CS267
Applications of Parallel Computers

www.cs.berkeley.edu/~demmel/cs267_Spr12/

Lecture 1: Introduction

Jim Demmel

EECS & Math Departments

demmel@cs.berkeley.edu

Outline

all

- Why ~~powerful~~ computers must be parallel processors

  Including your laptops and handhelds

- Large Computational Science and Engineering (CSE) problems require powerful computers

  Commercial problems too

- Why writing (fast) parallel programs is hard

  But things are improving

- Structure of the course

Units of Measure

- **High Performance Computing (HPC) units are:**
  - Flop: floating point operation, usually double precision unless noted
  - Flop/s: floating point operations per second
  - Bytes: size of data (a double precision floating point number is 8)
- **Typical sizes are millions, billions, trillions…**

Mega	Mflop/s = 10^6 flop/sec	Mbyte = 2^{20} = 1048576 ~ 10^6 bytes
Giga	Gflop/s = 10^9 flop/sec	Gbyte = 2^{30} ~ 10^9 bytes
Tera	Tflop/s = 10^{12} flop/sec	Tbyte = 2^{40} ~ 10^{12} bytes
Peta	Pflop/s = 10^{15} flop/sec	Pbyte = 2^{50} ~ 10^{15} bytes
Exa	Eflop/s = 10^{18} flop/sec	Ebyte = 2^{60} ~ 10^{18} bytes
Zetta	Zflop/s = 10^{21} flop/sec	Zbyte = 2^{70} ~ 10^{21} bytes
Yotta	Yflop/s = 10^{24} flop/sec	Ybyte = 2^{80} ~ 10^{24} bytes

- **Current fastest (public) machine ~ 11 Pflop/s**
  - Up-to-date list at www.top500.org

all (2007)

Why ~~powerful~~ computers are parallel

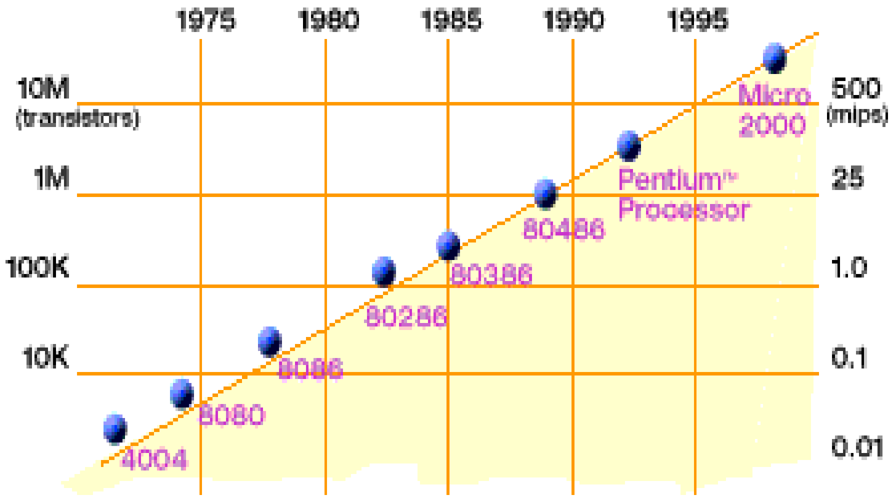
circa 1991-2006

Tunnel Vision by Experts

- "I think there is a world market for maybe five computers."
  - Thomas Watson, chairman of IBM, 1943.
- "There is no reason for any individual to have a computer in their home"
  - Ken Olson, president and founder of Digital Equipment Corporation, 1977.
- "640K [of memory] ought to be enough for anybody."
  - Bill Gates, chairman of Microsoft,1981.
- "On several recent occasions, I have been asked whether parallel computing will soon be relegated to the trash heap reserved for promising technologies that never quite make it."
  - Ken Kennedy, CRPC Directory, 1994

Slide source: Warfield et al.

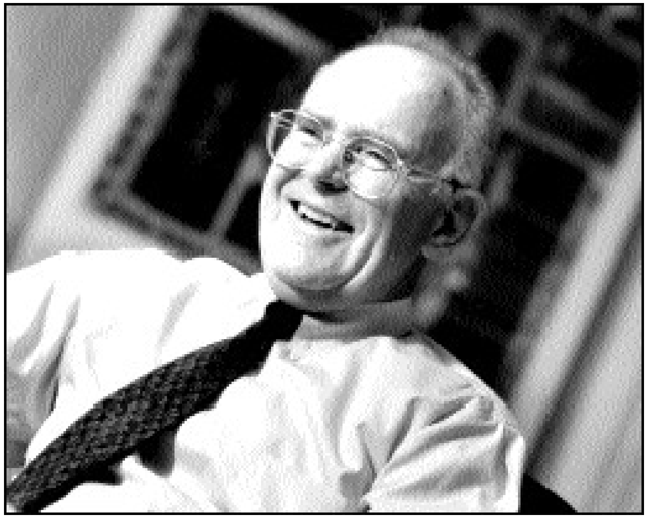
Technology Trends: Microprocessor Capacity



2X transistors/Chip Every 1.5 years

Called "Moore's Law"

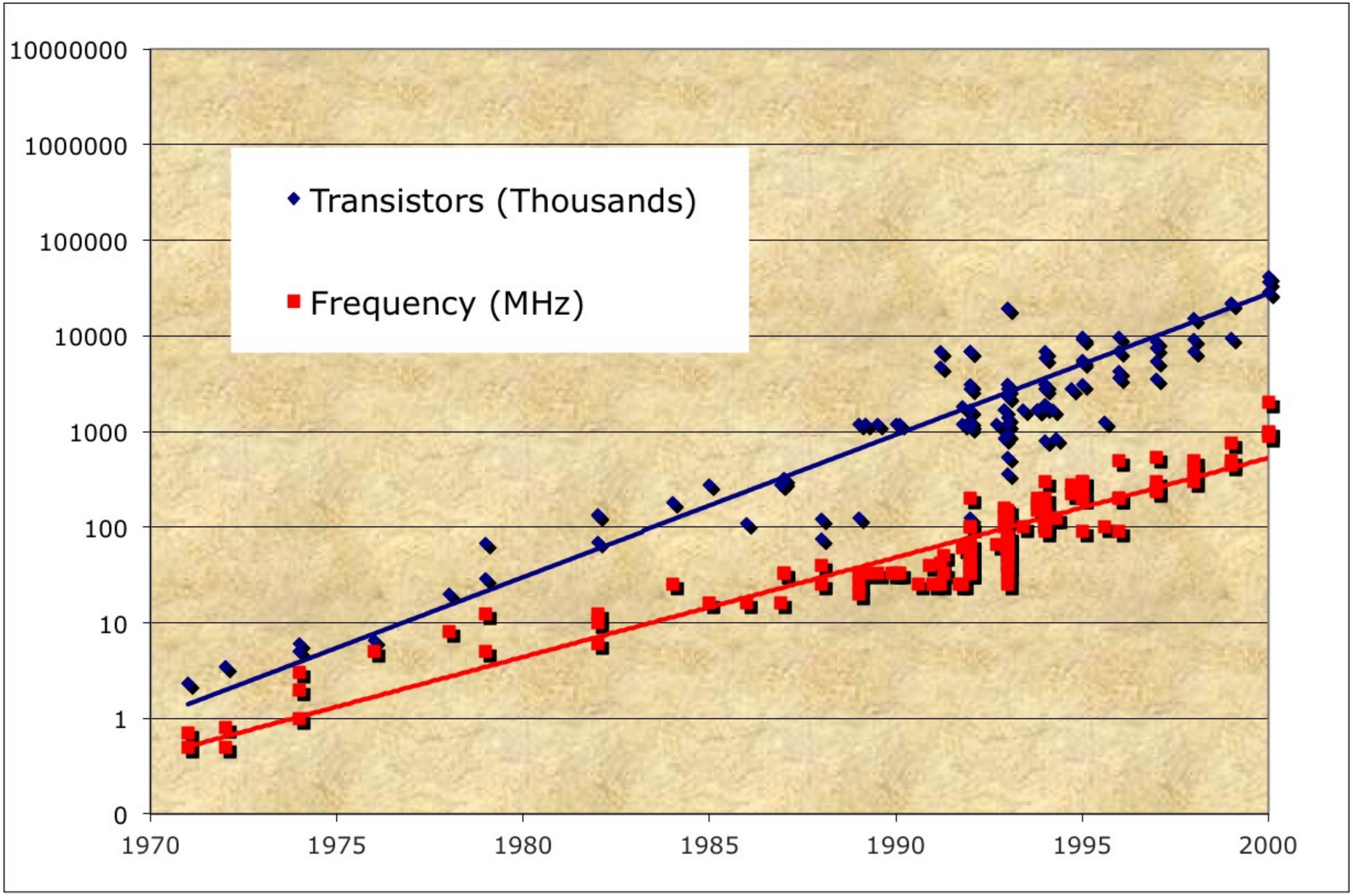
Microprocessors have become smaller, denser, and more powerful.



Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

Slide source: Jack Dongarra

Microprocessor Transistors / Clock (1970-2000)

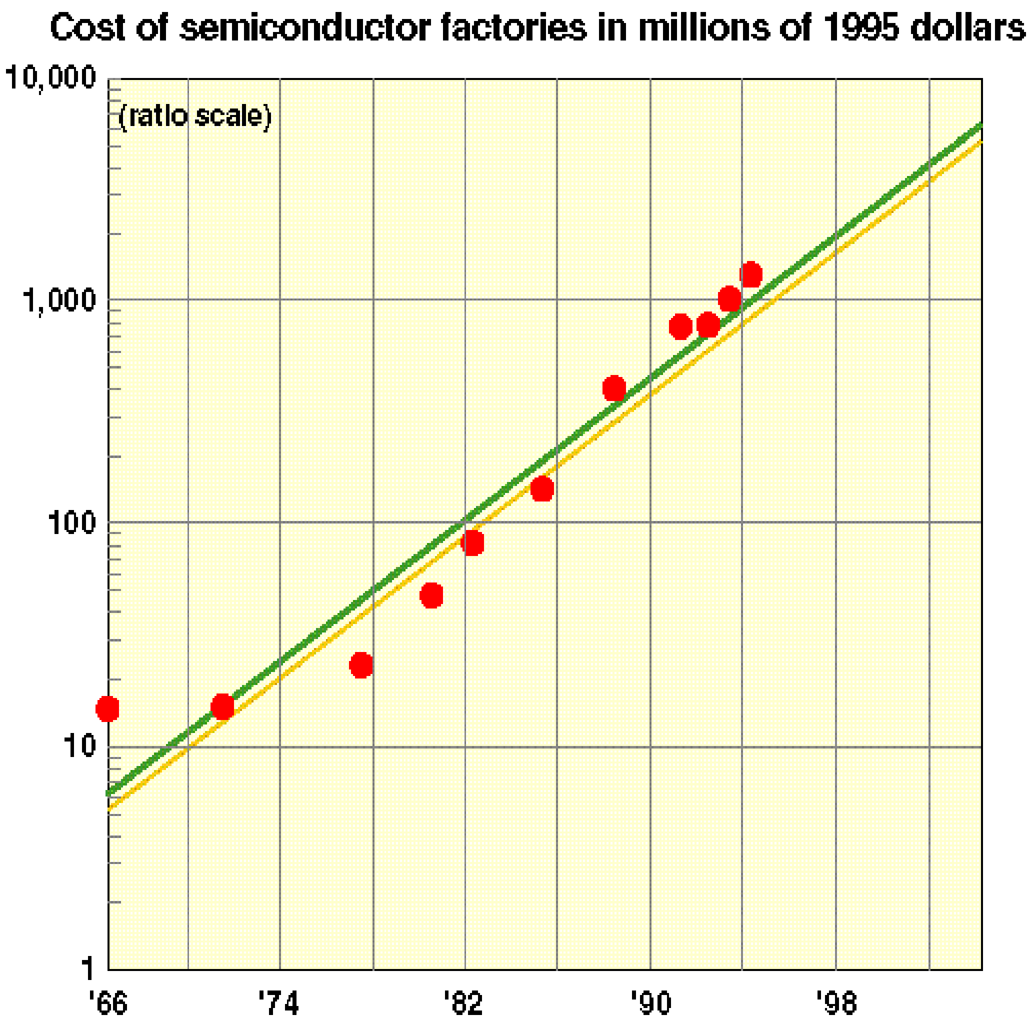


Impact of Device Shrinkage

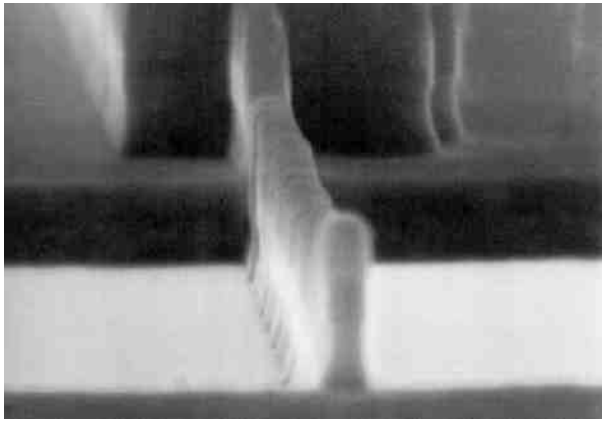
- What happens when the feature size (transistor size) shrinks by a factor of x ?
- Clock rate goes up by x because wires are shorter
  - actually less than x, because of power consumption
- Transistors per unit area goes up by x^2
- Die size also tends to increase
  - typically another factor of ~x
- Raw computing power of the chip goes up by ~ x^4 !
  - typically x^3 is devoted to either on-chip
    - parallelism: hidden parallelism such as ILP
    - locality: caches
- So most programs x^3 times faster, without changing them

Manufacturing Issues Limit Performance

Manufacturing costs and yield problems limit use of density



- Moore's 2nd law (Rock's law): costs go up



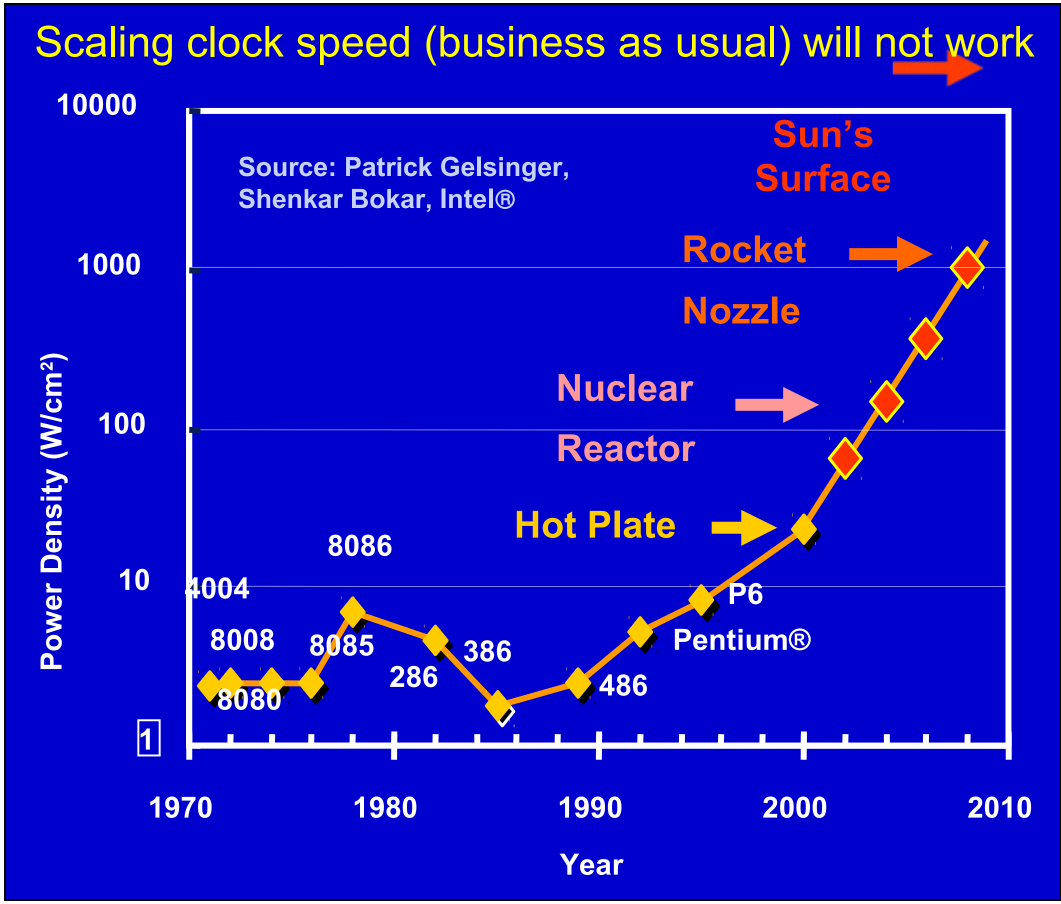
Demo of 0.06 micron CMOS

Source: Forbes Magazine

- **Yield**
  - -What percentage of the chips are usable?
  - -E.g., Cell processor (PS3) is sold with 7 out of 8 "on" to improve yield

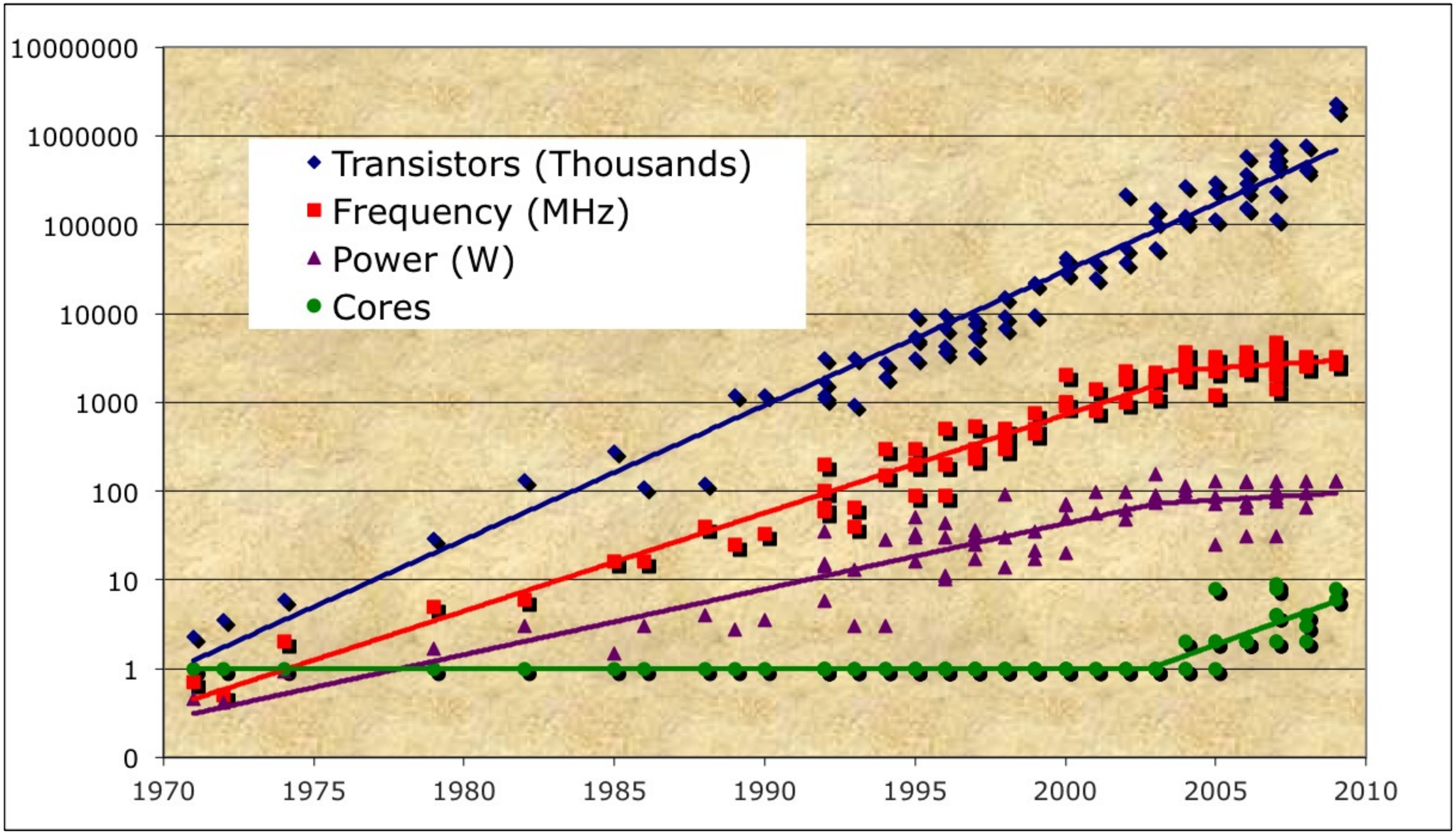
Power Density Limits Serial Performance

- Concurrent systems are more power efficient
  - Dynamic power is proportional to V^2fC
  - Increasing frequency (f) also increases supply voltage (V) → cubic effect
  - Increasing cores increases capacitance (C) but only linearly
  - Save power by lowering clock speed



- High performance serial processors waste power
  - Speculation, dynamic dependence checking, etc. burn power
  - Implicit parallelism discovery
- More transistors, but not faster serial processors

Revolution in Processors



- Chip density is continuing increase ~2x every 2 years
- Clock speed is not
- Number of processor cores may double instead
- Power is under control, no longer growing

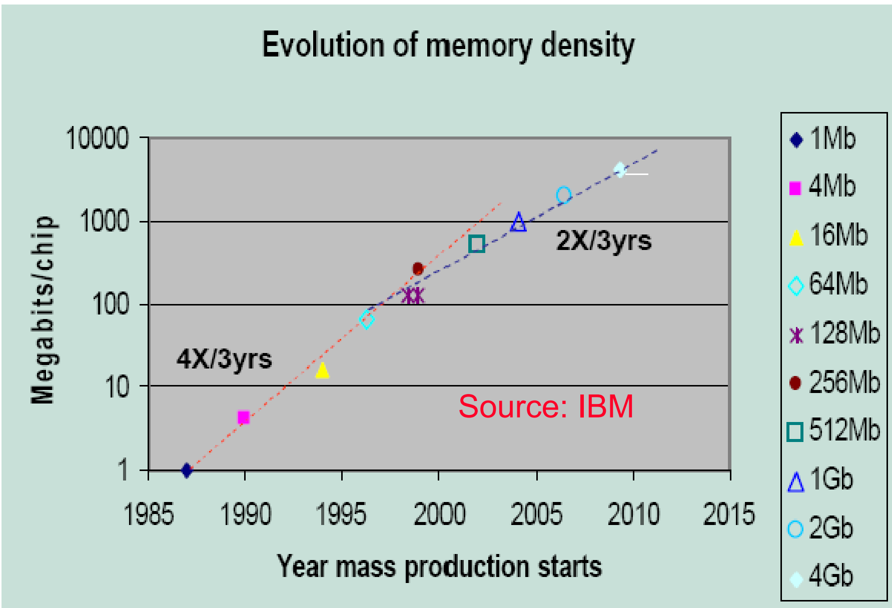
Parallelism in 2012?

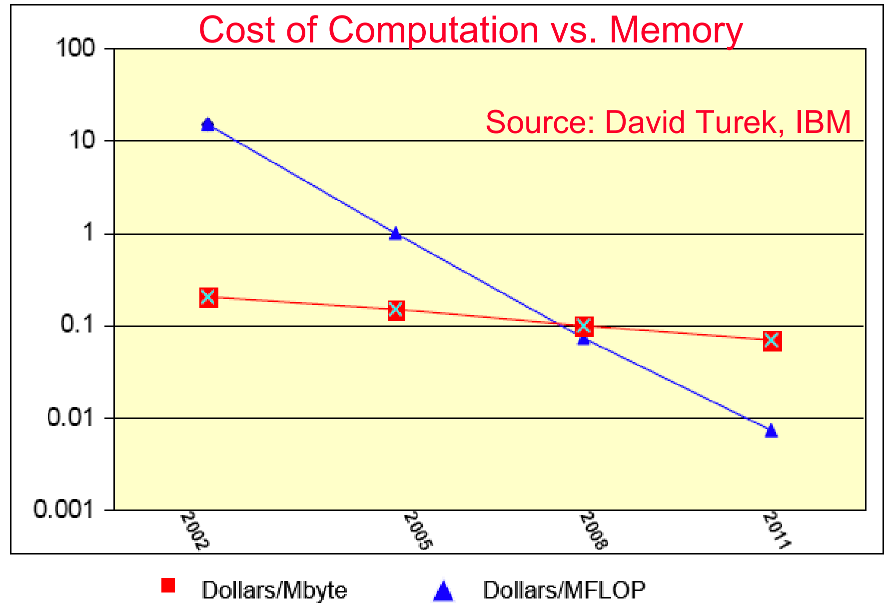
- These arguments are no longer theoretical
- All major processor vendors are producing *multicore* chips
  - Every machine will soon be a parallel machine
  - To keep doubling performance, parallelism must double
- Which (commercial) applications can use this parallelism?
  - Do they have to be rewritten from scratch?
- Will all programmers have to be parallel programmers?
  - New software model needed
  - Try to hide complexity from most programmers – eventually
  - In the meantime, need to understand it
- Computer industry betting on this big change, but does not have all the answers
  - Berkeley ParLab established to work on this

Memory is Not Keeping Pace

Technology trends against a constant or increasing memory per core

- Memory density is doubling every three years; processor logic is every two
- Storage costs (dollars/Mbyte) are dropping gradually compared to logic costs





The cost to sense, collect, generate and calculate data is declining much faster than the cost to access, manage and store it

Question: Can you double concurrency without doubling memory?

- **Strong scaling:** fixed problem size, increase number of processors
- **Weak scaling:** grow problem size proportionally to number of processors

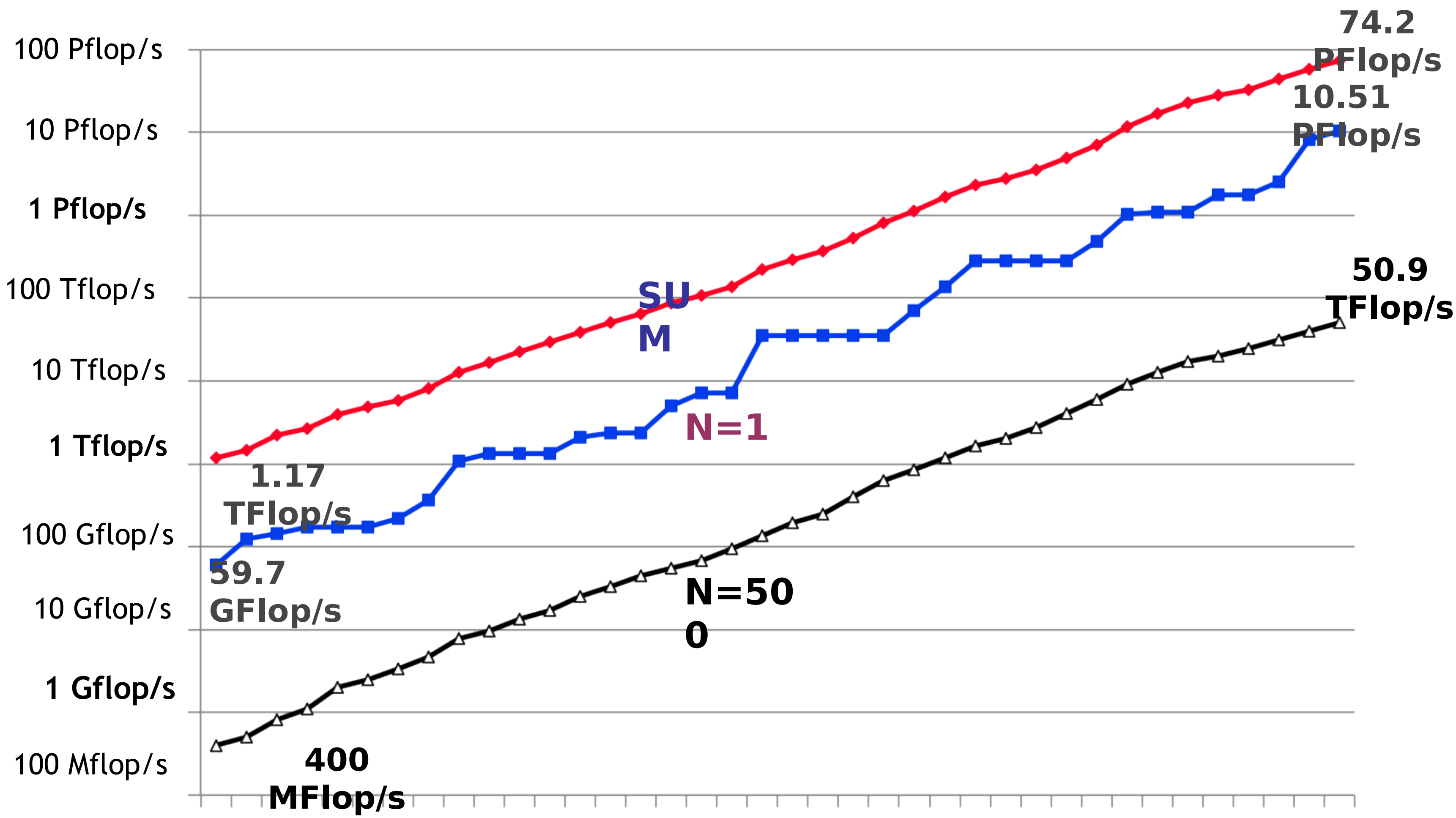
The TOP500 Project

- Listing the 500 most powerful computers in the world
- Yardstick: Rmax of Linpack
  - Solve Ax=b, dense problem, matrix is random
  - Dominated by dense matrix-matrix multiply
- Update twice a year:
  - ISC'xy in June in Germany
  - SCxy in November in the U.S.
- All information available from the TOP500 web site at: www.top500.org

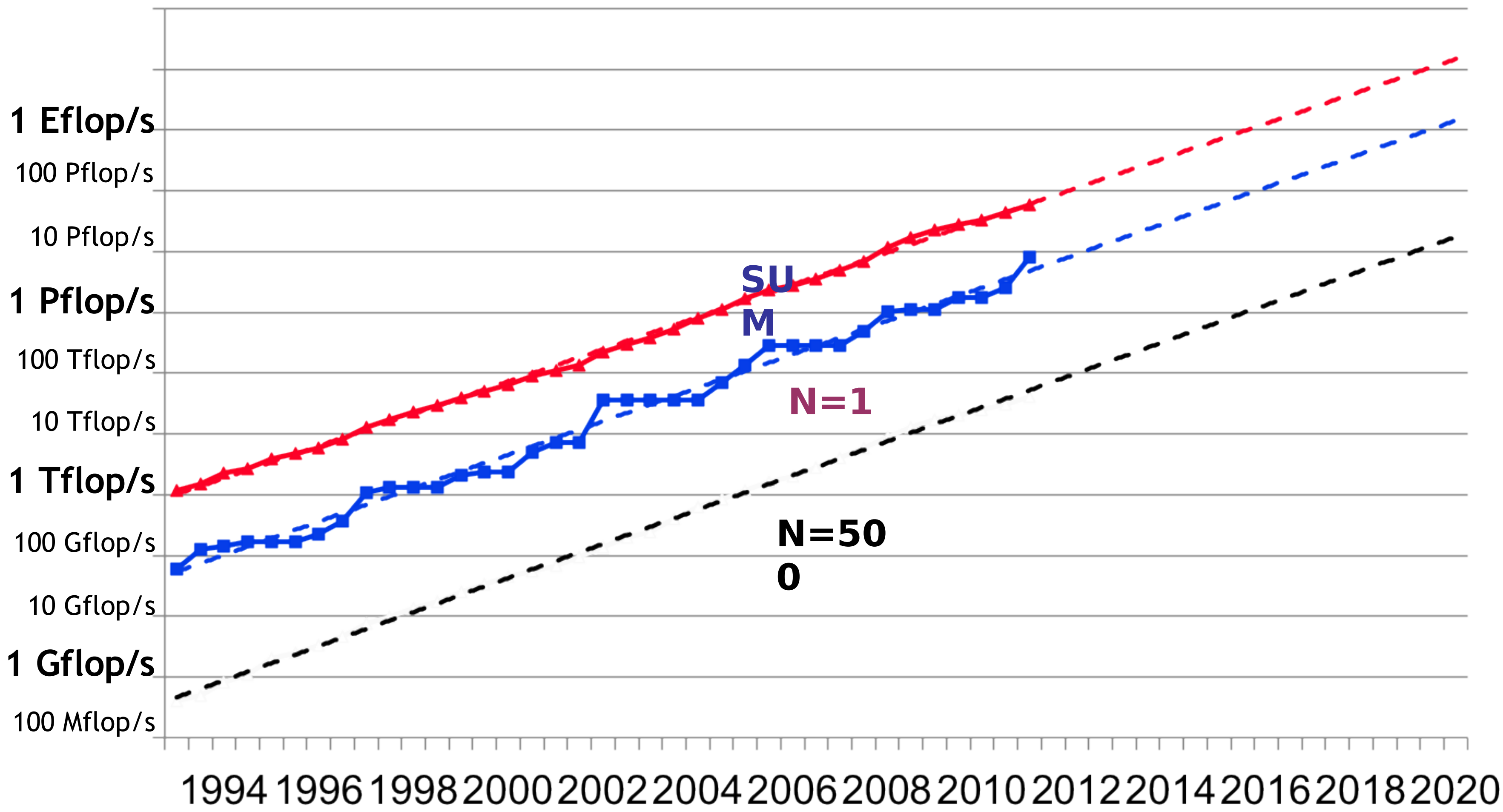
38th List: The TOP10

	Site	*Manufacturer*	*Computer*	*Country*	*Cores*	*Rmax [Pflops]*	*Power [MW]*
1	RIKEN Advanced Institute for Computational Science	Fujitsu	K Computer SPARC64 VIIIfx 2.0GHz, Tofu Interconnect	Japan	795,024	10.51	12.66
2	National SuperComputer Center in Tianjin	NUDT	Tianhe-1A NUDT TH MPP, Xeon 6C, NVidia, FT-1000 8C	China	186,368	2.566	4.04
3	Oak Ridge National Laboratory	Cray	Jaguar Cray XT5, HC 2.6 GHz	USA	224,162	1.759	6.95
4	National Supercomputing Centre in Shenzhen	Dawning	Nebulae TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU	China	120,640	1.271	2.58
5	GSIC, Tokyo Institute of Technology	NEC/HP	TSUBAME-2 HP ProLiant, Xeon 6C, NVidia, Linux/Windows	Japan	73,278	1.192	1.40
6	DOE/NNSA/LANL/SNL	Cray	Cielo Cray XE6, 8C 2.4 GHz	USA	142,272	1.110	3.98
7	NASA/Ames Research Center/NAS	SGI	Pleiades SGI Altix ICE 8200EX/8400EX	USA	111,104	1.088	4.10
8	DOE/SC/ LBNL/NERSC	Cray	Hopper Cray XE6, 6C 2.1 GHz	USA	153,408	1.054	2.91
9	Commissariat a l'Energie Atomique (CEA)	Bull	Tera 100 Bull bullx super-node S6010/S6030	France	138.368	1.050	4.59
10	DOE/NNSA/LANL	IBM	Roadrunner BladeCenter QS22/LS21	USA	122,400	1.042	2.34

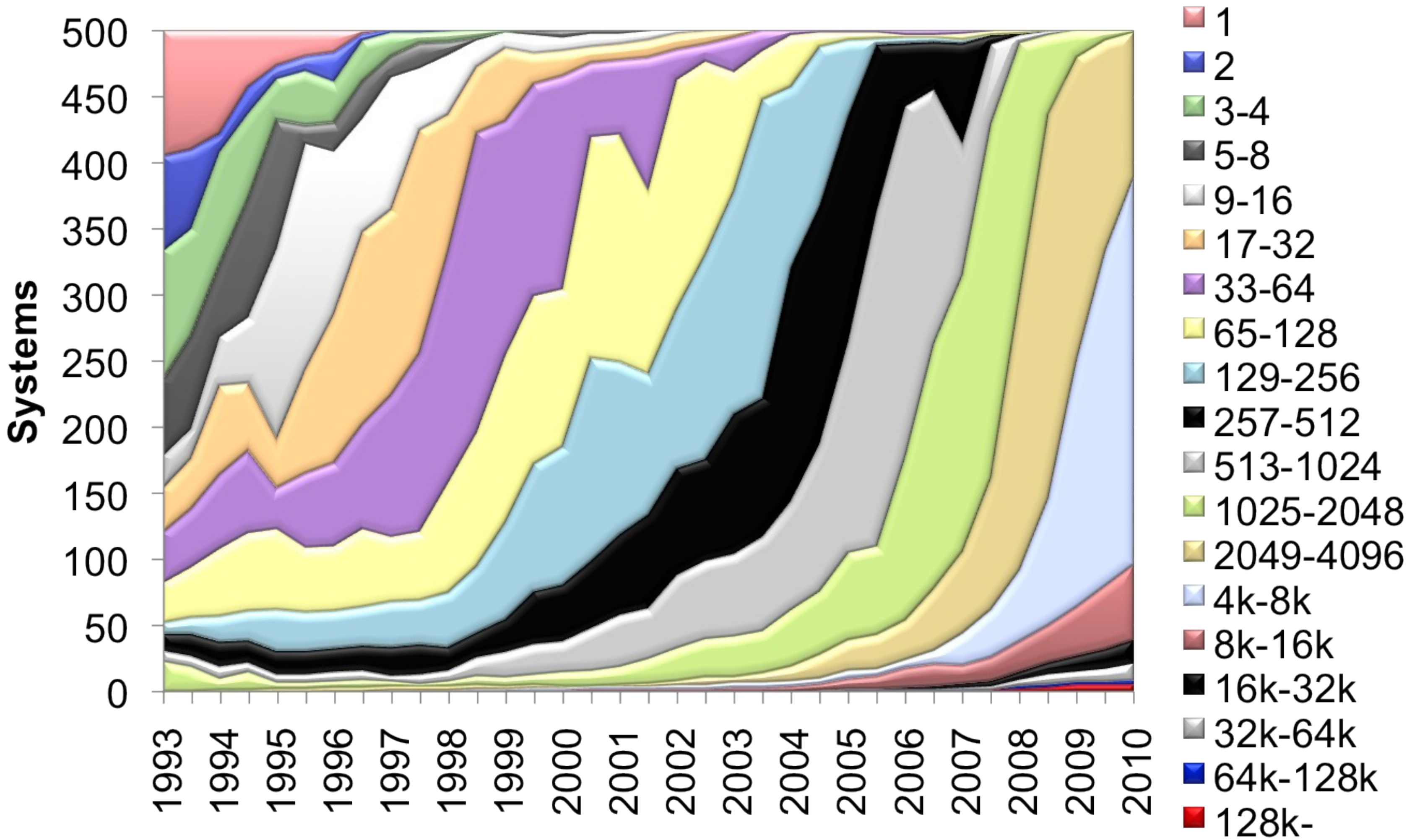
Performance Development



Projected Performance Development



Core Count



Systems
500
450
400
350
300
250
200
150
100
50
0
1993
1994
1995
1996
1997
1998
1999
2000
2001
2002
2003
2004
2005
2006
2007
2008
2009
2010
1
2
3-4
5-8
9-16
17-32
33-64
65-128
129-256
257-512
513-1024
1025-2048
2049-4096
4k-8k
8k-16k
16k-32k
32k-64k
64k-128k
128k-

Moore's Law reinterpreted

- Number of cores per chip can double every two years
- Clock speed will not increase (possibly decrease)
- Need to deal with systems with millions of concurrent threads
- Need to deal with inter-chip parallelism as well as intra-chip parallelism

Outline

all

- Why ~~powerful~~ computers must be parallel processors
  - Including your laptops and handhelds
- Large CSE problems require powerful computers
  - Commercial problems too
- Why writing (fast) parallel programs is hard
  - But things are improving
- Structure of the course

Computational Science - News

"An important development in sciences is occurring at the intersection of computer science and the sciences that has the potential to have a profound impact on science. It is a leap from the application of computing … to the *integration of computer science concepts, tools, and theorems* into the very fabric of science." -*Science* 2020 Report, March 2006

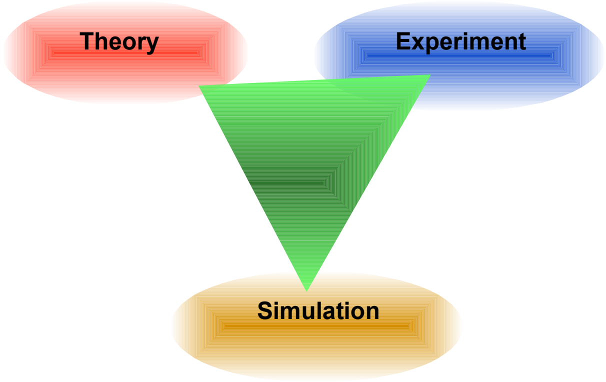
Nature, March 23, 2006

Drivers for Change

- Continued exponential increase in computational power
  - Can simulate what theory and experiment can't do
- Continued exponential increase in experimental data
  - Moore's Law applies to sensors too
  - Need to analyze all that data

Simulation: The Third Pillar of Science

- **Traditional scientific and engineering method:**
  - (1) Do theory or paper design
  - (2) Perform experiments or build system

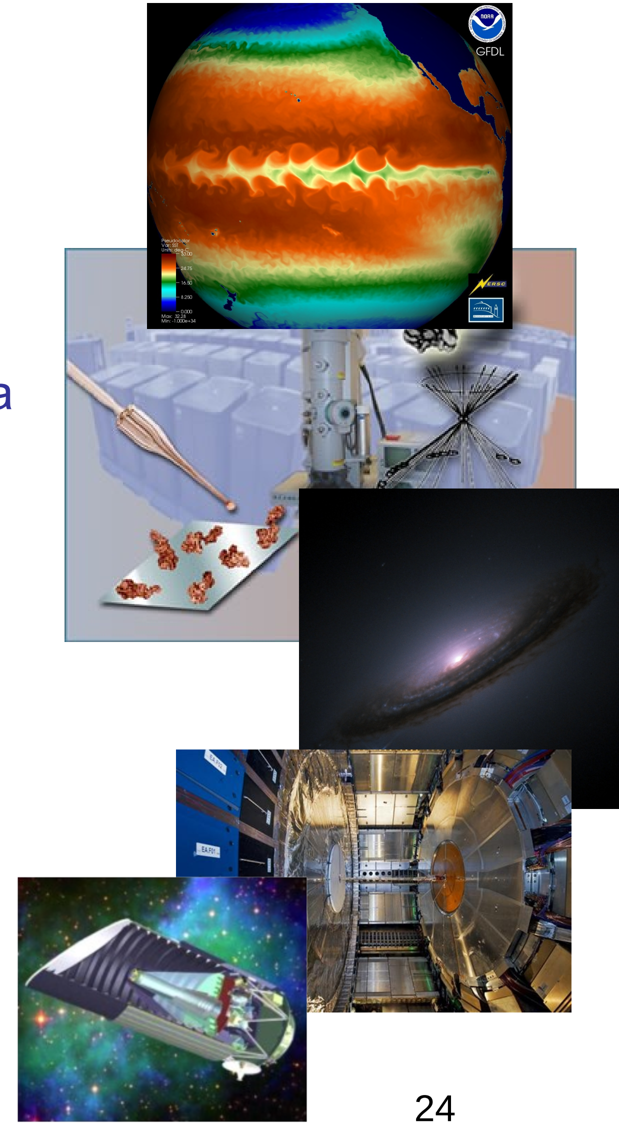


- **Limitations:**
  - –Too difficult—build large wind tunnels
  - –Too expensive—build a throw-away passenger jet
  - –Too slow—wait for climate or galactic evolution
  - –Too dangerous—weapons, drug design, climate experimentation

- **Computational science and engineering paradigm:**
  - (3) Use computers to simulate and analyze the phenomenon
  - \- Based on known physical laws and efficient numerical methods
  - \- Analyze simulation results with computational tools and methods beyond what is possible manually

Data Driven Science

- Scientific data sets are growing exponentially
  - Ability to generate data is exceeding our ability to store and analyze
  - Simulation systems and some observational devices grow in capability with Moore’s Law
- Petabyte (PB) data sets will soon be common:
  - *Climate modeling:* estimates of the next IPCC data is in 10s of petabytes
  - *Genome:* JGI alone will have .5 petabyte of data this year and double each year
  - *Particle physics*: LHC is projected to produce 16 petabytes of data per year
  - *Astrophysics*: LSST and others will produce 5 petabytes/year (via 3.2 Gigapixel camera)
- Create scientific communities with “Science Gateways” to data



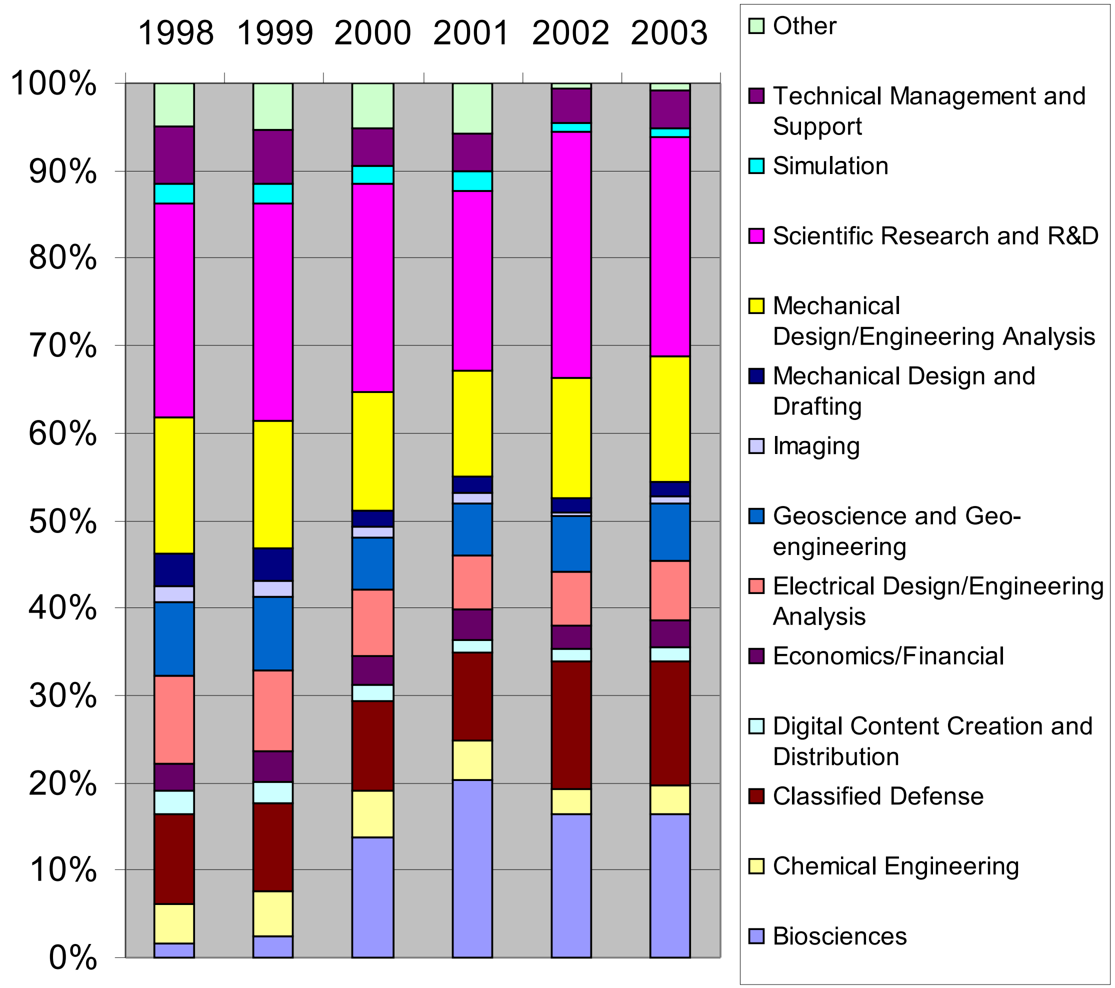
Some Particularly Challenging Computations

- **Science**
  - Global climate modeling
  - Biology: genomics; protein folding; drug design
  - Astrophysical modeling
  - Computational Chemistry
  - Computational Material Sciences and Nanosciences
- **Engineering**
  - Semiconductor design
  - Earthquake and structural modeling
  - Computation fluid dynamics (airplane design)
  - Combustion (engine design)
  - Crash simulation
- **Business**
  - Financial and economic modeling
  - Transaction processing, web services and search engines
- **Defense**
  - Nuclear weapons -- test by simulations
  - Cryptography

Economic Impact of HPC

- **Airlines:**
  - System-wide logistics optimization systems on parallel systems.
  - Savings: approx. $100 million per airline per year.
- **Automotive design:**
  - Major automotive companies use large systems (500+ CPUs) for:
    - CAD-CAM, crash testing, structural integrity and aerodynamics.
    - One company has 500+ CPU parallel system.
  - Savings: approx. $1 billion per company per year.
- **Semiconductor industry:**
  - Semiconductor firms use large systems (500+ CPUs) for
    - device electronics simulation and logic validation
  - Savings: approx. $1 billion per company per year.
- **Energy**
  - Computational modeling improved performance of current nuclear power plants, equivalent to building two new power plants.

$5B World Market in Technical Computing in 2004



Source: IDC 2004, from NRC Future of Supercomputing Report

What Supercomputers Do – Two Examples

- **Climate modeling**
  - **simulation replacing experiment that is too slow**
- **Cosmic microwave background radition**
  - **analyzing massive amounts of data with new tools**

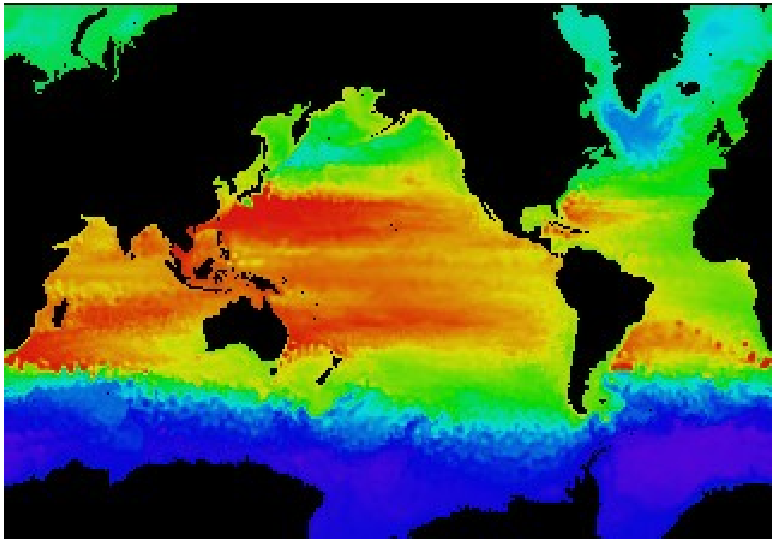
Global Climate Modeling Problem

- Problem is to compute:

  f(latitude, longitude, elevation, time) → “weather” =

  (temperature, pressure, humidity, wind velocity)

- Approach:
  - *Discretize* the domain, e.g., a measurement point every 10 km
  - Devise an algorithm to predict weather at time $t+\delta t$ given t
- Uses:
  - Predict major events, e.g., El Nino
  - Use in setting air emissions standards
  - Evaluate global warming scenarios



Source: http://www.epm.ornl.gov/chammp/chammp.html

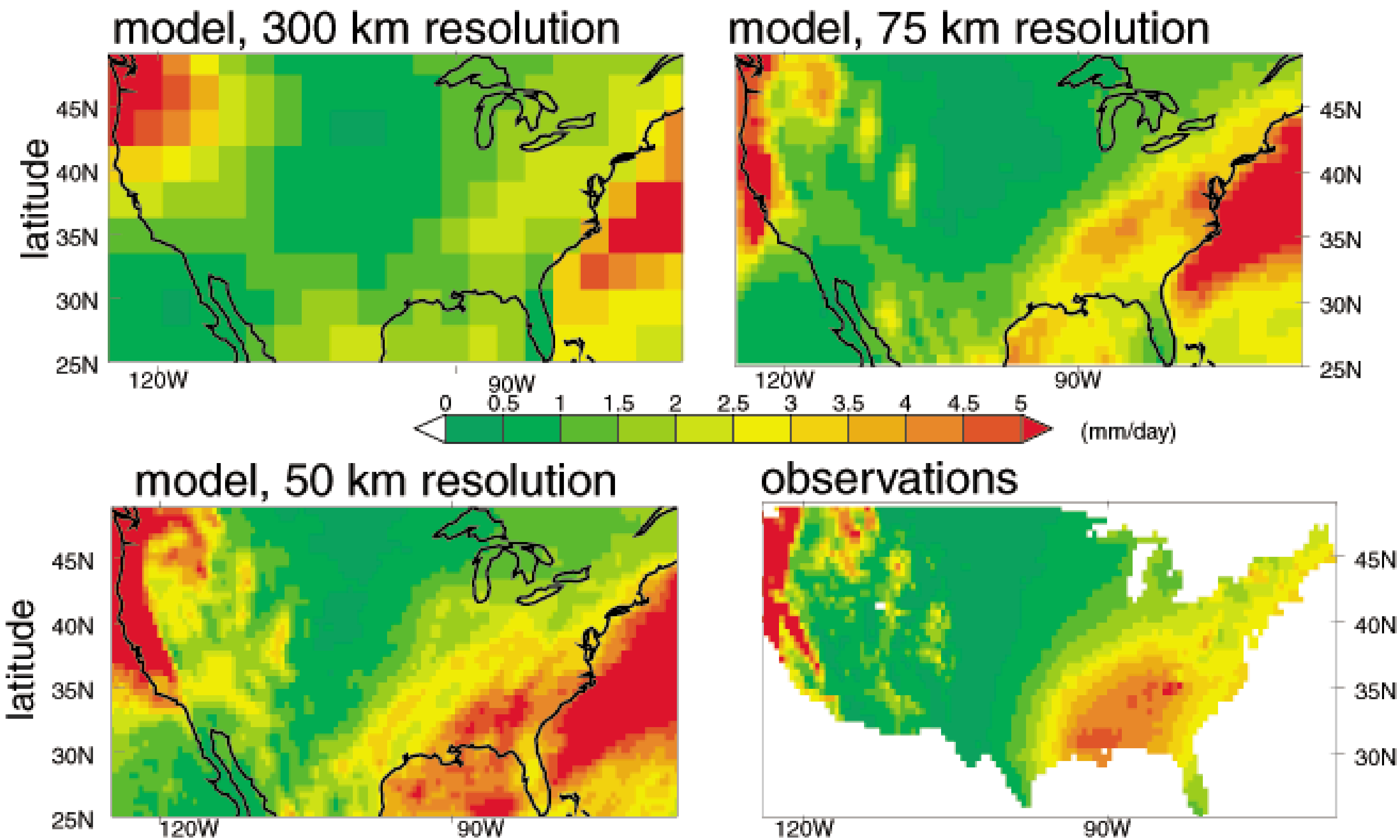
Global Climate Modeling Computation

- One piece is modeling the fluid flow in the atmosphere
  - Solve Navier-Stokes equations
    - Roughly 100 Flops per grid point with 1 minute timestep
- Computational requirements:
  - To match real-time, need 5 x 10^{11} flops in 60 seconds = 8 Gflop/s
  - Weather prediction (7 days in 24 hours) → 56 Gflop/s
  - Climate prediction (50 years in 30 days) → 4.8 Tflop/s
  - To use in policy negotiations (50 years in 12 hours) → 288 Tflop/s
- To double the grid resolution, computation is 8x to 16x
- State of the art models require integration of atmosphere, clouds, ocean, sea-ice, land models, plus possibly carbon cycle, geochemistry and more
- Current models are coarser than this

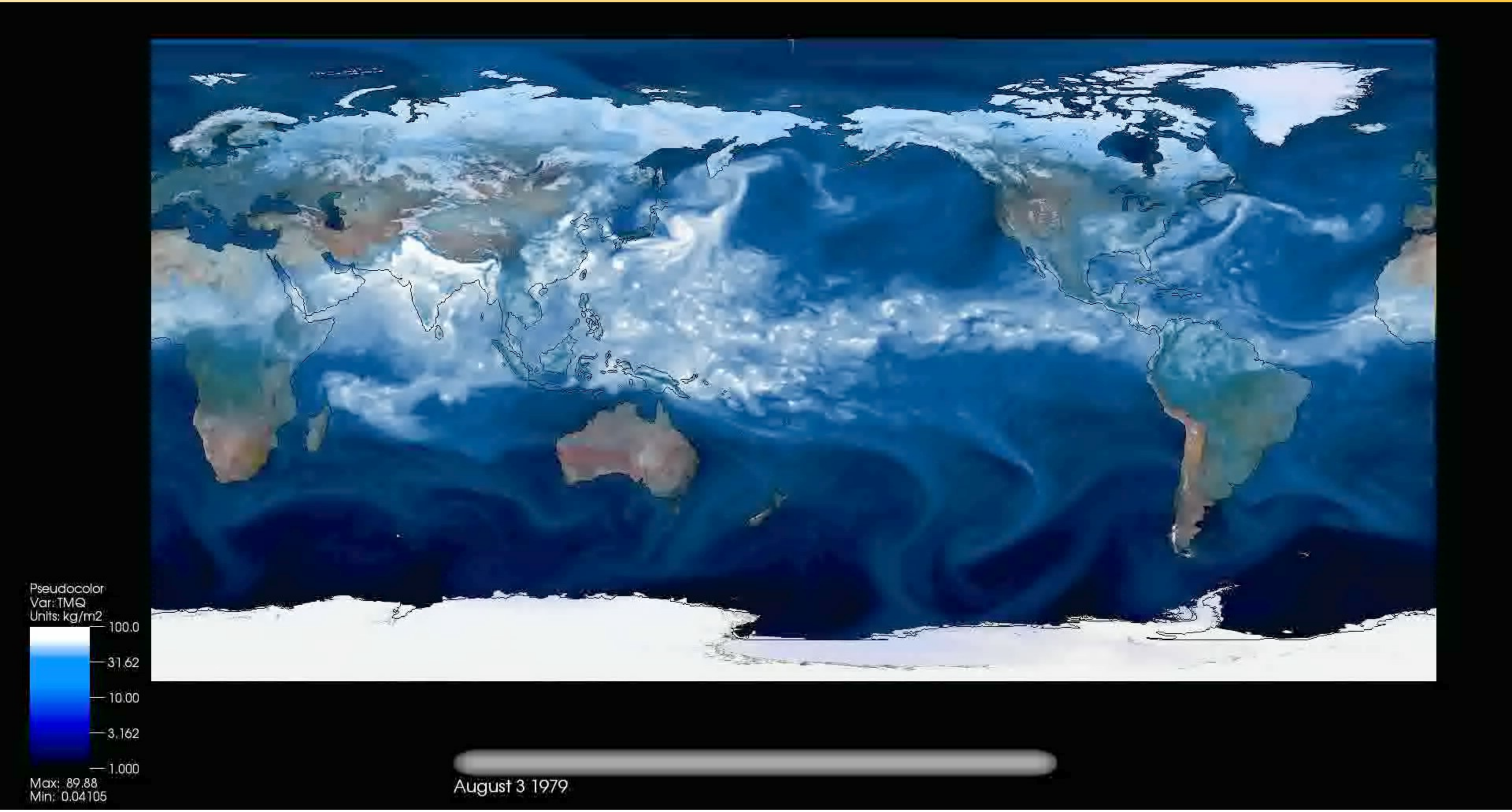
High Resolution Climate Modeling on NERSC-3 – P. Duffy, et al., LLNL

Wintertime Precipitation

As model resolution becomes finer, results converge towards observations

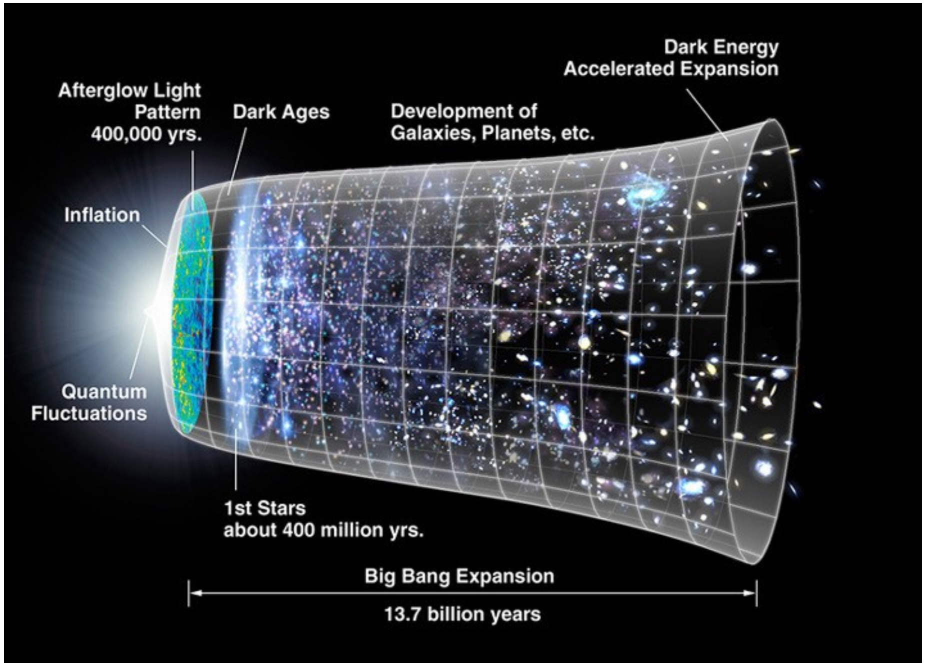


U.S.A. Hurricane



Source: Data from M.Wehner, visualization by Prabhat, LBNL

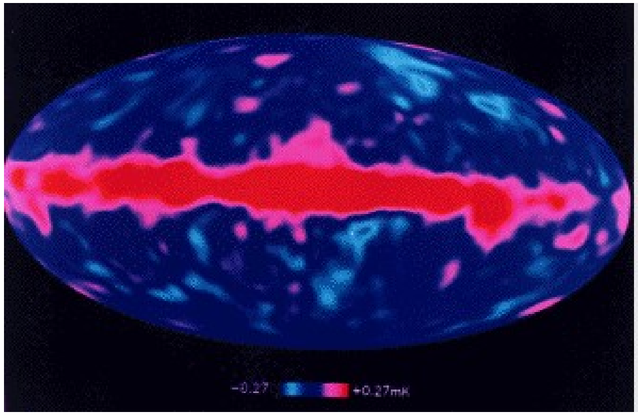
NERSC User George Smoot wins 2006 Nobel Prize in Physics



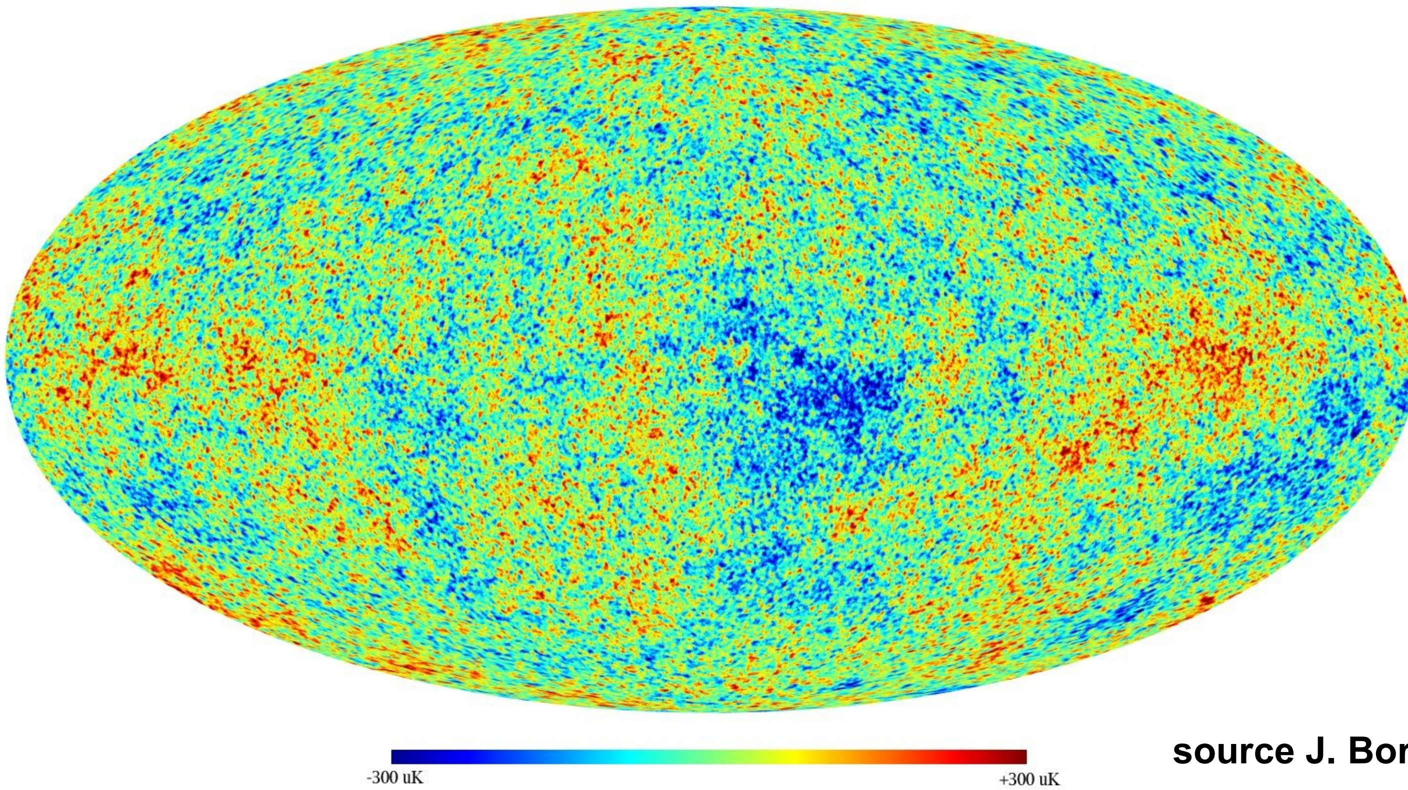
Cosmic Microwave Background Radiation (CMB): an image of the universe at 400,000 years

Smoot and Mather 1992

COBE Experiment showed anisotropy of CMB



The Current CMB Map



source J. Borrill, LBNL

- Unique imprint of primordial physics through the tiny anisotropies in temperature and polarization.
- Extracting these μKelvin fluctuations from inherently noisy data is a serious computational challenge.

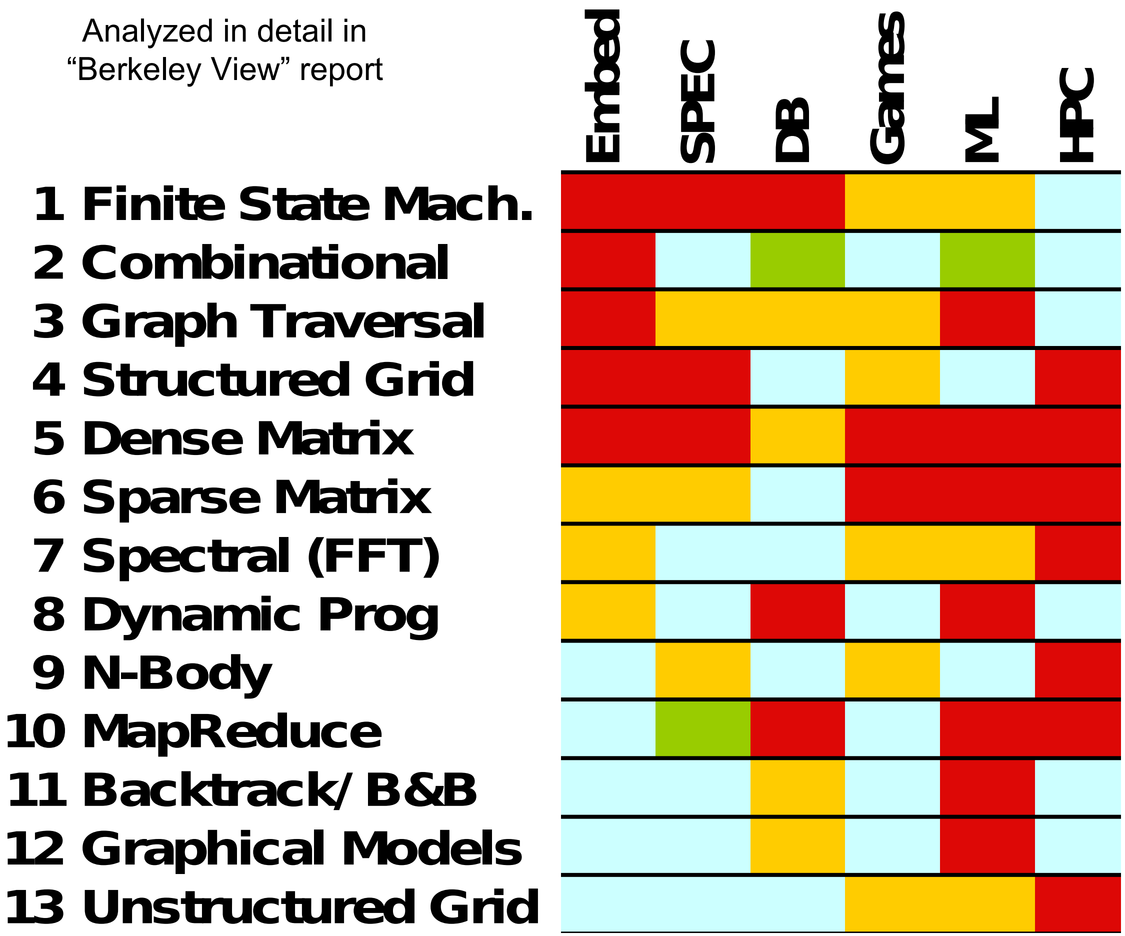
Evolution Of CMB Data Sets: Cost > O(Np^3)

Experiment	N_t	N_p	N_b	Limiting Data	Notes
COBE (1989)	$2x10^9$	$6x10^3$	$3x10^1$	Time	Satellite, Workstation
BOOMERanG (1998)	$3x10^8$	$5x10^5$	$3x10^1$	Pixel	Balloon, 1st HPC/NERSC
(4yr) WMAP (2001)	$7x10^{10}$	$4x10^7$	$1x10^3$	?	Satellite, Analysis-bound
Planck (2007)	$5x10^{11}$	$6x10^8$	$6x10^3$	Time/ Pixel	Satellite, Major HPC/DA effort
POLARBEAR (2007)	$8x10^{12}$	$6x10^6$	$1x10^3$	Time	Ground, NG-multiplexing
CMBPol (~2020)	10^{14}	10^9	10^4	Time/ Pixel	Satellite, Early planning/design

data compression

Which commercial applications *require* parallelism?

Analyzed in detail in
"Berkeley View" report

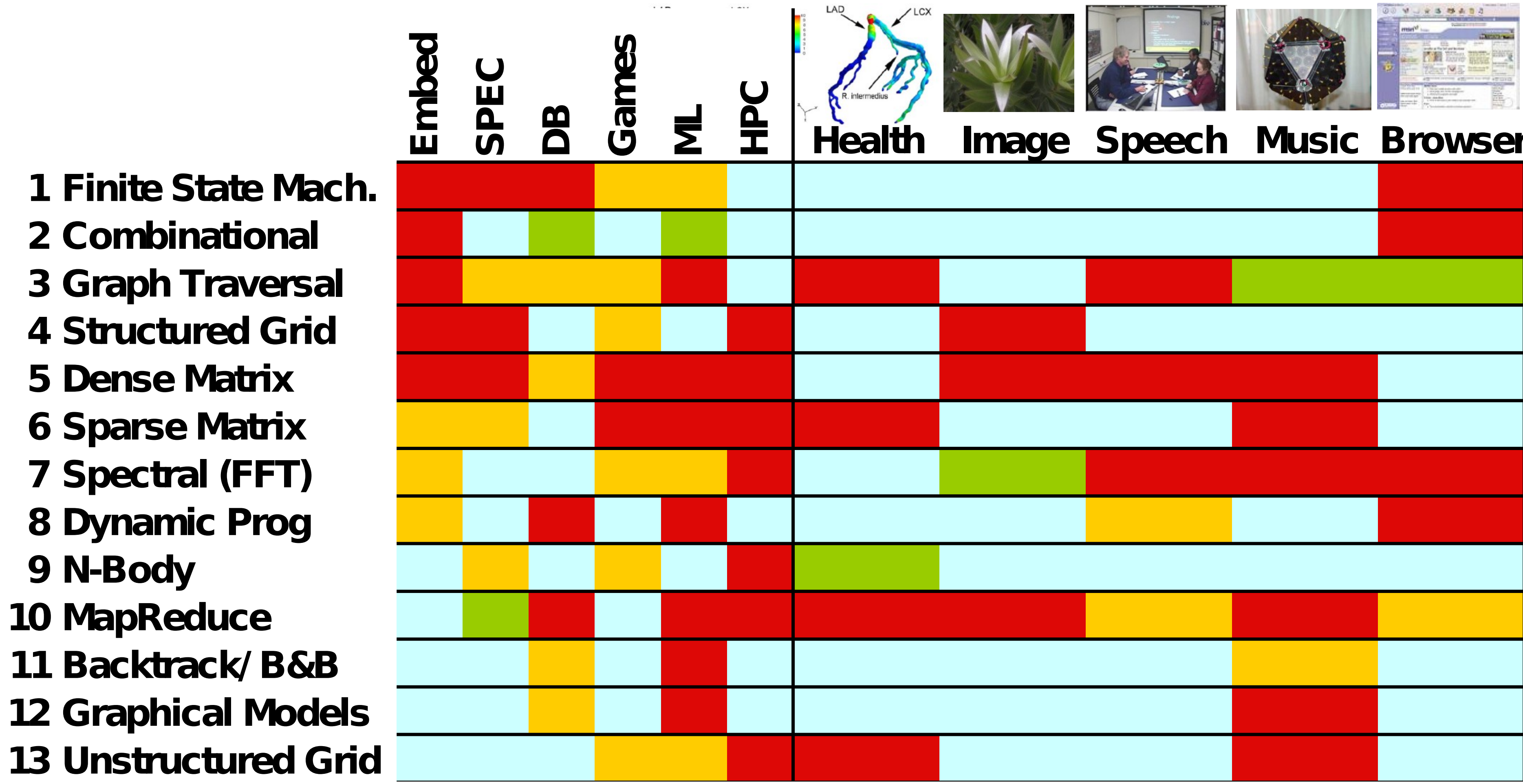


Analyzed in detail in "Berkeley View" report
www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.html

What do commercial and CSE applications have in common?

Motif/Dwarf: Common Computational Methods

(Red Hot → Blue Cool)



Outline

all

- Why ~~powerful~~ computers must be parallel processors
  - Including your laptops and handhelds
- Large CSE problems require powerful computers
  - Commercial problems too
- Why writing (fast) parallel programs is hard
  - But things are improving
- Structure of the course

Principles of Parallel Computing

- Finding enough parallelism (Amdahl's Law)
- Granularity – how big should each parallel task be
- Locality – moving data costs more than arithmetic
- Load balance – don't want 1K processors to wait for one slow one
- Coordination and synchronization – sharing data safely
- Performance modeling/debugging/tuning

⇨ All of these things makes parallel programming even harder than sequential programming.

“Automatic” Parallelism in Modern Machines

- Bit level parallelism
  - within floating point operations, etc.
- Instruction level parallelism (ILP)
  - multiple instructions execute per clock cycle
- Memory system parallelism
  - overlap of memory operations with computation
- OS parallelism
  - multiple jobs run in parallel on commodity SMPs

Limits to all of these -- for very high performance, need user to identify, schedule and coordinate parallel tasks

Finding Enough Parallelism

- Suppose only part of an application seems parallel
- Amdahl's law
  - let s be the fraction of work done sequentially, so (1-s) is fraction parallelizable
  - P = number of processors

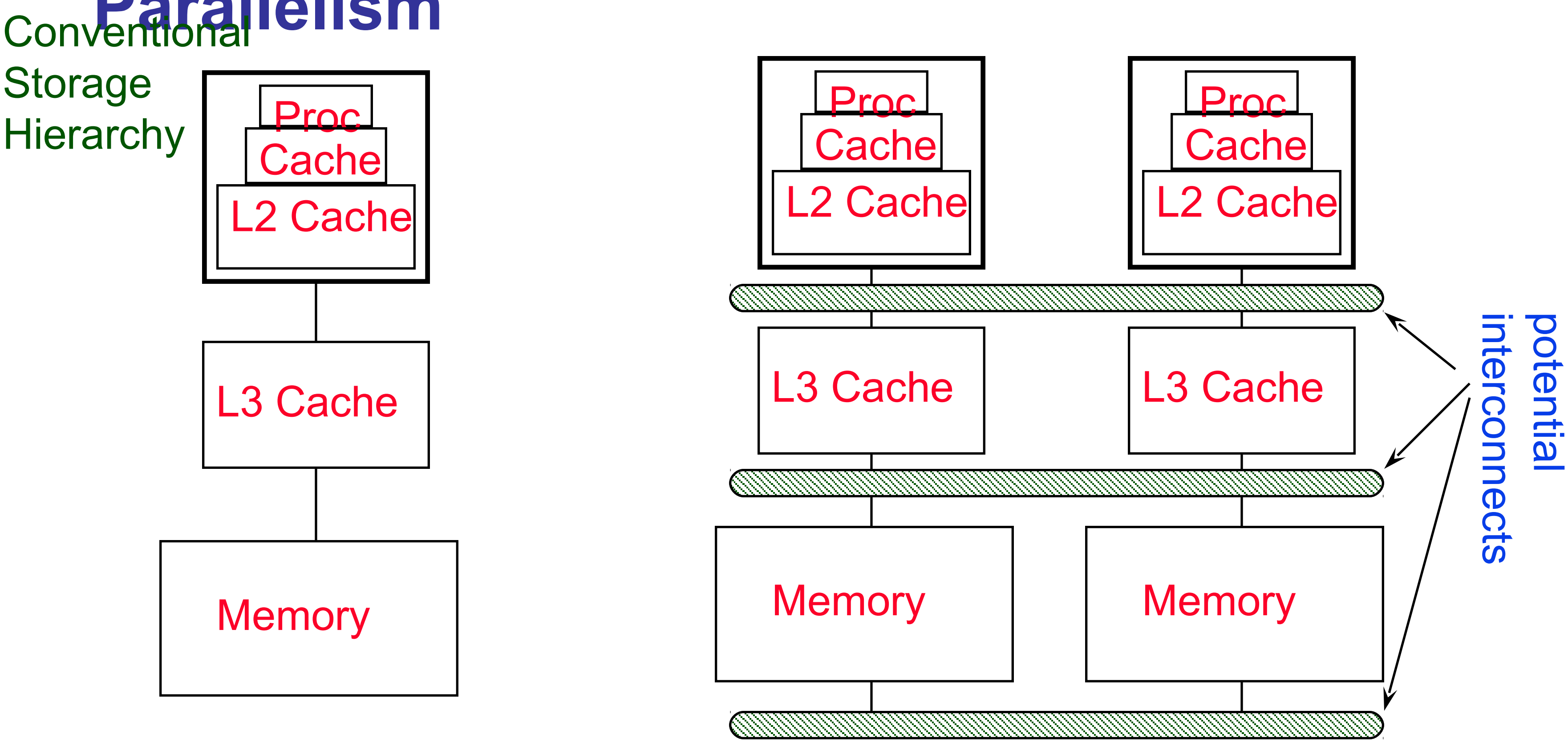
$$\text{Speedup}(P) = \text{Time}(1)/\text{Time}(P)$$
$$\le 1/(s + (1-s)/P)$$
$$\le 1/s$$

- Even if the parallel part speeds up perfectly performance is limited by the sequential part
- Top500 list: currently fastest machine has P~705K; 2nd fastest has ~186K+GPUs

Overhead of Parallelism

- Given enough parallel work, this is the biggest barrier to getting desired speedup
- Parallelism overheads include:
    - cost of starting a thread or process
    - cost of communicating shared data
    - cost of synchronizing
    - extra (redundant) computation
- Each of these can be in the range of milliseconds (=millions of flops) on some systems
- Tradeoff: Algorithm needs sufficiently large units of work to run fast in parallel (i.e. large granularity), but not so large that there is not enough parallel work

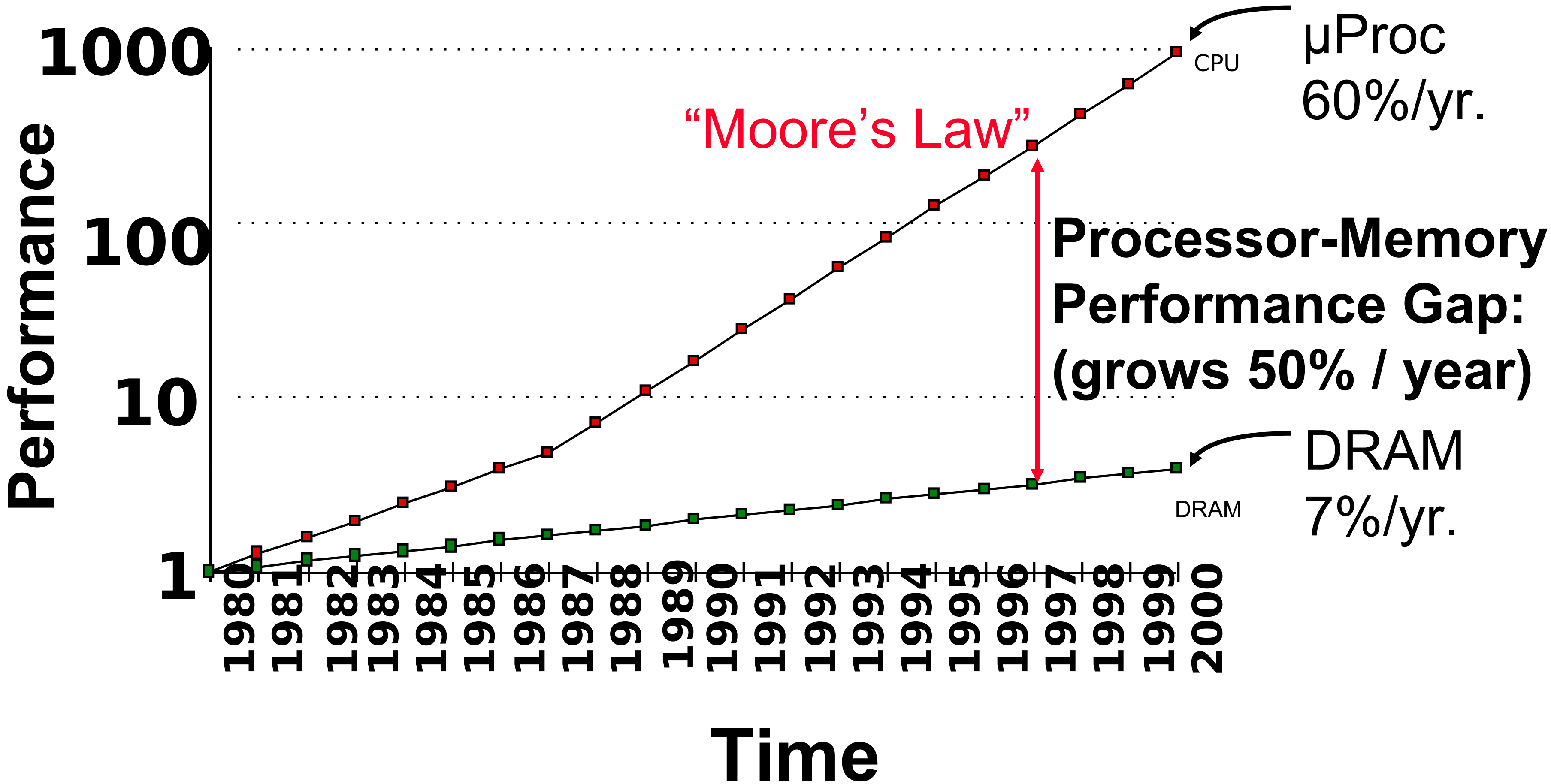
Locality and Parallelism



- Large memories are slow, fast memories are small
- Storage hierarchies are large and fast on average
- Parallel processors, collectively, have large, fast cache
  - the slow accesses to "remote" data we call "communication"
- Algorithm should do most work on local data

Processor-DRAM Gap (latency)

Goal: find algorithms that minimize communication, not necessarily arithmetic



Load Imbalance

- Load imbalance is the time that some processors in the system are idle due to
  - insufficient parallelism (during that phase)
  - unequal size tasks
- Examples of the latter
  - adapting to “interesting parts of a domain”
  - tree-structured computations
  - fundamentally unstructured problems
- Algorithm needs to balance load
  - Sometimes can determine work load, divide up evenly, before starting
    - “Static Load Balancing”
  - Sometimes work load changes dynamically, need to rebalance dynamically
    - “Dynamic Load Balancing”

Parallel Software Eventually – ParLab view

- 2 types of programmers ➔ 2 layers of software
- **Efficiency Layer** (10% of programmers)
  - Expert programmers build Libraries implementing kernels, “Frameworks”, OS, ….
  - Highest fraction of peak performance possible
- **Productivity Layer** (90% of programmers)
  - Domain experts / Non-expert programmers productively build parallel applications by composing frameworks & libraries
  - Hide as many details of machine, parallelism as possible
  - Willing to sacrifice some performance for productive programming
- Expect students may want to work at either level
  - In the meantime, we all need to understand enough of the efficiency layer to use parallelism effectively

Outline

- Why ~~powerful~~ all computers must be parallel processors
    - Including your laptops and handhelds
- Large CSE problems require powerful computers
    - Commercial problems too
- Why writing (fast) parallel programs is hard
    - But things are improving
- Structure of the course

Course Mechanics

- Web page:

  http://www.cs.berkeley.edu/~demmel/cs267_Spr12/

- Normally a mix of CS, EE, and other engineering and science students
- **Please fill out survey on web page (to be posted)**
- Grading:
  - Warmup assignment (homework 0 on the web)
    - **Build a web page on an interest of yours in CSE**
  - Three programming assignments
  - Final projects
    - Could be parallelizing an application, building or evaluating a tool, etc.
    - We encourage interdisciplinary teams, since this is the way parallel scientific software is generally built
- Class computer accounts on Hopper, Dirac at NERSC
  - Fill out forms next time

Remote instruction – preparing an experiment

- Lectures will be webcast, archived, as in past semesters
  - See class webpage for details
- XSEDE is nationwide project supporting users of NSF supercomputer facilities
  - XSEDE plans to offer CS267 to students nationwide
  - This semester, lectures will be editted, and homework ported to NSF supercomputer faciltiies
  - Challenges to “scaling up” education
    - Q&A – piazza or google groups? See class web page
    - Autograding
      - For correctness – run test cases (not as easy as it sounds)
      - For performance – timing on suitable platform
  - Ditto for Kurt Keutzer’s CS194 class
  - Thanks for participating in the development of this experiment!

Rough List of Topics

- Basics of computer architecture, memory hierarchies, performance
- Parallel Programming Models and Machines
  - Shared Memory and Multithreading
  - Distributed Memory and Message Passing
  - Data parallelism, GPUs
  - Cloud computing
- Parallel languages and libraries
  - Shared memory threads and OpenMP
  - MPI
  - Other Languages , frameworks (UPC, CUDA, PETSC, “Pattern Language”, …)
- “Seven Dwarfs” of Scientific Computing
  - Dense & Sparse Linear Algebra
  - Structured and Unstructured Grids
  - Spectral methods (FFTs) and Particle Methods
- 6 additional motifs
  - Graph algorithms, Graphical models, Dynamic Programming, Branch & Bound, FSM, Logic
- General techniques
  - Autotuning, Load balancing, performance tools
- Applications: climate modeling, nanoscience, biology … (guest lecturers)

Reading Materials

- What does Google recommend?
- Pointers on class web page
- Must read:
  - "The Landscape of Parallel Processing Research: The View from Berkeley"
    - http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf
- Some on-line texts:
  - Demmel's notes from CS267 Spring 1999, which are similar to 2000 and 2001. However, they contain links to html notes from 1996.
    - http://www.cs.berkeley.edu/~demmel/cs267_Spr99/
  - Ian Foster's book, "Designing and Building Parallel Programming".
    - http://www-unix.mcs.anl.gov/dbpp/
- Potentially useful texts:
  - "Sourcebook for Parallel Computing", by Dongarra, Foster, Fox, ..
    - A general overview of parallel computing methods
  - "Performance Optimization of Numerically Intensive Codes" by Stefan Goedecker and Adolfy Hoisie
    - This is a practical guide to optimization, mostly for those of you who have never done any optimization

Reading Materials (cont.)

- Recent books with papers about the current state of the art
  - David Bader (ed.), "Petascale Computing, Algorithms and Applications", Chapman & Hall/CRC, 2007
  - Michael Heroux, Padma Ragahvan, Horst Simon (ed.),"Parallel Processing for Scientific Computing", SIAM, 2006.
  - M. Sottile, T. Mattson, C. Rasmussen, Introduction to Concurrency in Programming Languages, Chapman & Hall/CRC, 2009.
- More pointers on the web page

Instructors

- Jim Demmel, EECS & Mathematics
- GSIs:
  - Nick Knight, EECS
  - Brian Van Straalen, LBNL & EECS
- Contact information on web page

Students

- 64 registered or on the waitlist (57 grad, 7 undergrad)
- 32 CS or EECS, rest from
  - Applied Mathematics
  - Applied Science & Technology (AS&T)
  - Astronomy
  - Bioengineering
  - Civil & Environmental Engineering
  - Industrial Eng and Operations Research
  - Information Management Systems
  - Integrative Biology
  - Mathematics
  - Mechanical Engineering
  - Nuclear Engineering
  - Physics

What you should get out of the course

In depth understanding of:

- When is parallel computing useful?
- Understanding of parallel computing hardware options.
- Overview of programming models (software) and tools, and experience using some of them
- Some important parallel applications and the algorithms
- Performance analysis and tuning
- Exposure to various open research questions

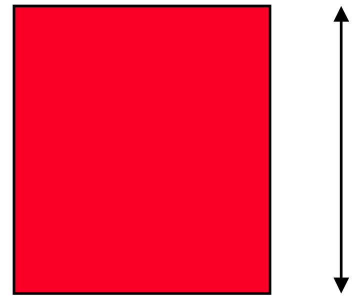
Extra slides

More Exotic Solutions on the Horizon

- Graphics and Game processors
  - Graphics Processing Units (GPUs), e.g., NVIDIA and ATI/AMD
  - Game processors, e.g., Cell for PS3
  - Parallel processor attached to main processor
  - Originally special purpose, getting more general
  - Programming model not yet mature
- FPGAs – Field Programmable Gate Arrays
  - Inefficient use of chip area
  - More efficient than multicore for some domains
  - Programming challenge now includes hardware design, e.g., layout
  - Wire routing heuristics still troublesome;
- Dataflow architectures
  - Have considerable experience with dataflow from 1980's
  - Programming with functional languages?

More Limits: How fast can a serial computer be?

1 Tflop/s, 1 Tbyte sequential machine



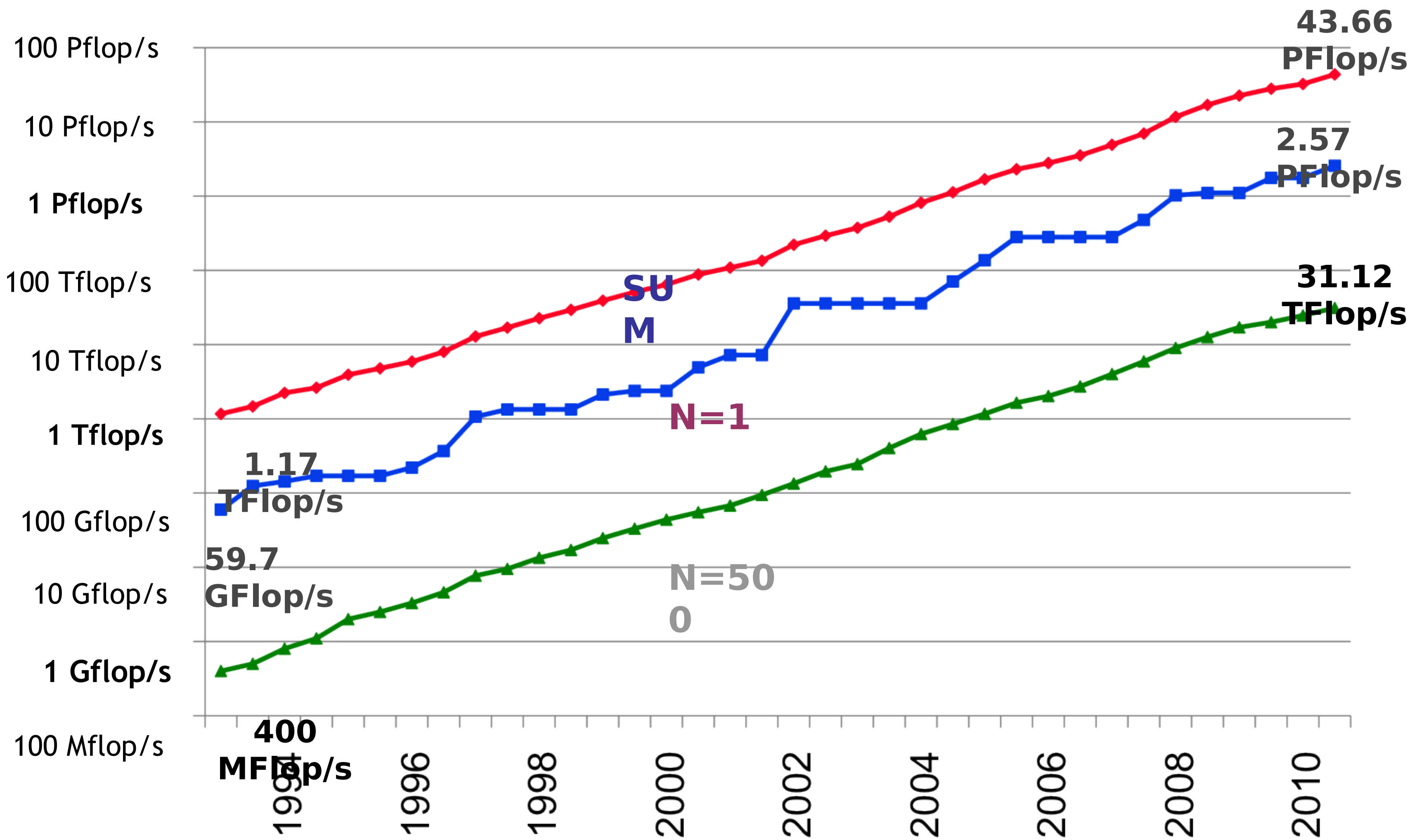
r = 0.3 mm

- Consider the 1 Tflop/s sequential machine:
  - Data must travel some distance, r, to get from memory to processor.
  - To get 1 data element per cycle, this means 10^{12} times per second at the speed of light, $c = 3\text{x}10^8$ m/s. Thus $r < c/10^{12} = 0.3$ mm.
- Now put 1 Tbyte of storage in a 0.3 mm x 0.3 mm area:
  - Each bit occupies about 1 square Angstrom, or the size of a small atom.
- No choice but parallelism

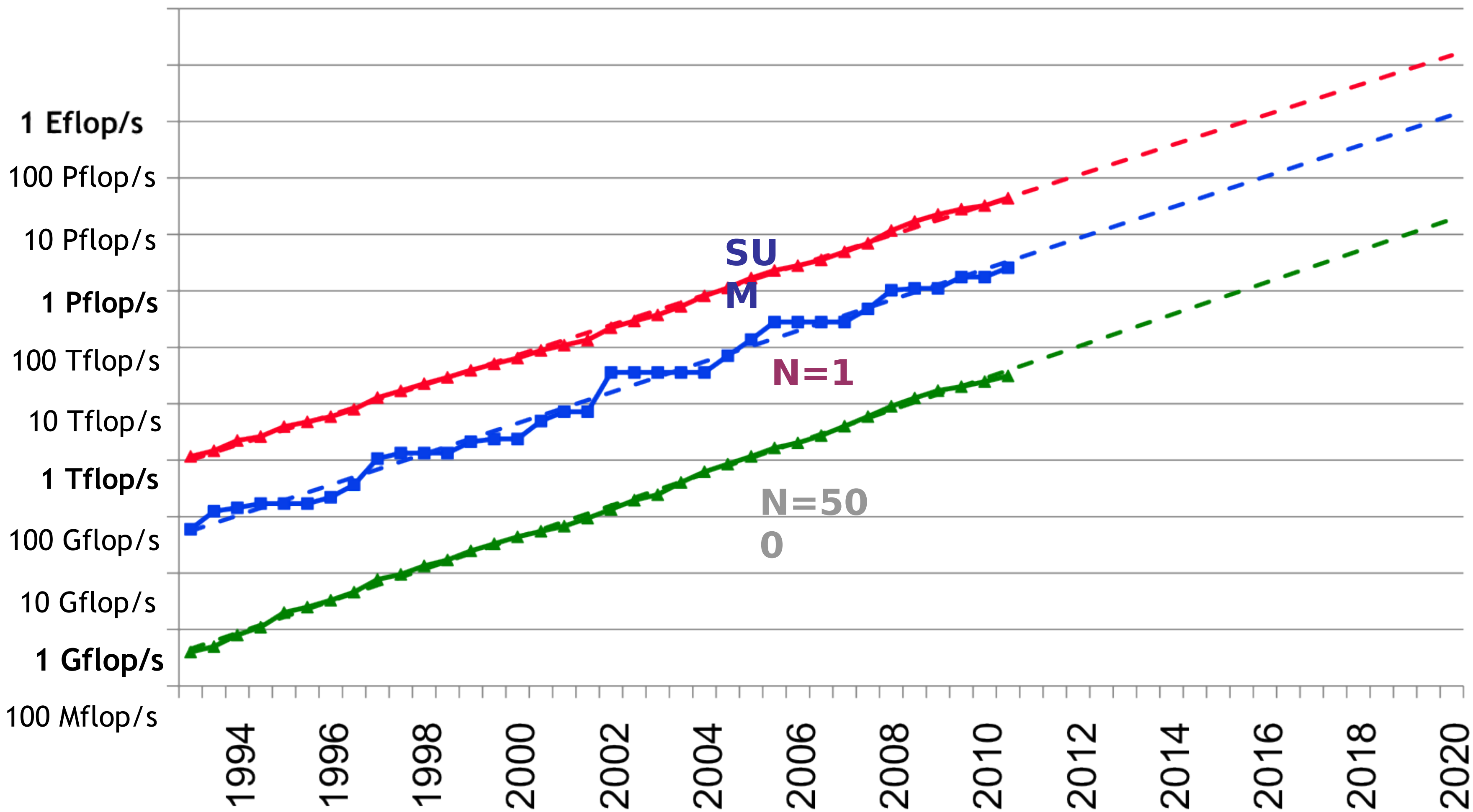
36th List: The TOP10

Rank	Site	Manufacturer	Computer	Country	Cores	Rmax [Tflops]	Power [MW]
1	National SuperComputer Center in Tianjin	NUDT	Tianhe-1A NUDT TH MPP, Xeon 6C, NVidia, FT-1000 8C	China	186,368	2,566	4.04
2	Oak Ridge National Laboratory	Cray	Jaguar Cray XT5, HC 2.6 GHz	USA	224,162	1,759	6.95
3	National Supercomputing Centre in Shenzhen	Dawning	Nebulae TC3600 Blade, Intel X5650, NVidia Tesla C2050 GPU	China	120,640	1,271	2.58
4	GSIC, Tokyo Institute of Technology	NEC/HP	TSUBAME-2 HP ProLiant, Xeon 6C, NVidia, Linux/Windows	Japan	73,278	1,192	1.40
5	DOE/SC/ LBNL/NERSC	Cray	Hopper Cray XE6, 6C 2.1 GHz	USA	153,408	1.054	2.91
6	Commissariat a l'Energie Atomique (CEA)	Bull	Tera 100 Bull bullx super-node S6010/S6030	France	138.368	1,050	4.59
7	DOE/NNSA/LANL	IBM	Roadrunner BladeCenter QS22/LS21	USA	122,400	1,042	2.34
8	University of Tennessee	Cray	Kraken Cray XT5 HC 2.36GHz	USA	98,928	831.7	3.09
9	Forschungszentrum Juelich (FZJ)	IBM	Jugene Blue Gene/P Solution	Germany	294,912	825.5	2.26
10	DOE/NNSA/ LANL/SNL	Cray	Cielo Cray XE6, 6C 2.4 GHz	USA	107,152	816.6	2.95

Performance Development



Projected Performance Development



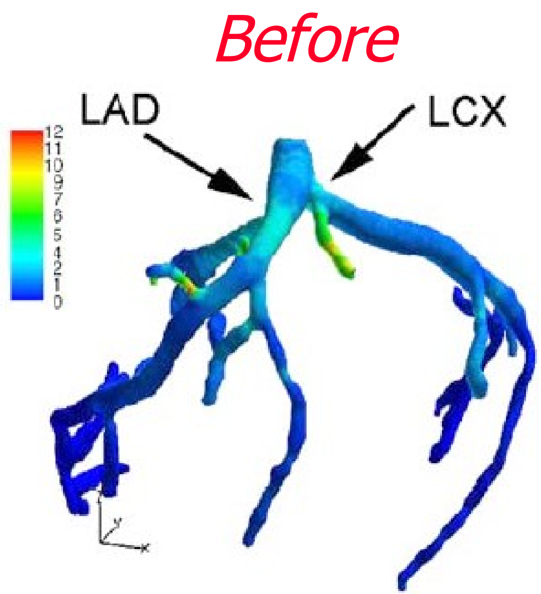
Compelling Laptop/Handheld Apps (David Wessel)

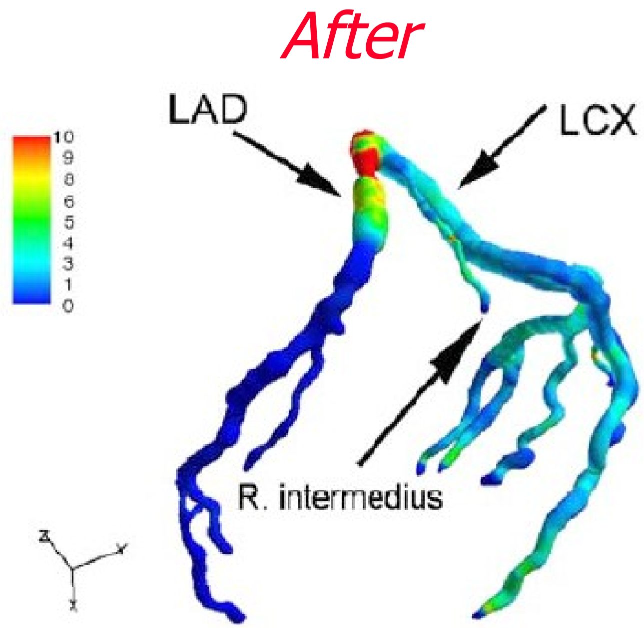
- Musicians have an insatiable appetite for computation + real-time demands
  - More channels, instruments, more processing, more interaction!
  - Latency must be low (5 ms)
  - Must be reliable (No clicks)

1. Music Enhancer
   - Enhanced sound delivery systems for home sound systems using large microphone and speaker arrays
   - Laptop/Handheld recreate 3D sound over ear buds
1. Hearing Augmenter
   - Laptop/Handheld as accelerator for hearing aide
1. Novel Instrument User Interface
   - New composition and performance systems beyond keyboards
   - Input device for Laptop/Handheld

Berkeley Center for New Music and Audio Technology (CNMAT) created a compact loudspeaker array: 10-inch-diameter icosahedron incorporating 120 tweeters.

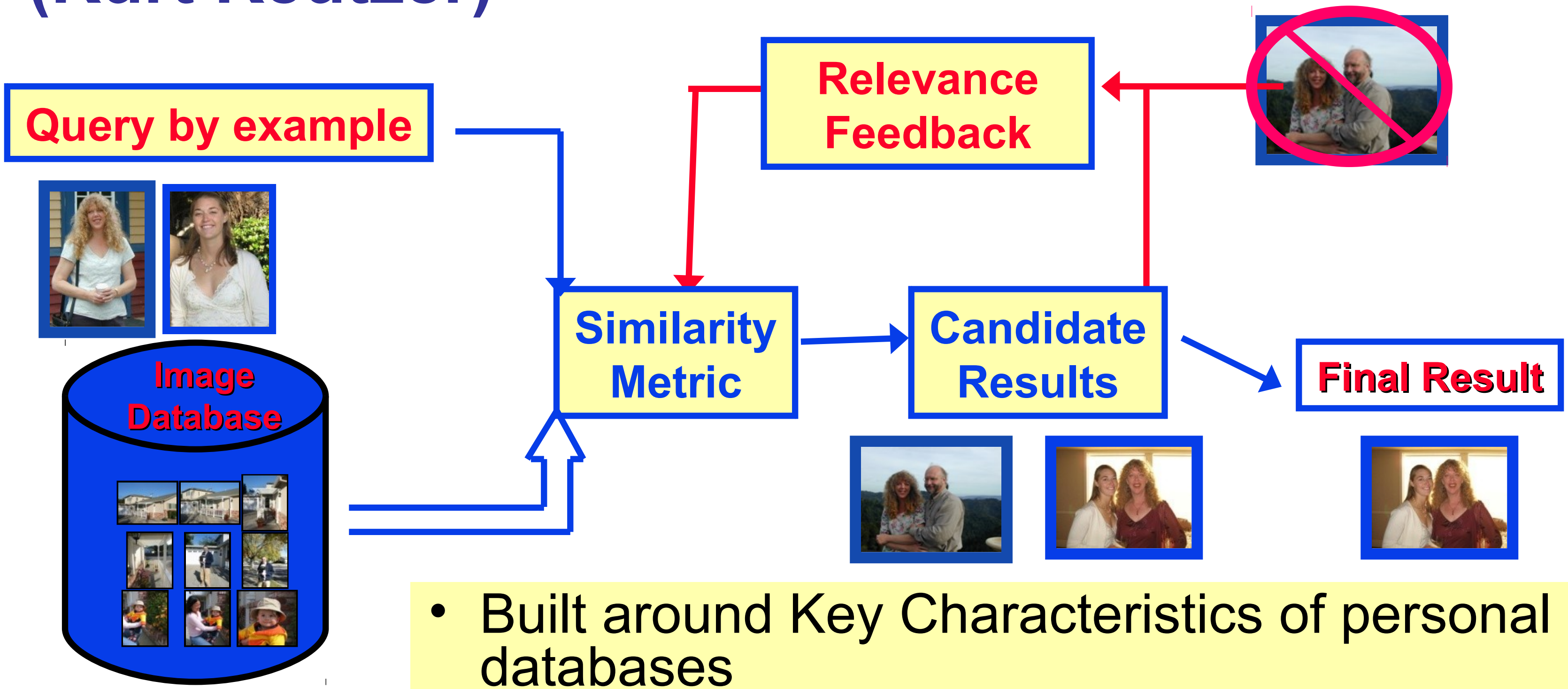
Coronary Artery Disease (Tony Keaveny)





- Modeling to help patient compliance?
  - 450k deaths/year, 16M symptomatic, 72M High BP
- Massively parallel, Real-time variations
  - CFD FE solid (non-linear), fluid (Newtonian), pulsatile
  - Blood pressure, activity, habitus, cholesterol

Content-Based Image Retrieval (Kurt Keutzer)

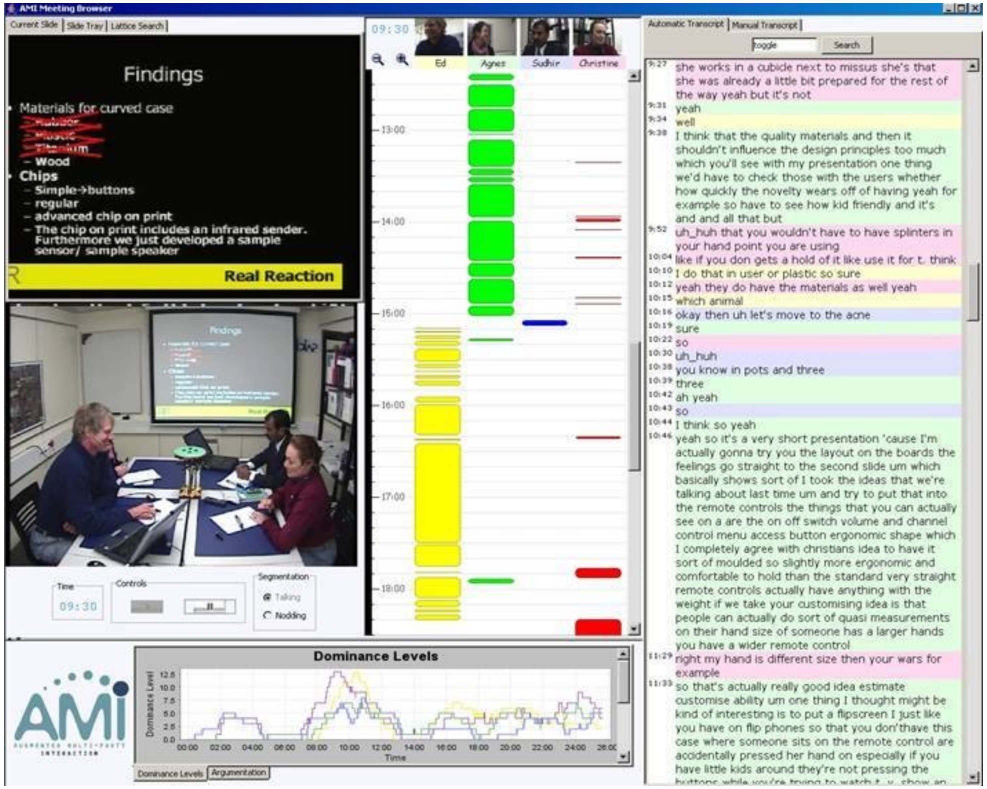


1000's of images

- Built around Key Characteristics of personal databases
  - Very large number of pictures (>5K)
  - Non-labeled images
  - Many pictures of few people
  - Complex pictures including people, events, places, and objects

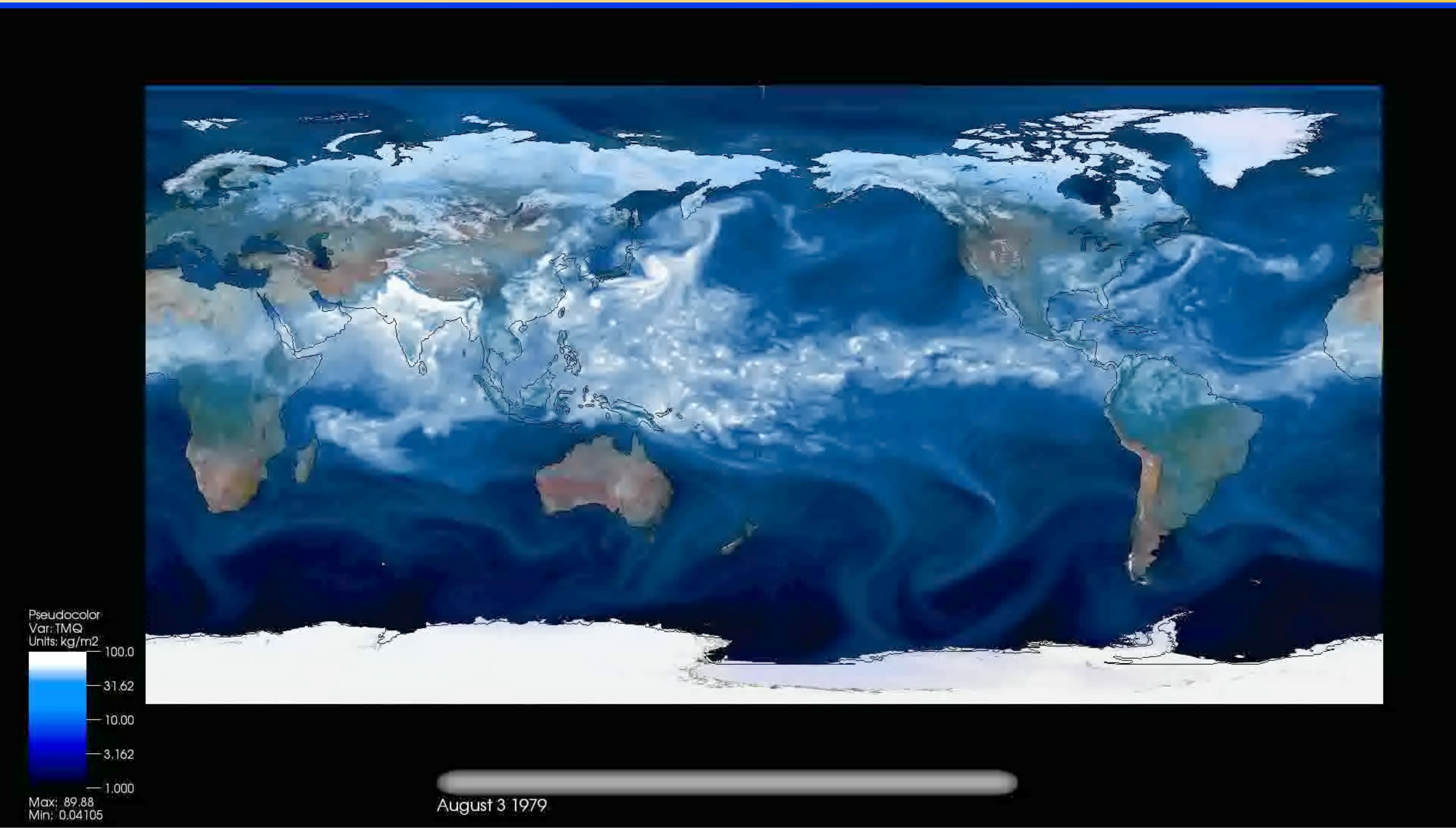
Compelling Laptop/Handheld Apps (Nelson Morgan)

- Meeting Diarist
  - Laptops/ Handhelds at meeting coordinate to create speaker identified, partially transcribed text diary of meeting



- Teleconference speaker identifier, speech helper
  - L/Hs used for teleconference, identifies who is speaking, “closed caption” hint of what being said

U.S.A. Hurricane



Source: Data from M.Wehner, visualization by Prabhat, LBNL

Outline

- Why ~~powerful~~ all computers must be parallel processors
    - Including your laptops and handhelds
- Large CSE problems require powerful computers
    - Commercial problems too
- Why writing (fast) parallel programs is hard
    - But things are improving
- Principles of parallel computing performance
- Structure of the course

Improving Real Performance

Peak Performance grows exponentially, a la Moore's Law

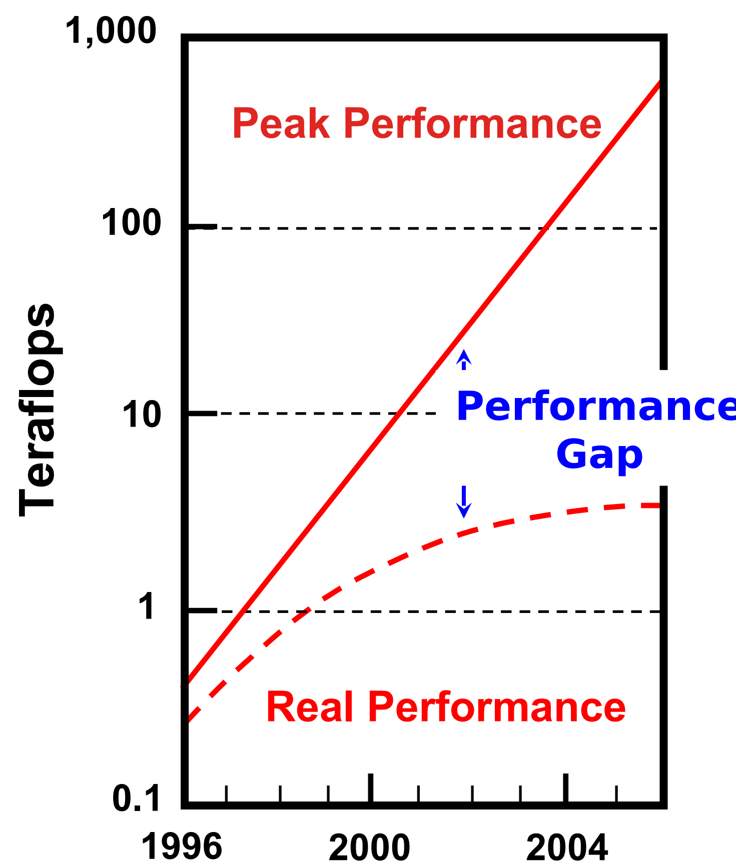
- In 1990's, peak performance increased 100x; in 2000's, it will increase 1000x

But efficiency (the performance relative to the hardware peak) has declined

- was 40-50% on the vector supercomputers of 1990s
- now as little as 5-10% on parallel supercomputers of today

Close the gap through ...

- Mathematical methods and algorithms that achieve high performance on a single processor and scale to thousands of processors
- More efficient programming models and tools for massively parallel supercomputers



Performance Levels

- Peak performance
  - Sum of all speeds of all floating point units in the system
  - You can't possibly compute faster than this speed
- LINPACK
  - The "hello world" program for parallel performance
  - Solve Ax=b using Gaussian Elimination, highly tuned
- Gordon Bell Prize winning applications performance
  - The right application/algorithm/platform combination plus years of work
- Average sustained applications performance
  - What one reasonable can expect for standard applications

When reporting performance results, these levels are often confused, even in reviewed publications

Performance Levels (for example on NERSC-5)

- Peak advertised performance (PAP): 100 Tflop/s
- LINPACK (TPP): 84 Tflop/s
- Best climate application: 14 Tflop/s
  - WRF code benchmarked in December 2007
- Average sustained applications performance: ? Tflop/s
  - Probably less than 10% peak!
- We will study performance
  - Hardware and software tools to measure it
  - Identifying bottlenecks
  - Practical performance tuning (Matlab demo)

Computational Science and Engineering (CSE)

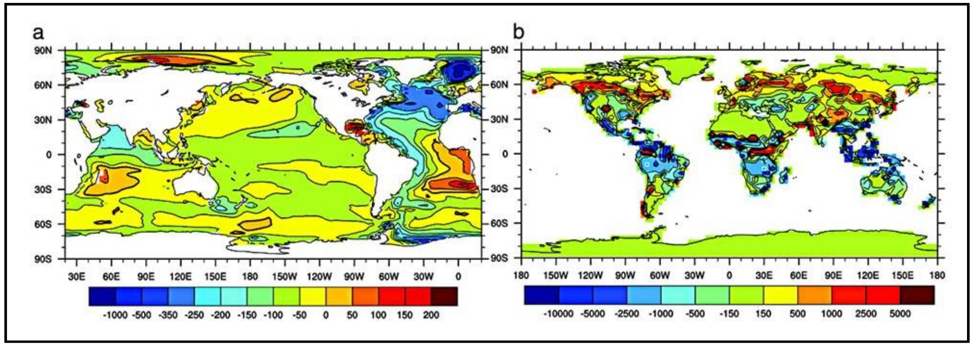
- CSE is a widely accepted label for an evolving field concerned with the science of and the engineering of systems and methodologies to solve computational problems arising throughout science and engineering
- CSE is characterized by
  - Multi - disciplinary
  - Multi - institutional
  - Requiring high-end resources
  - Large teams
  - Focus on community software
- CSE is not "just programming" (and not CS)
- Fast computers necessary but not sufficient
- Graduate program in CSE at UC Berkeley

Reference: Petzold, L., *et al.*, Graduate Education in CSE, *SIAM Rev.*, 43(2001), 163-177

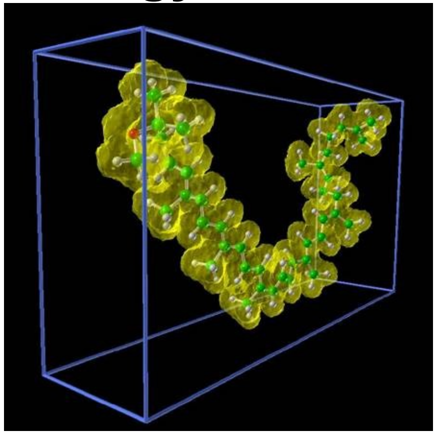
SciDAC - First Federal Program to Implement CSE

- **SciDAC (Scientific Discovery through Advanced Computing) program created in 2001**
  - **About $50M annual funding**
  - **Berkeley (LBNL+UCB) largest recipient of SciDAC funding**

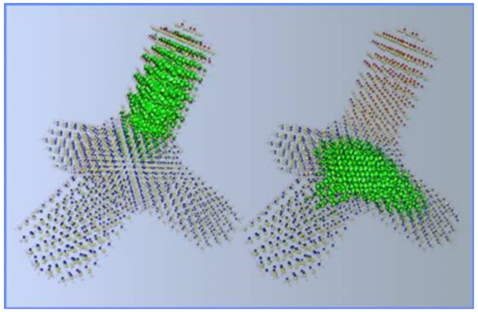
Global Climate



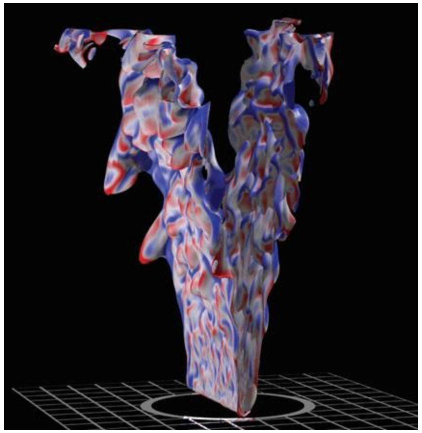
Biology



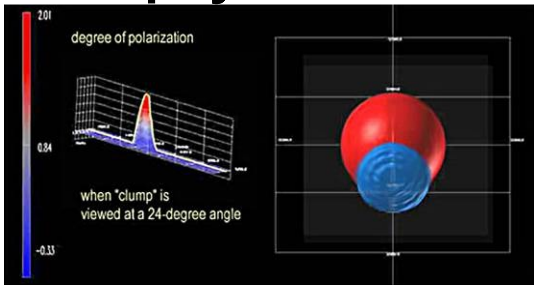
Nanoscience



Combustion

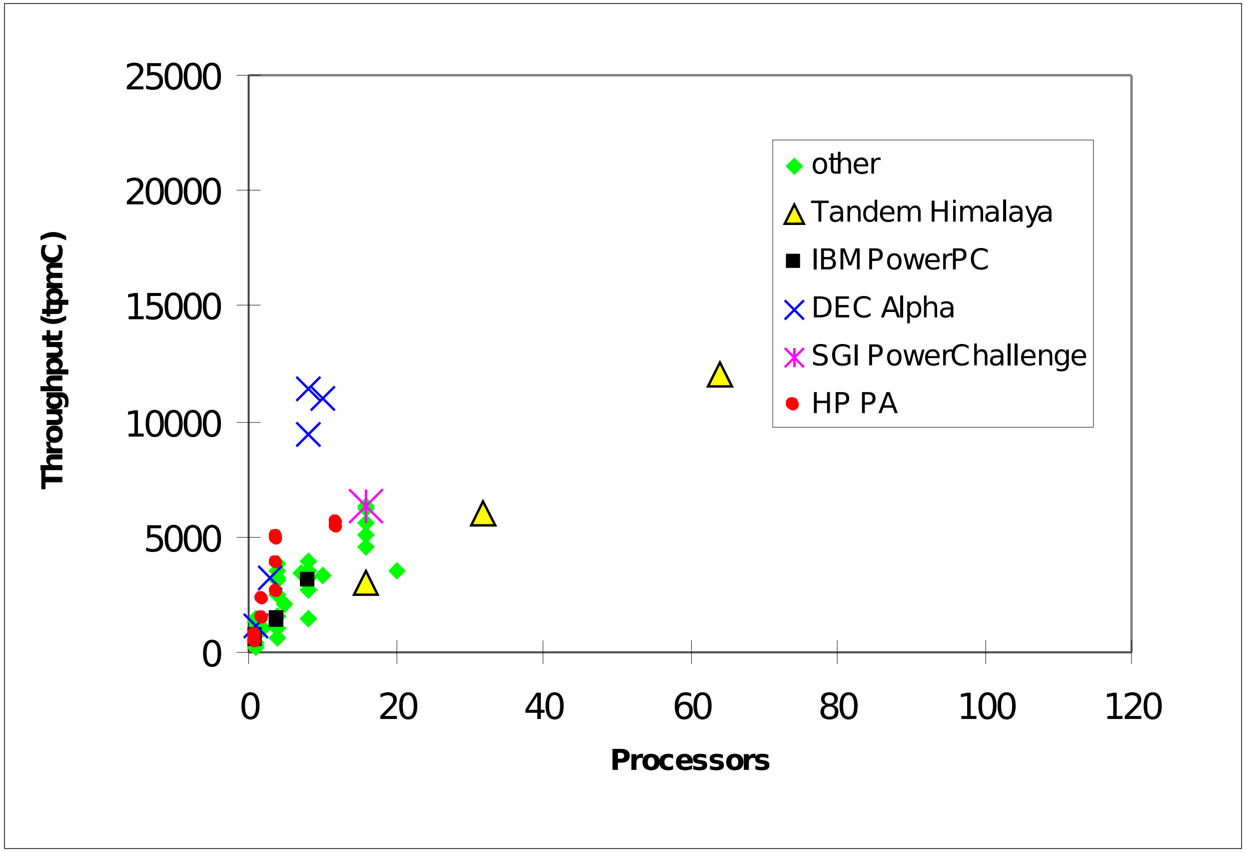


Astrophysics



Transaction Processing

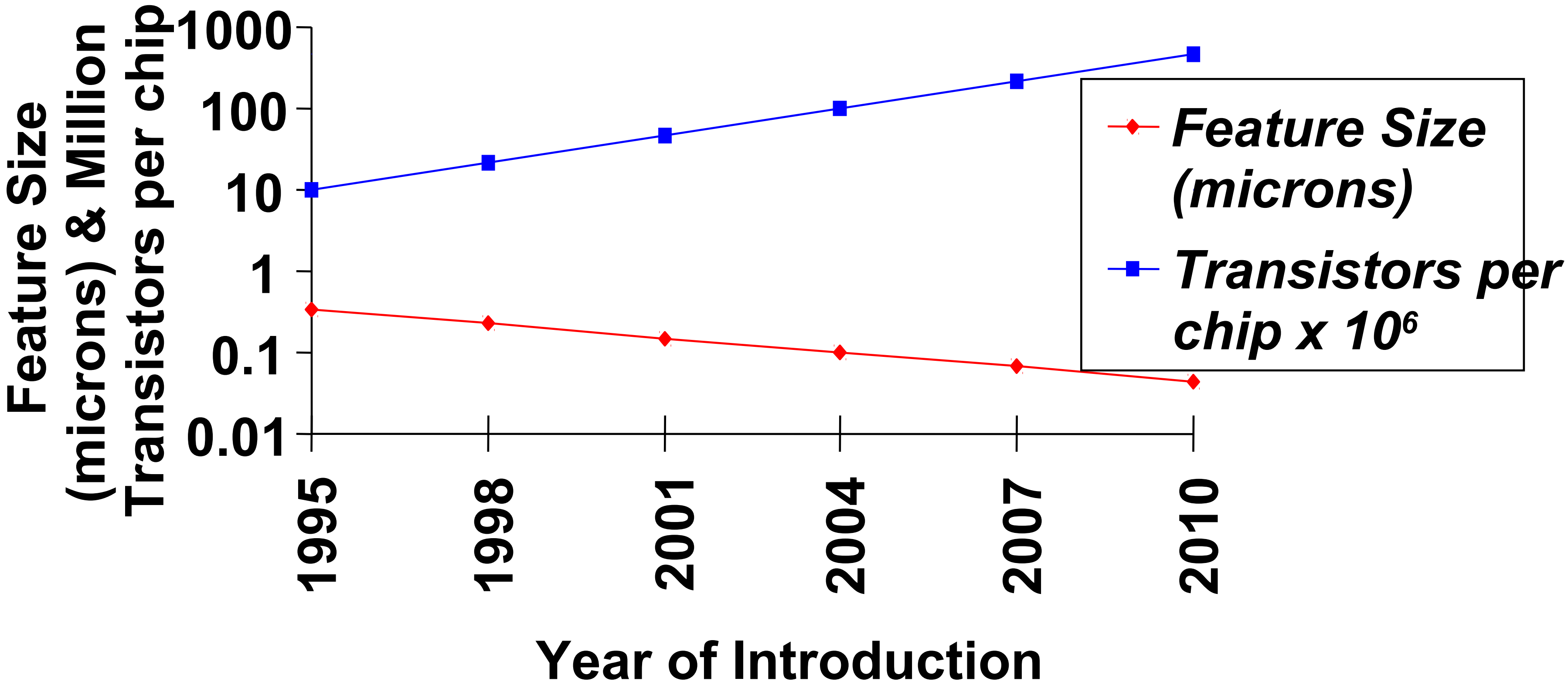
(mar. 15, 1996)



- Parallelism is natural in relational operators: select, join, etc.
- Many difficult issues: data partitioning, locking, threading.

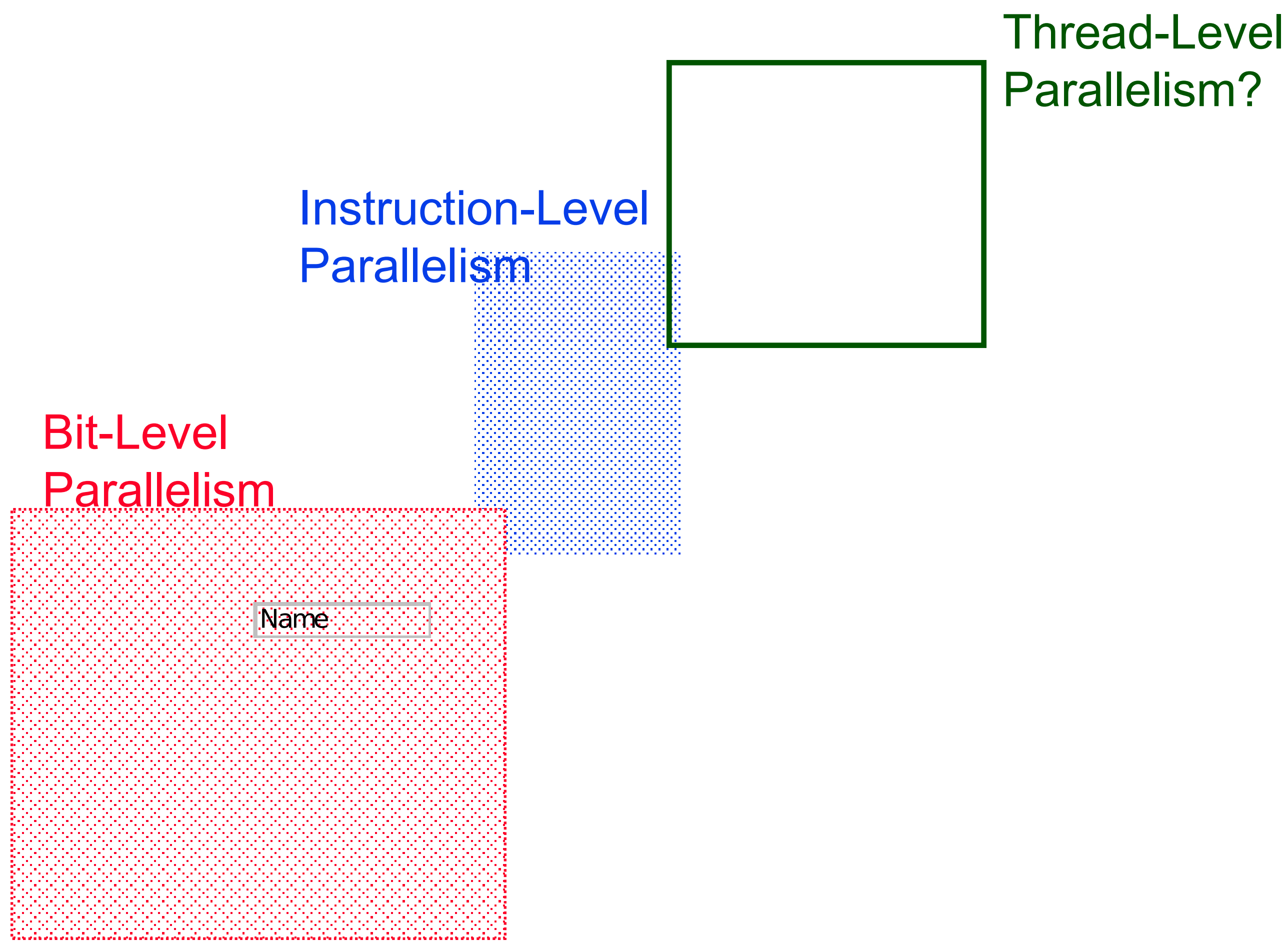
SIA Projections for Microprocessors

Compute power ~1/(Feature Size)^3

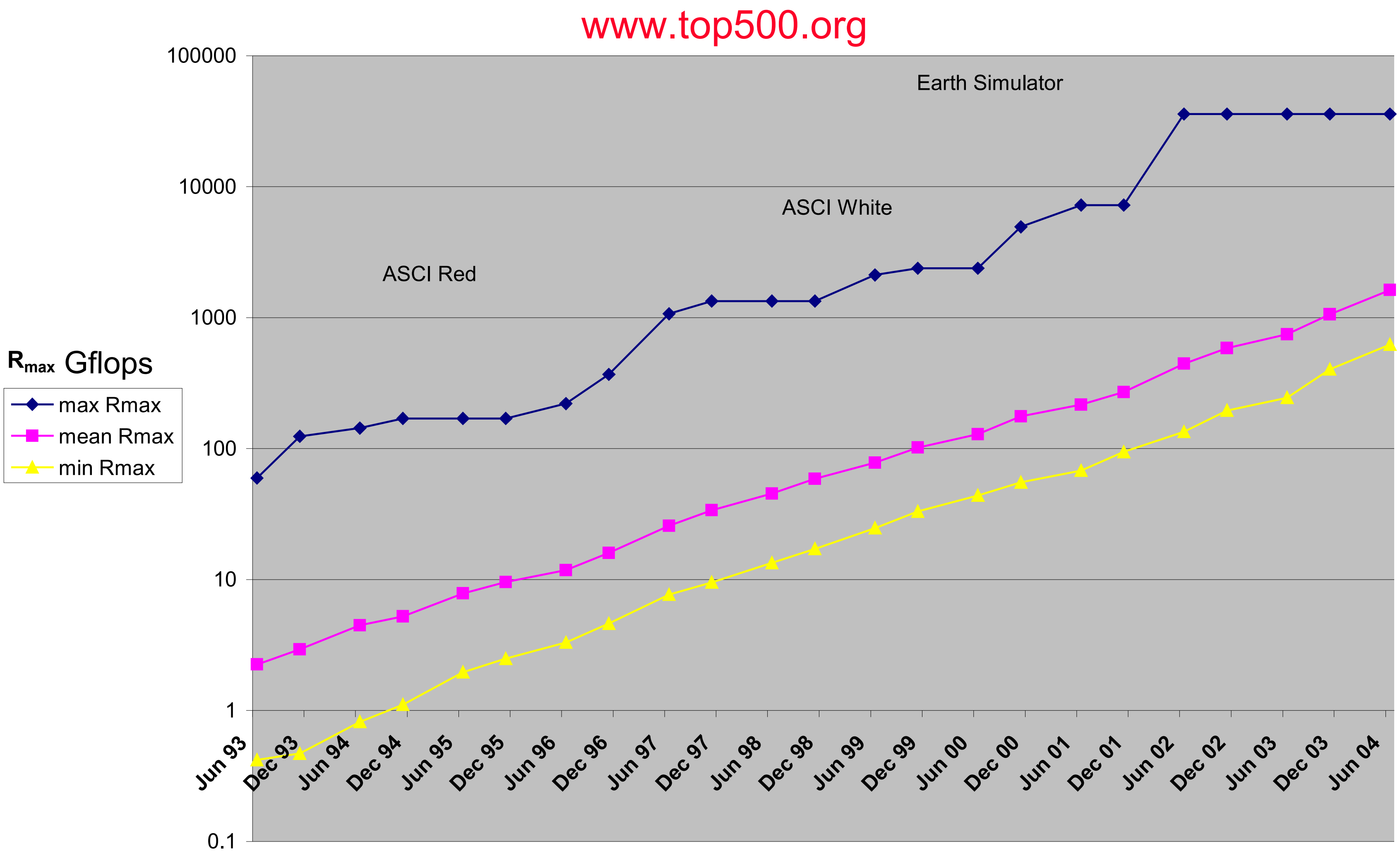


based on F.S.Preston, 1997

Much of the Performance is from Parallelism

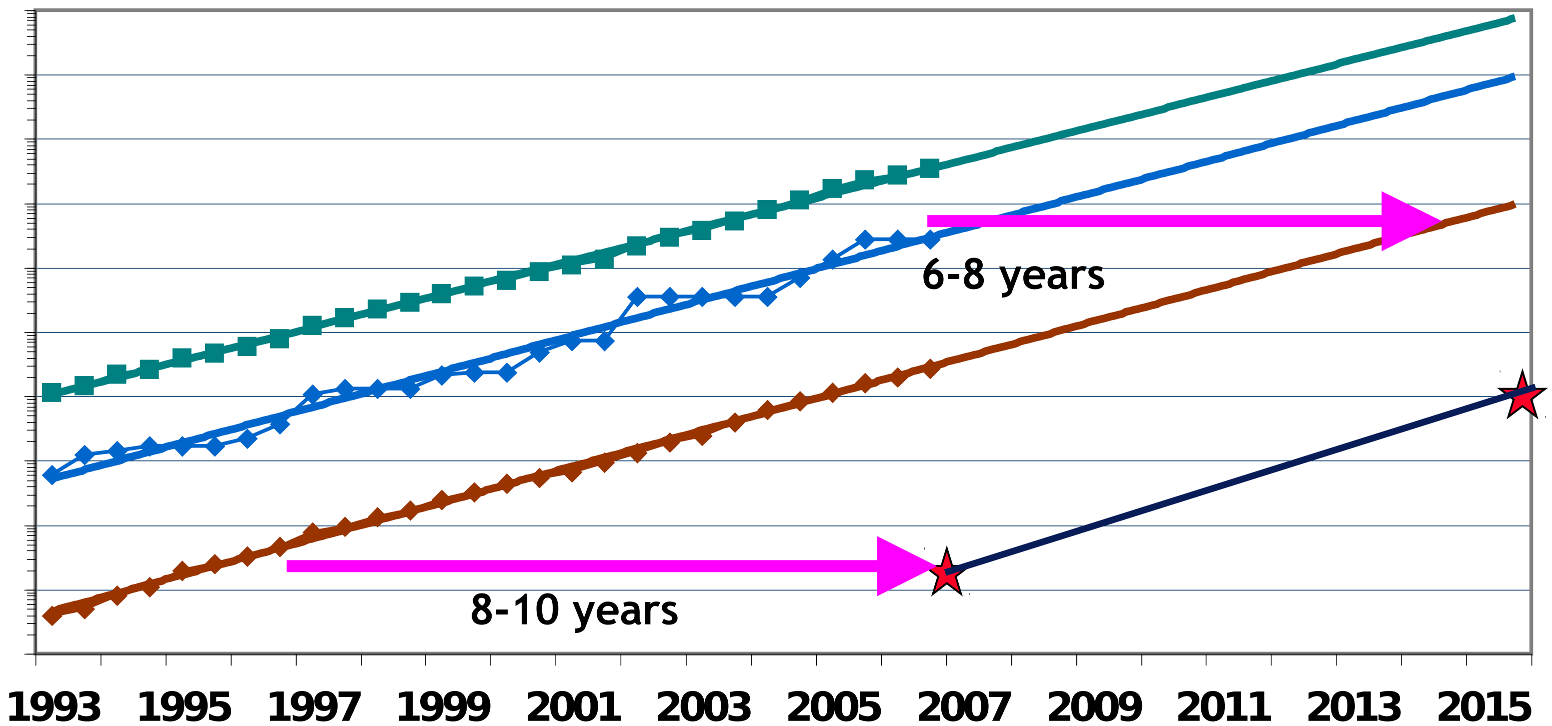


Performance on Linpack Benchmark



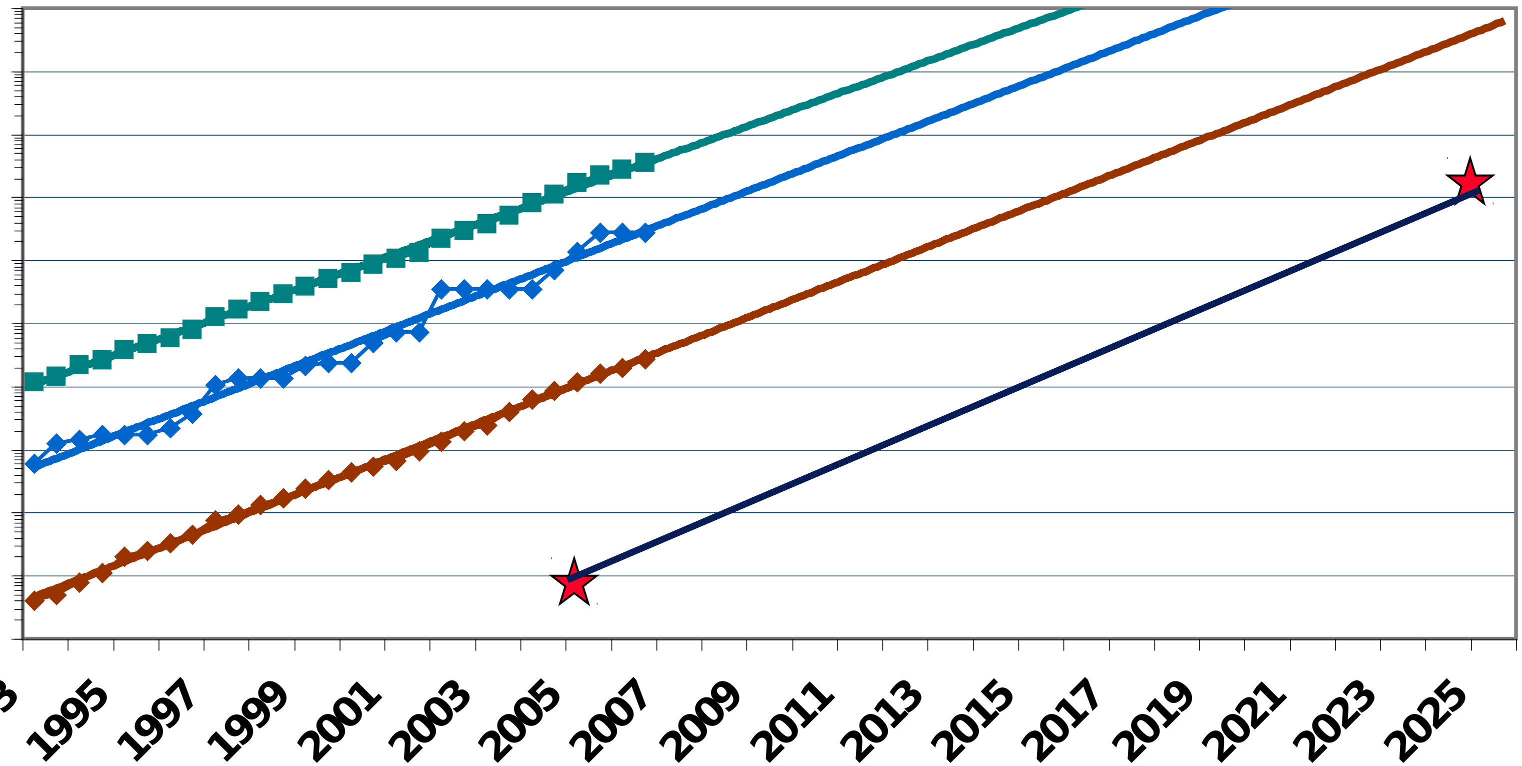
Nov 2004: IBM Blue Gene L, 70.7 Tflops Rmax

Performance Projection



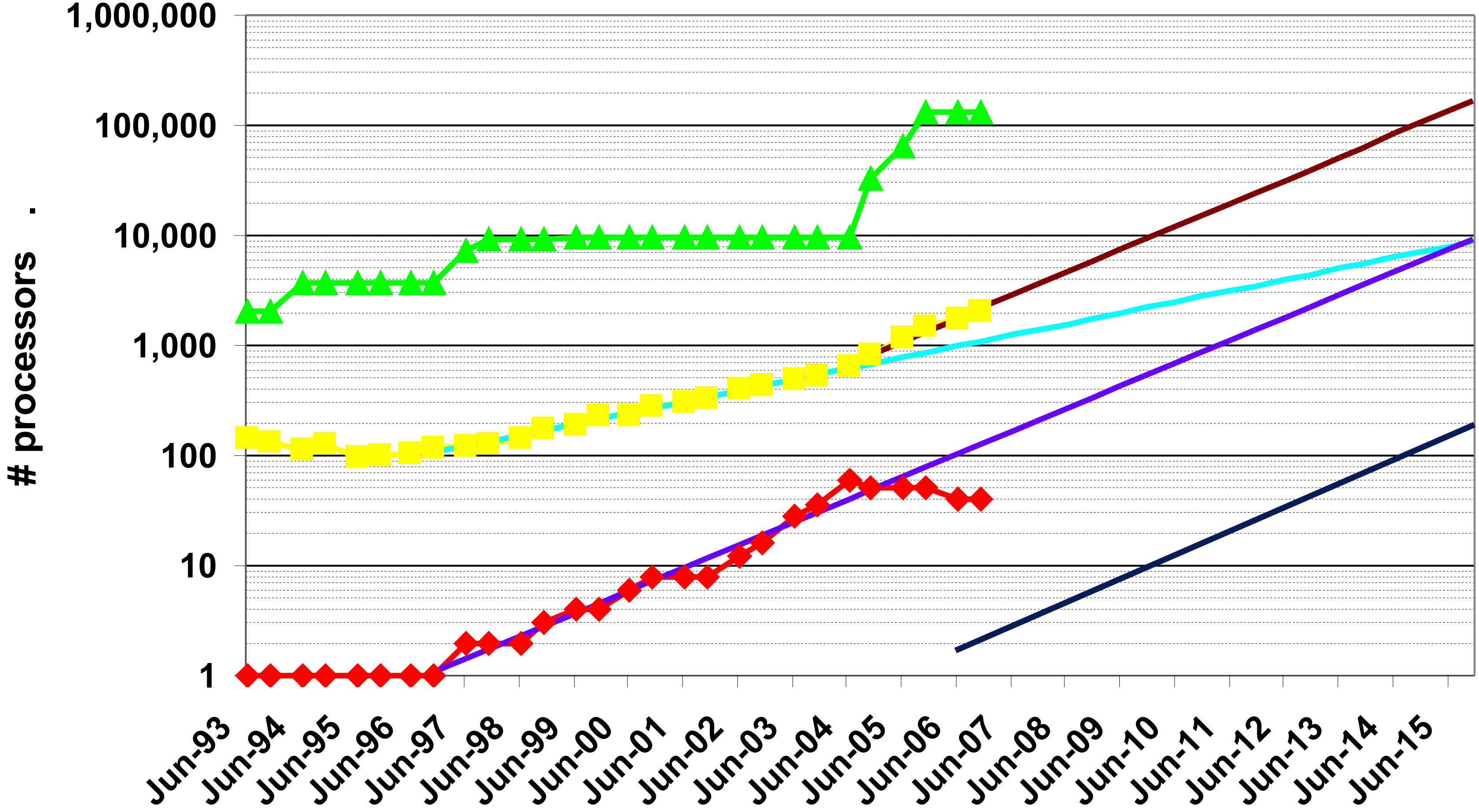
Slide by Erich Strohmaier, LBNL

Performance Projection



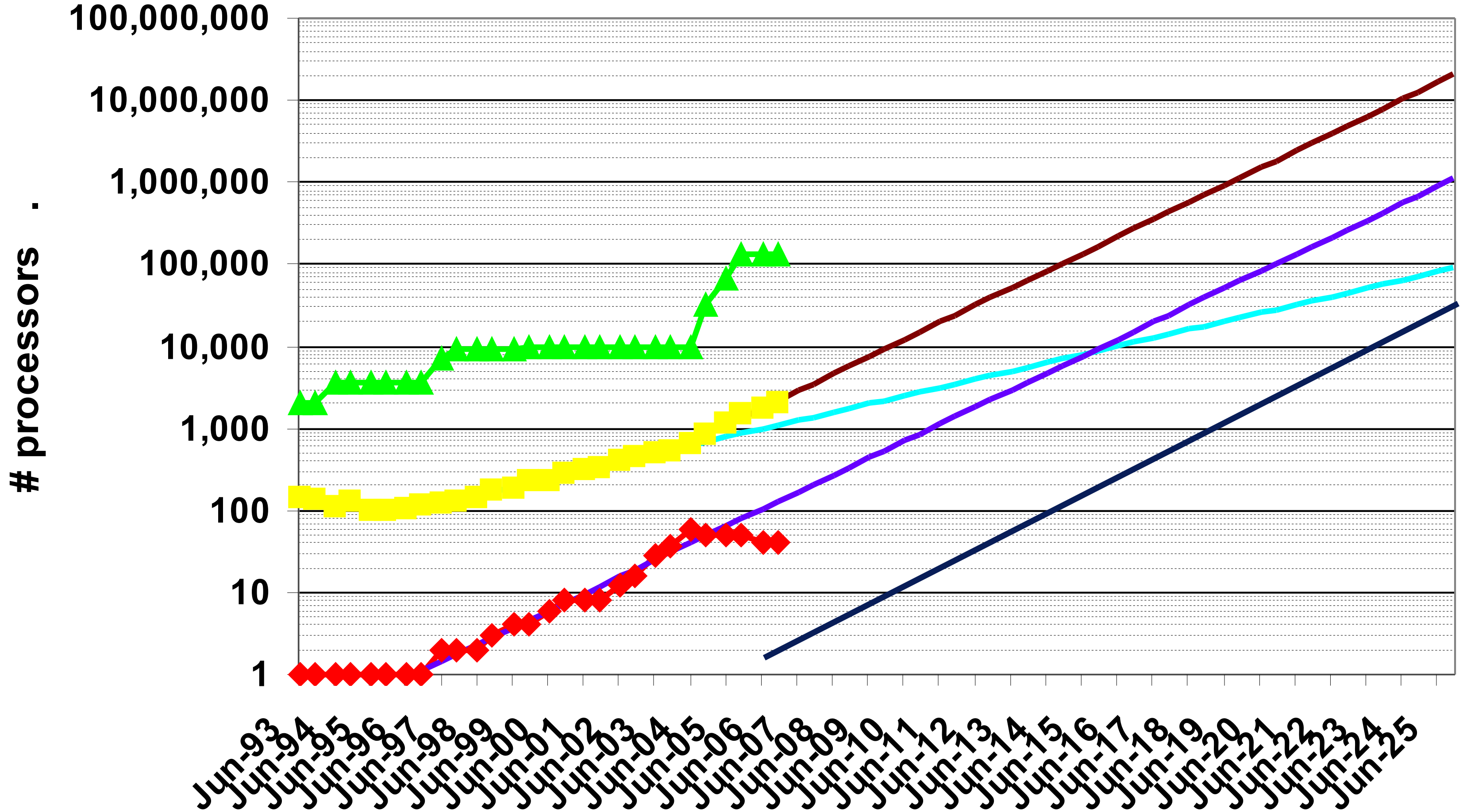
Slide by Erich Strohmaier, LBNL

Concurrency Levels



Slide by Erich Strohmaier, LBNL

Concurrency Levels- There is a Massively Parallel System Also in Your Future



Slide by Erich Strohmaier, LBNL

Supercomputing Today

- Microprocessors have made desktop computing in 2007 what supercomputing was in 1995.
- Massive Parallelism has changed the “high-end” completely.
- Most of today's standard supercomputing architecture are “hybrids”, clusters built out of commodity microprocessors and custom interconnects.
- The microprocessor revolution will continue with little attenuation for at least another 10 years
- The future will be massively parallel, based on multicore

Outline

- Why ~~powerful~~ all computers must be parallel computers

  Including your laptop and handhelds

- Large important problems require powerful computers

  Even computer games

- Why writing (fast) parallel programs is hard

  But things are improving

- Principles of parallel computing performance

- Structure of the course

Is Multicore the Correct Response?

- **Kurt Keutzer:** “This shift toward increasing parallelism is not a triumphant stride forward based on breakthroughs in novel software and architectures for parallelism; instead, this plunge into parallelism is actually a retreat from even greater challenges that thwart efficient silicon implementation of traditional uniprocessor architectures.”

- **David Patterson:** “Industry has already thrown the hail-mary pass. . . But nobody is running yet.”

Community Reaction

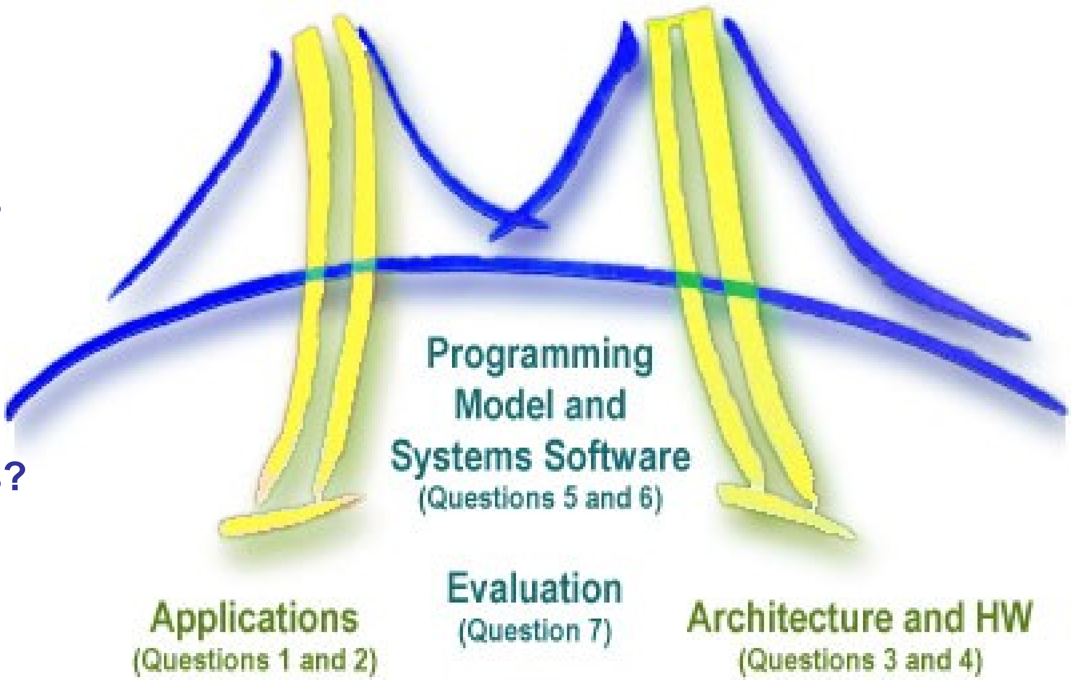
- Desktop/Consumer
  - **Move from almost no parallelism to parallelism**
  - **But industry is already betting on parallelism (multicore) for its future**

- HPC
  - **Modest growth in parallelism is giving way to exponential growth curve**
  - **Have Parallel programming tools and algorithms, but driven by experts (unlikely to be adopted by broader software development community)**

- The first hardware is here, but have no consensus on hardware details or software model necessary to program it
  - Reaction: Widespread Panic!

The View from Berkeley: Seven Questions for Parallelism

- Applications:
  - **1. What are the apps?**
  - **2. What are kernels of apps?**
- Hardware:
  - **3. What are the HW building blocks?**
  - **4. How to connect them?**
- Programming Model / Systems Software:
  - **5. How to describe apps and kernels?**
  - **6. How to program the HW?**
- Evaluation:
  - **7. How to measure success?**



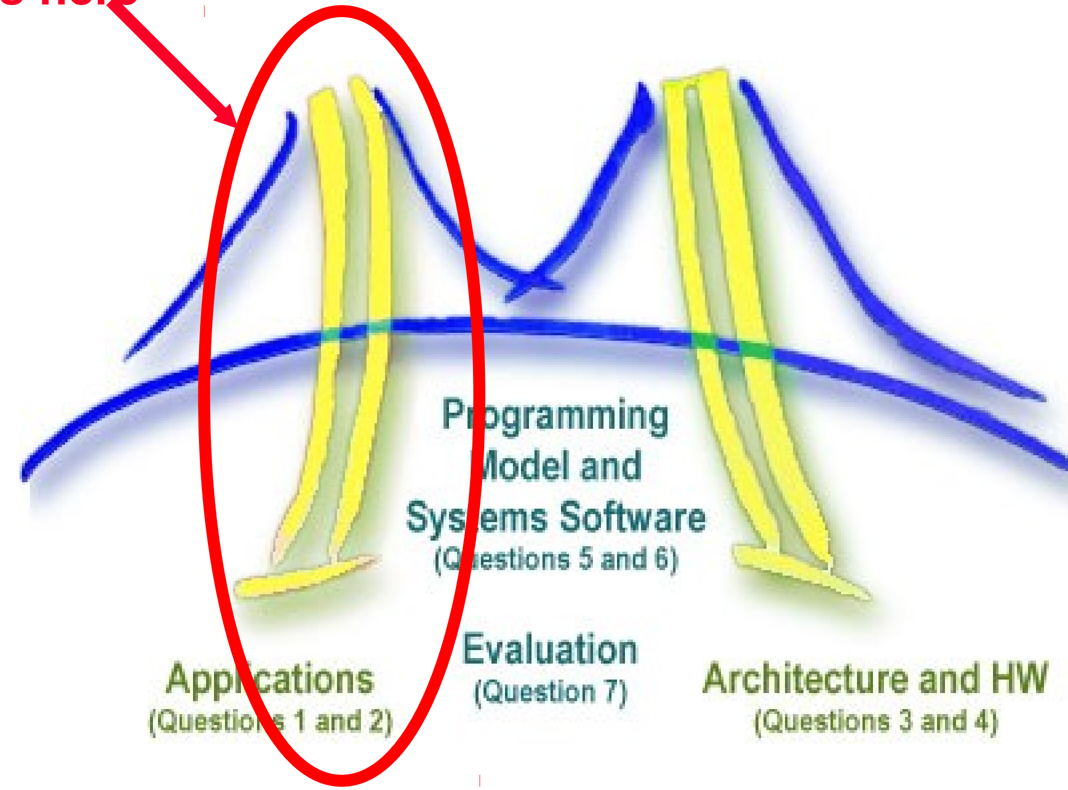
(Inspired by a view of the Golden Gate Bridge from Berkeley)

http://www.eecs.berkeley.edu/Pubs/TechRpts/2006/EECS-2006-183.pdf

Applications

- Applications:
  - **1. What are the apps?**
  - **2. What are kernels of apps?**
- Hardware:
  - **3. What are the HW building blocks?**
  - **4. How to connect them?**
- Programming Model / Systems Software:
  - **5. How to describe apps and kernels?**
  - **6. How to program the HW?**
- Evaluation:
  - **7. How to measure success?**

CS267 focus is here



(Inspired by a view of the Golden Gate Bridge from Berkeley)

Much Ado about ~~Dwarves~~ Motifs

High-end simulation in the physical sciences = 7 numerical methods:

1. **Structured Grids (including locally structured grids, e.g. Adaptive Mesh Refinement)**
2. **Unstructured Grids**
3. **Fast Fourier Transform**
4. **Dense Linear Algebra**
5. **Sparse Linear Algebra**
6. **Particles**
7. **Monte Carlo** → Map Reduce

- ***Benchmarks enable assessment of hardware performance improvements***
- ***The problem with benchmarks is that they enshrine an implementation***
- ***At this point in time, we need flexibility to innovate both implementation and the hardware they run on!***
- ***Dwarves provide that necessary abstraction***

Slide from "Defining Software Requirements for Scientific Computing", Phillip Colella, 2004

Do dwarfs work well outside HPC?

- Examine effectiveness 7 dwarfs elsewhere
  1. **Embedded Computing (EEMBC benchmark)**
  2. **Desktop/Server Computing (SPEC2006)**
  3. **Data Base / Text Mining Software**
     - **Advice from Jim Gray of Microsoft and Joe Hellerstein of UC**
  1. **Games/Graphics/Vision**
  2. **Machine Learning**
     - **Advice from Mike Jordan and Dan Klein of UC Berkeley**

- Result: Added 7 more dwarfs, revised 2 original dwarfs, renumbered list

Destination is Manycore

- We need revolution, not evolution
- Software or architecture alone can't fix parallel programming problem, need innovations in both
- "Multicore" 2X cores per generation: 2, 4, 8, …
- "Manycore" 100s is highest performance per unit area, and per Watt, then 2X per generation:
64, 128, 256, 512, 1024 …
- **Multicore architectures & Programming Models good for 2 to 32 cores won't evolve to Manycore systems of 1000's of processors**
⇒ Desperately need HW/SW models that work for Manycore or will run out of steam
(as ILP ran out of steam at 4 instructions)

Units of Measure in HPC

- High Performance Computing (HPC) units are:
  - Flop: floating point operation
  - Flops/s: floating point operations per second
  - Bytes: size of data (a double precision floating point number is 8)
- Typical sizes are millions, billions, trillions…

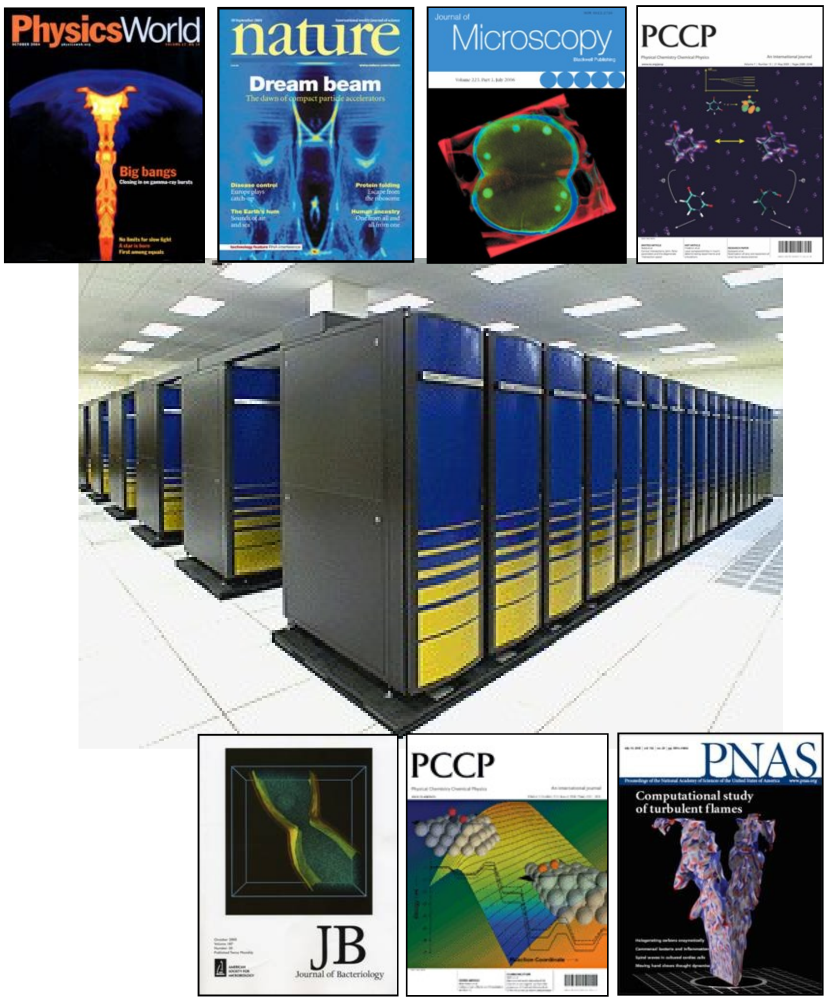
Mega	Mflop/s = 10^6 flop/sec	Mbyte = 2^{20} = 1048576 ~ 10^6 bytes
Giga	Gflop/s = 10^9 flop/sec	Gbyte = 2^{30} ~ 10^9 bytes
Tera	Tflop/s = 10^{12} flop/sec	Tbyte = 2^{40} ~ 10^{12} bytes
Peta	Pflop/s = 10^{15} flop/sec	Pbyte = 2^{50} ~ 10^{15} bytes
Exa	Eflop/s = 10^{18} flop/sec	Ebyte = 2^{60} ~ 10^{18} bytes
Zetta	Zflop/s = 10^{21} flop/sec	Zbyte = 2^{70} ~ 10^{21} bytes
Yotta	Yflop/s = 10^{24} flop/sec	Ybyte = 2^{80} ~ 10^{24} bytes

- See www.top500.org for current list of fastest machines

30th List: The TOP10

	Manufacturer	Computer	Rmax [TF/s]	Installation Site	Country	Year	#Cores
1	IBM	BlueGene/L eServer Blue Gene	478.2	DOE/NNSA/LLNL	USA	2007	212,992
2	IBM	JUGENE BlueGene/P Solution	167.3	Forschungszentrum Juelich	Germany	2007	65,536
3	SGI	SGI Altix ICE 8200	126.9	New Mexico Computing Applications Center	USA	2007	14,336
4	HP	Cluster Platform 3000 BL460c	117.9	Computational Research Laboratories, TATA SONS	India	2007	14,240
5	HP	Cluster Platform 3000 BL460c	102.8	Swedish Government Agency	Sweden	2007	13,728
6 3	Sandia/Cray	Red Storm Cray XT3	102.2	DOE/NNSA/Sandia	USA	2006	26,569
7 2	Cray	Jaguar Cray XT3/XT4	101.7	DOE/ORNL	USA	2007	23,016
8 4	IBM	BGW eServer Blue Gene	91.29	IBM Thomas Watson	USA	2005	40,960
9	Cray	Franklin Cray XT4	85.37	NERSC/LBNL	USA	2007	19,320
10 5	IBM	New York Blue eServer Blue Gene	82.16	Stony Brook/BNL	USA	2007	36,864

New 100 Tflops Cray XT-4 at NERSC



Cray XT-4 “Franklin”

19,344 compute cores

102 Tflop/sec peak

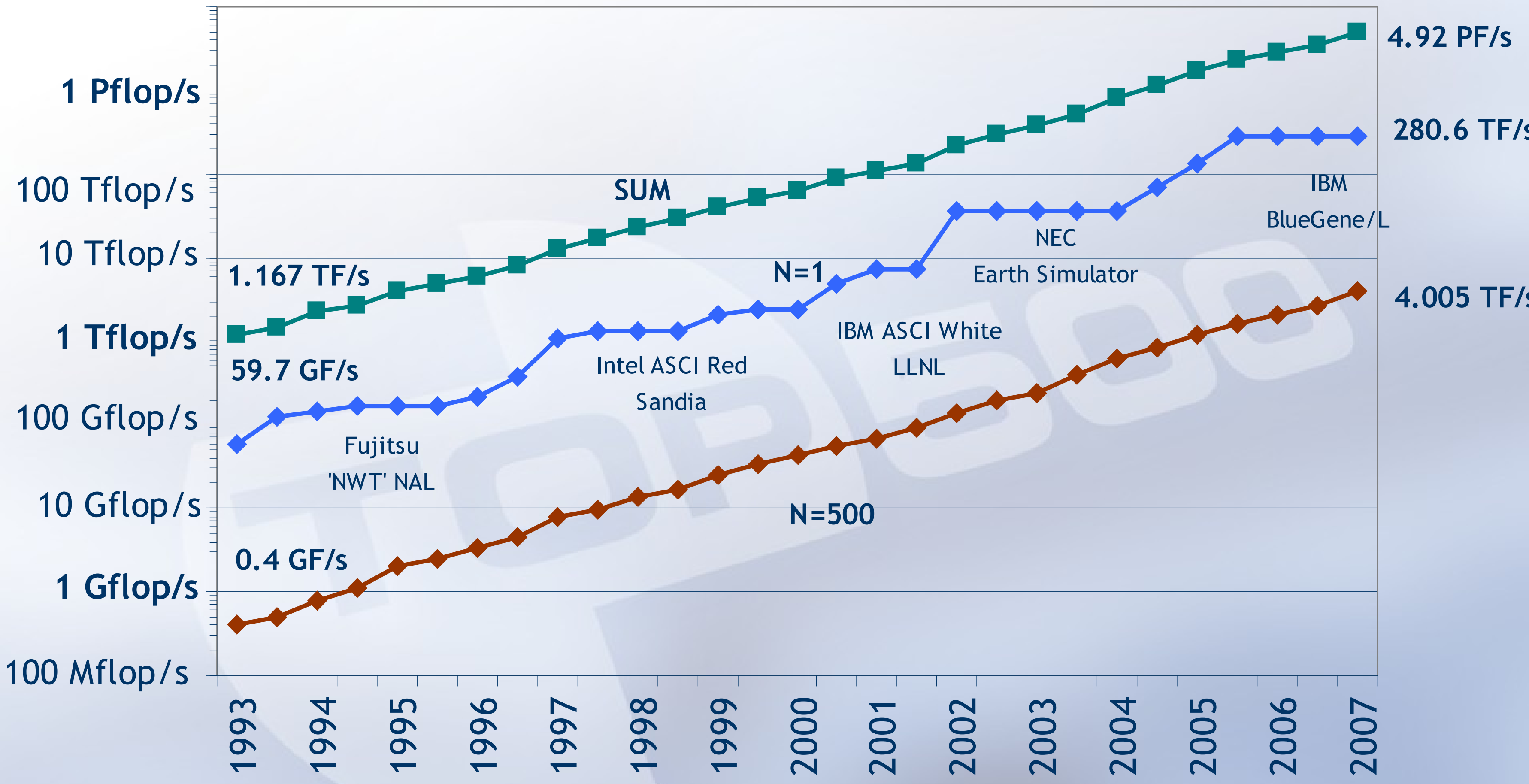
39 TB memory

350 TB usable disk space

50 PB storage archive

NERSC is enabling new science

Performance Development



Signpost System in 2005

IBM BG/L @ LLNL

- 700 MHz
- 65,536 nodes
- 180 (360) Tflop/s peak
- 32 TB memory
- 135 Tflop/s LINPACK
- 250 m² floor space
- 1.8 MW power

Outline

all

- Why ~~powerful~~ computers must be parallel processors

  Including your laptop

- Large important problems require powerful computers

  Even computer games

- Why writing (fast) parallel programs is hard

- Principles of parallel computing performance

- Structure of the course

Why we need powerful computers

New Science Question: Hurricane Statistics

What is the effect of different climate scenarios on number and severity of tropical storms?

	1979	1980	1981	1982	Obs
Northwest Pacific Basin	>25	~30			40
Atlantic Basin	~6	~12			?

Work in progress—results to be published

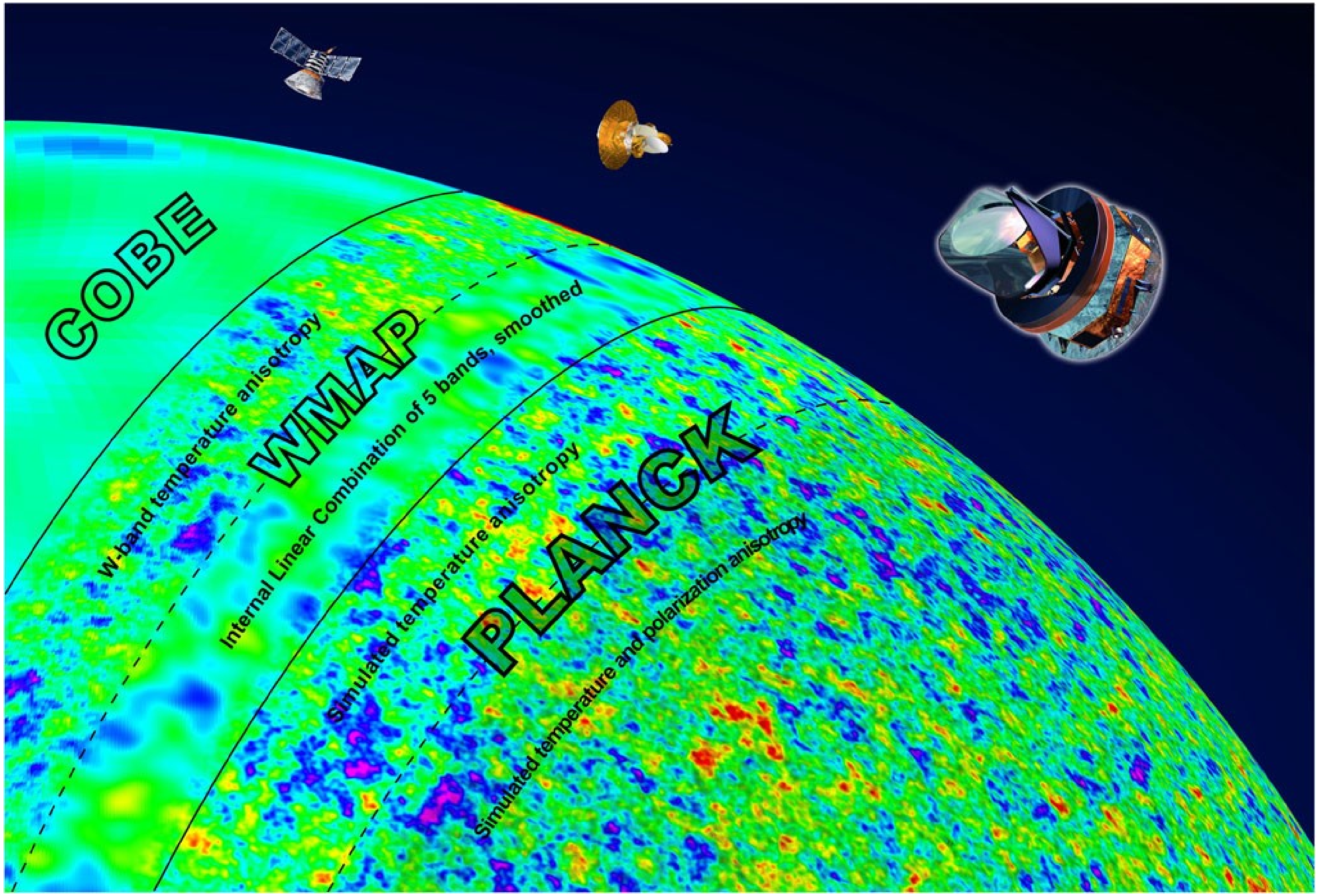
Source: M.Wehner, LBNL

CMB Computing at NERSC

- **CMB data analysis presents a significant and growing computational challenge, requiring**
  - **well-controlled approximate algorithms**
  - **efficient massively parallel implementations**
  - **long-term access to the best HPC resources**
- **DOE/NERSC has become the leading HPC facility in the world for CMB data analysis**
  - **O(1,000,000) CPU-hours/year**
  - **O(10) Tb project disk space**
  - **O(10) experiments & O(100) users (rolling)**

source J. Borrill, LBNL

Evolution Of CMB Satellite Maps



Algorithms & Flop-Scaling

- Map-making
  - Exact maximum likelihood : $O(N_p^3)$
  - PCG maximum likelihood : $O(N_i\, N_t \log N_t)$
  - Scan-specific, e.g.. destriping : $O(N_t \log N_t)$
  - Naïve : $O(N_t)$

Speed

Accuracy

- Power Spectrum estimation
  - Iterative maximum likelihood : $O(N_i\, N_b\, N_p^3)$
  - Monte Carlo pseudo-spectral :
    - Time domain : $O(N_r\, N_i\, N_t \log N_t)$, $O(N_r\, l_{max}^3)$
    - Pixel domain : $O(N_r\, N_t)$
    - Simulations
      - exact simulation > approximate analysis !

Speed

Accuracy

CMB is Characteristic for CSE Projects

- Petaflop/s and beyond computing requirements
- Algorithm and software requirements
- Use of new technology, e.g. NGF
- Service to a large international community
- Exciting science

Parallel Browser (Ras Bodik)

- Web 2.0: Browser plays role of traditional OS
  - Resource sharing and allocation, Protection
- Goal: Desktop quality browsing on handhelds
  - Enabled by 4G networks, better output devices
- Bottlenecks to parallelize
  - Parsing, Rendering, Scripting
- “SkipJax”
  - Parallel replacement for JavaScript/AJAX
  - Based on Brown’s FlapJax