cf.* Sec. 2.9. Certainly, one must establish and use standard methods in parallel software to justify expense of a new parallel implementation. These standards must apply across a range of architectures.

3.1.6 In many cases (*cf.* Sec. 2.2), it is easier to decompose for a parallel machine than to vectorize for a conventional supercomputer. This is especially true for small (university) codes.

We can superficially abstract from this that universities should purchase parallel computers and industry vector supercomputers!

3.1.7 The importance of parallel machines for artificial intelligence (AI) is unclear to me. If the AI is implemented with neural networks, then the relevance and use of parallel computers is clear. More traditional AI systems parallelize less easily although chess (Sec. 2.11) is a good example which both parallelizes and needs high performance. How many other such AI applications are there?

3.2 Performance Lessons

3.2.1 One can get high performance on essentially all scientific computations. As shown in [Fox:84c], [Fox:85c], [Fox:88a], [Gustafson:88a], and [Fox:89b]

- Performance scales linearly in number of nodes at constant grain size (problem size proportional to machine size)
- Fixed problem size does not scale; this can be viewed as Amdahl's Law

3.2.2 Some initial disappointments can be traced to imbalance in early commercial machines, such as the iPSC/1 and FPS T Series hypercubes.

3.2.3 We saw in Secs. 2.2, 2.3 and Sec. 2.10 that machines like an NCUBE or transputer arrays look particularly attractive compared to the CRAY XMP class machines on irregular problems where one finds

- it is more natural and easier to decompose than to vectorize
- the NCUBE efficiency is "low"; maybe 50%
- but the CRAY efficiency even lower; maybe 5%.

Note that the average CRAY-XMP performance in computer center operation is about 25 megaflops with a 12% efficiency. The NCAR CRAY realizes a sustained 50 megaflops, which is perhaps the peak average performance. Perhaps we have too high a standard for the efficiency of parallel machines!

3.2.4 Scalar node MIMD machines are natural general purpose machines with reasonable performance over a range of problems. As seen in Table 6, hypercubes have high efficiency on regular (e.g., full matrix and QCD) problems, but so does the CRAY XMP class machine. Hypercubes with vector nodes or SIMD machines are attractive for regular problems.

3.2.5 Different programming methodologies and lack of standards handicap performance studies. As FTN-8X is yet to be implemented uniformly, it is hard even to port between CRAY XMP and the ETA-10. This, for instance, is handicapping the "grand challenge" that I mentioned in Sec. 2.1. Of course, porting between parallel and sequential machines is hard and [Messina:89a] essentially reimplemented from scratch many algorithms for their performance evalution. The different software methodologies for shared and distributed memory machines cannot be avoided as some sense it is the more convenient environment that motivates shared memory machines. It would be unreasonable to require shared memory machines to always use message passing!

3.3 Decomposition Lessons

Here we refer to issues concerning the decomposition or dividing up of problems to minimize communication and equalize load on processors.

3.3.1 Three years ago, I thought decomposition or load balancing was a key problem, but as described in Sec 2.4, it is surprisingly easy!

- Usually, the application scientist can specify it from the natural geometric structure of the problem
- Several heuristic methods provide automatic decomposition. These include recursive bisection, simulated annealing and neural networks.

3.3.2 Current hardware trends have emphasized transparent message routing where the user need not be aware of machine topology. This is clearly convenient, but Sec 3.3.1 indicates it is not strictly necessary for a broad class of problems. We can note that:

- Most problems can be mapped with software to any reasonable bandwidth topology with modest routing overheads.
- In particular, for current Caltech codes (say those described in Sec. 2) there is an average overhead of less than 5% due to routing. This should be compared to overheads of perhaps 50% due to poor compilers (e.g., for NCUBE and WEITEK), 50% as system is overall rather flaky and 25% due to node to neighboring hypercube node communication.
- We must emphasize that these software solutions have yet to be "packaged" nicely for general use. Not every programmer is comfortable with simulated annealing. Thus, automatic routing hardware is certainly convenient in the "real world".

3.3.3 In current systems, message start up time, which includes hardware and software effects is a much more serious overhead than either node to node through routing, or channel transmission between neighboring nodes.

3.4 Hardware Lessons

3.4.1 So far, high performance computations on moderately or massively parallel machines ($\gtrsim 8$ nodes) has been confined to distributed memory machines. The comparison between distributed and shared memory architecture is hard because of the lack of comparable machines and experience.

3.4.2 Five years ago, there were many university projects building novel machines, but in the future commercial systems will dominate the parallel field as they now do with conventional supercomputers.

3.4.3 The U. S. entrepreneurial environment will guarantee a wide range of architectures even without new university projects. Portable software will be at a premium to exploit these machines.

3.4.4 Many or perhaps all the current commercial parallel systems are disappointing in some ways. For instance, our NCUBE is now in full use as a "production supercomputer" but this took two years and at least $150K costs in software development at Caltech. If we remember Bill Joy's law that sequential computers improve a factor of two per year in cost-performance, we see that a two-year delay translates into a factor of four loss in cost effectiveness compared to the conventional competition. Novel computers are bound to need extra time to develop viable software than their sequential competition; this certainly handicaps their ability to compete.

3.4.5 As implied by Sec. 3.4.4, systems integration is not yet well addressed in the parallel machines. This includes issues such as,

- General multiuser operating systems especially for distributed memory machines; debugging
- Adherence to standards

- Input/Output for disks and graphics. The architecture of the I/O system gets surprisingly little attention in the literature.
- High performance appropriate hosts (not PC's or workstations) for a parallel supercomputer

3.4.6 For MIMD machines, we have already discussed some issues in Secs. 3.1.1, 3.2.3, and 3.2.4.

- Currently, machines like the NCUBE or transputer with scalar (floating point) nodes seem more successful than machines with vector nodes. The poor old user finds it hard to vectorize and decompose problems! One such optimization is enough.
- Given that such scalar node systems are particularly attractive for irregular problems, maybe one should consider adding specialized support for the data structures like linked lists needed for irregular problems.
- Several megabytes (but not arbitrarily large) memory per node is needed to hold program, decomposed data, databases and reused communication data (see Sec. 2.10).
- One can expect the differences between shared and distributed memory architectures to lessen as both are based on low latency networks. Either local memory (for machines like the hypercube) or caches on shared memory machines will require data locality for good performance.

3.4.7 SIMD machines can support at least 50% of university scientific applications as shown in Table 1. They currently give the peak performance for regular problems (see Tables 4, 6). Perhaps the commercial applications are more irregular and will show a lower fraction appropriate for the SIMD architecture.
3.4.8 In Figure 17, we show a possible structure of an integrated high performance novel architecture computer environment. Simple scalar node MIMD machines support general problems with either vector or SIMD architectures as an "accelerator" for regular problems.

3.5 Software Lessons

3.5.1 Whereas the role of universities in developing hardware systems may be limited in the future, we expect universities to have a critical role in the software for parallel machines where we cannot hope for the commercial systems to be adequate.

3.5.2 A key question is: "What is the appropriate productive standard programming environment for parallel machines?" This could be based on [Fox:88u], [Fox:88w]:

- New languages
- Compiler generated parallelism
- Application specific high level environments
- Explicit user decomposition

3.5.3 Note that essentially all successful reasonable performance use of parallel machines have used explicit user decomposition which is low level and machine dependent. We expect we must find more portable attractive methods if parallel computers are to take over from the conventional architectures.

3.5.4 Approaches like LINDA ("shared message space") or the new language, OCCAM, appear not to address enough of the issues to be the solution of the question posed in Sec. 3.5.2.

3.5.5 When I started work on the hypercube in 1981, I had great confidence that they would be successful as I could make a simple performance model and prove "mathematically" that for dedicated users, hypercubes would work [Fox:83a]. I don't have any way of making a similar prediction for the software environment!

3.5.6 The development of parallel computing has involved collaborations between several academic fields — in particular computer scientists, applied mathematicians and application scientists such as computational physicists and chemists. I see large gulfs have developed between these fields with, for instance, no agreed common language and little global understanding or sympathy of the issues and goals for each academic field. I suggest that universities need to reach out to support interdisciplinary research and education. In particular, we need to fill the gap designated as computational science in Figure 18 [Rheinboldt:85a], [Raveche:87a].

Acknowledgements

I would like to thank Paul Messina and the many others whose research is reviewed in this paper.

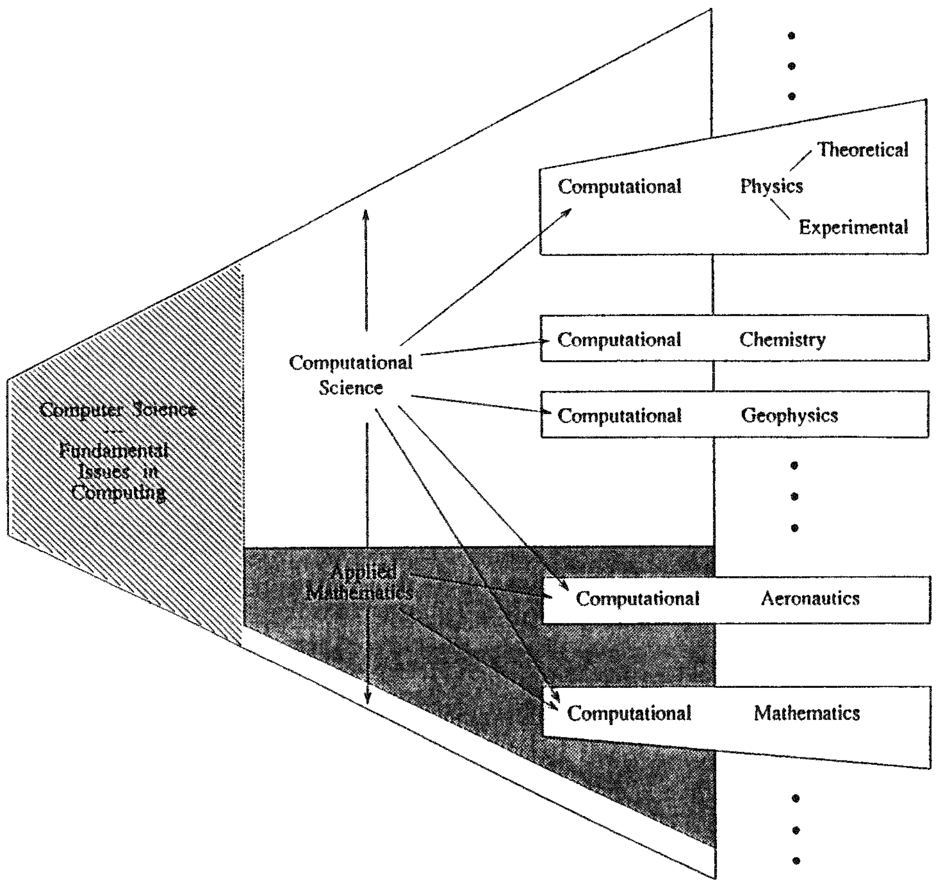


Figure 18: The gaps between traditional disciplines; Computer Science, Applied Mathematics, Physics, Chemistry, ...; and the role of Computational Science in filling this gap.

References

[Appel:85a] Appel, A. W. "An efficient program for many-body simulation," *Sci. Stat. Comput.*, 6:85, 1985.

[Baden:87a] Baden, S. B. *Run-Time Partitioning of Scientific Continuum Calculations Running on Multiprocessors.* PhD thesis, University of California, Berkeley, 1987.

[Baillie:89b] Baillie, C. F., and Fox, G. C. "Parallel computing comes of age: Supercomputer calculations for lattice QCD and spin models on advanced architecture computers." Technical Report C3P-711, California Institute of Technology, January 1989. Presented at European Symposium on High Performance Computing, Montpellier, France 22-24 March, 1989.

[Baillie:89c] Baillie, Clive, F. "Lattice spin models and new algorithms." Technical Report C3P-777, California Institute of Technology, April 1989.

[Baillie:89e] Baillie, C. F., Brickner, R. G., Gupta, R., and Johnsson, L. "QCD with dynamical fermions on the Connection Machine." Technical Report C3P-786, California Institute of Technology, May 1989. To be presented at the Supercomputing 89 in Reno, Nevada.

[Barhen:88a] Barhen, J., Gulati, S., and Iyengar, S. S. "The pebble crunching model for load balancing in concurrent hypercube ensembles," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 189–199. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-610.

[Barnes:86a] Barnes, J., and Hut, P. "A hierarchical $O(N \log N)$ force calculation algorithm," *Nature*, 324:446, 1986.

[Barnes:88a] Barnes, T. "Numerical solution of high temperature superconductor spin systems." Technical Report C3P-638, California Institute of Technology, June 1988. To appear in Proceedings of the Oak Ridge Meeting on Computational Atomic and Nuclear Physics at One Gigaflop, Toronto preprint UTPT-88-07.

[Barnes:88b] Barnes, T., Kotchan, D., and Swanson, E. S. "Evidence for a phase transition in the zero temperature anistrophic 2D Heisenberg antiferromagnet." Technical Report C3P-653, California Institute of Technology, July 1988. To appear in Phys. Rev. B.; Toronto preprint UTPT-88-09.

[Barnes:89a] Barnes, T., Cappon, K. J., Dagotto, E., Kotchan, D., and Swanson, E. S. "Critical behavior of the 2D anisotropic Heisenberg antiferromagnet: A numerical test of spin-wave theory." Technical Report C3P-722, University of Toronto, March 1989. UTPT-89-01.

[Berry:88a] Berry, M., Chen, D., Koss, P., Kuck, D., Lo, S., Pang, Y., Roloff, R., Sameh, A., Clementi, E., Chin, S., Schneider, D., Fox, G., Messina, P., Walker, D., Hsiung, C., Schwarzmeier, J., Lue, K., Orszag, S., Seidl, F., Johnson, O., Swanson, G., Goodrum, R., and Martin, J. "The perfect clube benchmarks: Effective performance evaluation of supercomputers." Technical Report CSRD Rpt. No. 827, The Performance Evaluation Club (Perfect), November 1988.

[Brickner:89a] Brickner, R., and Baillie, C. "Pure gauge QCD on the Connection Machine." Technical Report C3P-710, California Institute of Technology, February 1989.

[Burns:88a] Burns, P., Crichton, J., Curkendall, D., Eng, B., Goodhart, C., Lee, R., Livingston, R., Peterson, J., Pniel, M., Tuazon, J., and Zimmerman, B. "The JPL/Caltech Mark IIIfp hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 872–884. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-607.

[Callahan:88a] Callahan, S. "Non-local path integral Monte Carlo on the hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1296–1302. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-589.

[Callahan:88b] Callahan, S. *Exchange Interactions in Solid ^{3}He on a Parallel Computer.* PhD thesis, California Institute of Technology, August 1988. Caltech Report C3P-645.

[Chandy:81a] Chandy, K. M., and Misra, J. "Asynchronous distributed simulation via a sequence of parallel computations," *Comm. ACM*, 24:1981, 1981.

[Chen:88a] Chen, W. K., and Gehringer, E. F. "A graph-oriented mapping strategy for a hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 200–209. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988.

[Chiu:86a] Chiu, T. W., "Fermion spectrum on the random block lattice." Contributed paper to XXIII International Conference on High Energy Physics, Berkeley, 1986.

[Chiu:88e] Chiu, T. W. "Vacuum polarization on 4-d random block lattice." Technical Report C3P-693, California Institute of Technology, 1988.

[Cuccaro:88a] Cuccaro, S. A., Hipes, P. G., and Kuppermann, A. "Hyper-spherical coordinate reactive scattering using variational surface functions." Technical Report C3P-720, California Institute of Technology, 1988. Accepted for publication in Chem. Phys. Letters.

[Decyk:88a] Decyk, V. K. "Supercomputer," 27:33, 1988.

[Ding:89a] Ding, H. Q., Baillie, C. F., and Fox, G. C. "Heavy quark potential at large distances." Technical Report C3P-779, California Institute of Technology, April 1989.

[Ercal:88a] Ercal, F., Ramanujam, J., and Sadayappan, P. "Task allocation onto a hypercube by recursive mincut bipartitioning," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 210–221. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988.

[Felten:88g] Felten, E. W., and Otto, S. W. "Chess on a hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1329–1341. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-579.

[Felten:88h] Felten, E. W., and Otto, S. W. "Chess on a hypercube." Technical Report C3P-579b, California Institute of Technology, 1988. In Proceedings of IEE Symposium on The Design and Application of Parallel Digital Processors, April 1988, Lisbon, 30–42.

[Felten:88i] Felten, E. W., and Otto, S. W. "A highly parallel chess program," in *Proceedings of International Conference on Fifth Generation Computer Systems 1988*, pages 1001–1009. ICOT, November 1988. Tokyo, Japan, November 28 – December 2. Caltech Report C3P-579c.

[Fox:83a] Fox, G. C. "Decomposition of scientific problems for concurrent processors." Technical Report C3P-028, California Institute of Technology, 1983. CALT-68-986.

[Fox:84c] Fox, G. "On the sequential component of computation." Technical Report C3P-130, California Institute of Technology, 1984.

[Fox:85c] Fox, G. "The performance of the Caltech hypercube in scientific calculations: A preliminary analysis," in F. A. Matsen and T. Tajima, editors, *SuperComputers — Algorithms, Architectures, and Scientific Computation*. University of Texas Press, 1987. Caltech Report C3P-161.

[Fox:88a] Fox, G. C., Johnson, M. A., Lyzenga, G. A., Otto, S. W., Salmon, J. K., and Walker, D. W. *Solving Problems on Concurrent Processors*, volume 1. Prentice-Hall, Inc., Englewood Cliffs, NJ 07632, 1988.

[Fox:88b] Fox, G. C. "What have we learnt from using real parallel machines to solve real problems?," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 897–955. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-522.

[Fox:88c] Fox, G. C., editor. *The Third Conference on Hypercube Concurrent Computers and Applications*. Jet Propulsion Laboratory of the California Institute of Technology, ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Volume 1 - Architecture, Software, Computer Systems and General Issues; Volume 2 - Applications.

[Fox:88e] Fox, G. C., and Furmanski, W. "Load balancing loosely synchronous problems with a neural network," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 241–278. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-363b.

[Fox:88f] Fox, G. C., and Furmanski, W. "A string theory for time dependent complex systems and its application to automatic decomposition," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 285–305. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-521.

[Fox:88g] Fox, G. C., and Furmanski, W. "Hypercube algorithms for neural network simulation the Crystal_Accumulator and the Crystal_Router," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 714–724. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-405b.

[Fox:88u] Fox, G. C. "Issues in software development for concurrent computers," in G. J. Knafl, editor, *Proceedings of The Twelfth Annual International Computer Software and Applications Conference*, pages 302–305. Computer Society Press of the IEEE, 1730 Massachusetts Avenue, N.W., Washington, D.C. 20036-1903, October 1988. Caltech Report C3P-640.

[Fox:88v] Fox, G. C., and Messina, P. "Report for 1988 on the Caltech Concurrent Computation Program." Annual Report C3P-685, California Institute of Technology, December 1988.

[Fox:88w] Fox, G. C. "Theory and practice of concurrent systems," in *Proceedings of International Conference on Fifth Generation Computer Systems 1988*, pages 157–160. ICOT, November 1988. Caltech Report C3P-664.

[Fox:88ll] Fox, G. C., and Frey, A. "High performance parallel supercomputing application, hardware, and software issues for a teraflop computer." Technical Report C3P-451c, California Institute of Technology, December 1988.

[Fox:88nn] Fox, G. C. "A graphical approach to load balancing and sparse matrix vector multiplication on the hypercube," in M. Schultz, editor, *Numerical Algorithms for Modern Parallel Computer Architectures*, pages 37–62. Springer-Verlag, 1988. Caltech Report C3P-327b.

[Fox:89b] Fox, G. C. "Experience on the hypercube." Technical Report C3P-716, California Institute of Technology, February 1989.

[Fox:89h] Fox, G. C. "A note on neural networking for trackfinding." Technical Report C3P-748, California Institute of Technology, March 1989.

[Frisch:86a] Frisch, U., Hasslacher, B., and Pomeau, Y. "Lattice-gas automata for the Navier-Stokes equations," *Phys. Rev. Lett.*, 56:1505–1508, 1986.

[Furmanski:87a] Furmanski, W., Bower, J. M., Nelson, M. E., Wilson, M. A., and Fox, G. "Piriform (Olfactory) cortex model on the hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 977–999. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-404b.

[Gottlieb:86a] Gottlieb, A. "An overview of the NYU ultracomputer project," in J. J. Dongarra, editor, *Experimental Parallel Computing Architectures*. North-Holland, Amsterdam, 1987.

[Gottschalk:87f] Gottschalk, T. D. "Multiple track initiation on a hypercube," in *Proceedings of the Second International Conference on Supercomputing*, St. Petersburg, Florida, May 1987. International Supercomputing Institute Inc. Caltech Report C3P-398.

[Gottschalk:88a] Gottschalk, T. D. "Concurrent multiple target tracking," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1247–1268. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-567.

[Greengard:87a] Greengard, L. "The rapid evaluation of potential fields in particle systems." Yale research report YALEU/DCS/RR-533, Yale University, April 1987.

[Gupta:88a] Gupta, R., DeLapp, J., Batrouni, G., Fox, G. C., Baillie, C., and Apostolakis, J. "The phase transition in the 2-d XY model," *Phys. Rev. Lett.*, 61:1996, 1988. Caltech Report C3P-643.

[Gurnis:88a] Gurnis, M., Raefsky, A., Lyzenga, G. A., and Hager, B. H. "Finite element solution of thermal convection on a hypercube concurrent computer," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1176–1179. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-595.

[Gustafson:88a] Gustafson, J. L., Montry, G. R., and Benner, R. E. "Development of parallel methods for a 1024-processor hypercube," *SIAM J. Sci. Stat. Comput.*, 9(4):609–638, July 1988.

[Gutt:89a] Gutt, G. M. *The Physics of Granular Systems.* PhD thesis, California Institute of Technology, May 1989. Caltech Report C3P-785.

[Hernquist:87a] Hernquist, L. "Performance characteristics of tree codes," *Astrophysical Journal Supplement*, 64(4):715–734, 1987.

[Hillis:86a] Hillis, D., and Steele, G. "Data parallel algorithms," *Comm. ACM*, 29:1170, 1986.

[Hillis:87b] Hillis, D., and Barnes, J. "Programming a highly parallel computer," *Nature*, 326:27, 1987.

[Hipes:88a] Hipes, P. G., and Kuppermann, A. "Gauss-Jordan inversion with pivoting on the Caltech Mark II hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1621–1634. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-578.

[Hipes:88b] Hipes, P., Mattson, T., Wu, M., and Kuppermann, A. "Chemical reaction dynamics: Integration of coupled sets of ordinary differential equations on the Caltech hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1051–1061. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-570.

[Ito:88a] Ito, N., and Karata, Y. "An effective algorithm for the Monte Carlo simulation of the Ising model on a vector processor," *Supercomputer 25*, 5(3):31–48, May 1988.

[Johnson:73a] Johnson, B. R. "The multi-channel log-derivative method for scattering calculations," *Journal of Computational Physics*, 49:23, 1973.

[Koller:88a] Koller, J. "A dynamic load balancer on the Intel hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 279–284. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-497.

[Kosterlitz:73a] Kosterlitz, J. M., and Thouless, D. J. "Ordering, metastability and phase transitions in two-dimensional systems," *J. Phys. C*, 6:1181, 1973.

[Kuppermann:86a] Kuppermann, A., and Hipes, P. G. "Three-dimensional quantum mechanical reactive scattering using symmetrized hyperspherical coordinates," *J. Chem. Phys.*, 84(10):5962–5964, 1986. Caltech Report C3P-343.

[Liewer:88b] Liewer, P. C., Decyk, V. K., Dawson, J. D., and Fox, G. C. "A universal concurrent algorithm for plasma particle-in-cell simulation codes," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1101–1107. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-562.

[Liewer:88e] Liewer, P. C., and Decyk, V. K. "A general concurrent algorithm for plasma particle-in-cell simulation codes." Technical Report C3P-649b, California Institute of Technology/JPL, 1988. To be published in Journal of Computational Physics.

[Liewer:89a] Liewer, P. C., Zimmerman, B. A., Decyk, V. K., and Dawson, J. M. "Application of hypercube computers to plasma particle-in-cell simulation codes." Technical Report C3P-717, California Institute of Technology, 1989. Paper presented at the Fourth International Conference on Supercomputing, Santa Clara.

[Livingston:88a] Livingston, M., and Stout, Q. F. "Distributing resources in hypercube computers," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 1*, pages 222–231. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988.

[Lyzenga:85a] Lyzenga, G. A., Raefsky, A., and Hager, B. H. "Finite elements and the method of conjugate gradients on a concurrent processor," in *Proceedings of the 1985 ASME International Computers in Engineering*, August 1985. Boston. Caltech Report C3P-164.

[Lyzenga:88a] Lyzenga, G. A., Raefsky, A., and Nour-Omid, B. "Implementing finite element software on hypercube machines," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1755–1761. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-594.

[Meier:89a] Meier, D. L., Cloud, K. C., Horvath, J. C., Allan, L. D., Hammond, W. H., and Maxfield, H. A. "A general framework for complex time-driven simulations on hypercubes." Technical Report C3P-761, California Institute of Technology, March 1989. Paper presented at the Fourth Conference on Hypercubes, Concurrent Computers and Applications.

[Messina:89a] Messina, P., Baillie, C. F., Felten, E. W., Hipes, P. G., Walker, D. W., Williams, R. D., Pfeiffer, W., Alagar, A., Kamrath, A., Leary, R. H., and Rogers, J. "Benchmarking advanced architecture computers." Technical Report C3P-712, California Institute of Technology, February 1989. To be published in Concurrency: Practice and Experience.

[Metropolis:53a] Metropolis, N., Rosenbluth, A. W., Rosenbluth, M. N., Teller, A. H., and Teller, E. "Equation of state calculations by fast computing machines," *J. Chem. Phys.*, 21:1087, 1953.

[Niedermayer:88a] Niedermayer, F. "General cluster updating method for Monte Carlo simulations," *Phys. Rev. Lett.*, 61:2026, 1988.

[Nour-Omid:87b] Nour-Omid, B., Raefsky, A., and Lyzenga, G. "Solving finite element equations on concurrent computers," in *Proceedings of the Symposium on Parallel Computations and Their Impact on Mechanics*. ASME, December 1987. Caltech Report C3P-463.

[Otto:84a] Otto, S. W., and Stack, J. D. "SU(3) heavy-quark potential with high statistics," *Phys. Rev. Lett.*, 52:2328, 1984. Caltech Report C3P-067.

[Patterson:88a] Patterson, J. E., and Zimmerman, B. A. "Hypercube workstations for real-time signal processing and analysis." Proposal C3P-516, California Institute of Technology, January 1988.

[Pfeiffer:88a] Pfeiffer, W., Alagar, A., Kamrath, A., Leary, R., and Rogers, J. "Benchmarking and optimization of scientific codes on the CRAY X-MP, CRAY-2, and SCS-40 vector computers." Technical Report C3P-699, California Institute of Technology and San Diego Supercomputer Center, November 1988. Submitted for publication in The Journal of Supercomputing.

[Raefsky:88a] Raefsky, A., Gurnis, M., Hager, B. H., and Lyzenga, G. A. "Finite element solutions of thermal convection on a hypercube concurrent computer," *EOS, Trans. Amer. Geophy. Union*, 69:463, 1988. Caltech Report C3P-737.

[Raefsky:88b] Raefsky, A., Lyzenga, G. A., Gurnis, M., and Hager, B. H. "Solid earth continuum calculations on a hypercube concurrent computer." Technical Report C3P-738, California Institute of Technology, 1988. To appear in Geophysical Research Letters.

[Raveche:87a] Raveché, H. J., Lawrie, D. H., and Despain, A. M. "A national computing initiative: The agenda for leadership," in *Report of the Panel on Research Issues in Large-Scale Computational Science and Engineering*. SIAM, 1987. SIAM Workshop held at Leesburg, Virginia, February 2–3, 1987.

[Rheinboldt:85a] Rheinboldt, W. C., "A report of the panel on future directions in computational mathematics algorithms and scientific software," 1985.

[Salmon:87a] Salmon, J. "CUBIX: Programming hypercubes without programming hosts," in M. T. Heath, editor, *Hypercube Multiprocessors*, pages 3–9. SIAM, Philadelphia, 1987. Caltech Report C3P-378.

[Salmon:89a] Salmon, J., Quinn, P., and Warren, M. "Using parallel computers for very large N-body simulations: Shell formation using 180K particles." Technical Report C3P-780, California Institute of Technology, April 1989.

[Seiler:88a] Seiler, E., Stamatescu, I. O., Patrascioiu, A., and Linke, V. "Critical-behavior, scaling and universality in some two-dimensional spin models," *Nucl. Phys. B*, 305(4):623–660, 1988.

[Seitz:85a] Seitz, C. L. "The cosmic cube," *Communications of the ACM*, 28:22, 1985.

[Walker:89a] Walker, D. W. "The implementation of a three-dimensional PIC code on a hypercube concurrent processor." Technical Report C3P-739, California Institute of Technology, March 1989. Presented at Fourth Hypercube Conference.

[Warren:88b] Warren, M., and Salmon, J. "An O(N log N) hypercube N-body integrator," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 971–975. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-593.

[Warren:88c] Warren, M. "An $O(N \log N)$ hypercube N-body integrator." Technical Report C3P-639, California Institute of Technology, May 1988. Caltech Undergraduate Senior Thesis.

[Werner:87a] Werner, B. T. *A Physical Model of Wind-Blown Sand Transport.* PhD thesis, California Institute of Technology, 1987. Caltech Report C3P-425.

[Werner:88a] Werner, B. T., and Haff, P. K. "Dynamical simulations of granular materials using the Caltech hypercube," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1313–1318. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-612.

[Williams:88a] Williams, R. D. "DIME: A programming environment for unstructured triangular meshes on a distributed-memory parallel processor," in G. C. Fox, editor, *The Third Conference on Hypercube Concurrent Computers and Applications, Volume 2*, pages 1770–1787. ACM Press, 11 West 42nd Street, New York, NY 10036, January 1988. Caltech Report C3P-502.

[Williams:88d] Williams, R. "Free-lagrange hydrodynamics with a distributed-memory parallel processor," *Parallel Computing*, 7:439–443, 1988. North-Holland. Caltech Report C3P-424b.

[Williams:88e] Williams, R. D. "Supersonic flow in parallel with an unstructured mesh." Technical Report C3P-636, California Institute of Technology, May 1988.

[Williams:89a] Williams, R., and Glowinski, R. "Distributed irregular finite elements." Technical Report C3P-715, California Institute of Technology, February 1989.

[Wolff:89a] Wolff, U. "Collective Monte Carlo updating for spin systems," *Phys. Rev. Lett.*, 62:361, 1989.