Hypercubes are General–Purpose Multiprocessors with High Speedup

contribution for

The Great Race

Marina Chen, Yale
Erik DeBenedictis, Bell Labs*
Geoffrey Fox, Caltech
Jingke Li, Yale
David Walker, Caltech

presented by
Erik DeBenedictis

*Now with Ansoft Corporation.

Outline

- Introduction
- Physics Research
- Programming Technique
- Parallel Compilers

Introduction

- Since we did not have the largest machine, we were not in a good position to demonstrate the highest speedup.

- We believe the acceptability of hypercubes depends on factors other than speedup:

  - Ability to train scientists and engineers to use the technology

  - Potential for general purpose programming techniques

  - Potential for parallel compilers

- Therefore, we attempt to show the *generality* of hypercube programming.

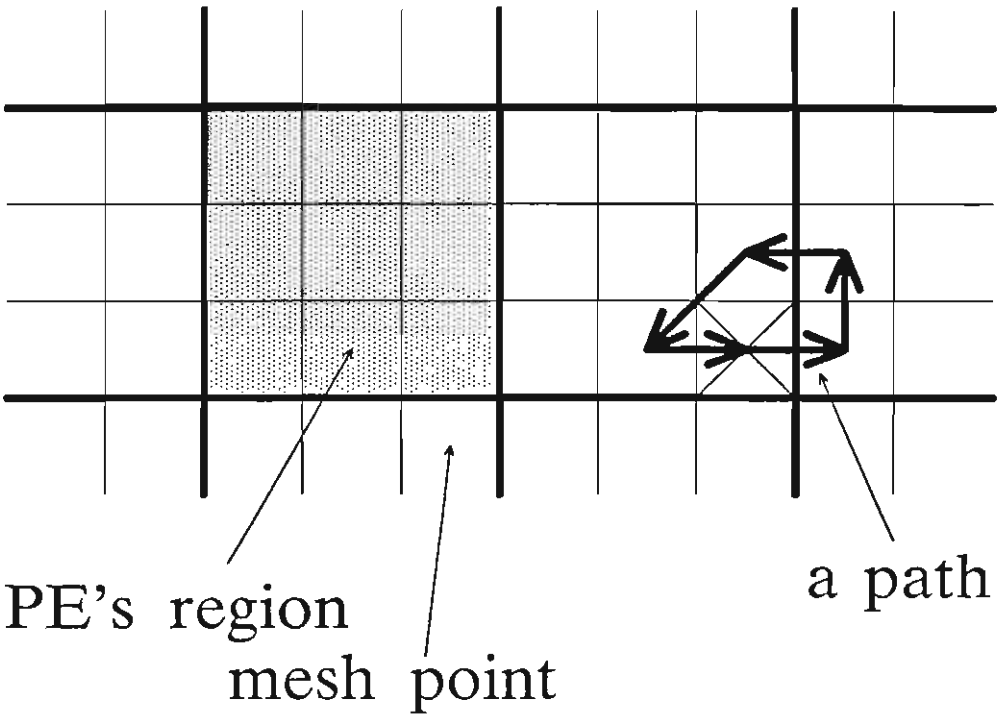
QCD Calculations – Caltech

- A problem in physics research is to evaluate certain differential equations relating the quark interactions.

- The CPU time required to evaluate these equations is around a CRAY–year.

- Since this is research, the physicists will run the program repeatedly, adjusting it each time.

Picture of QCD Solver

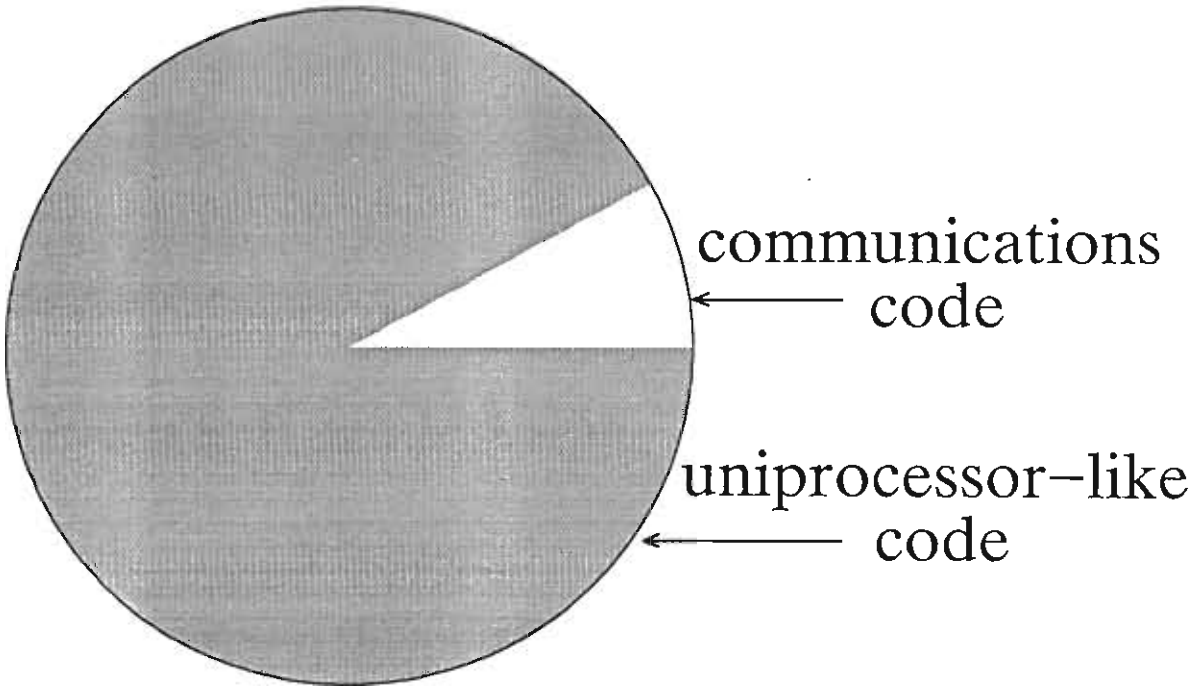


- The real problem is four dimensional.

- The values at the mesh points are matrices.

- Updating a mesh point involves touring neighboring mesh points via *a number of* paths.

Structure of QCD Code



- The bulk of the code -- and the part that contains the physics -- is written in a uniprocessor style.

- Physicists can therefore adjust the program as easily as on a conventional computer.

QCD Performance

Total size	N_{sites}	T_{512}	T_1	*Speedup*	*Overhead*
16x8x8x8	16	5.202	2037.033	391.7	0.308
16x16x8x8	32	10.123	4073.586	402.4	0.272
16x16x16x8	64	19.598	8146.693	415.7	0.232
16x16x16x16	128	37.995	16292.905	429.1	0.194
32x16x16x16	256	73.386	32585.330	443.9	0.153
32x32x16x16	512	144.493	65170.179	451.1	0.135
32x32x32x16	1024	284.819	130339.879	457.7	0.119

Various QCD Applications

This lists Caltech research using the hypercube for numerical quantum field theory in approximately chronological order.

Authors	Project		Hypercube	Reference
Brooks, Fox, Otto Randeria, Athas De Benedictis, Newton, Seitz	Glueball mass		Mk I 4 node	1
Otto, Randeria	Glueball mass, modified action		Mk I 4 node	2
Otto, Stack	Static quark potential (Meson) $12^3 \times 16$ lattice		Mk I 64 node	3
Otto, Stolorz	Glueball mass, enhanced statistics $12^3 \times 16$ lattice		Mk I 64 node	4
Fucito, Soloman	Chiral symmetry breaking | Deconfinement transition | Mass spectrum	finite temperature	pseudo fermion	Mk II 64 node	5 5 6
Patel, Otto, Gupta	Monte Carlo renormalization group. Nonperturbative β-function		Mk I 64 node	7
Flower, Otto, Martin	Finite temperature deconfinement | 4 light quark flowers	Langevin	Mk II 32 node	Unpublished
Flower, Otto	Energy density, heavy meson		Mk II 32 node	8
Flower, Otto	Static quark potential (Meson) 20^4 lattice. Scaling		Mk II 128 node	9
Kolawa, Furmanski	Glueball mass (su(2)). Hamiltonian "loop" formalism		Mk II 32 node	10
Stolorz, Otto	Microcanonical renormalization Group		Mk I 64 node	11
Flower	Restoration of rotational symmetry		Mk II 128 node	12
Flower	Static quark potential (Baryon) 20^4 lattice		Mk II 128 node	12
Flower	Energy density (Baryon)		Mk II 32 and 128 node	12
Flower, Otto, Martin, Apostolakis	Static quark potential (Meson). Four light flavors by Langevin method		Mark III 32 node	In progress
Baillie, Ding, Gupta	QCD with fermions		FPS T-series 128-node	In progress

Other Caltech Applications

Geophysics

42	I:	Finite-Difference Wave Propagation	
43	I:	Normal Modes of the Earth	
44	I:	Finite-Element Flow Modelling	(191)

Physics

45	I:	Lattice Gauge Theory with Fermions on the Hypercube	(184)
46	B:	Random Lattice Calculations	(183)
47	C:	Two Dimensional Melting	(95)
48	B:	Non Local Path Integral Monte Carlo for Helium	(177)
49	I:	N log N Algorithm for Astrophysical Particle Dynamics	(164)
50	I:	The Hypercube for Astronomical Data Analysis	(215)
51	I:	Statistical Gravitational Lensing	(184)
51A	I:	Multichannel Schrödinger Equation	

General Algorithms and Numerical Analysis

52	C:	LU Decomposition of Banded Matrices and the Solution of Linear Systems	(4)
53	C:	Optimal Matrix Algorithms and Communication Strategies for Homogeneous Hypercubes	(C^3P-314, 386)
54	I:	Adaptive Multigrid on the Mark III	(159)
55	I:	Finite Element Methods in Coherent Parallel C	(56)
56	C:	Communication Strategies for Network Simulations	(C^3P-405)
57	C:	A Concurrent Implementation of the Prime Factor Algorithm	(6)
58	C:	Concurrent Tracking Algorithms with Kalman Filters	(186)
59	I:	Chess on a Hypercube	383
60	C:	Shift Register Sequence Random Number Generators on the Hypercube	(182)
61	B:	Transaction Analysis on the NCUBE	

34 Applications from Caltech Computation Program as of November 1987. (Labelled 28-61) (I= In Progress, B= Begin, C= Complete).

Biology and CNS

28	I	Structural Simulations of Neural Networks Using a General-Purpose Neural Network Simulator and a Hypercube Concurrent Computer	(C^3P-404)
28A	B:	Periodic Orbits in the Pitiform Cortex	
29	C:	Back Propagation Algorithms for Character Recognition and Computer Games	
30	I:	Pattern Recognition by Neural Networks on Hypercubes	(207)
31	I:	Collective Stereopsis	(16)
32	B:	Mapping the Human Genome	
32A	B:	Modeling Complex Neurons	

Chemistry and Chemical Engineering

33	I:	Integration of Coupled Sets of Ordinary Differential Equations on the Caltech Hypercubes: Matrix Inversion	(85)
34	B:	Polymer Simulations on the Hypercube	
35	B:	Concurrent Optimization and Dynamic Simulation in Chemical Engineering	
35A	B:	Quantum Lattice System for High T_c Superconductivity and Monte Carlo Simulation	

Engineering (See also Numerical Analysis)

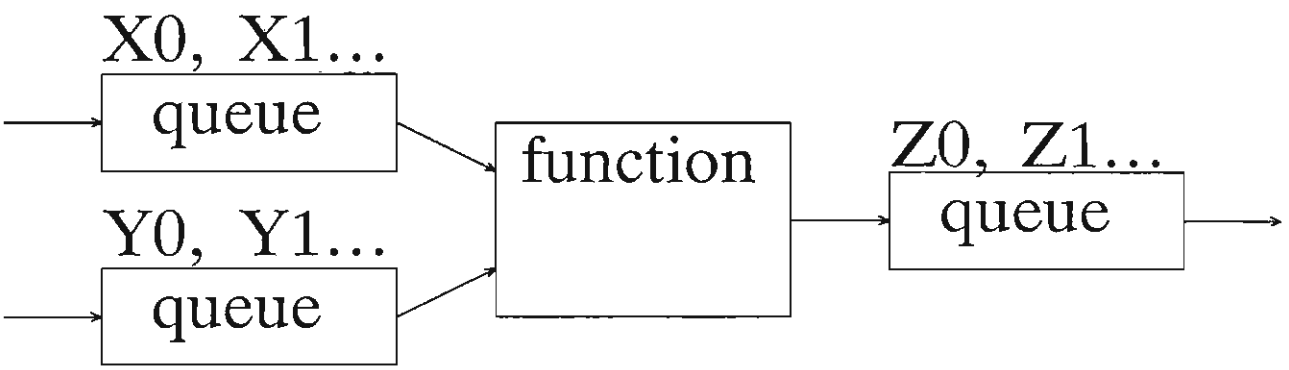
36	C:	Ray Tracing on the Hypercube	(73)
37	I:	Plasma Simulations on the Mark III Hypercube Computer	(108)
38	I:	Vortex Dynamics	(131)
39	I:	Synthetic Aperture Radar (SAR) Analysis on the Hypercube	(468) (303D)
40	I:	Flux-Corrected Transport on the NCUBE	
41	I:	Parallel Free-Langrange Hydrodynamics with a Distributed-Memory Parallel Processor	(189)

Other Programming Techniques

- The EMU Circuit Simulator example demonstrates dataflow programming and the *master–server plan*.

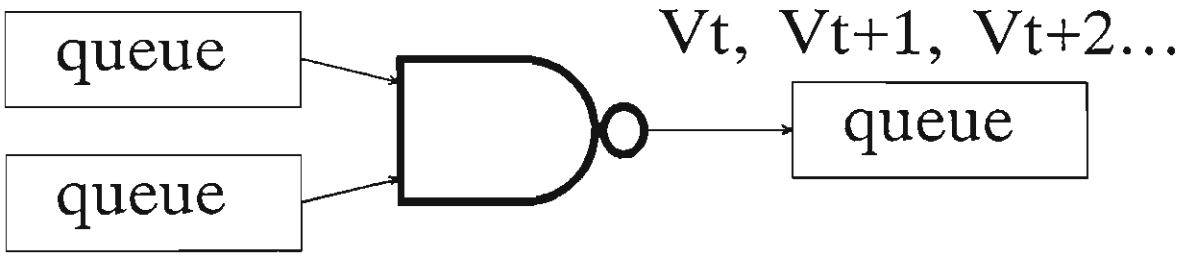
- Other research demonstrates other techniques.

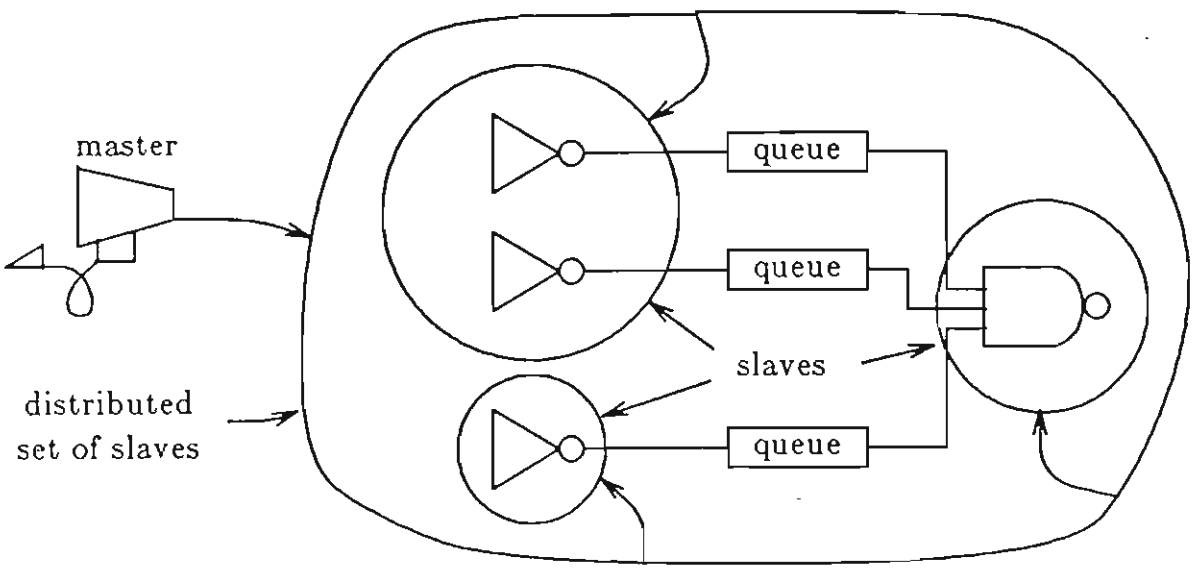
Dataflow Programming



$$Z_i = f(X_i, Y_i)$$

Experiment: EMU Simulator





Performance of Simulator

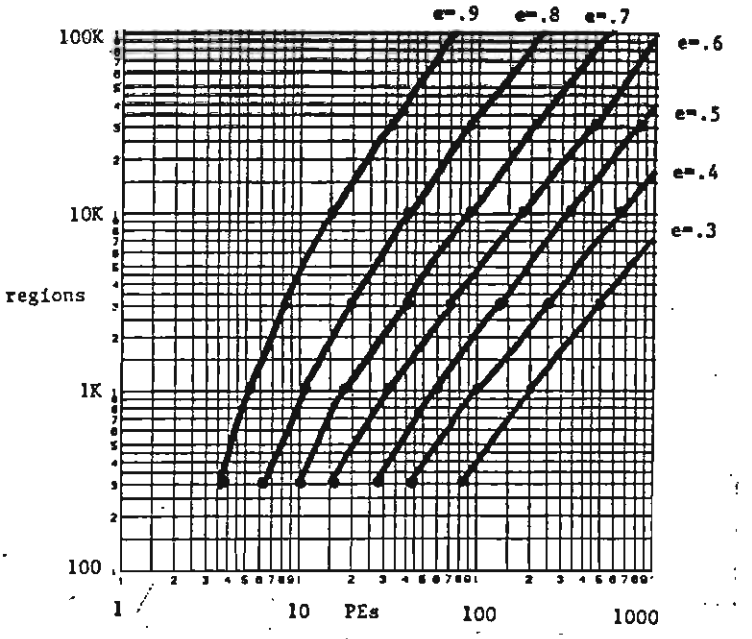


Figure 8: Load Balance Efficiency

- Vertical axis is circuit size in regions, where a region is about 8 transistors.

- Horizontal axis is the number of PE's in multiprocessor.

- Plots give lines of equal efficiency.

- This is an analytic model, but it has been validated by actual runs.

- Speedup for Bell Award: 39; highest speedup to date: 68 (129 PE's).

Parallel Compilers

- Many of the functions in languages and compilers were done manually until they were well enough understood to be coded into a compiler.

- The parallel behavior of Hypercube programs is currently written manually, but some of it is becoming well understood.

- Therefore, perhaps parallel compilers for Hypercubes are possible now.

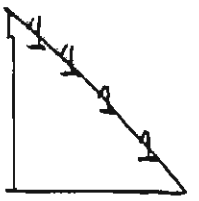
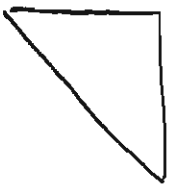
- Crystal is a parallel language. The Crystal compiler incorporates some Hypercube programming methods to generate efficient parallel code.

- The LU–decomposition program demonstrates the parallel compiler.

LU Decomposition in Crystal

```
LU_decomposition(A0) = [L_matrix, U_matrix]
 where (
 n = ||A0||,  ! dimension of the square matrix

 a(i, j, k) over D3 =
    << k = 0 -> A0[i-1, j-1],
      (0 < k) and (k <= n) -> a(i, j , k-1) - (L(i,  k) * ( U(k, j)))
    >>,

 L(i, k) over D1 =
    << (1 <= i) and (i < k) -> 0,
      i = k -> 1,
      k < i -> a(i, k, k-1) % U( k, k)
    >>,

 U(k, j) over D2 =
    << (1 <= j) and (j < k) -> 0,
      k <= j -> a(k, j, k-1)
    >>,

 ! define a 3-dimensional domain
 D = {(i,j,k) | 1 <= i < n+1, 1 <= j < n+1, 1 <= k < n+1},
 ! projection of D along the 1'st axis
 D1 = D proj 1,
 ! projection of D along the 0'th axis, then transpose the domain
 D2 = transpose(D proj 0),
 ! join of two domains into one
 D3 = {(i,j,0) | (i,j) in (D proj 2)} + D

 L_matrix = [ L(i,k) | (i,k) in D1],
 U_matrix = [ U(k,j) | (k,j) in D2]
 )
```

LU Decomposition from a Textbook

$A = LU$ L:  U: 

$$L_{ij} = \frac{A_{ij} - \sum_{k=1}^{j-1} L_{ik} U_{kj}}{U_{jj}}$$

$$U_{ij} = A_{ij} - \sum_{k=1}^{i-1} L_{ik} U_{kj}$$

Let $a_{ijk} = A_{ij} - \sum_{k=1}^{x} L_{ik} U_{kj}$

then $U_{ij} = a_{i,j,i-1}$

$$L_{ij} = \frac{a_{i,j,j-1}}{U_{jj}}$$

Crystal Results

- Speedup on 500x500 matrix: 98 on 128 PEs.

- The compiler generated code that interleaved data (for load balance) and also blocked the data (for communications performance).

Conclusions

- We believe that Hypercubes will evolve into general purpose computers.

- Speedup has been adequately demonstrated for Hypercubes; therefore Hypercubes will evolve into *scalable* general purpose computers.

- Some important areas for future research on Hypercubes are:

  – Methods for informing and teaching scientists and engineers about Hypercube programming.

  – Devising more programming techniques to extend the range of applications that are considered amenable to Hypercubes.

  – Automating programming techniques through parallel compilers.