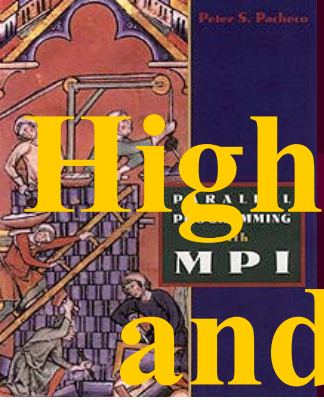
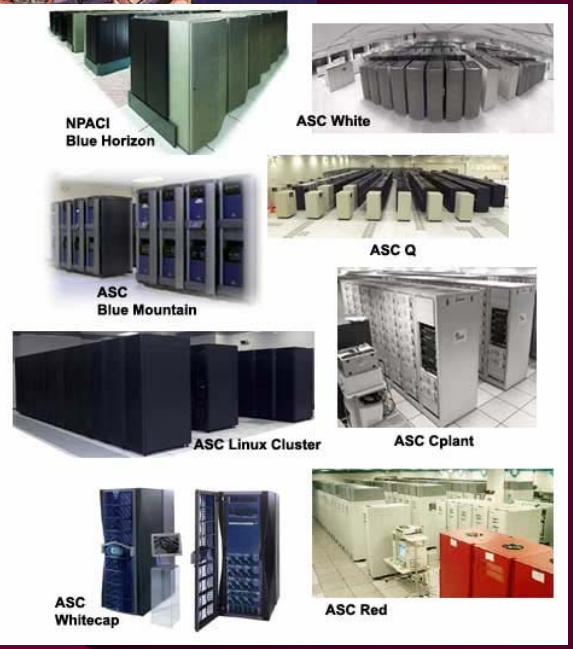
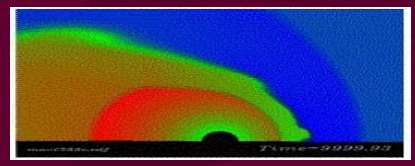
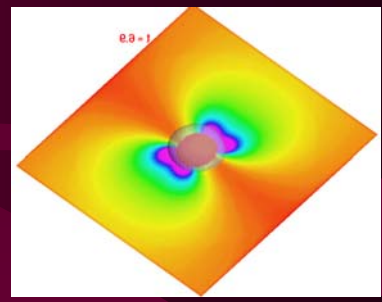
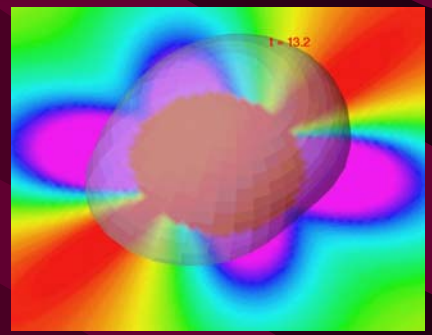


High Performance Computing and Computational Science



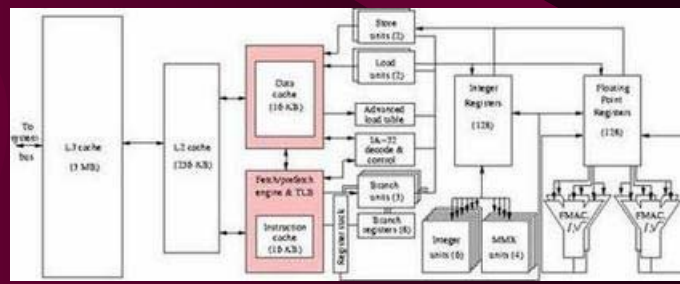






Spring Semester 2005

Geoffrey Fox
Community Grids Laboratory
Indiana University
505 N Morton
Suite 224
Bloomington IN
gcf@indiana.edu



Abstract of Introduction to HPC & Computational Science (HPCCS)

- Course Logistics
- Exemplar applications
- Status of High Performance Computing and Computation HPCC nationally
- Application Driving Forces
  - Some Case Studies -- Importance of algorithms, data and simulations
- Parallel Processing in Society
- Technology and Commodity Driving Forces
  - Inevitability of Parallelism in different forms
  - Moore's law and exponentially increasing transistors
  - Dominance of Commodity Implementation

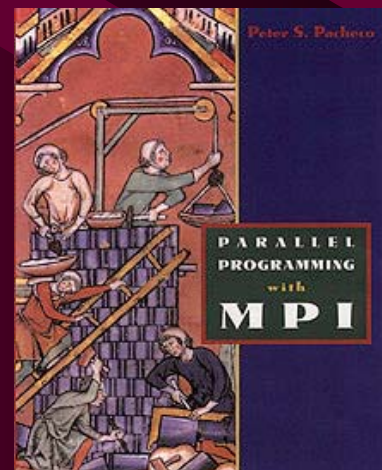
Basic Course Logistics

- Instructor: Geoffrey Fox -- gcf@indiana.edu, 8122194643
- Backup: Marlon Pierce – mpierce@cs.indiana.edu,
- Home Page is: http://grids.ucs.indiana.edu/ptliupages/jsucourse2005/
- A course with similar scope was given Spring 2000 at http://www.old-npac.org/projects/cps615spring00/
  - The machines have got more powerful and there are some architectural innovations but base ideas and software techniques are largely unchanged
- There is a two volume CD of resource material prepared in 1999 which we can probably make available

Books For Course

- **The Sourcebook of Parallel Computing,** Edited by Jack Dongarra, Ian Foster, Geoffrey Fox, William Gropp, Ken Kennedy, Linda Torczon, Andy White, October 2002, 760 pages, ISBN 1-55860-871-0, Morgan Kaufmann Publishers. http://www.mkp.com/books_catalog/catalog.asp?ISBN=1-55860-871-0

- **Parallel Programming with MPI**, Peter S. Pacheco, Morgan Kaufmann, 1997. Book web page: http://fawlty.cs.usfca.edu/mpi/



Course Organization

- Graded on the basis of approximately 8 Homework sets which will be due Thursday of the week following day (Monday or Wednesday given out)
- There will be one project -- which will start after message passing (MPI) discussed
- Total grade is 70% homework, 30% project
- Languages will Fortran or C
- All homework will be handled via email to gcf@indiana.edu

Useful Recent Courses on the Web

- Arvind Krishnamurthy, Parallel Computing, Yale
  - http://lambda.cs.yale.edu/cs424/notes/lecture.html Fall 2004
- Jack Dongarra, Understanding Parallel Computing, Tennessee
  http://www.cs.utk.edu/%7Edongarra/WEB-PAGES/cs594-2005.html Spring 2005
  http://www.cs.utk.edu/%7Edongarra/WEB-PAGES/cs594-2003.html Spring 2003
- Alan Edelman, Applied Parallel Computing, MIT
  http://beowulf.lcs.mit.edu/18.337/ Spring 2004
- Kathy Yelick, Applications of Parallel Computers, UC Berkeley
  http://www.cs.berkeley.edu/~yelick/cs267/ Spring 2004
- Allan Snavely, CS260: Parallel Computation, UC San Diego
  http://www.sdsc.edu/~allans/cs260/cs260.html Fall 2004
- John Gilbert, Applied Parallel Computing, UC Santa Barbara
  http://www.cs.ucsb.edu/~gilbert/cs240aSpr2004/ Spring 2004
- Old course from Geoffrey Fox
  http://www.old-npac.org/projects/cps615spring00/ Spring 2000

Generally Useful Links

- Summary of Processor Specifications
  http://www.geek.com/procspec/procspec.htm
- Top 500 Supercomputers updated twice a year
  http://www.top500.org/list/2003/11/
  http://www.top500.org/ORSC/2004/overview.html
- Past Supercomputer Dreams
  http://www.paralogos.com/DeadSuper/
- OpenMP Programming Model
  http://www.openmp.org/
- Message Passing Interface
  http://www.mpi-forum.org/

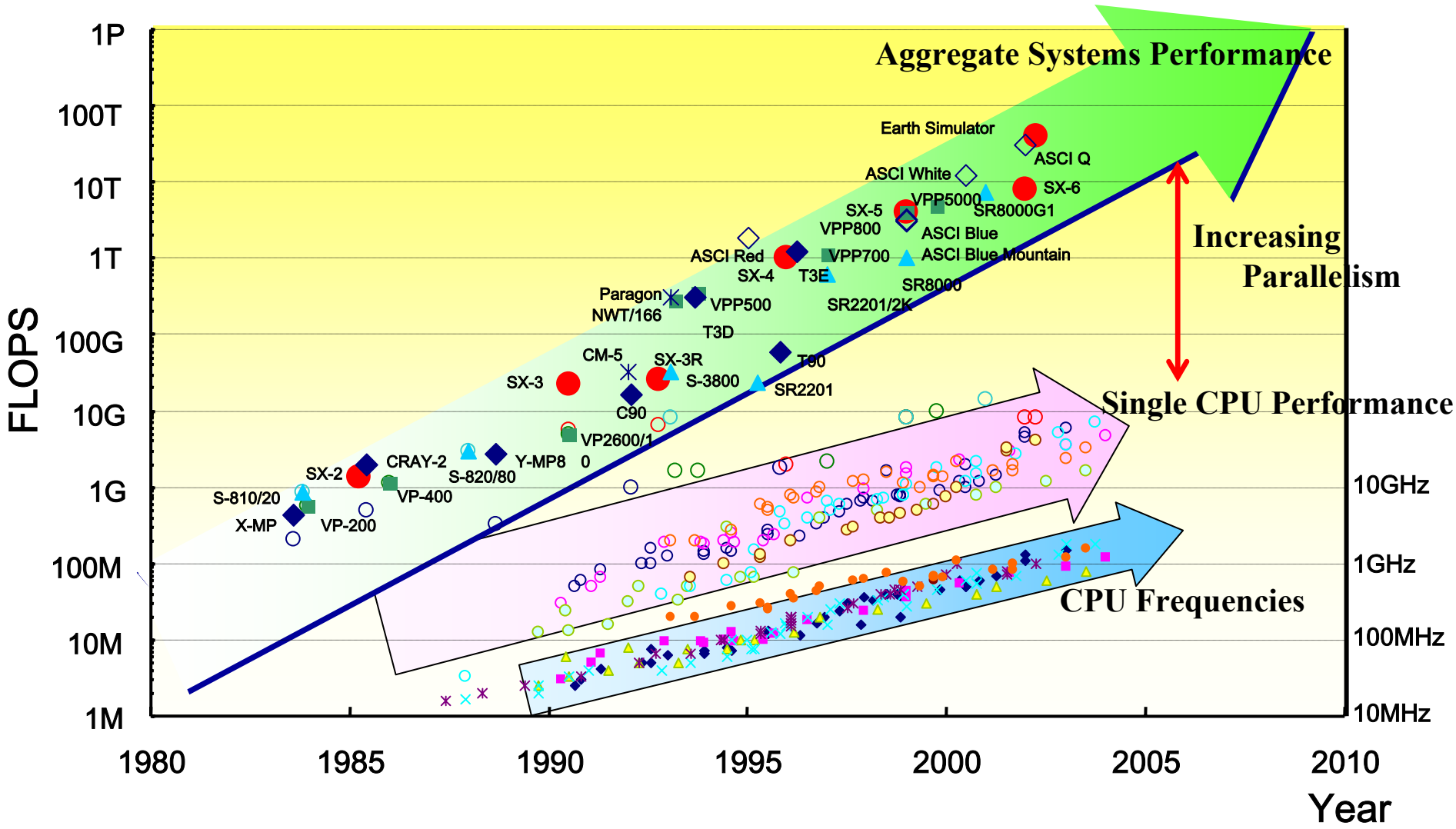
Very Useful Old References

- David Bailey and Bob Lucas CS267 Applications of Parallel Computers
  - http://www.nersc.gov/~dhbailey/cs267/ Taught 2000
- Jim Demmel's Parallel Applications Course: http://www.cs.berkeley.edu/~demmel/cs267_Spr99/
- Dave Culler's Parallel Architecture course: http://www.cs.berkeley.edu/~culler/cs258-s99/
- David Culler and Horst Simon 1997 Parallel Applications: http://now.CS.Berkeley.edu/cs267/
- Michael Heath Parallel Numerical Algorithms: http://www.cse.uiuc.edu/cse412/index.html
- Willi Schonauer book (hand written): http://www.uni-karlsruhe.de/Uni/RZ/Personen/rz03/book/index.html
- Parallel computing at CMU: http://www.cs.cmu.edu/~scandal/research/parallel.html

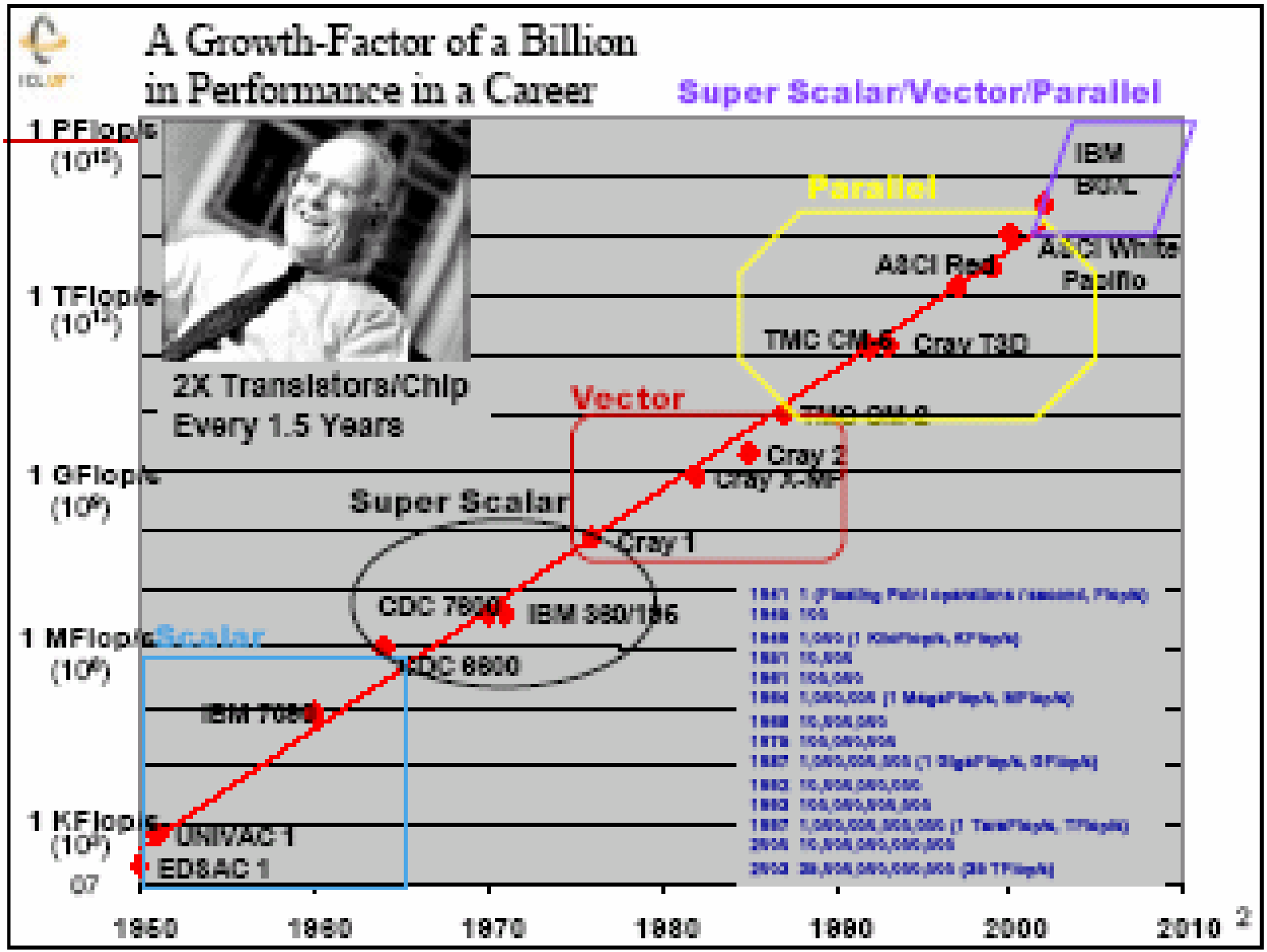
Essence of Parallel Computing

- When you want to solve a large or hard problem, you don't hire superperson, you hire lots of ordinary people
  - Palaces and Houses have same building material (roughly); you use more on a Palace
- Parallel Computing is about using lots of computers together to compute large computations
  - Issues are organization (architecture) and orchestrating all those CPUs to work together properly
  - What mangers and CEOs do in companies

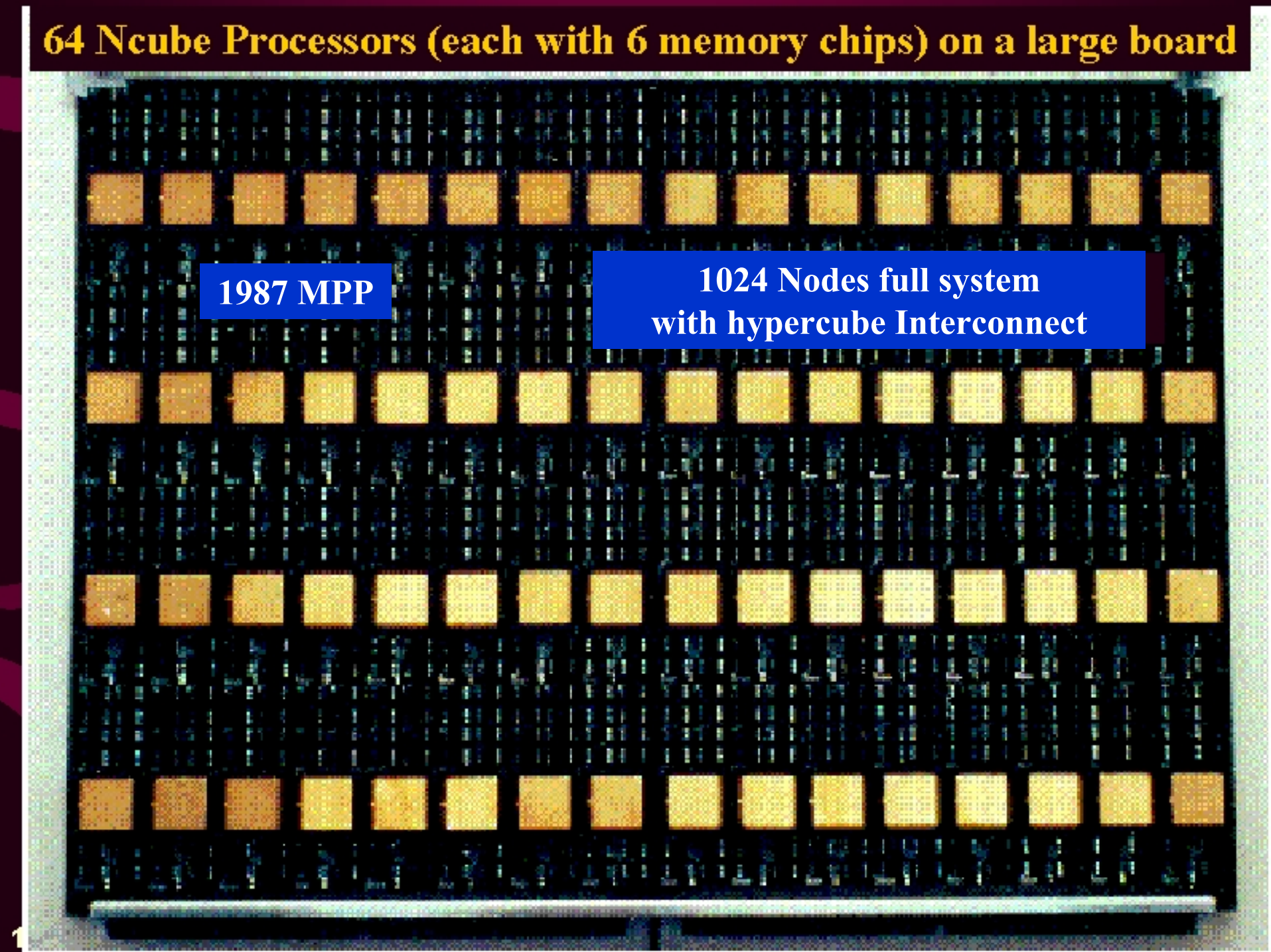
History of High Performance Computers



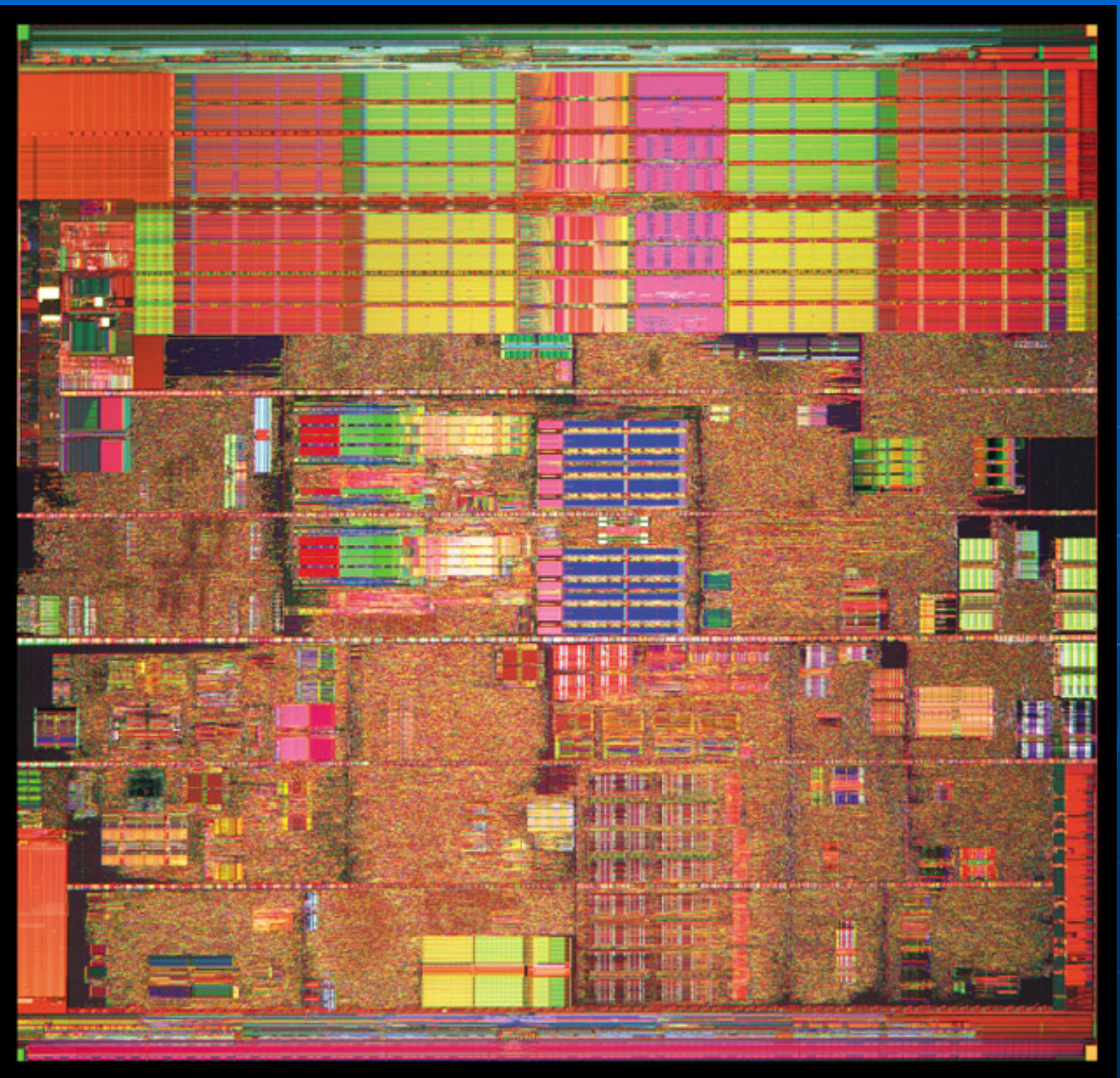
Performance from 1960 to 2010



64 Ncube Processors (each with 6 memory chips) on a large board



Prescott has 125 Million Transistors



Compared to Ncube
100X Clock
500X Density

50000X Potential Peak Performance Improvement

Probably more like
1000X
Realized Performance Improvement

So not so easy to organize all those transistors to work together

Consequences of Transistor Deluge

- The increase in performance of PC's and Supercomputer's comes from the continued improvement in the capability to build chips with more and more transistors
  - Moore's law describes this increase which has been a constant exponential for 50 years
- This translates to more performance and more memory for a given cost or a given space
  - Better communication networks and more powerful sensors driven by related technology (and optical fibre)
- The ability to effectively use all these transistors is central problem in parallel computing
- Software methodology has advanced much more slowly than the hardware
  - The MPI approach we will describe is over 20 years old

Some Comments on Simulation and HPCC

- HPCC is a maturing field with many organizations installing large scale systems
- These include NSF (academic computations) with TeraGrid activity, DoE (Dept of Energy) with ASCI and DoD (Defense) with Modernization
  - New High End Computing efforts partially spurred by Earth Simulator
- There are new applications with new algorithmic challenges
  - Web Search and Hosting Applications
  - ASCI especially developed large linked complex simulations with if not new much better support in areas like adaptive meshes
  - On earthquake simulation, new "fast multipole" approaches to a problem not tackled this way before
  - On financial modeling, new Monte Carlo methods for complex options
- Integration of Grids and HPCC to build portals (problem solving Environments) and to supporting increasing interest in embarrassingly or pleasingly parallel problems

Application Driving Forces

4 Exemplars

Selection of Motivating Applications

- Large Scale Simulations in Engineering
  - Model airflow around an aircraft
  - Study environmental issues -- flow of contaminants
  - Forecast weather
  - Oil Industry: Reservoir Simulation and analysis of Seismic data
- Large Scale Academic Simulations (Physics, Chemistry, Biology)
  - Study of Evolution of Universe
  - Study of fundamental particles: Quarks and Gluons
  - Study of protein folding
  - Study of catalysts
  - Forecast Earthquakes (has real world relevance)
- "Other Critical Real World Applications"
  - Transaction Processing
  - Web Search Engines and Web Document Repositories
  - Run optimization and classification algorithms in datamining of Enterprise Information Systems
  - Model Financial Instruments

Units of HPCC

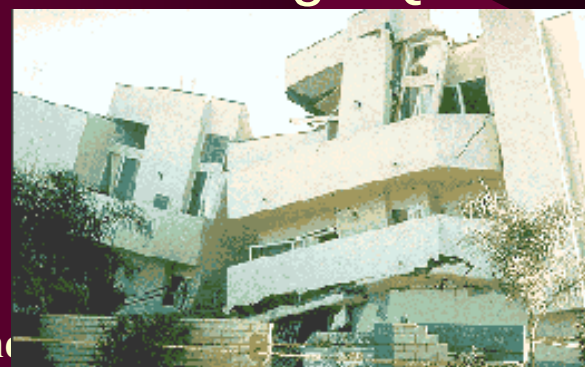
- **From Jim Demmel we need to define:**

1 Mflop	1 Megaflop	10^6 Flop/sec
1 Gflop	1 Gigaflop	10^9 Flop/sec
1 Tflop	1 Teraflop	10^12 Flop/sec
1 MB	1 Megabyte	10^6 Bytes
1 GB	1 Gigabyte	10^9 Bytes
1 TB	1 Terabyte	10^12 Bytes
1 PB	1 Petabyte	10^15 Bytes

Application Motivation I: Earthquakes

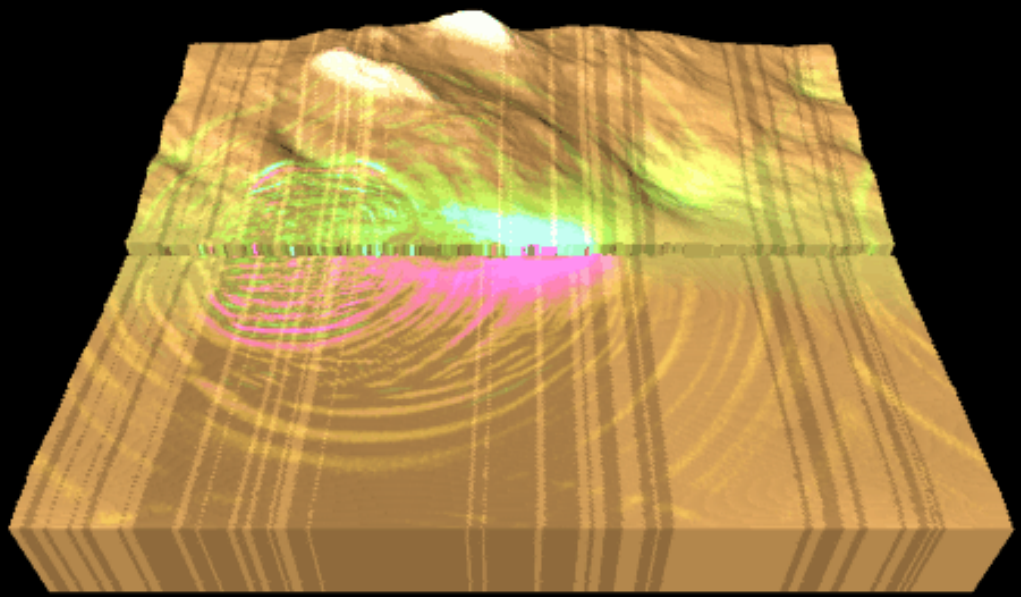
- Kobe 1995 Earthquake caused $200 Billion in damage and was quite unexpected -- the big one(s) in California are expected to be worse
- Field Involves Integration of simulation (of earth dynamics) with sensor data (e.g. new GPS satellite measurements of strains http://www.scign.org) and with information gotten from pick and shovel at the fault line.
  - Laboratory experiments on shaking building and measurement of frictions between types of rock materials at faults

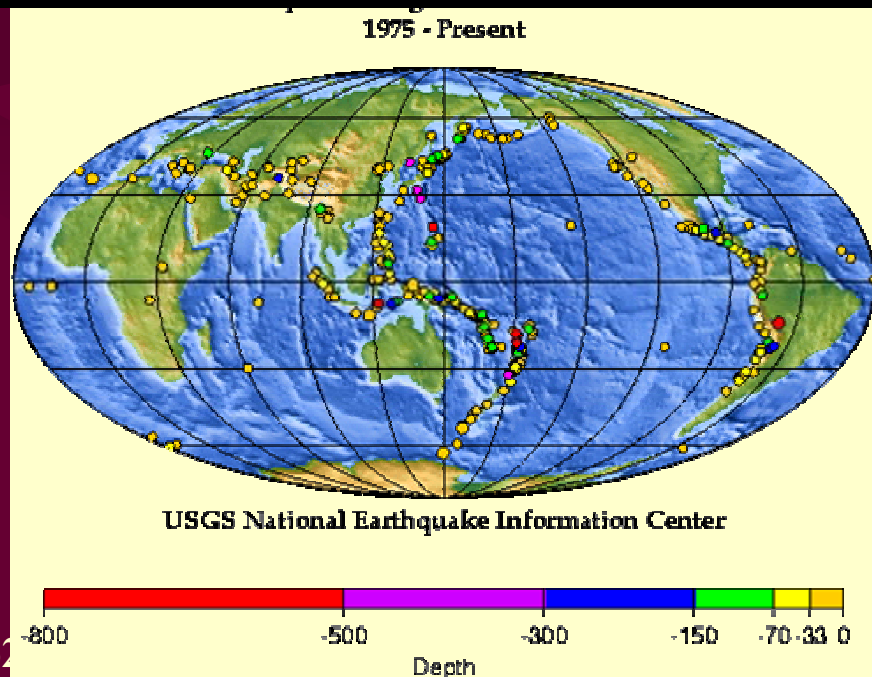
Northridge Quake

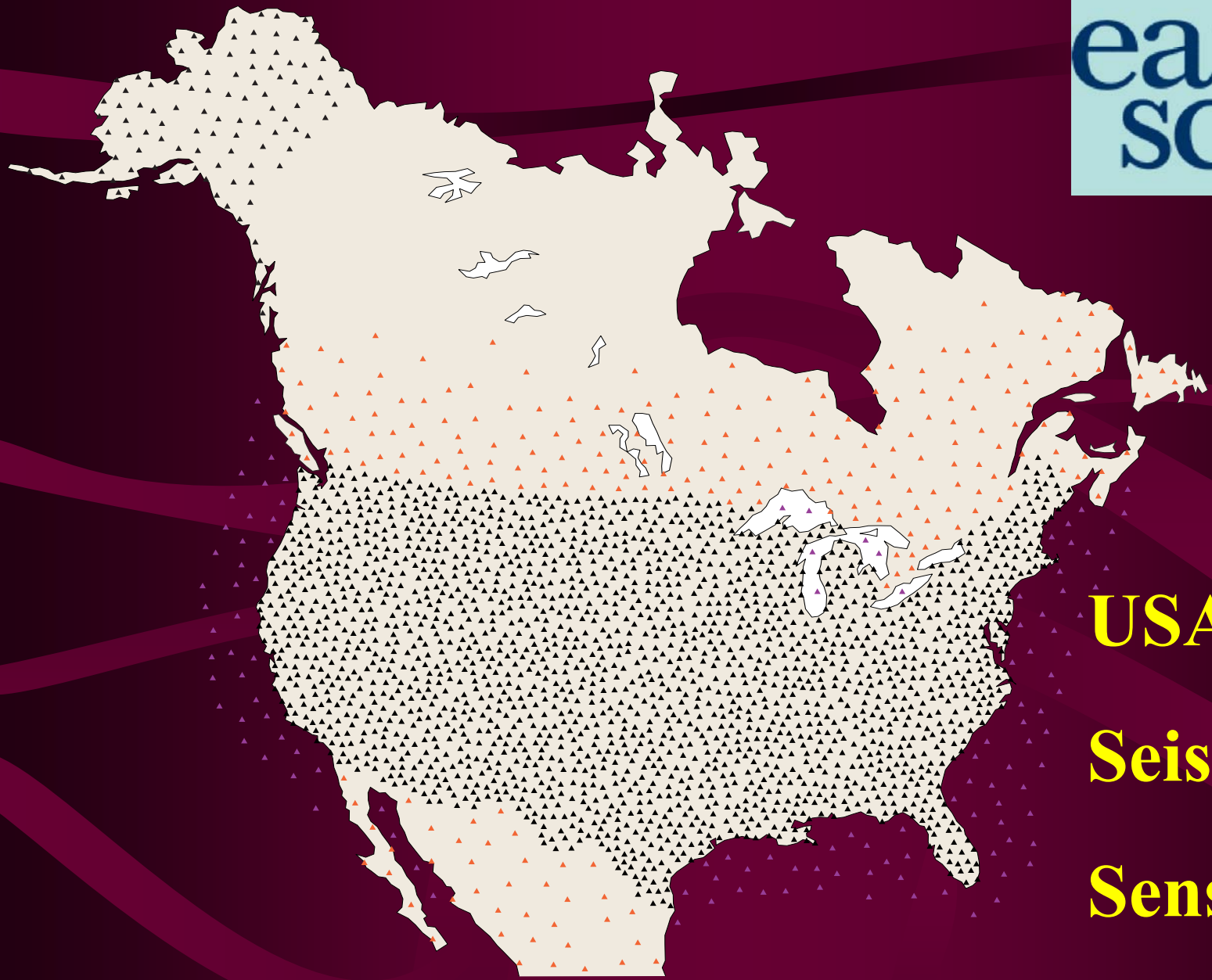


Application Motivation I: Earthquakes (Contd.)

- Technologies include data-mining (is dog barking really correlated with earthquakes) as well as PDE solvers where both finite element and fast multipole methods (for Green's function problems) are important
- Multidisciplinary links of ground motion to building response simulations
- Applications include real-time estimates of after-shocks used by scientists and perhaps crisis management groups
- http://www.servogrid.org







earthscope

USArray

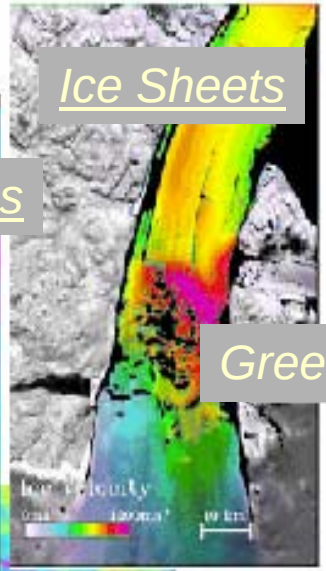
Seismic

Sensors

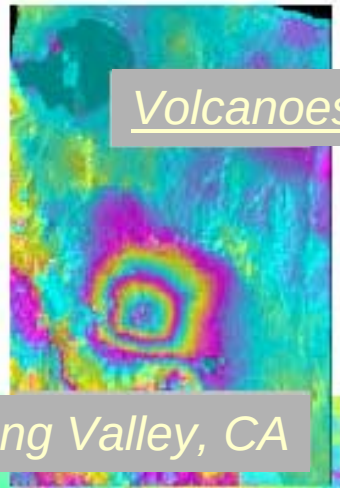
Interferometric SAR Observations: Present and Future

Site-specific Irregular Scalar Measurements

Ice Sheets

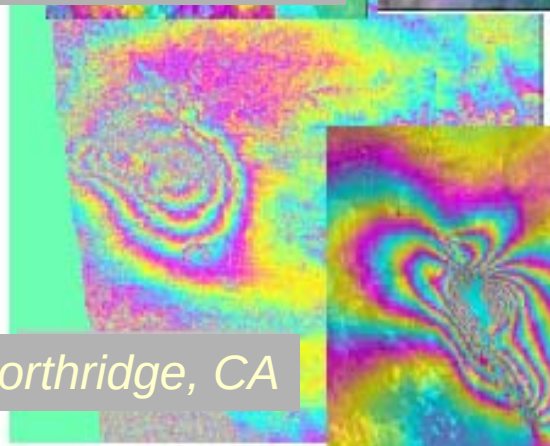


Volcanoes

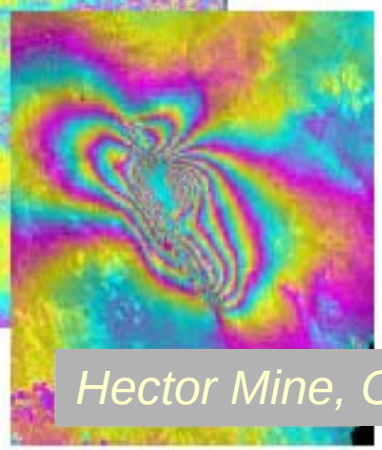


Greenland

Long Valley, CA



Northridge, CA

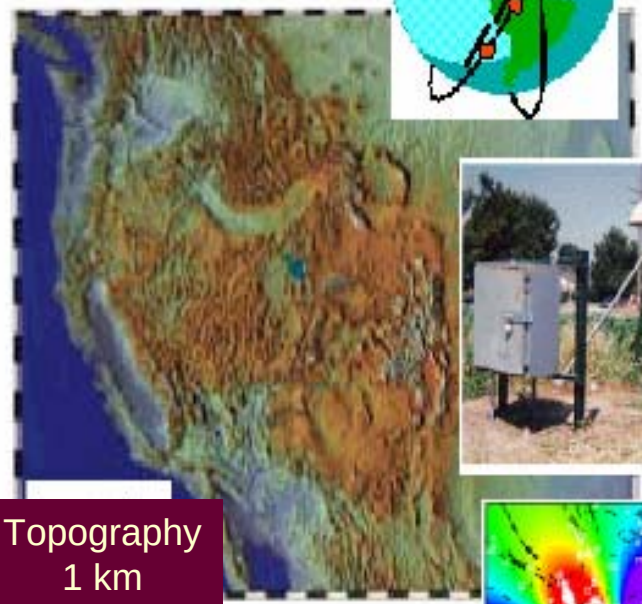


Hector Mine, CA

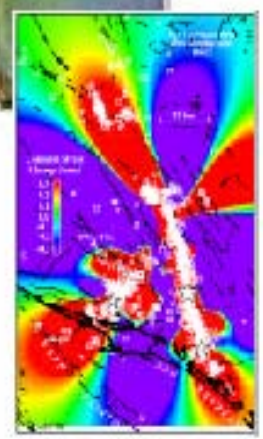
Earthquakes

Constellations for Plate Boundary-Scale Vector Measurements

PBO



Topography 1 km



Stress Change

Computing Requirements for Weather

	2002 System	2010+ System	

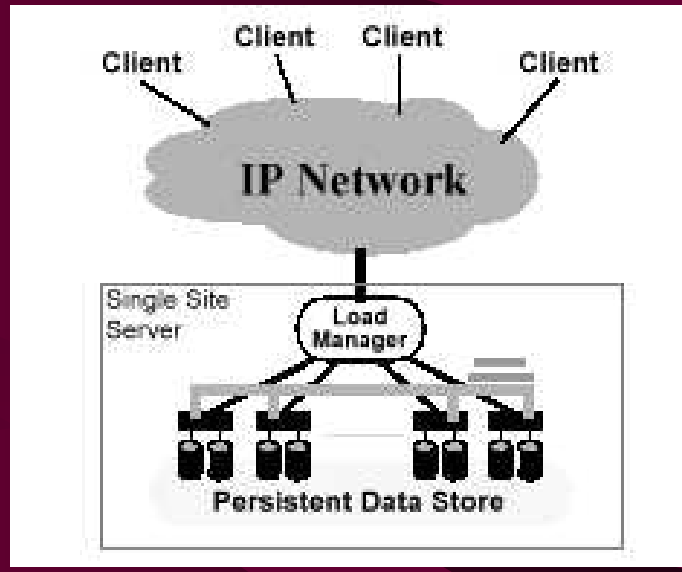
Resolution
• Horizontal
• Vertical levels
• Time step
• Observations
  ○ Ingested
  ○ Assimilated
	
	100 km
	55
	30 minutes
	
	10^7 / day
	10^5 / day
		
		10 km
		100
		6 minutes
		
		10^{11} / day
		10^8/ day	

System Components:	Atmosphere Land-surface Data assimilation	Atmosphere, Land-surface, Ocean, Sea-ice, Next-generation data assimilation Chemical constituents (100)	
Computing: • Capability (single image system) • Capacity (includes test, validation, reanalyzes, development)	 10 GFlops 100 GFlops	Must Have 20 TFlops (2000x) 400 TFlop (4000x)	Important 50 TFlops 1 PFlops
Data Volume: • Input (observations) • Output (gridded)	 400 MB / day 2 TB / day	1 PB / day 10 PB / day	

Networking/Storage
• Data movement
  ○ Internal
  ○ External
• Archival
	
	4 TB / day
	5 GB / day
	1 TB / day
		
		20 PB / day
		10 TB / day
		10 PB / day	

Application Motivation II: Web Search

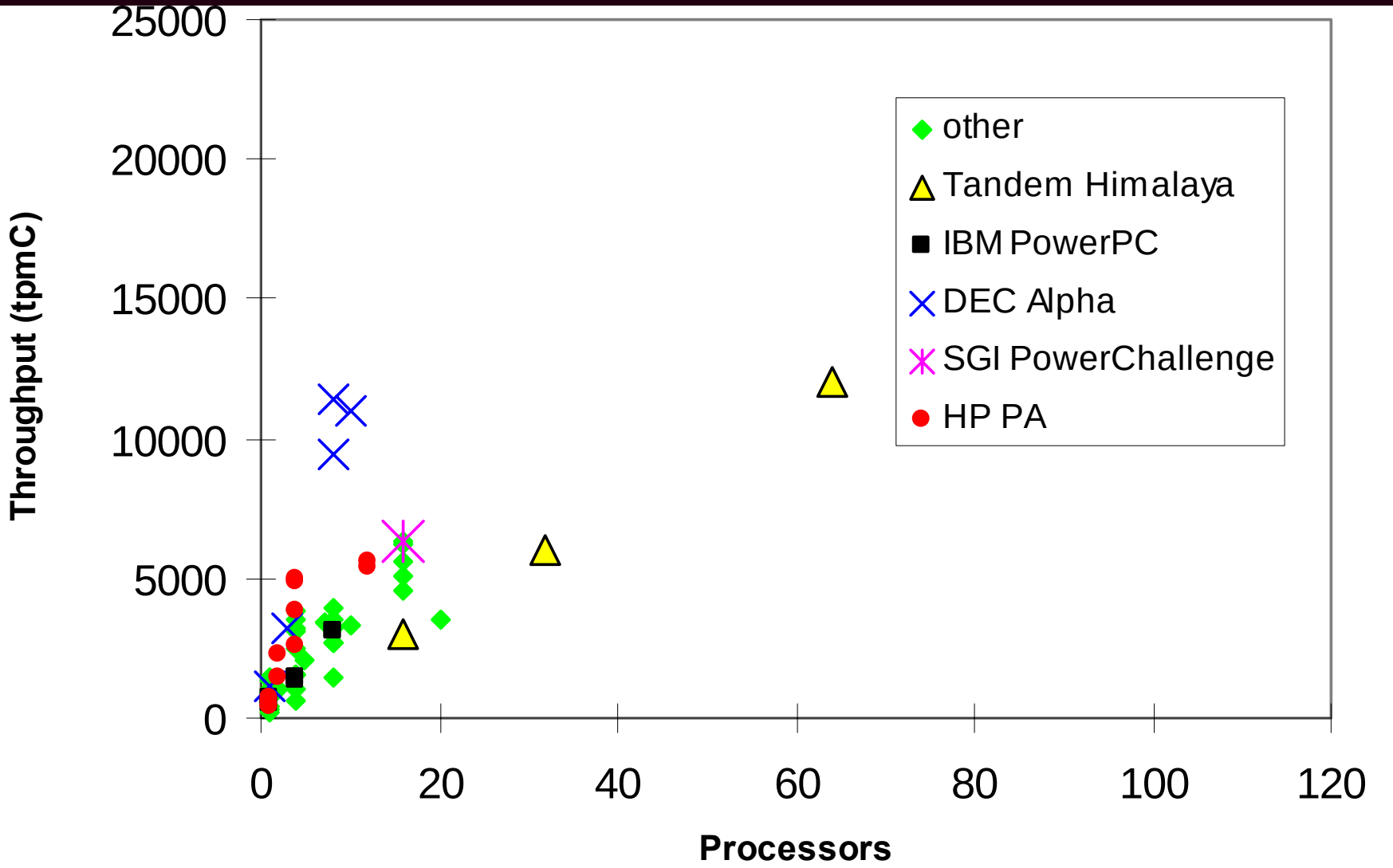
- Note **Web Search**, like transaction analysis has "obvious" **parallelization** (over both users and data)with modest synchronization issues
- Critical issues are: **fault-tolerance** (.9999 to .99999 reliability); **bounded search time** (a fraction of a second); **scalability** (to the world); **fast system upgrade times** (days)



Service	Nodes	Queries	Node HW
AOL web cache	>300	>3.3B/day	4-CPU DEC 4100s
Inktomi Search Engine	500	>40M/day	2-CPU Sun Workstations
Geocities	>300	>25M/day	PCs
Anonymous web-based e-mail	>400	50M/day	PCs

Exemplar III: Database transaction processing

- TPC-C Benchmark Results from March 96
- Parallelism is pervasive (more natural in SQL than Fortran)
- Small to moderate scale parallelism very important



64 Processors

2004 TPC-C Results

Rank	Company	System	tpmC	Price/tpmC	System Availability	Database	Operating System	TP Monitor	Date Submitted	Cluster
1	IBM	IBM eServer p5 595 64p	3,210,540	5.19 US $	05/14/05	IBM DB2 UDB 8.2	IBM AIX 5L V5.3	Microsoft COM+	11/18/04	N
2	HP	HP Integrity rx5670 Cluster 64P	1,184,893	5.52 US $	04/30/04	Oracle Database 10g Enterprise Edition	Red Hat Enterprise Linux AS 3	BEA Tuxedo 8.1	12/08/03	Y
3	IBM	IBM eServer pSeries 690 Model 7040-681	1,025,486	5.43 US $	08/16/04	IBM DB2 UDB 8.1	IBM AIX 5L V5.2	Microsoft COM+	02/17/04	N
4	HP	HP Integrity Superdome	1,008,144	8.33 US $	04/14/04	Oracle Database 10g Enterprise Edition	HP UX 11.iv2 64-Bit Base OS	BEA Tuxedo 8.0	11/04/03	N
5	IBM	IBM eServer p5 570 16P	809,144	4.95 US $	09/30/04	IBM DB2 UDB 8.1	IBM AIX 5L V5.3	Microsoft COM+	07/12/04	N
6	HP	HP Integrity Superdome	786,646	6.49 US $	10/23/03	Microsoft SQL Server 2000 Enterprise Ed. 64-bit	Microsoft Windows Server 2003 Datacenter Edition 64-bit	Microsoft COM+	08/27/03	N
7	IBM	IBM eServer pSeries 690 Turbo 7040-681	768,839	8.55 US $	02/29/04	Oracle Database 10g Enterprise Edition	IBM AIX 5L V5.2	TXSeries Developers for AIX V5	09/12/03	N
8	IBM	IBM eServer pSeries 690 Turbo 7040-681	763,898	8.25 US $	11/08/03	IBM DB2 UDB 8.1	IBM AIX 5L V5.2	BEA Tuxedo 8.0	06/30/03	N
9	HP	HP Integrity Superdome	707,102	7.16 US $	10/23/03	Microsoft SQL Server 2000 Enterprise Ed. 64-bit	Microsoft Windows Server 2003 Datacenter Edition	Microsoft COM+	05/20/03	N
10	NEC	NEC Express5800/1320Xd C/S w/ Express5800/120Rf-2	683,575	5.99 US $	10/05/04	Oracle Database 10g Enterprise Edition	SUSE LINUX Enterprise Server 9	BEA Tuxedo 8.1	06/28/04	N

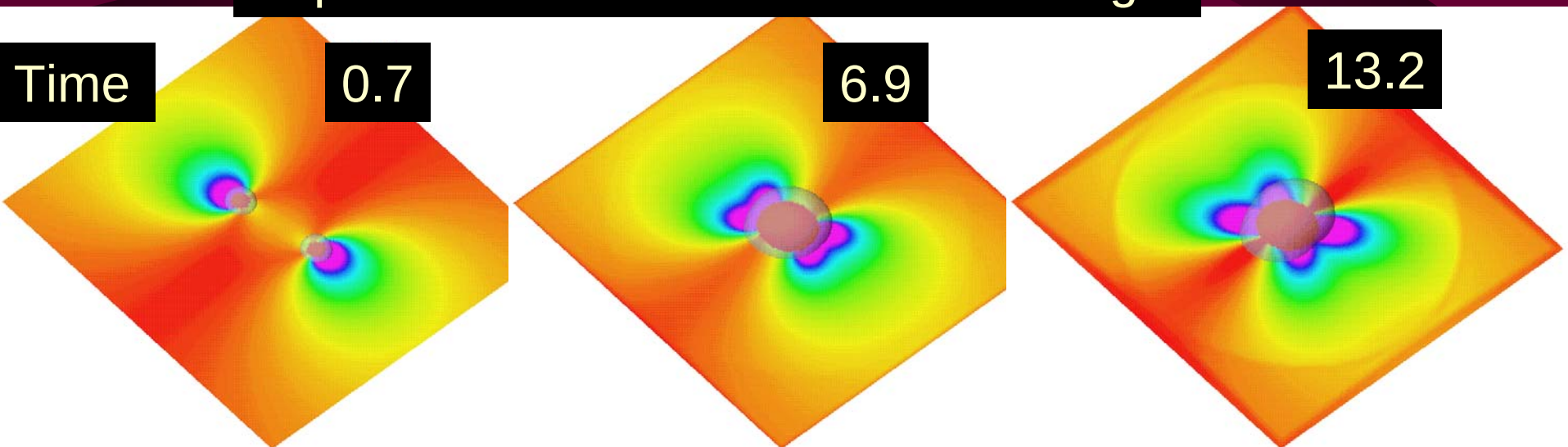
Application Motivation IV: Numerical Relativity

- As with all physical simulations, realistic 3D computations require "Teraflop" (10^12 operations per second) performance
- Numerical Relativity just solves the "trivial" Einstein equations $G_{\mu\nu} = 8\pi T_{\mu\nu}$ with indices running over 4 dimensions
- Apply to collision of two black holes which are expected to be a major source of gravitational waves for which US and Europe are building major detectors
- Unique features includes freedom to choose coordinate systems (Gauge freedom) in ways that changes nature of equations
- Black Hole has amazing boundary condition that no information can escape from it.
  - Not so clear how to formulate this numerically and involves interplay between computer science and physics
- At infinity, one has "simple" (but numerically difficult) wave equation; near black hole one finds very non linear system

Application Motivation IV: Numerical Relativity (Contd.)

- **Fortran90** (array syntax) very attractive to handle equations which are naturally written in Tensor (multi-dimensional) form
- **12 independent field values** defined on a mesh with black holes excised -- non trivial dynamic irregularity as holes rotate and spiral into each other in interesting domain
- **Irregular dynamic mesh** is not so natural in (parallel) Fortran 90 and one needs technology (including distributed data structures like **DAGH**) to support **adaptive finite difference** codes.

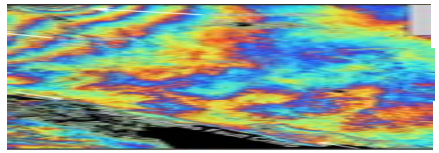
Separate Holes are simulated till Merger

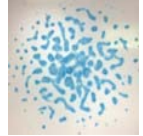


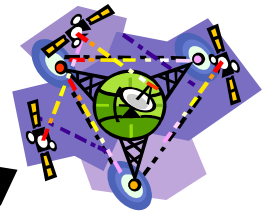
Summary of Application Trends

- There is a dynamic interplay between application needing more hardware and hardware allowing new/more applications
- Transition to parallel computing has occurred for scientific and engineering computing but this is 1-2% of computer market
  - Integration of Data/Computing
- Rapid progress in commercial computing
  - Database and transactions as well as financial modeling/oil reservoir simulation
  - Web servers including multi-media and search growing importance
  - Typically functional or pleasingly parallel
- Growing Importance of Observational Data
  - Sensors are increasing in capacity as fast as computers

Data Deluged Science Computing Paradigm







Data

Assimilation

Information

Simulation

Informatics

Model

Datamining

Ideas

Reasoning

Computational Science

Parallel Processing in Society

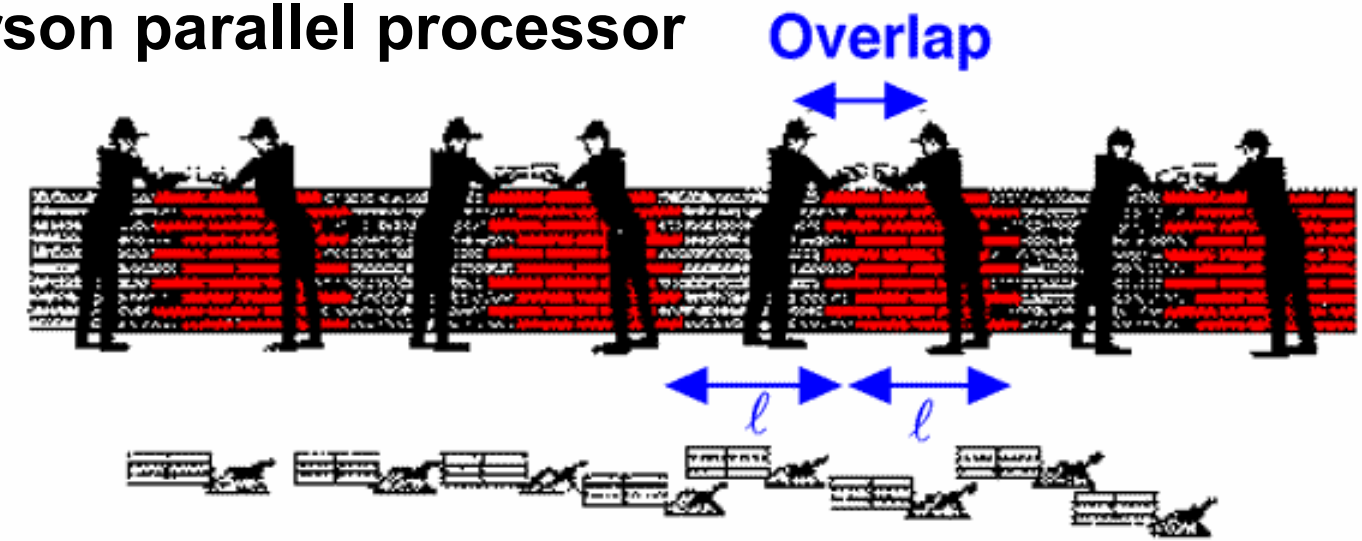
It's all well known ……

Parallel Processing and Society

- The fundamental principles behind the use of concurrent computers are identical to those used in society - in fact they are partly why society exists.
- If a problem is too large for one person, one does *not* hire a *SUPERman*, but rather puts together a team of ordinary people...
- cf. Construction of Hadrians Wall

Concurrent Construction of a Wall Using N = 8 Bricklayers *Decomposition by Vertical Sections*

8-person parallel processor

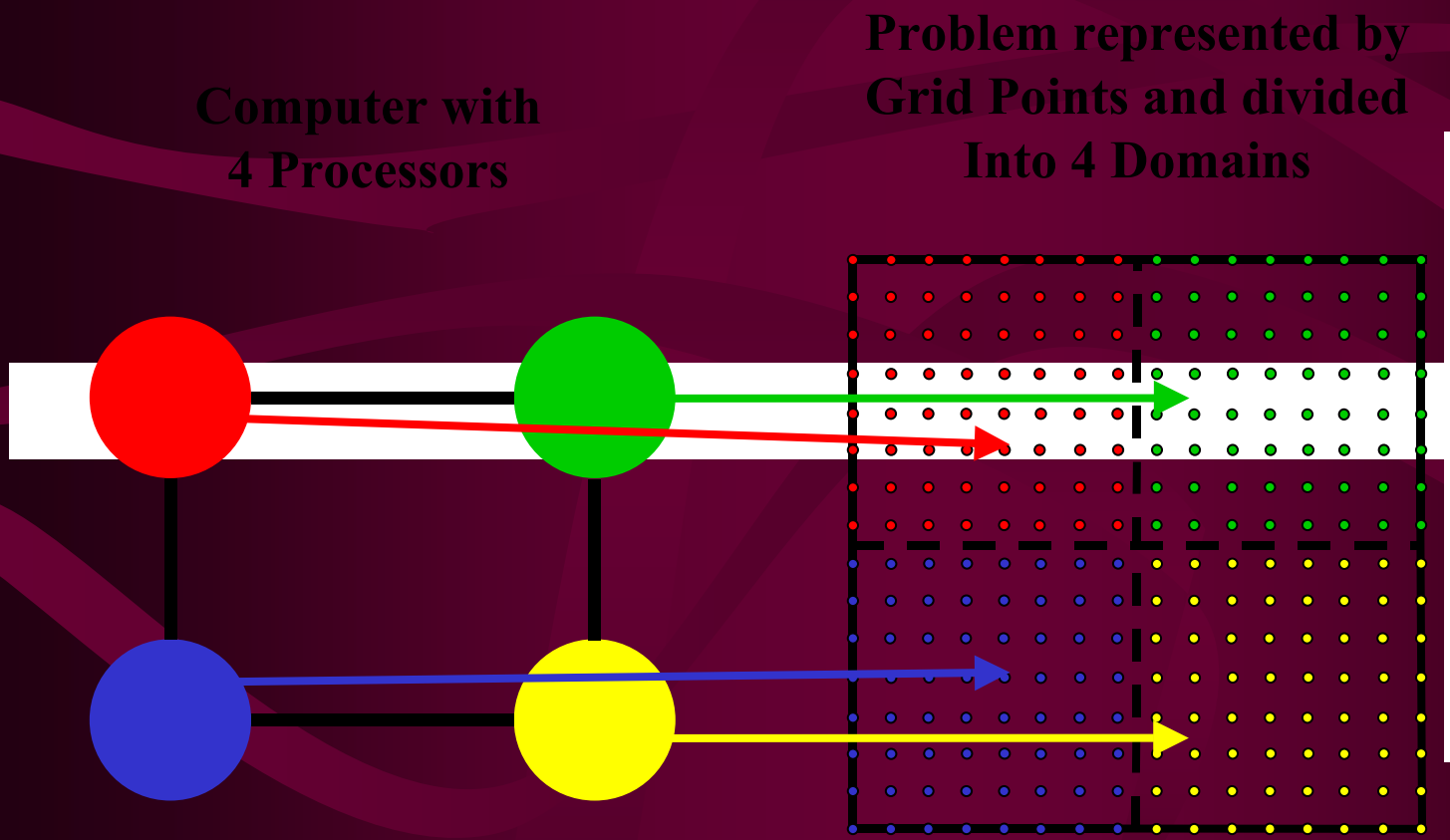


- **Domain Decomposition is Key to Parallelism**
  Need "Large" Subdomains $\ell >> \ell_{overlap}$

Divide problem into parts; one part for each processor

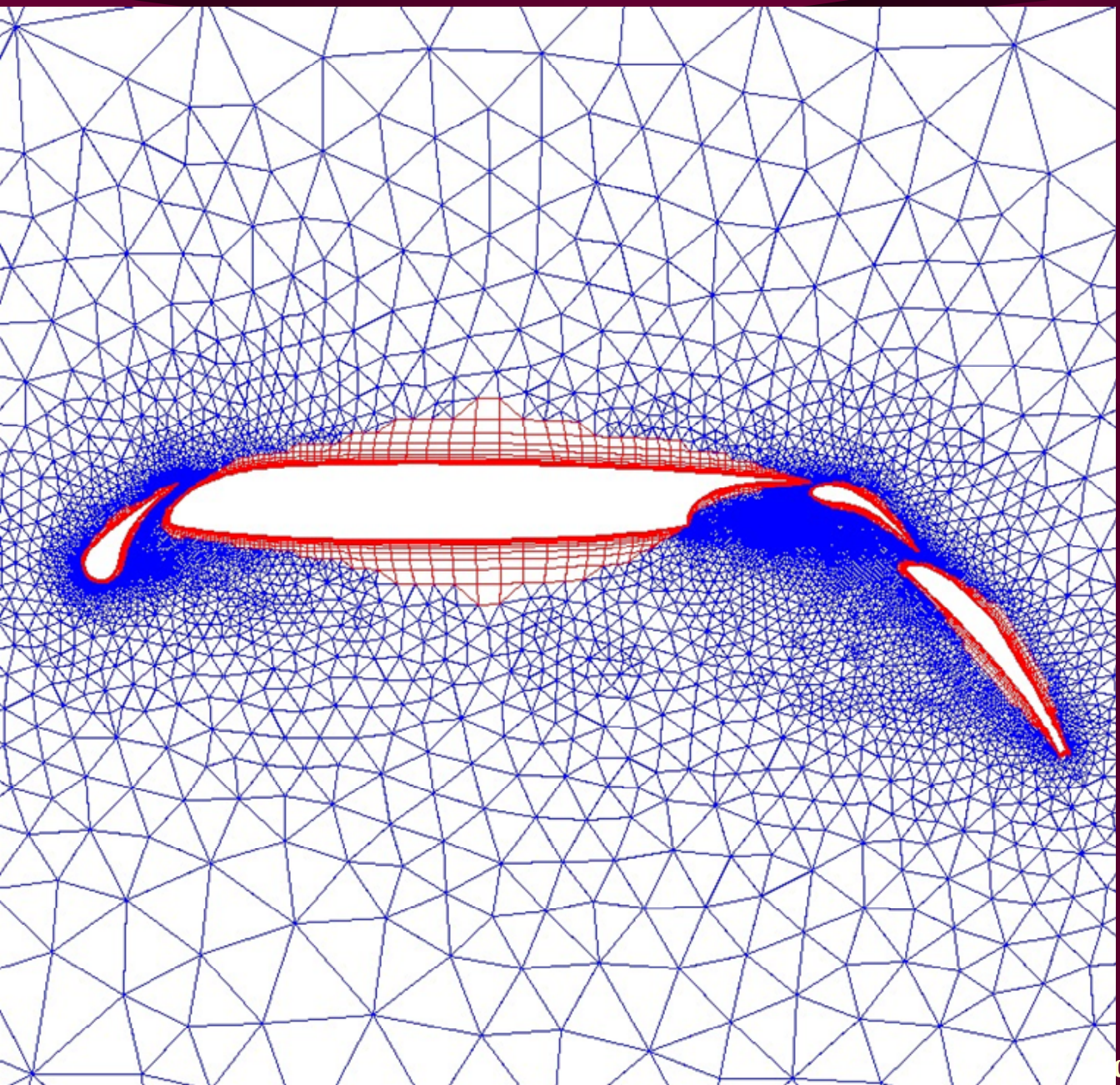
Seismic Simulation of Los Angeles Basin

- This is a (sophisticated) wave equation and you divide Los Angeles geometrically and assign roughly equal number of grid points to each processor



Divide surface into 4 parts and assign calculation of waves in each part to a separate processor

Irregular 2D Simulation -- Flow over an Airfoil



- The regular grid points become finite element mesh nodal points arranged as triangles filling space
- All the action (triangles) is near near wing boundary
- Use domain decomposition but no longer equal area as equal triangle count

Quantitative Speed-Up Analysis for Construction of Hadrian's Wall

- Quantitatively

S= Speed-up = N ε

efficiency

Number of Bricklayers

$\varepsilon \sim 1 - \text{constant} \ \frac{\ell_{overlap}}{\ell}$

ℓ = size (in metres) of wall assigned to each bricklayer

$\ell_{overlap}$ = overlap region

~ 6 metres in this case

Amdahl's Law of Parallel Processing

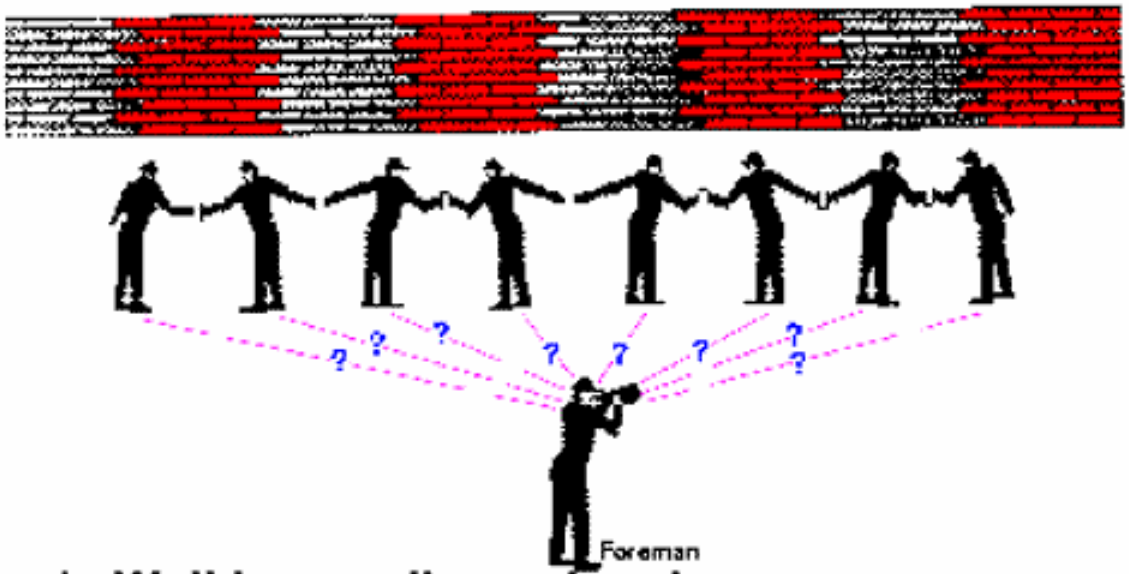
- Speedup S(N) is ratio Time(1 Processor)/Time(N Processors); we want S(N) ≥ 0.8 N
- Amdahl's law said no problem could get a speedup greater than about 10
- It is not correct as it was gotten by looking at wrong or small problems
- For Hadrian's wall S(N) satisfies our goal as long as l > about 60 meters *if* $l_{overlap}$ = about 6 meters
- If l is roughly same size as $l_{overlap}$ then we have "too many cooks spoil the broth syndrome"
  - One needs large problems to get good parallelism but only large problems need large scale parallelism

Pipelining --Another Parallel Processing Strategy for Hadrian's Wall

- *"Pipelining" or decomposition by horizontal section is:*
  - In general less effective
  - and leads to less parallelism
  - (N = Number of bricklayers must be < number of layers of bricks)

Exploit Aspect of problem which gives largest parallelism

Hadrian's Wall Illustrates that the Topology of Processor Must Include Topology of Problem



- Hadrian's Wall is one dimensional
- Humans represent a flexible processor node that can be arranged in different ways for different problems
- The lesson for computing is:
  Original MIMD machines used a hypercube topology. The hypercube includes several topologies including all meshes. It is a flexible concurrent computer that can tackle a broad range of problems. Current machines use different interconnect structure from hypercube but preserve this capability.

General Speed Up Analysis

- Comparing Computer and Hadrian's Wall Cases

$$\text{Speedup } S = \varepsilon N$$

$$\varepsilon = 1 - \frac{\text{constant}}{n^{1/d}} \cdot \frac{t_{comm}}{t_{calc}}$$

General	*Hadrian's Wall*
n = Grain Size	n = number of bricks laid by each mason $n \propto l$
d = Problem Dimension	d = 1 for a one dimensional wall (d = 2 for laying tiles on floor of Hadrian's Palace)
t_{calc} = Time to do each calculation	Time to lay one brick
t_{comm} = Time to communicate unit of information between nodes	Time to discuss/adjust brick laid at join between domains assigned adjacent masons

Nature's Concurrent Computers

- At the finest resolution, collection of neurons sending and receiving messages by axons and dendrites
- At a coarser resolution
  Society is a collection of brains sending and receiving messages by sight and sound
- Ant Hill is a collection of ants (smaller brains) sending and receiving messages by chemical signals
- **Lesson: All Nature's Computers Use Message Passing**
- **With several different Architectures**

Neural Network

The Web is also just message passing

Comparison of The Complete Problem to the subproblems formed in domain decomposition

For Hadrian's Wall. the complete problem:

is similar to the subtask performed by an individual bricklayer

Changed is

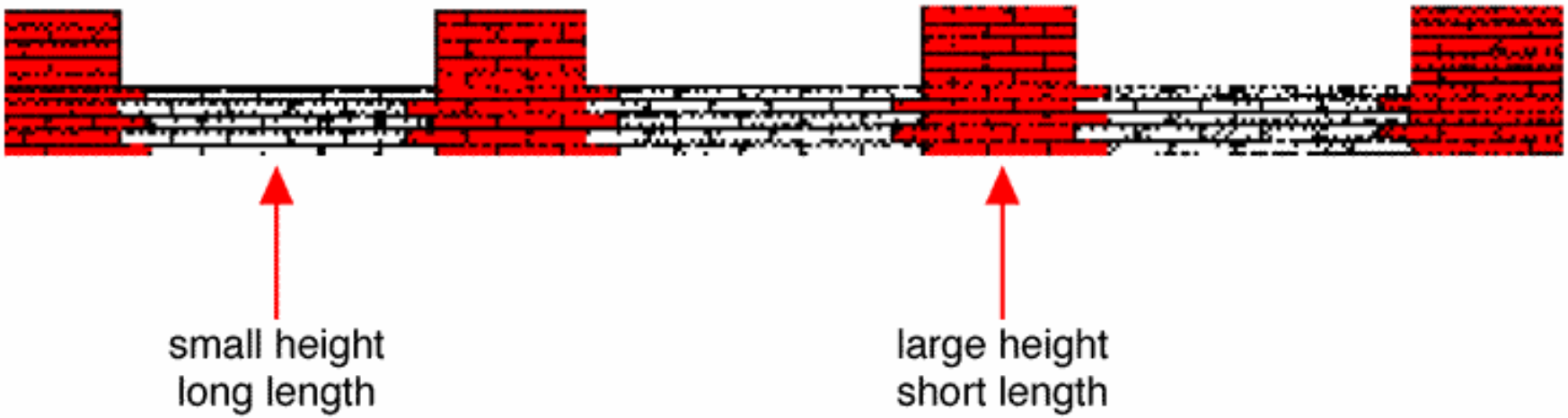
- Geometry
- Boundary Conditions

- *The case of Programming a Hypercube*
  - Each node runs software that is similar to sequential code
  - e.g., FORTRAN with geometry and boundary value sections changed

1984 Slide – today replace hypercube by cluster

Hadrian's Wall Illustrating an Irregular but Homogeneous Problem

- Geometry irregular but each brick takes about the same amount of time to lay.
- Decomposition of wall for an irregular geometry involves equalizing number of bricks per mason, not length of wall per mason.



Equal work not
Equal area of underlying domain
is load balancing requirement

Some Problems are Inhomogeneous Illustrated by:

An Inhomogeneous Hadrian Wall with Decoration

- Fundamental entities (bricks, gargoyles) are of different complexity
- Best decomposition dynamic

- Inhomogeneous problems run on concurrent computers but require dynamic assignment of work to nodes and strategies to optimize this
- (we use neural networks, simulated annealing, spectral bisection etc.)

Global and Local Parallelism Illustrated by Hadrian's Wall

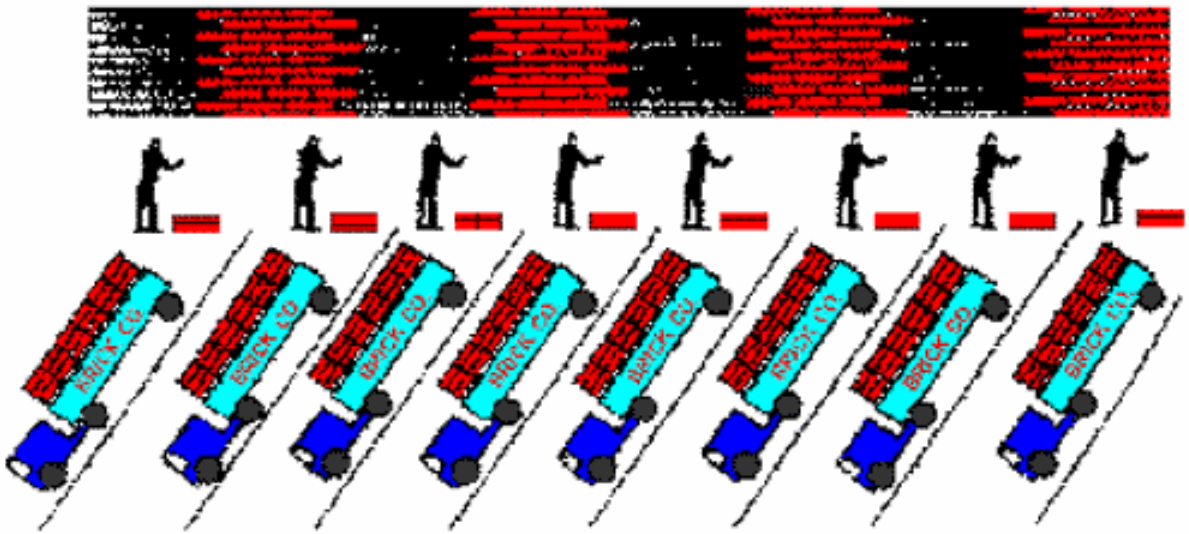
- Global Parallelism
  - Break up domain
  - Amount of Parallelism proportional to size of problem (and is usually large)
  - Unit is Bricklayer or Computer node
- Local Parallelism
  - Do in parallel local operations in the processing of basic entities
    - e.g. for Hadrian's problem, use two hands, one for brick and one for mortar while ...
    - for computer case, do addition at same time as multiplication
  - Local Parallelism is limited but useful
- Local and Global Parallelism Should both be Exploited

Between CPU's
Called Outer Parallelism

Inside CPU or Inner Parallelism

Parallel I/O Illustrated by Concurrent Brick Delivery for Hadrian's Wall

Bandwidth of Trucks and Roads Matches that of Masons



- Disk (input/output) Technology is better matched to several modest power processors than to a single sequential supercomputer
- Concurrent Computers natural in databases, transaction analysis

And today Sensors

Comparison of Concurrent Processing in Society and Computing

- Problems are large - use domain decomposition
  Overheads are edge effects
- Topology of processor matches that of domain - processor with rich flexible node/topology matches most domains
- Regular homogeneous problems easiest but
  irregular or Inhomogeneous } work with proper decomposition/planning
- Can use local and global parallelism
- Can handle concurrent calculation and I/O
- Nature always uses message passing as in parallel computers (at lowest level)

Technology Driving Forces

The commodity Stranglehold

TOP 500 from Dongarra, Meuer, Simon, Strohmaier

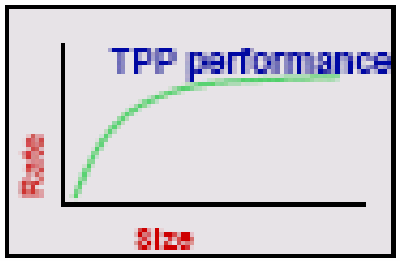
- http://www.top500.org

TOP500 SUPERCOMPUTER

H. Meuer, H. Simon, E. Strohmaier, & JD

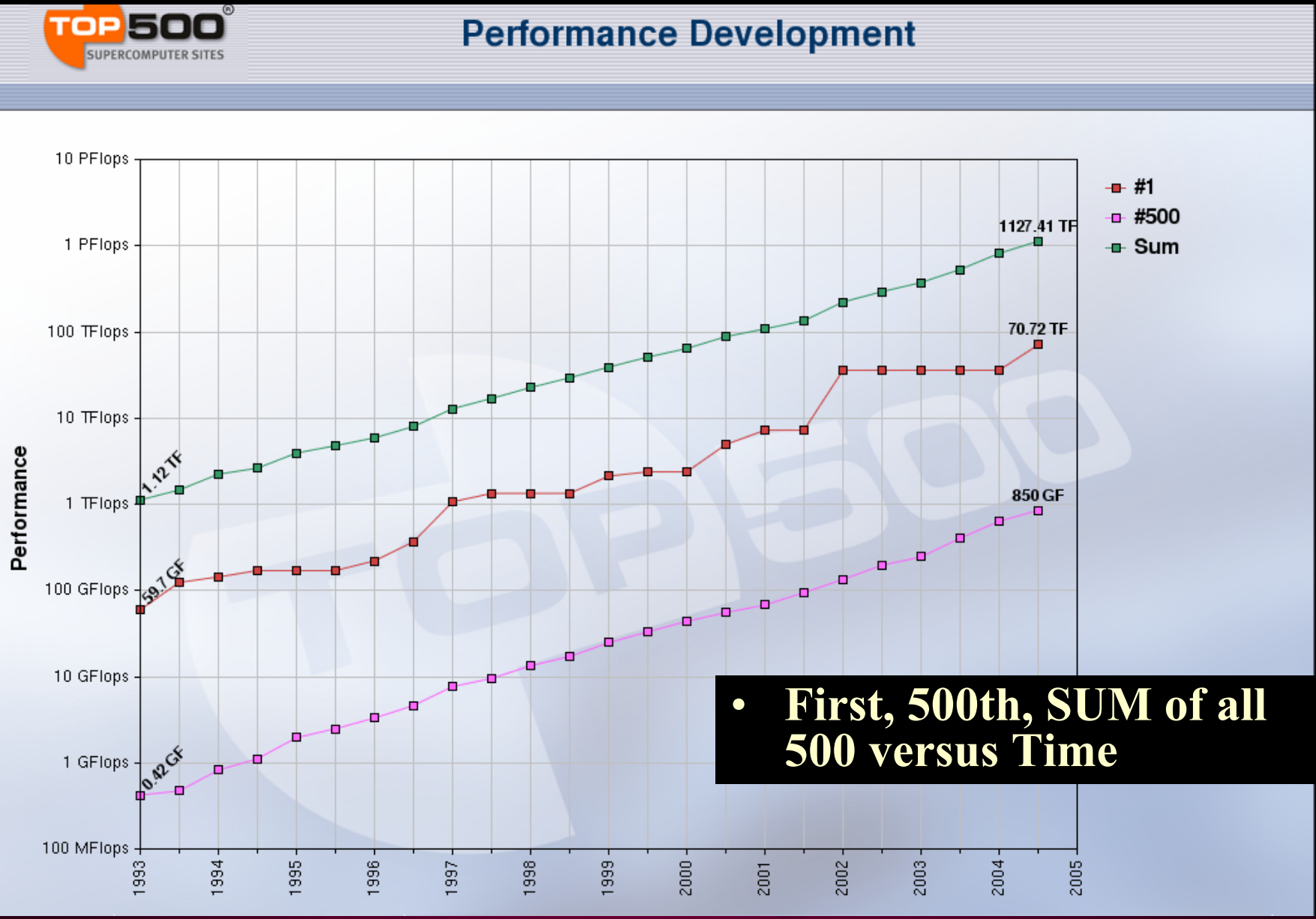
- Listing of the 500 most powerful Computers in the World
- Yardstick: Rmax from LINPACK MPP

$Ax=b$, *dense problem*

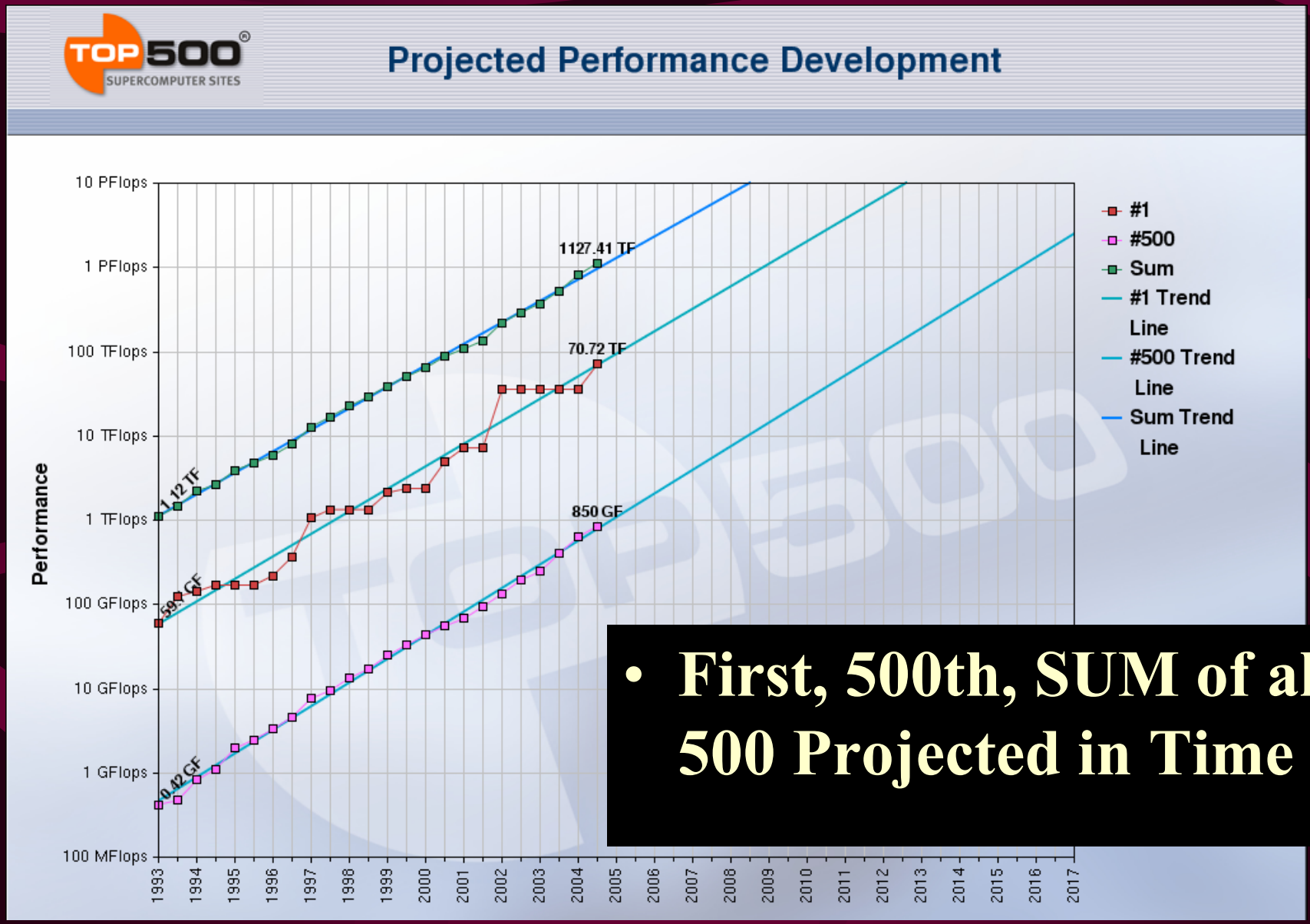


- Updated twice a year
  - SC'xy in the States in November
  - Meeting in Mannheim, Germany in June
- All data available from www.top500.org

Top 500 Performance versus time 93-99

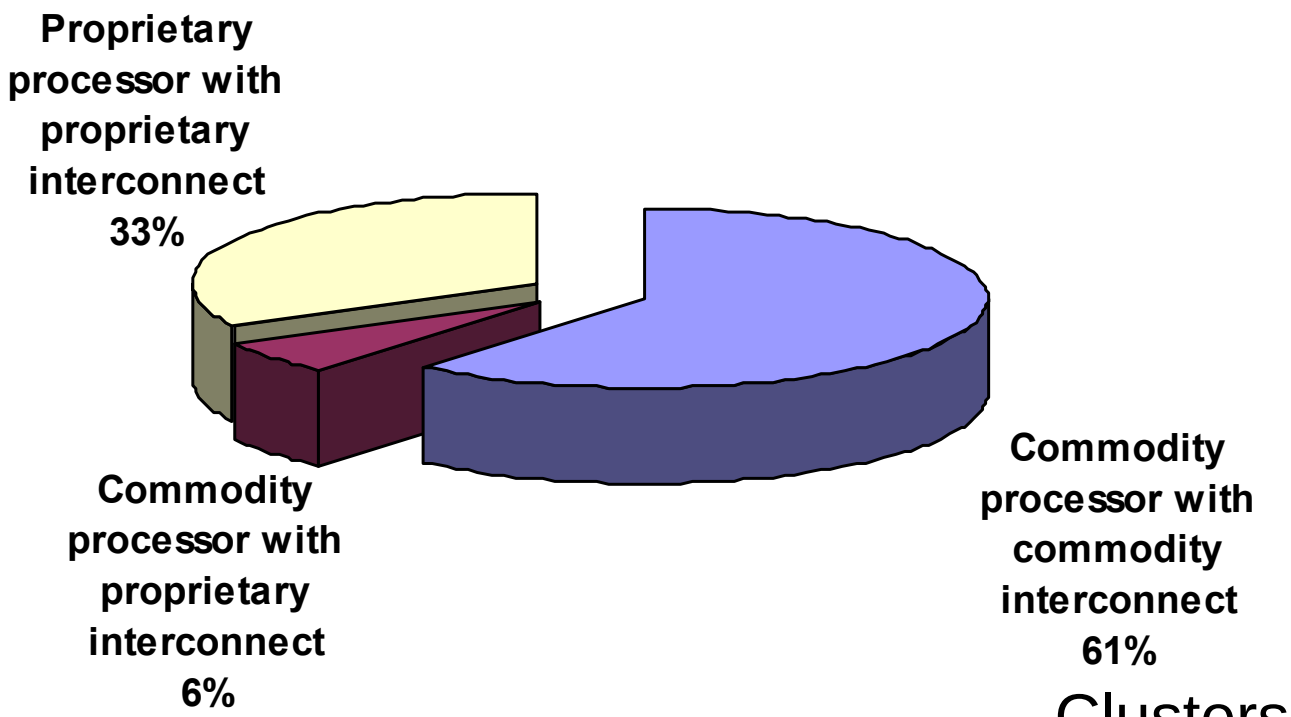


Projected Top 500 Until Year 2012



- First, 500th, SUM of all 500 Projected in Time

Architecture of Top 500 Computers



Architecture/Systems Continuum

Loosely Coupled

Tightly Coupled

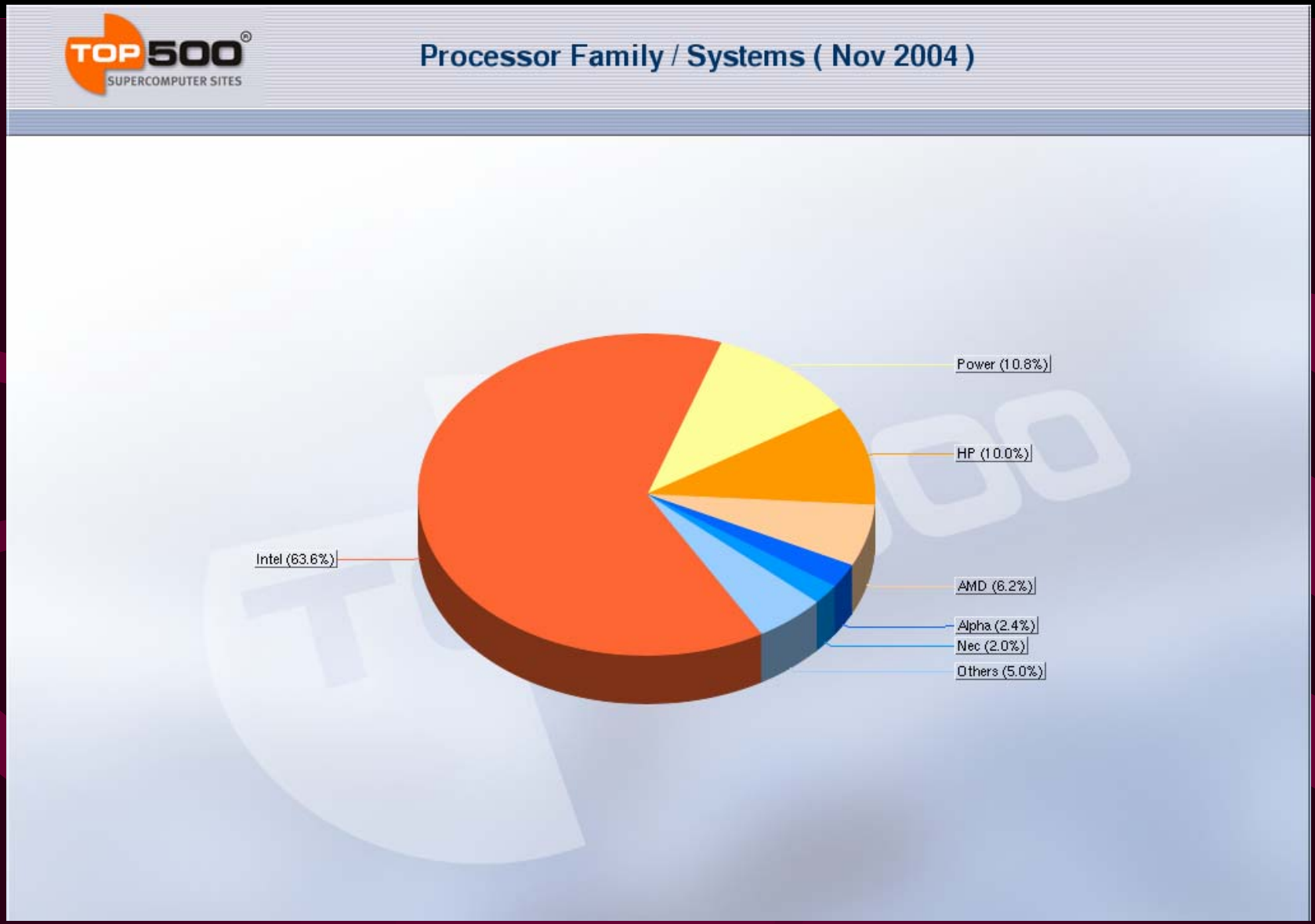
- **Commodity processor with commodity interconnect**
  - Clusters
    - Pentium, Itanium, Opteron, Alpha, PowerPC
    - GigE, Infiniband, Myrinet, Quadrics, SCI
  - NEC TX7
  - HP Alpha
  - Bull NovaScale 5160
- **Commodity processor with custom interconnect**
  - SGI Altix
    - Intel Itanium 2
  - Cray Red Storm
    - AMD Opteron
  - IBM Blue Gene/L (?)
    - IBM Power PC
- **Custom processor with custom interconnect**
  - Cray X1
  - NEC SX-7
  - IBM Regatta

JD2

check bgl status

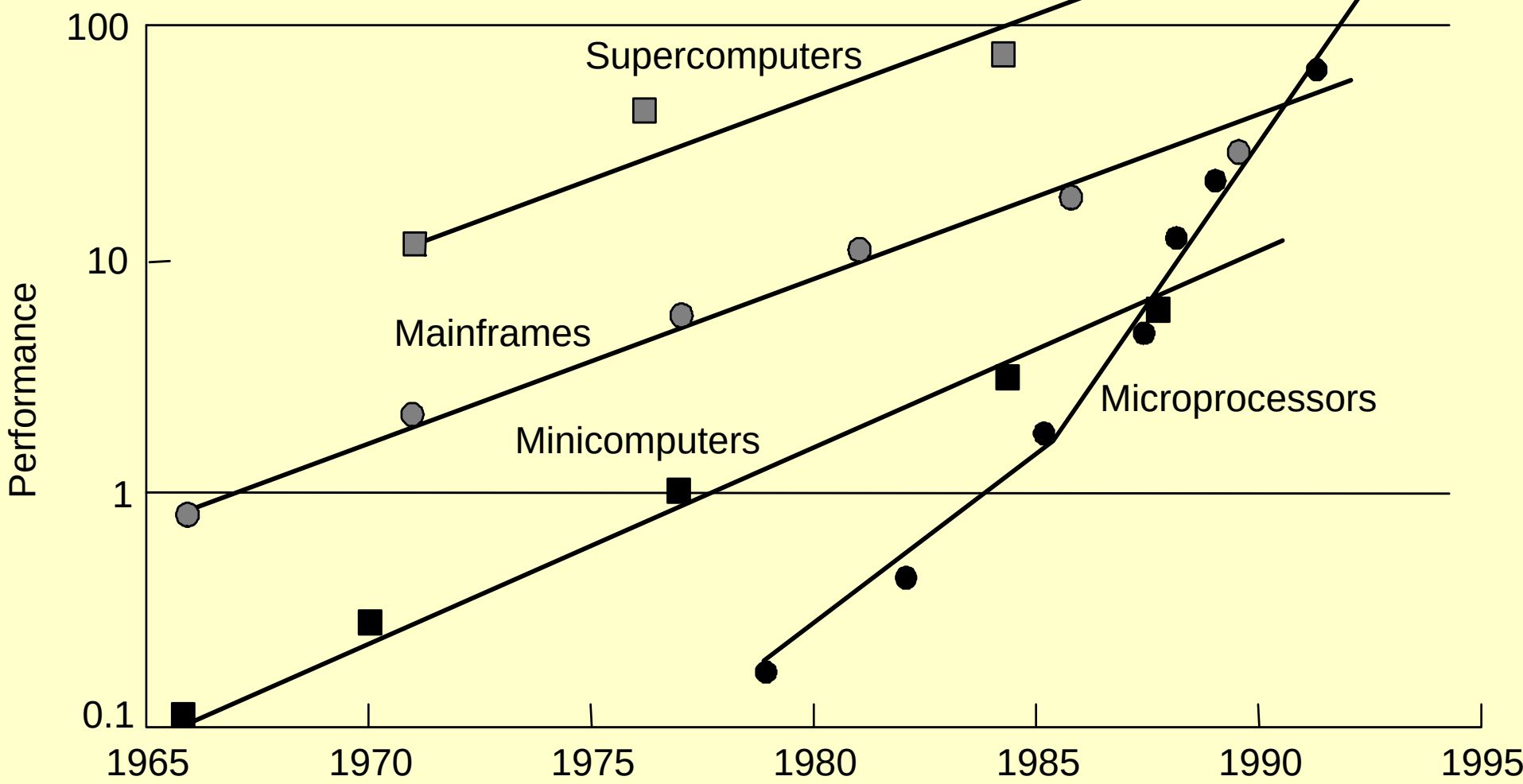
Jack Dongarra, 4/15/2004

CPU Chips of the TOP 500

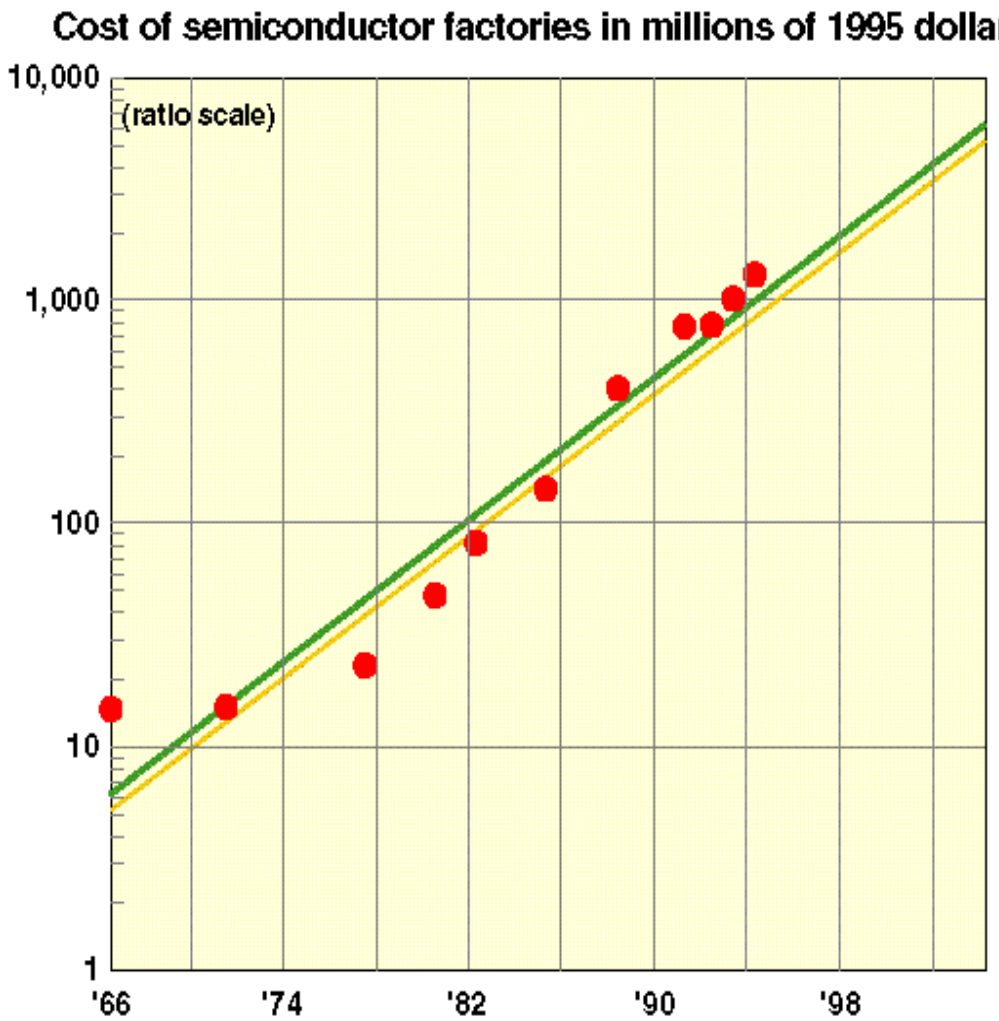


Technology Trends -- CPU's

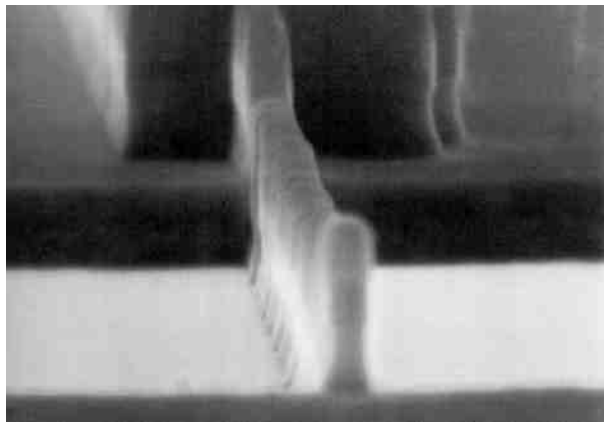
The natural building block for multiprocessors is now also the fastest! We don't make these plots any more



But there are limiting forces: Increased cost and difficulty of manufacturing



- **Moore's 2nd law (Rock's law)**



Demo of 0.06 micron CMOS

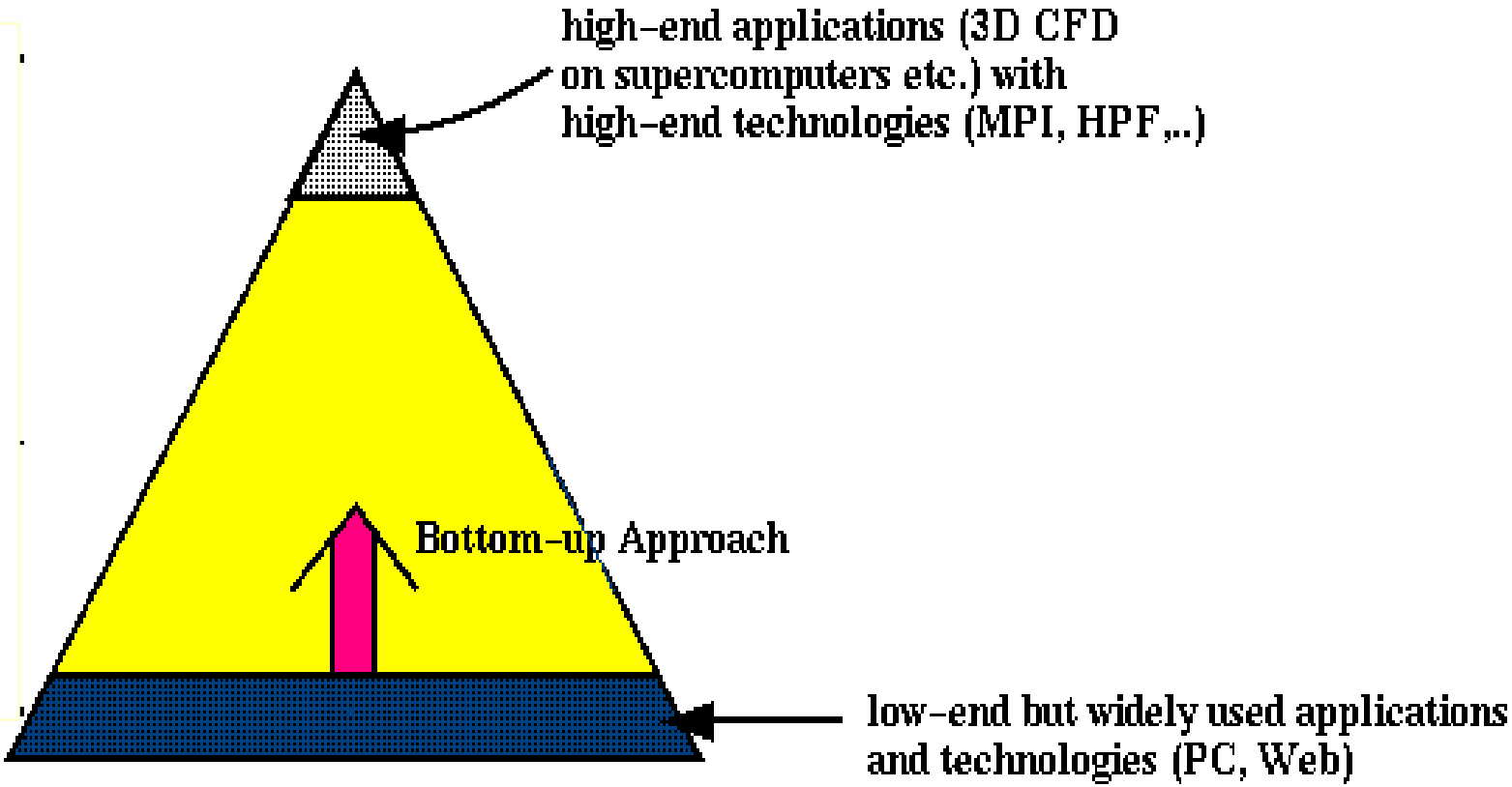
January 11 2005: Intel expects to spend $5B on new manufacturing equipment in 2005

CPU Technology

- 10-20 years ago we had many competing
  - CPU **Architectures** (Designs)
  - CPU **Base Technology** (Vacuum Tubes, ECL, CMOS, GaAs, Superconducting) either used or pursued
- Now all the architectures are about the same and there is only one viable technology – **CMOS**
  - Some approaches are just obsolete
  - Some (**superconducting**) we don't see how to make realistic computers out of
  - Others (**Gallium Arsenide**) might even be better but we can't afford to deploy infrastructure to support

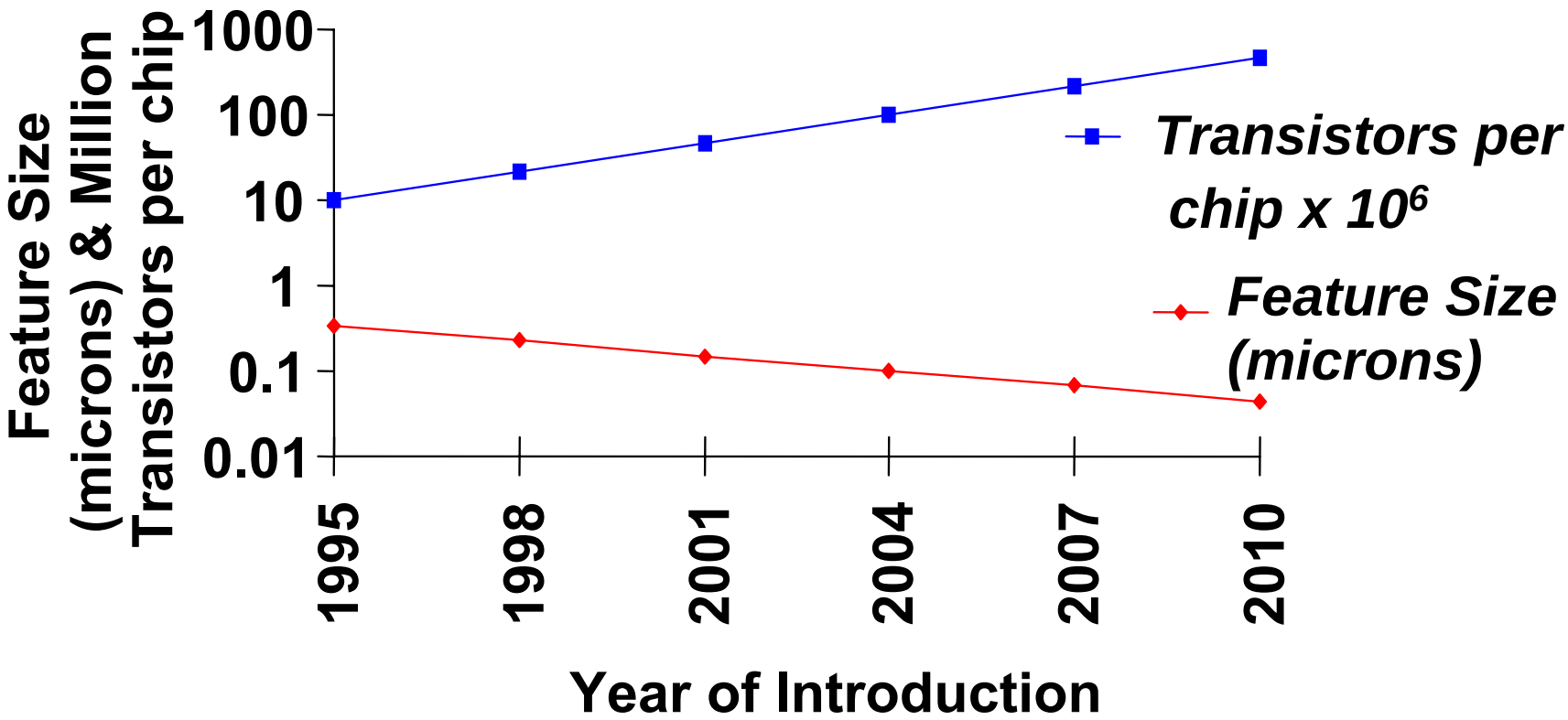
The Computing Pyramid

- Bottom of Pyramid has 100 times dollar value and 1000 times compute power of best supercomputer
- This motivates cluster computing and peer to peer (P2P) projects like SETI@Home



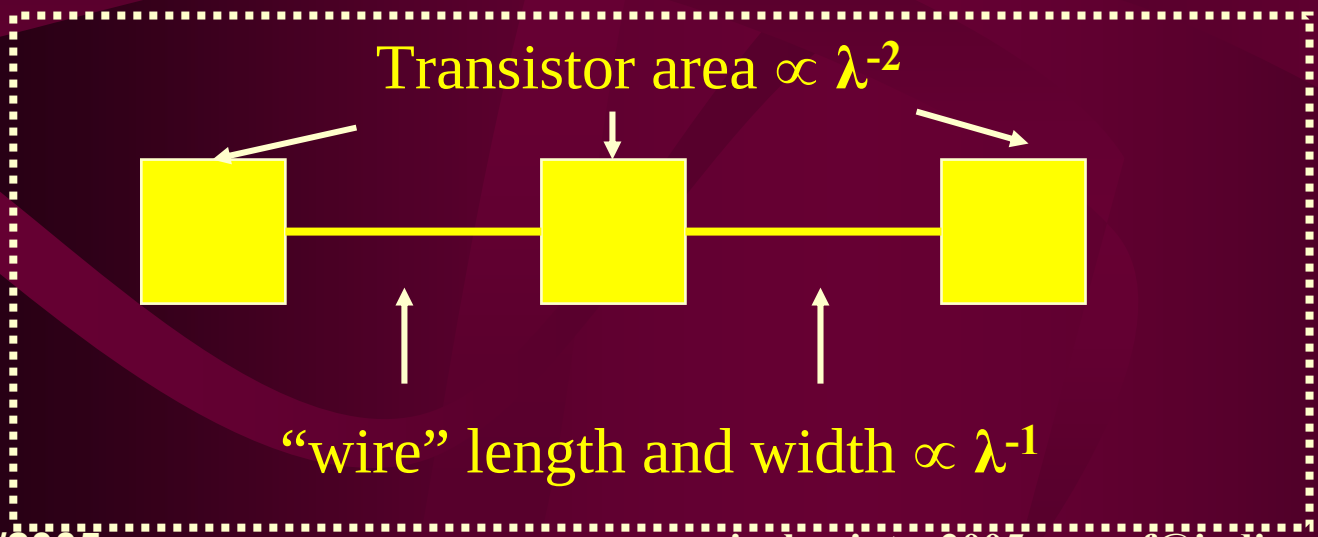
SIA Projections for Microprocessors

Compute power ~1/(λ = Feature Size)$^{3 \text{ to } 4}$



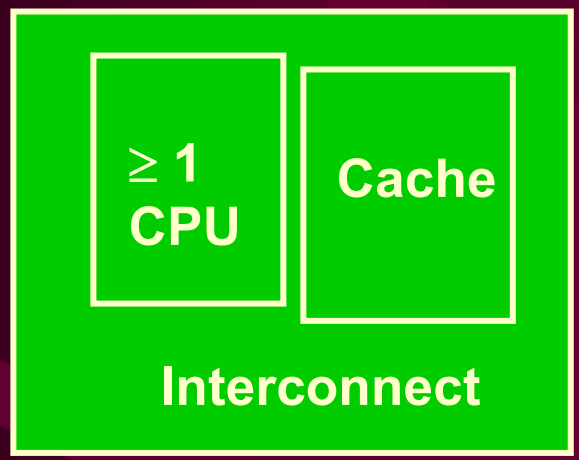
Chip Evolution I

- Basic driver of performance advances is decreasing feature size (λ); Circuits become either faster or lower in power
- The linear size of chips is growing too roughly like λ^{-1}
  - (area like λ^{-1})
- Clock rate improves roughly proportional to improvement in λ^{-1} (slower as speed decreases)
- Number of transistors improves like λ^{-2} (or faster like λ^{-3} as chip size increases)
- In total Performance grows like λ^{-4}

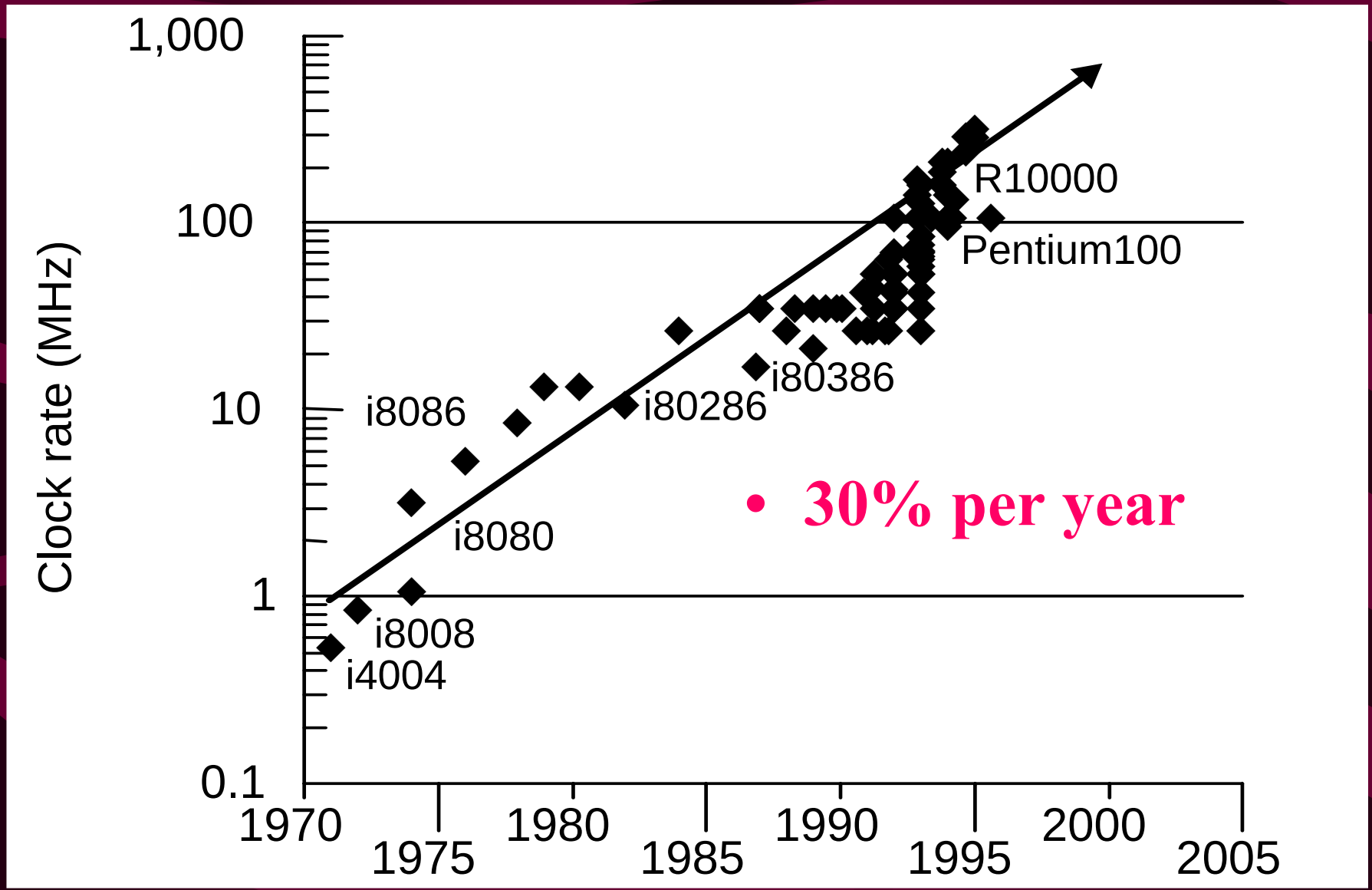


Chip Evolution II

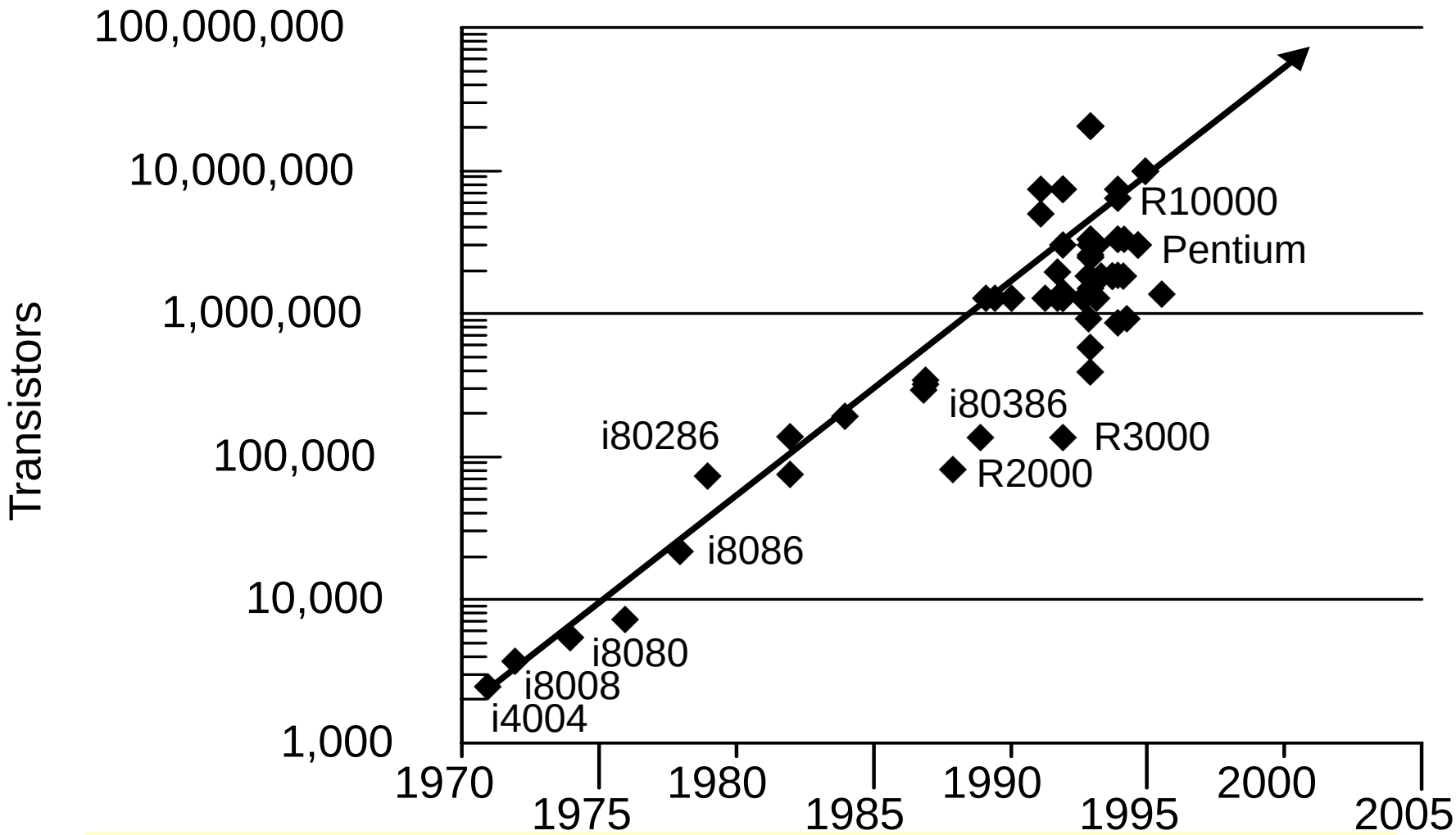
- Performance increases about 500x per decade; clock rate <10x, rest of increase is due to transistor count
- Current microprocessor transistors are used: 1/3 compute, 1/3 cache (on chip memory), 1/3 off-chip connect



Clock Frequency Growth Rate



Transistor Count Growth Rate

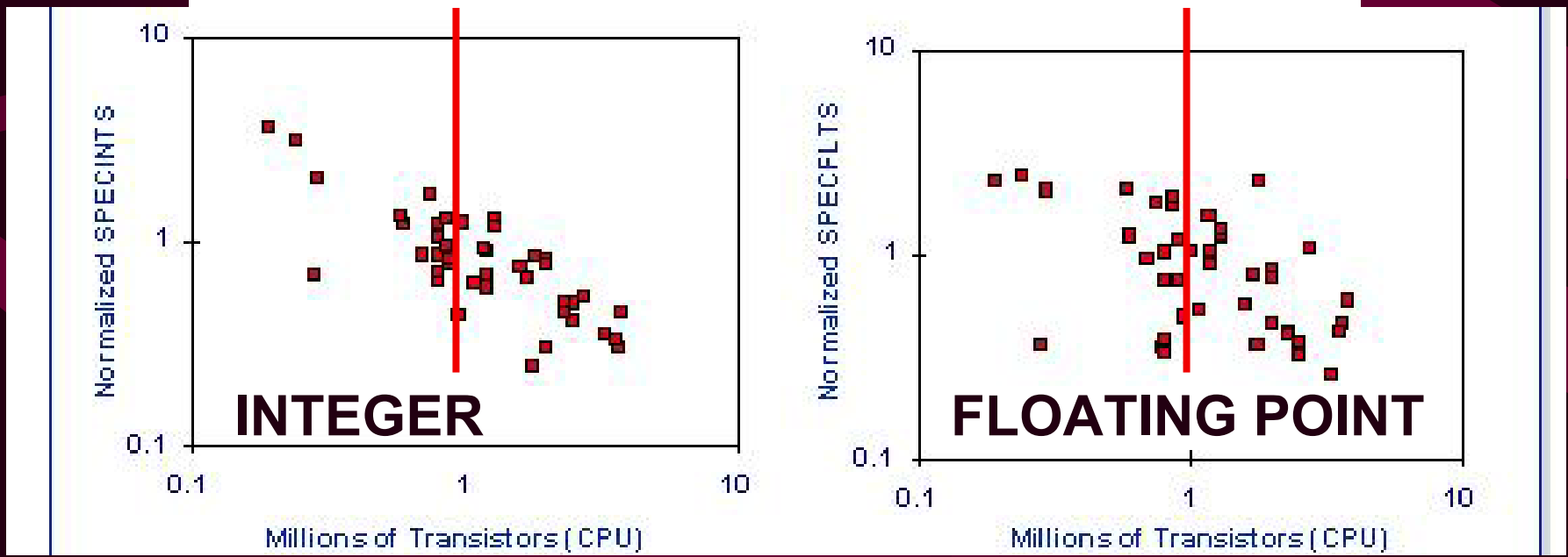


- **125 million transistors on Intel Prescott Chip Feb 2 2004.**
- **Transistor count grows faster than clock rate**
  - **Factor of 2 every 1.5 years is Moore's Law**

Architecture and Transistor Count

- When "real" microprocessor chips (1980) first appeared, they had < 50,000 transistors and there were simply not enough transistors to build a good architecture
- Once we reached around one million transistors (1988), we could build a good architecture and CMOS chips started to dominate computer market

Good Basic Architecture at Million Transistors



DRAM Expectations from SIA

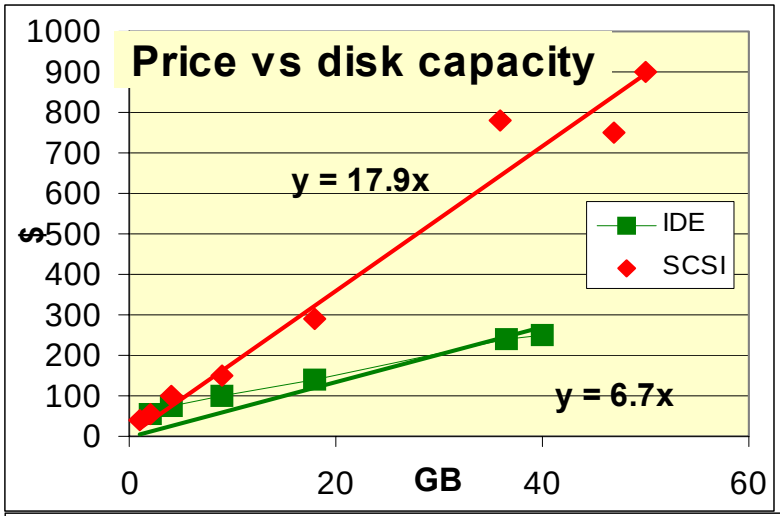
http://www.itrs.net/Common/2004Update/2004Update.htm

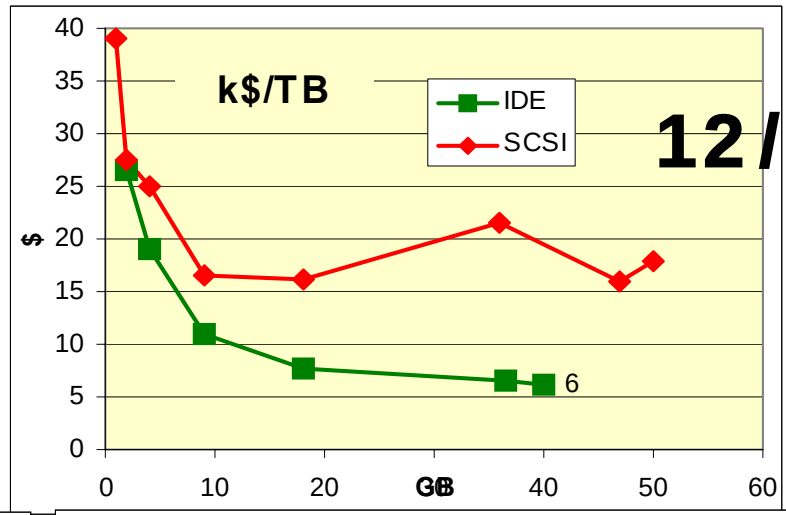
Table 1c DRAM Production Product Generations and Chip Size Model—Near-term Years

Year of Production	*2003*	*2004*	*2005*	*2006*	*2007*	*2008*	*2009*
Technology Node		*hp90*			*hp65*		
DRAM ½ Pitch (nm)	*100*	*90*	*80*	*70*	*65*	*57*	*50*
MPU/ASIC Metal 1 (M1) ½ Pitch (nm)	**120**	**107**	**95**	**85**	**76**	**67**	**60**
MPU/ASIC ½ Pitch (nm) (Un-contacted Poly)	**107**	**90**	**80**	**70**	**65**	**57**	**50**
MPU Printed Gate Length (nm) ††	**65**	**53**	**45**	**40**	**35**	**32**	**28**
MPU Physical Gate Length (nm)	**45**	**37**	**32**	**28**	**25**	**22**	**20**
Cell area factor [a]	**8**	**8**	**7.5**	**7**	**7**	**6**	**6**
Cell area [Ca = af^2] (mm^2)	**0.082**	**0.065**	**0.048**	**0.036**	**0.028**	**0.019**	**0.015**
Cell array area at production (% of chip size) §	**63.00%**	**63.00%**	**63.00%**	**63.00%**	**63.00%**	**63.00%**	**63.00%**
Generation at production §	**1G**	**1G**	**1G**	**2G**	**2G**	**4G**	**4G**
Functions per chip (Gbits)	**1.07**	**1.07**	**1.07**	**2.15**	**2.15**	**4.29**	**4.29**
Chip size at production (mm^2)§	**139**	**110**	**82**	**122**	**97**	**131**	**104**
Gbits/cm^2 at production §	**0.77**	**0.97**	**1.31**	**1.76**	**2.22**	**3.27**	**4.12**

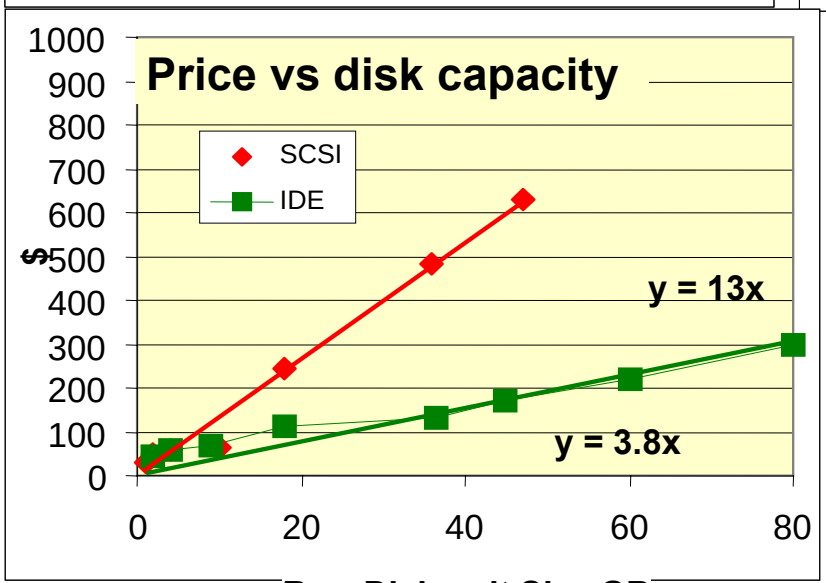
The Cost of Storage about 1K$/TB

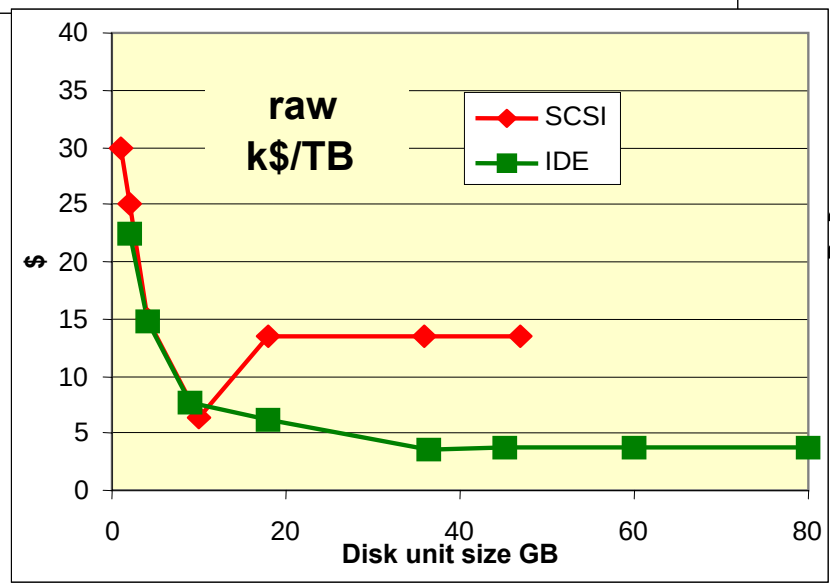
Jim Gray Microsoft





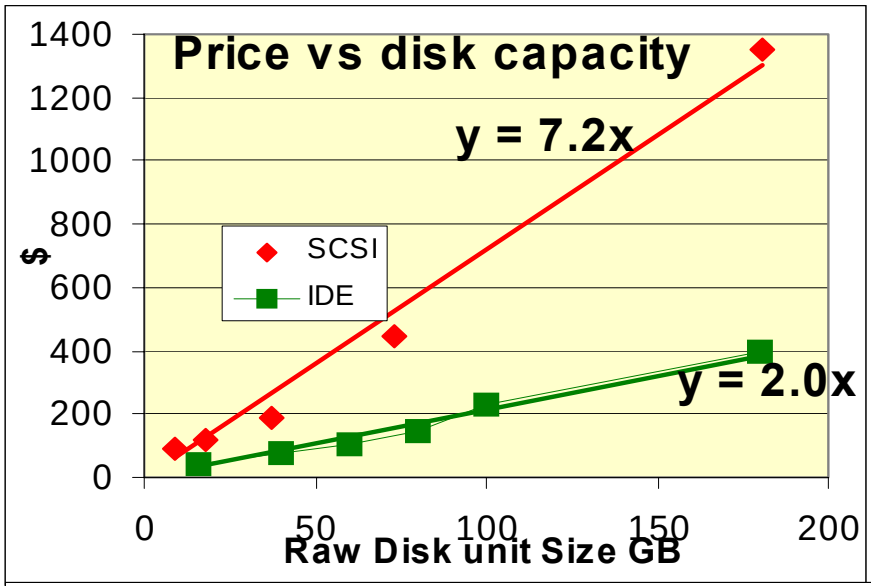
12/1/1999

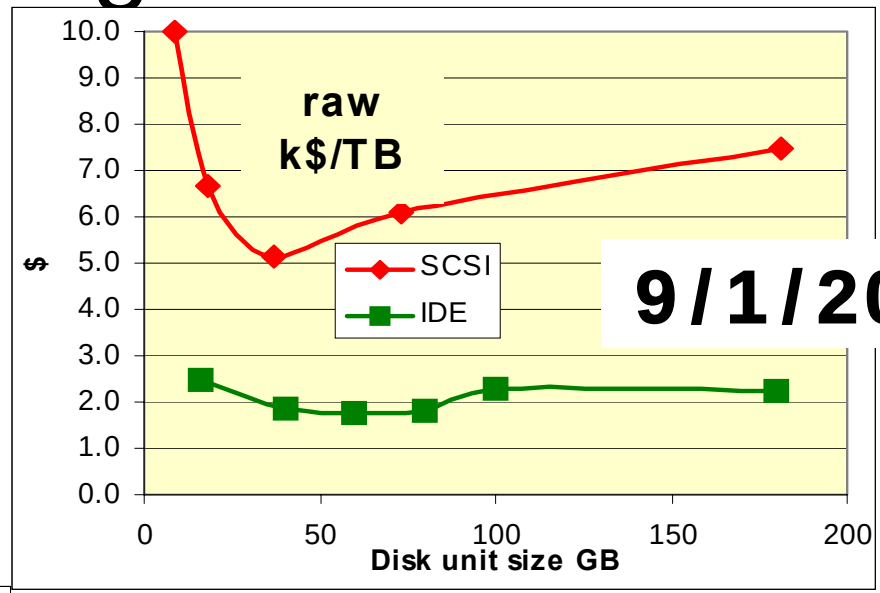




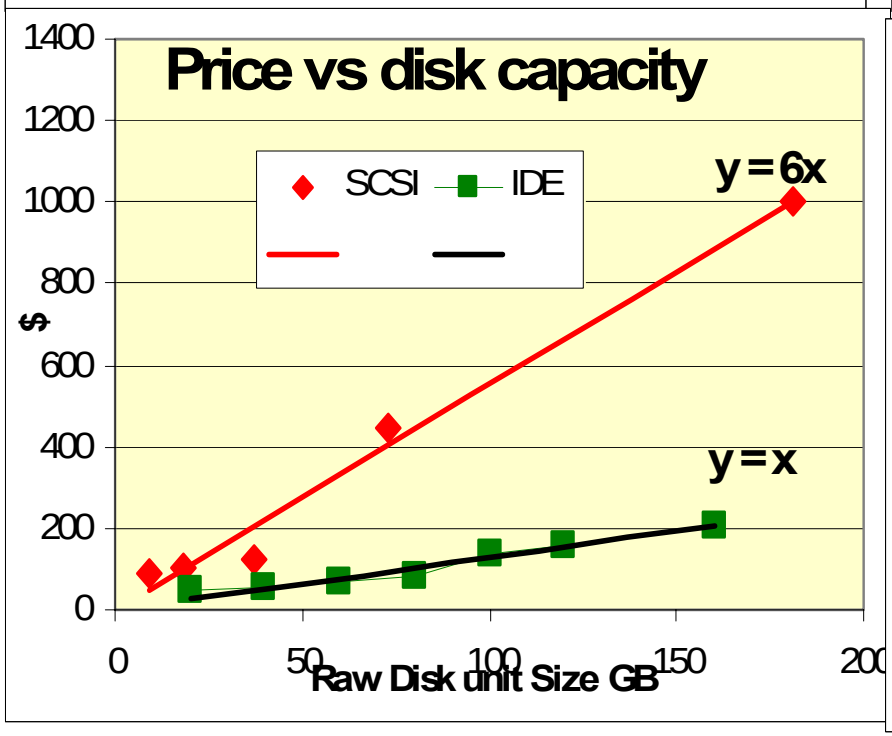
2000

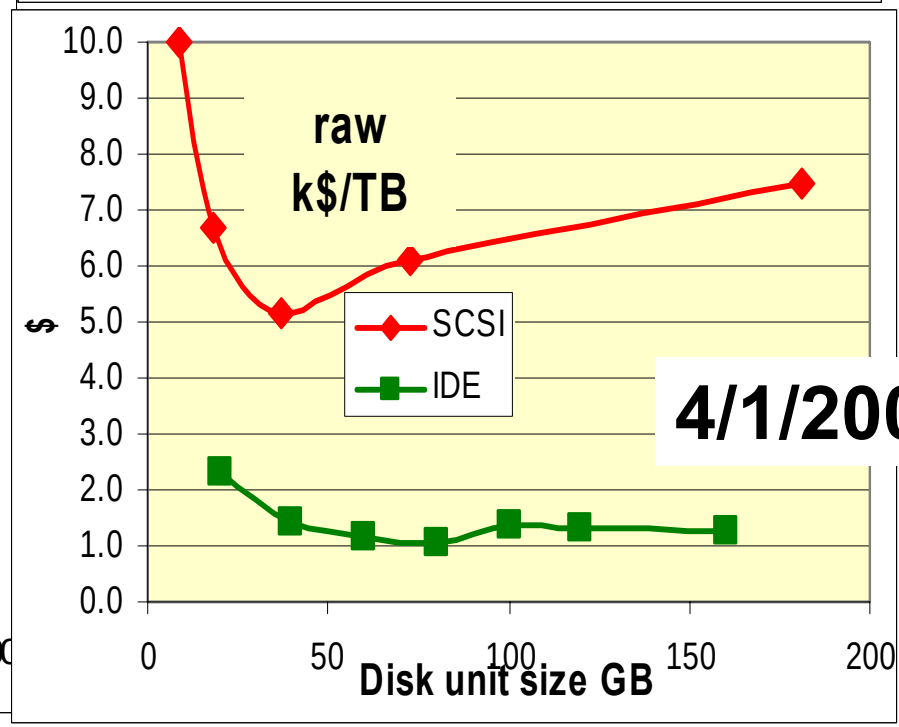
The Cost of Storage about 1K$/TB





9/1/2001





4/1/2002

Kilo

Mega

Giga

Tera

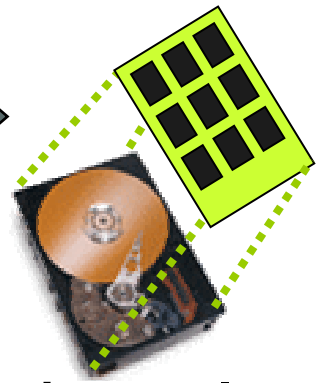
Peta

Exa

Zetta

Yotta

Disk Evolution



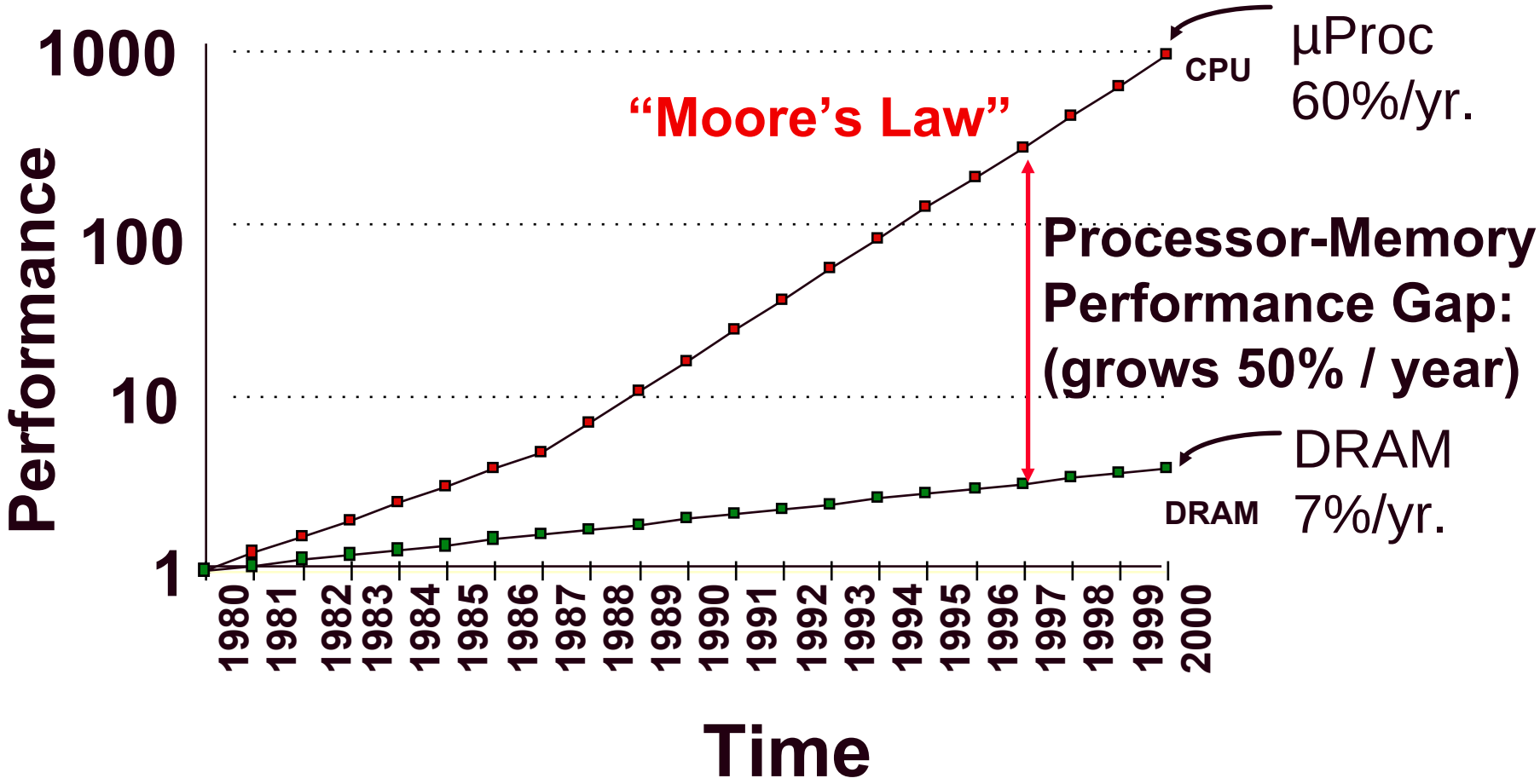
- Capacity:100x in 10 years
  1 TB 3.5” drive in 2005
  20 TB? in 2012?!
- System on a chip
- High-speed SAN
- Disk replacing tape
- Disk is super computer!

Importance of Memory Structure in High Performance Architectures

- Actually **memory bandwidth** is an essential problem in any computer as doing more computations per second requires accessing more memory cells per second!
  - Harder for **sequential** than **parallel** computers
- Key limit is that memory gets slower as it gets larger and one tries to keep information as near to CPU as possible (in necessarily small size storage)
- **This Data locality** is unifying concept of caches (sequential) and parallel computer multiple memory accesses
- Problem seen in extreme case for **Superconducting** CPU's which can be 100X current CPU's but seem to need to use conventional memory

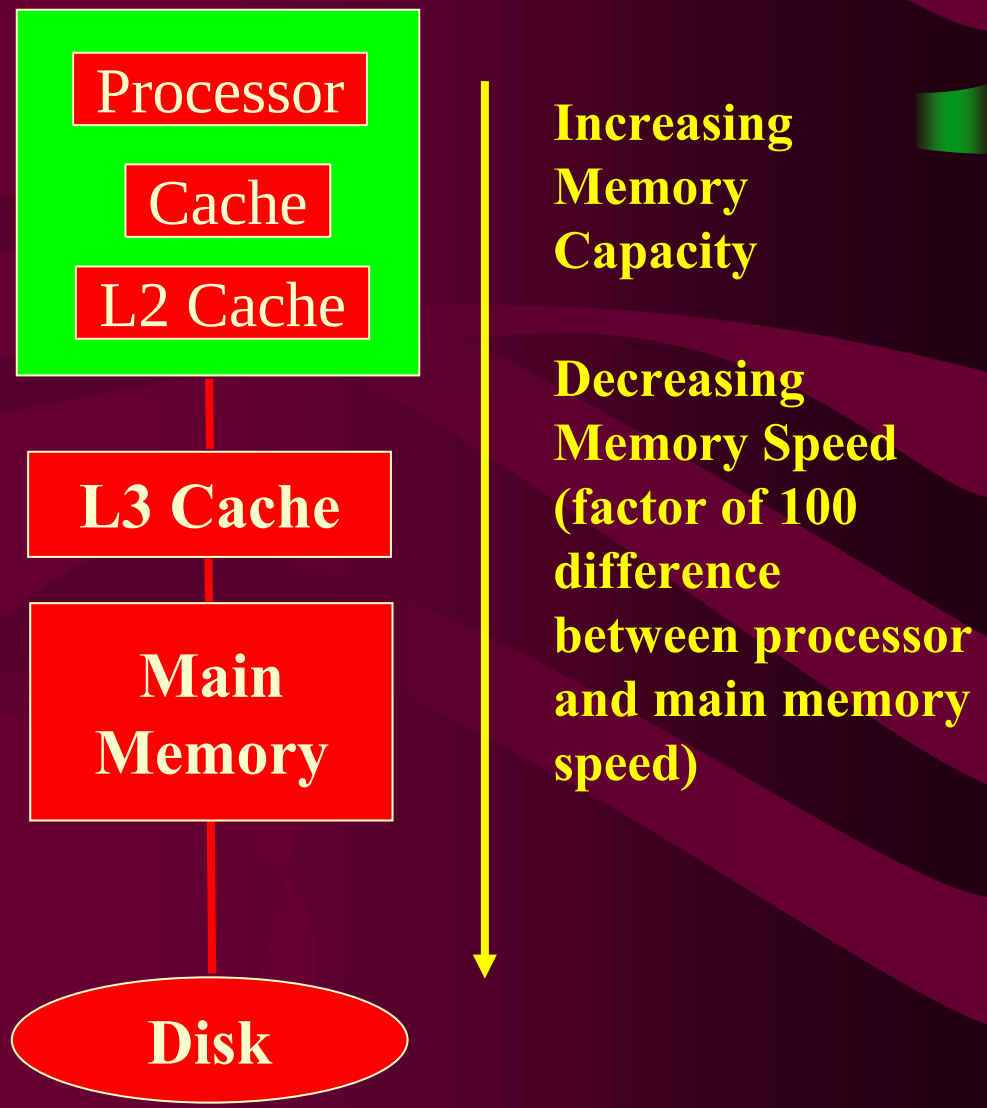
Processor-DRAM Growing Performance Gap (latency)

- This implies need for complex memory systems to hide memory latency

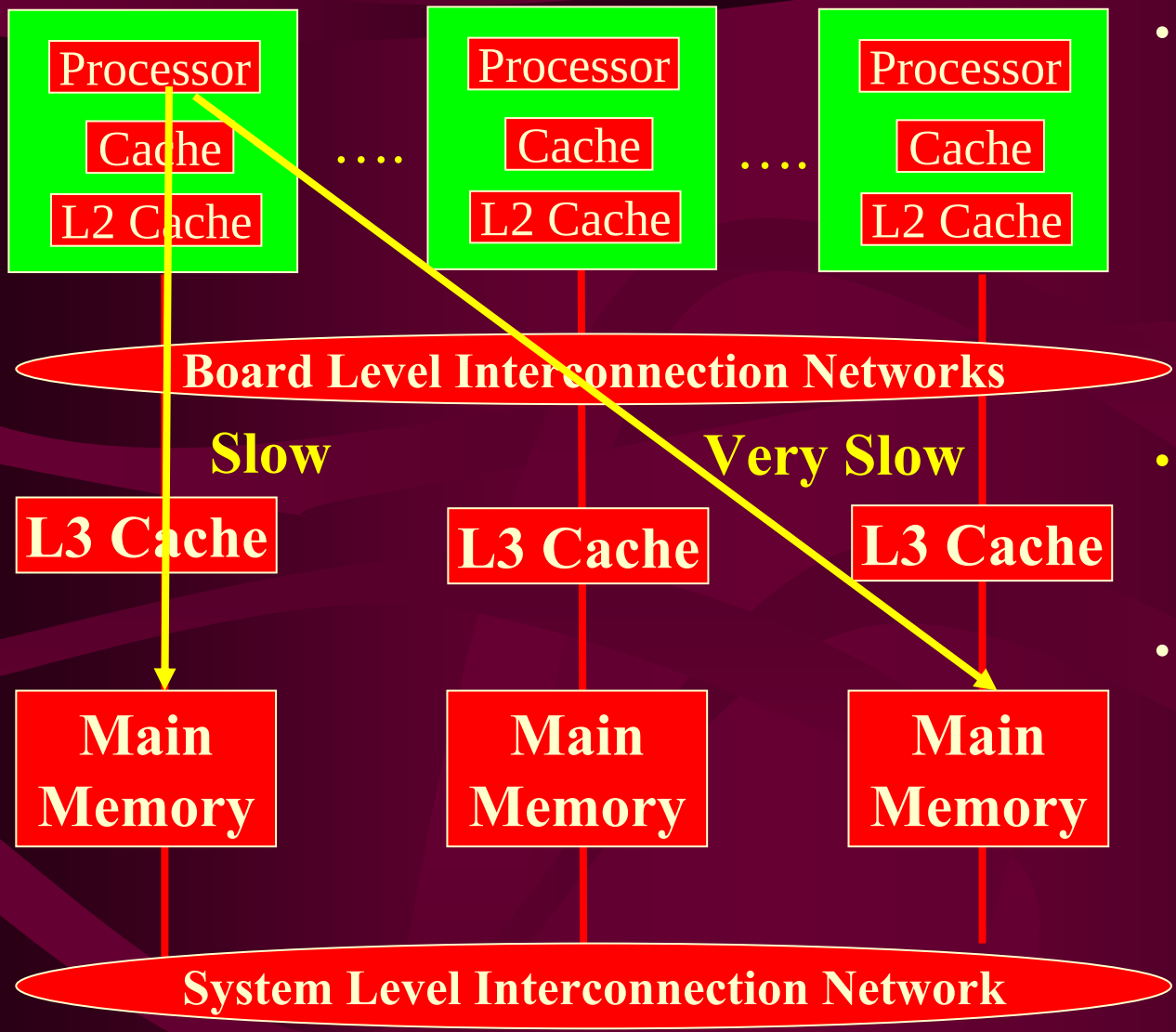


Sequential Memory Structure

- **Data locality** implies CPU finds information it needs in cache which stores most recently accessed information
- This means one **reuses a given memory reference** in many nearby computations e.g.
- A1 = B*C
- A2 = B*D + B*B
- …. Reuses B
- The more one uses any value fetched from memory, the higher the performance you will get



Parallel Computer Memory Structure



- For both parallel and sequential computers, cost is **accessing remote memories** with some form of "communication"
- **Data locality** addresses in both cases
- Differences are **quantitative size** of effect and what is done by user and what **automatically**

The cost of Accessing Data

- Both parallel and sequential computers must face the cost of data access and need for data locality
- Taking a 3 Ghz CPU, it does 3 operations every 10^{-9} seconds
  - Ignore multiple operations per clock cycle; makes memory-CPU gap worse
- Delay in fetching from data from memory is about 300 CPU operations
  - It can get several nearby data values simultaneously and so fetches blocks hoping you want nearby data
  - Data in on chip registers and cache is "instantaneous"
- Time to transfer data between 2 CPU's on a very optimized (expensive) parallel machine is about 3000 or more CPU operations
- Time to transfer data between 2 CPU's on a local network is about 3,000,000 CPU operations
- Time to transfer data between 2 CPU's across the world or continent is about 300,000,000 CPU operations

Outer versus Inner Parallelism

- Consider a classic HPC problem – weather prediction – your program would look like
  - a) for(all points in the atmosphere) {
  - b) Calculate new values for density, pressure, velocity, moisture and other chemical constituents based on fundamental equations }
- a) is outer and b) has inner or instruction or vector parallelism
- Both are important sources of parallelism
  - a) is focus of this course and is achieved by YOU
  - b) is "automatic" for CPU and compiler but can be aided by choice of programming styles

Outer Parallelism

- It corresponds to that achieved by bricklayers in Hadrian's wall
- For weather case, it could be three for loops over (x,y,z) – geographical position (x,y) and vertical distance z into atmosphere
- One can easily have 10^6 to 10^9 way parallelism for such 3D problems (100x100X100 or 1000X1000X1000)
- As in Hadrian's wall case, one needs to divide problem up into parts, so that each part is big enough that edge effects not important
  - 100,000 parts each with 10,000 for loop values (grid point values) would be a typical good choice

Inner Parallelism

- This corresponds to the arithmetic manipulating the values defined by outer parallelism for loop index values
- Whereas outer parallelism is huge and scales with size of problem
- Inner parallelism is modest (2→10) and largely independent of problem size
- Instruction Level Parallelism (ILP) executes statements like
  - x=10.; y=10.; z=10; simultaneously but has to worry that
  - x=10.; y=10.; fudge=0.2*x+0.8*y; cannot be done simultaneously
- Speculative execution boldly executes as many instructions as it can and redoes ones that have conflicts

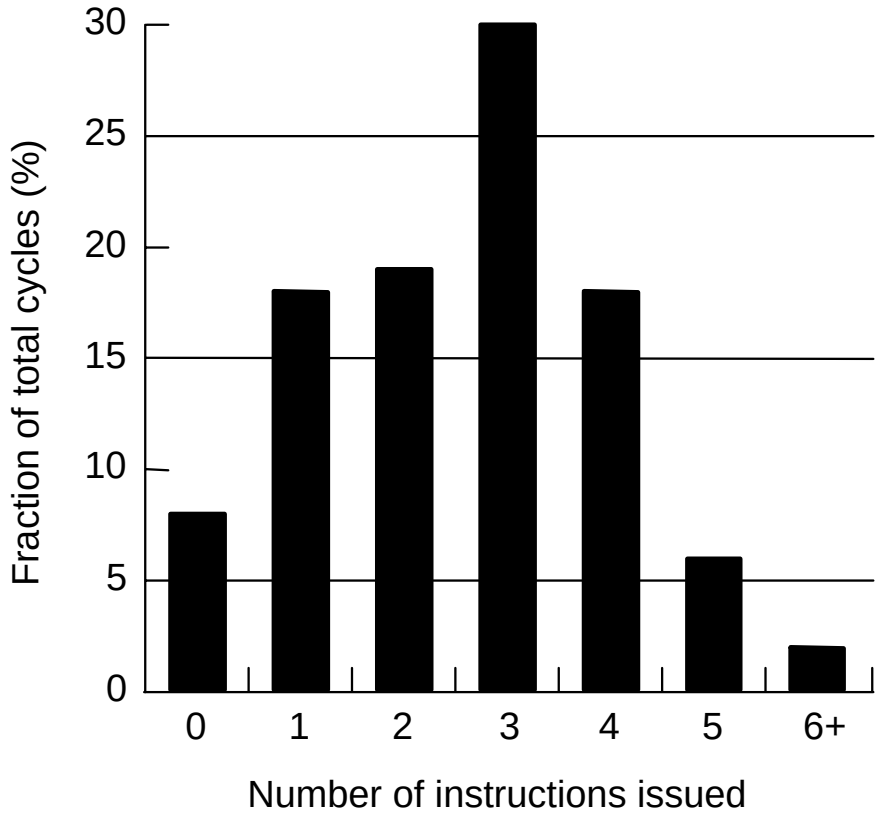
How to use more transistors?

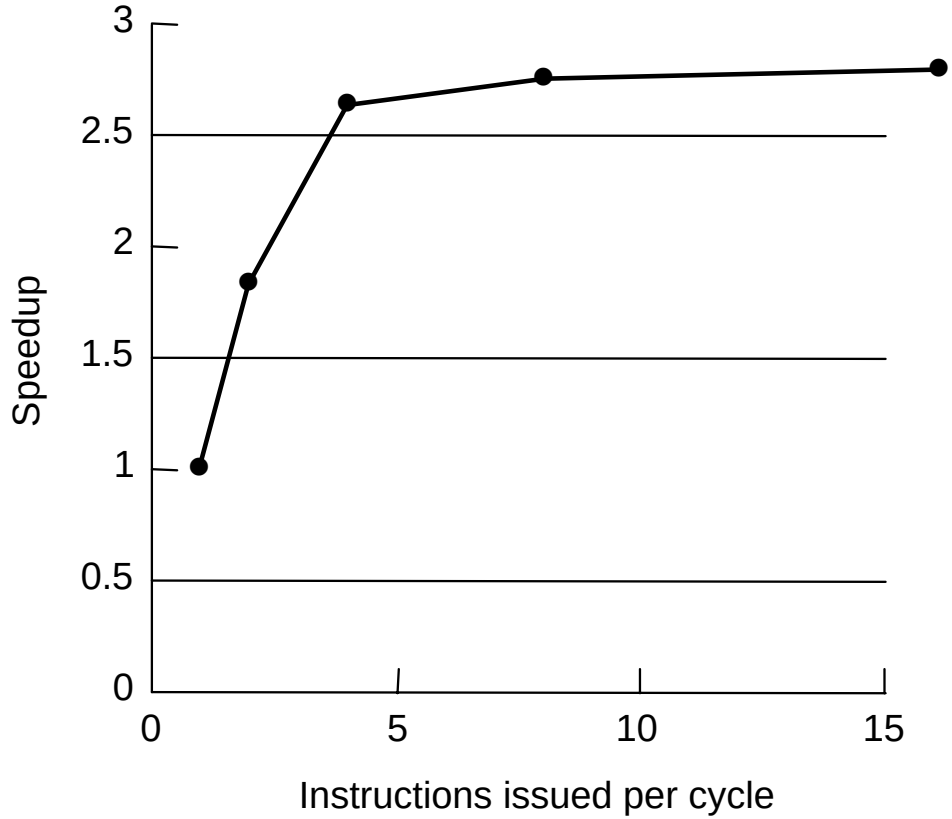
- **Parallelism in processing**
  - multiple operations per cycle reduces CPI
  - One cycle is 0.3 10^{-9} seconds
- **Cache** to give locality in data access
  - avoids latency and reduces CPI
  - also improves processor utilization
- Both need (transistor) resources, so tradeoff
- ILP (Instruction Loop Parallelism) drove performance gains of sequential microprocessors over last 15 years
- ILP Success was not expected by aficionado's of parallel computing and this "delayed" relevance of scaling "outer-loop" parallelism as user's just purchased faster "sequential machines"
- Outer loop parallelism would correspond to putting several CPU's on a chip but note we don't know how to automate use of multiple CPUs; ILP is attractive as "automatic"

CPI = Clock Cycles per Instruction

Possible Gain from ILP

- Hardware allowed many instructions per cycle using transistor budget for ILP parallelism
- Limited Speed up (average 2.75 below) and inefficient (50% or worse)
- However TOTALLY automatic (compiler generated)

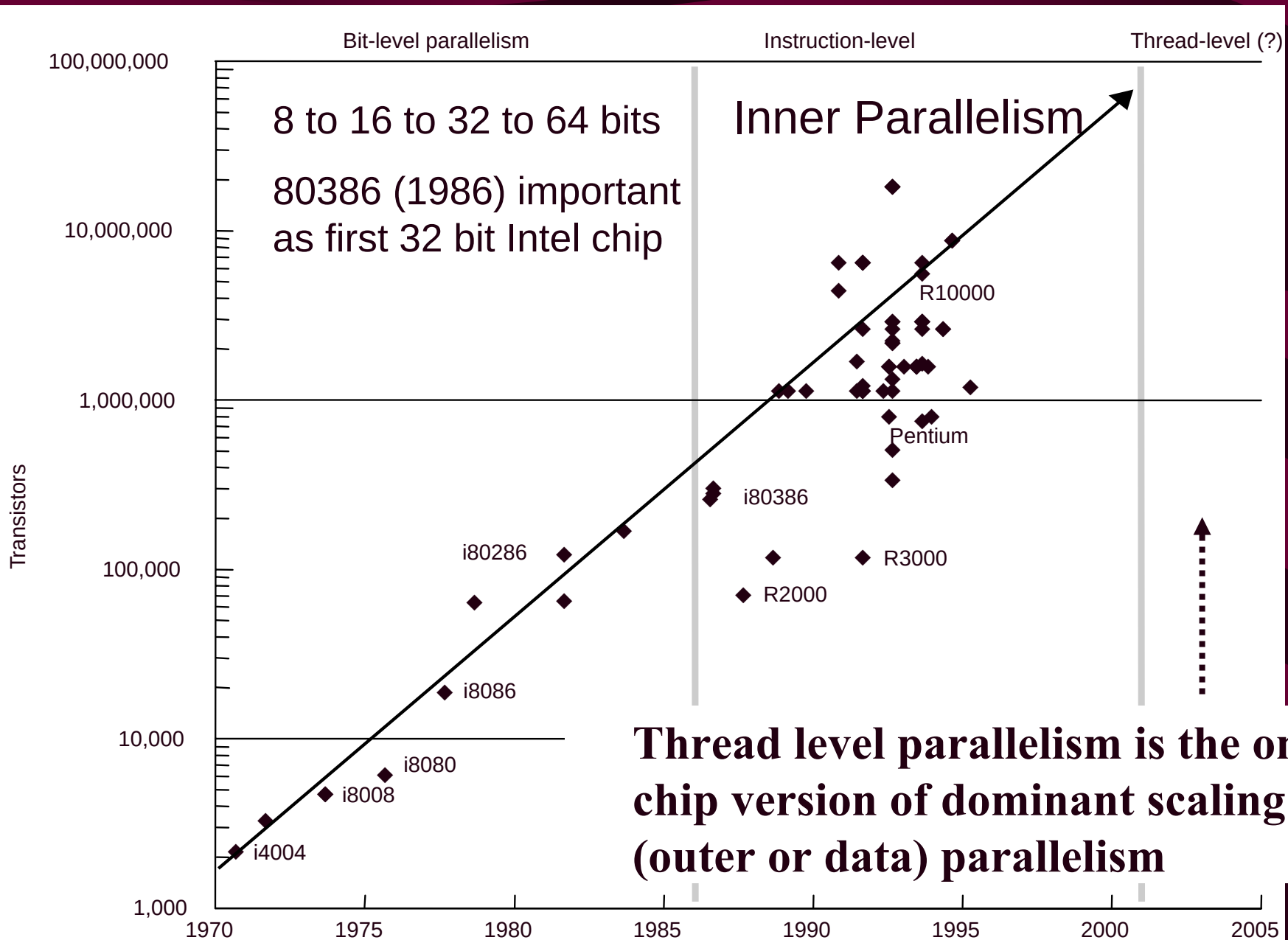




Parallel Computing Rationale

- Transistors are still getting cheaper and cheaper and it only takes some 0.5-1 million transistors to make a very high quality CPU
- This chip would have little ILP (or parallelism in "innermost loops")
- Thus next generation of processor chips more or less have to have multiple CPU's as gain from ILP limited
  - Might reconsider use of ILP once you have ability to exploit outer parallelism
- However getting much more speedup than this requires use of "outer loop" or data parallelism.
  - This is naturally implemented with threads on chip
- The March of Parallelism:
  One CPU on Multiple boards --> Multiple chips on a board --> Multiple CPU's on a chip
- Implies that "outer loop" Parallel Computing gets more and more important in dominant commodity market
- Use of "Outer Loop" parallelism can not (yet) be automated

Trends in Parallelism



Thread level parallelism is the on chip version of dominant scaling (outer or data) parallelism