CS240A: Applied Parallel Computing

Introduction

CS 240A Course Information

- Web page:

  http://www.cs.ucsb.edu/~tyang/class/240a17

- Class schedule: Tu/Th. 1pm-2:50pm Phelp 2510
- Instructor: Tao Yang (tyang at cs).
  - Office Hours: Tu/Th 12-1pm(or email me for appointments or just stop by my office). HFH building, Room 5113

Topics

- High performance computing
  - Basics of computer architecture, clusters/cloud systems.
  - Memory hierarchies,
  - High throughput computing
- Parallel Programming Models and Machines. Software/libraries
  - Shared memory vs distributed memory
  - Threads, OpenMP, MPI, MapReduce, Spark.
  - SIMD
- Patterns of parallelism. Optimization techniques for parallelization and performance
- Parallelism in Scientific Computing and Applications
  - Core algorithms (Dense & Sparse Linear Algebra)
- Parallelism in data-intensive applications and systems

What you should get out of the course

In depth understanding of:

- How to maximize the use of CPU/cache for high performance computing
- When is parallel computing useful?
- Understanding of parallel computing hardware options.
- Overview of programming models (software) and tools.
- Some parallel applications and the algorithms
- Performance analysis and tuning
- Exposure to various open research questions

Introduction: Outline

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

  Including your laptops and handhelds

- Why parallel processing?
  - Large Computational Science and Engineering (CSE) problems require powerful computers
  - Commercial data-oriented computing also needs.
- Basic parallel performance models
- Why writing (fast) parallel programs is hard

Metrics in Scientific Computing Worlds

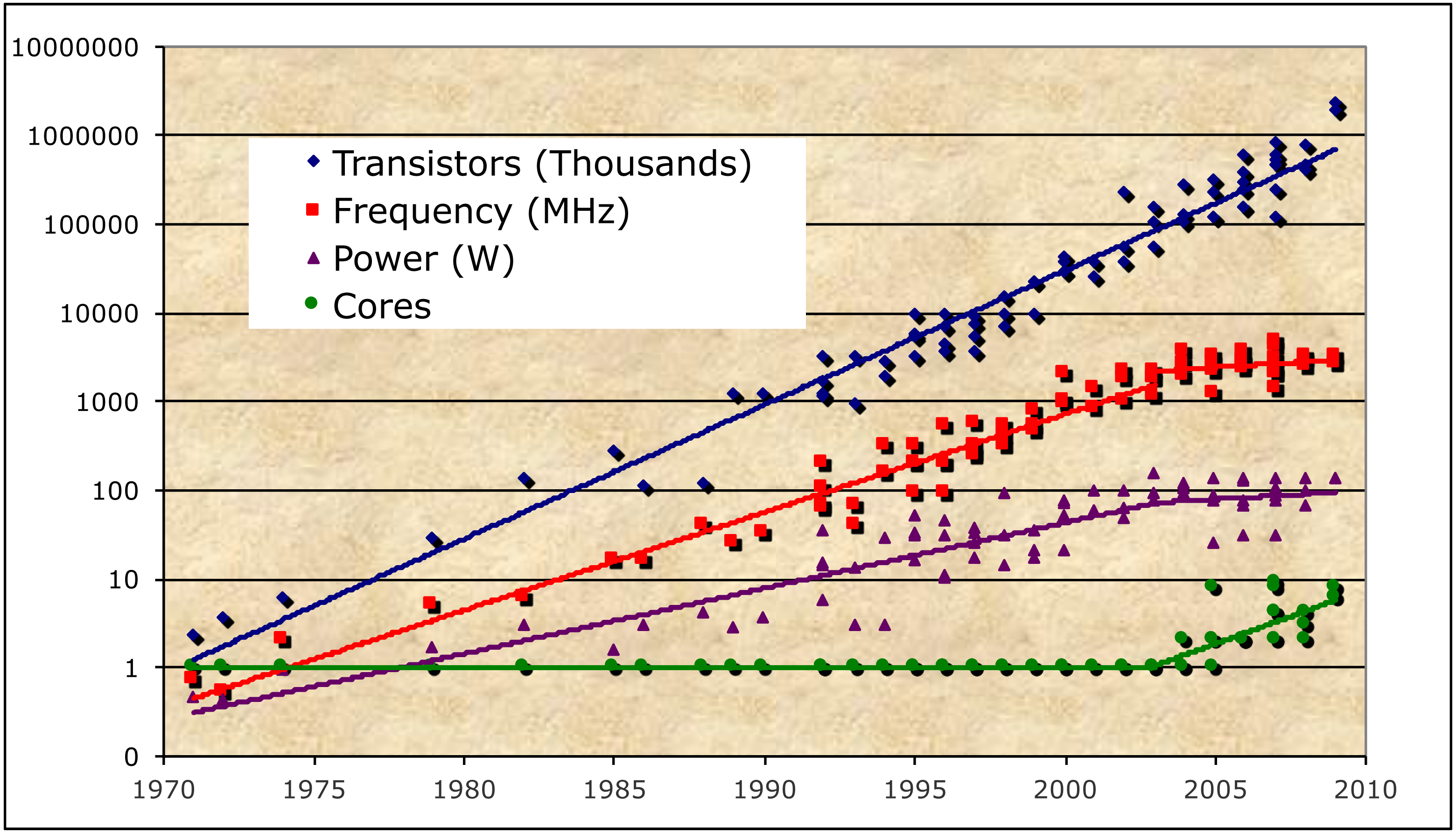
- **High Performance Computing (HPC) units are:**
  - Flop: floating point operation, usually double precision unless noted
  - Flop/s: floating point operations per second
  - Bytes: size of data (a double precision floating point number is 8)

- **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 ~ 27 Pflop/s**
  - Up-to-date list at www.top500.org

Revolution in Processors



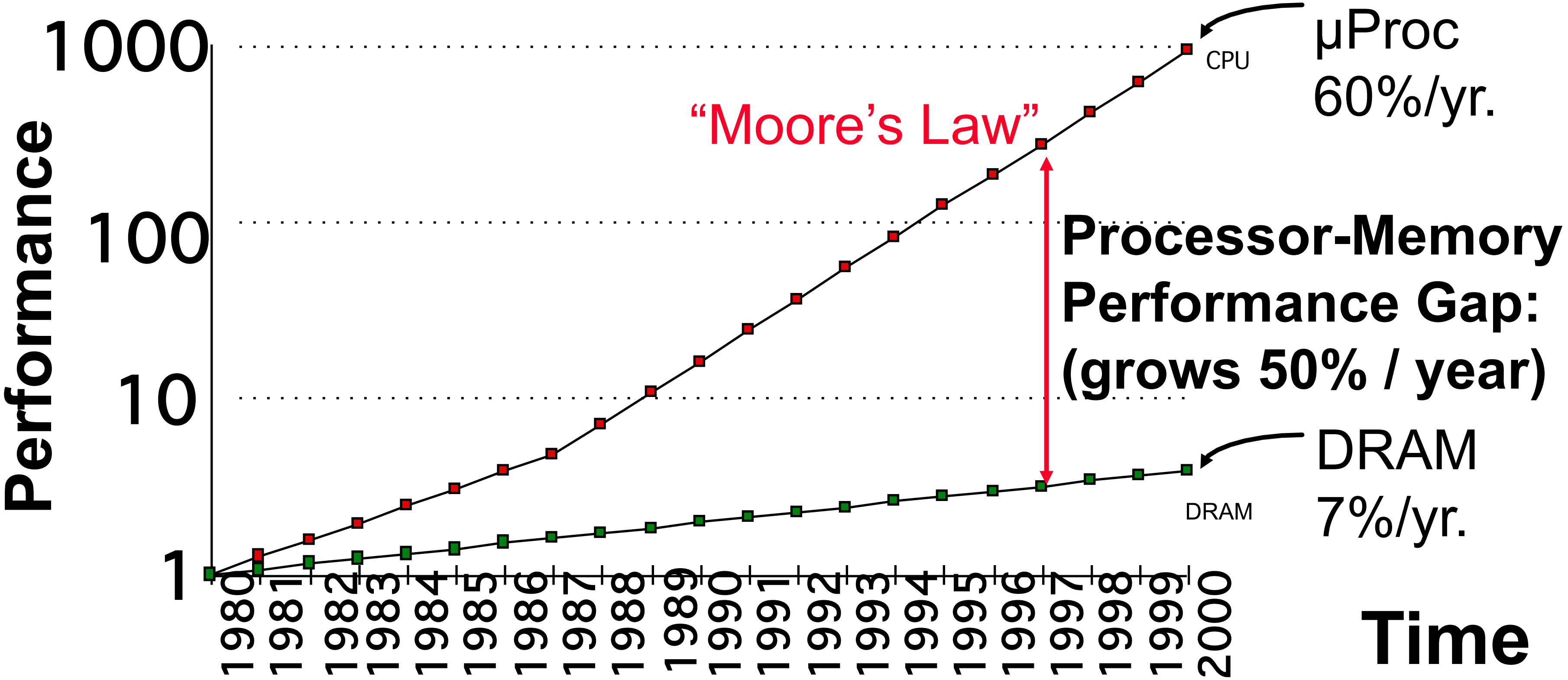
- Chip density is continuing increase ~2x every 2 years (Moore's Law)
- Clock speed is not
- Number of processor cores may double instead
- Power is under control, no longer growing

Impact of Parallelism

- 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
- Computer industry betting on this big change, but does not have all the answers

Memory is Not Keeping Pace

Memory density is doubling every three years; processor logic is every two



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
- Linpack performance
  - 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

From www.top500.org, Nov 2015

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National Super Computer Center in Guangzhou China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3M	33,862.7	54,902.4	17,808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x Cray Inc.	560,640	17,590.0	27,112.5	8,209

From www.top500.org, June 2017

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	**Sunway TaihuLight -** National Supercomputing Center in Wuxi China	**Sunway TaihuLight -** Sunway MPP, Sunway SW26010 260C 1.45GHz	10.6M	15,371	125,435	15,371
4	**Titan** DOE/SC/Oak Ridge National Laboratory United States	Cray XK7, Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, NVIDIA K20x	560,640	17,590	27,112	8,209

Moore’s Law reinterpreted

- Number of cores per chip will 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

- Why ~~powerful~~ all computers must be parallel processors
  - Including your laptops and handhelds
- Large Computational Science&Engineering and commercial problems require powerful computers
- Basic performance models
- Why writing (fast) parallel programs is hard

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.

Drivers for Changes in Computational Science

“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.” - ***Science*** **2020 Report, March 2006**

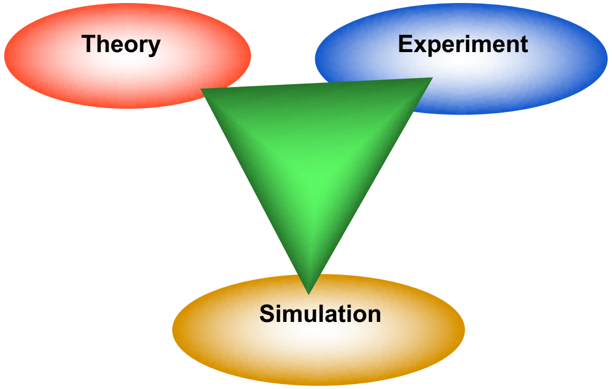
nature
LIFE ON EARTH
Methane-producing microbes arrived early
PLANT SPECIES
Do we still need them?
AVIAN FLU VIRUS
A molecular barrier to human-to-human transmission
2020 VISION
How computers will change the face of science
NATUREJOBS
The US Midwest

Nature, March 23, 2006

- Continued exponential increase in computational power → simulation is becoming third pillar of science, complementing theory and experiment
- Continued exponential increase in experimental data → Big data analysis w/o machine learning, visualization, and networked collaboration are becoming essential in all data rich scientific applications

Simulation: The Third Pillar of Science

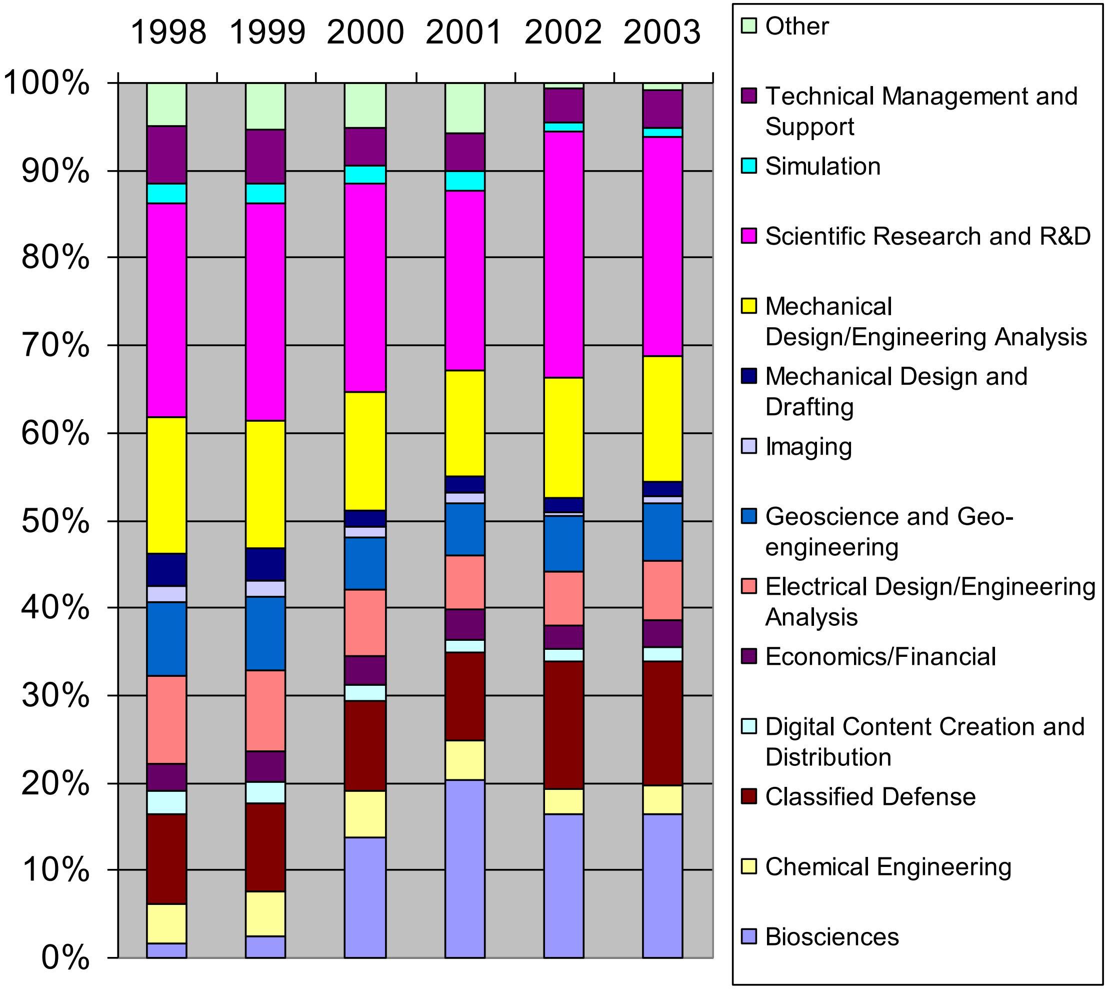
- **Traditional scientific and engineering method:**
  - (1) Do theory or paper design
  - (2) Perform experiments or build system



- **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

$5B World Market in Technical Computing

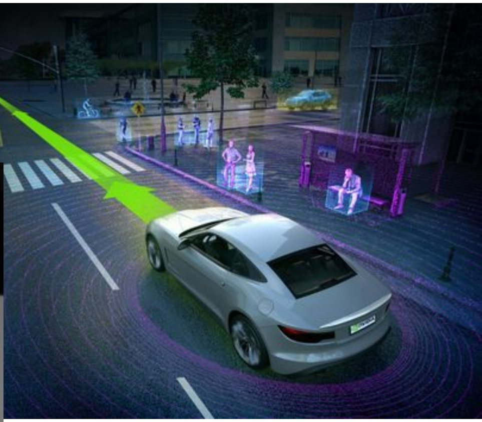


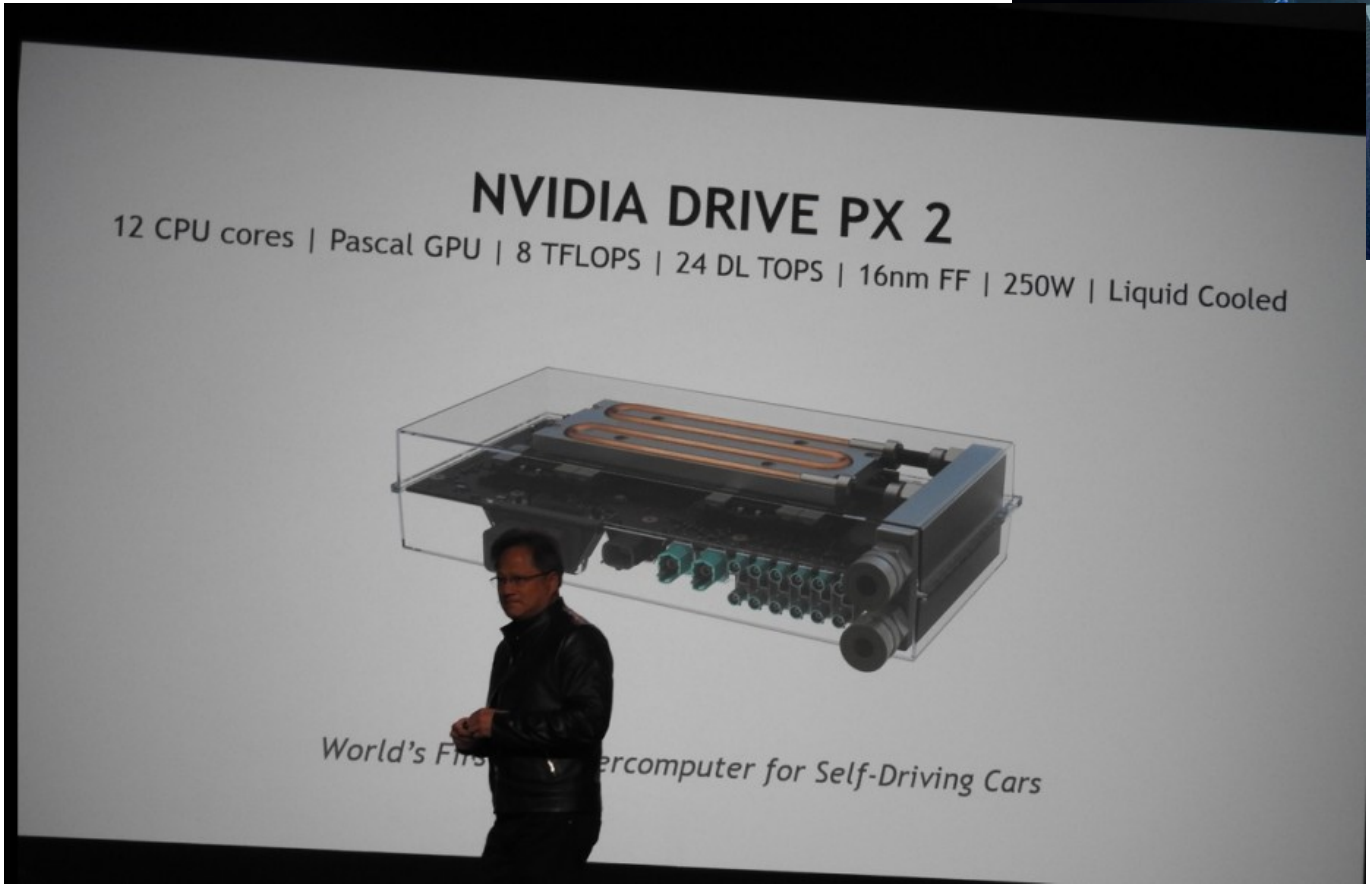
Source: IDC 2004, from NRC Future of Supercomputing Report

Supercomputing in Auto Industry

- NVIDIA Boosts IQ of Self-Driving Cars With World's First In-Car Artificial Intelligence Supercomputer.

Jan 4, 2016.





NVIDIA DRIVE PX 2

12 CPU cores | Pascal GPU | 8 TFLOPS | 24 DL TOPS | 16nm FF | 250W | Liquid Cooled

World's Firs ercomputer for Self-Driving Cars

Global Climate Modeling Problem

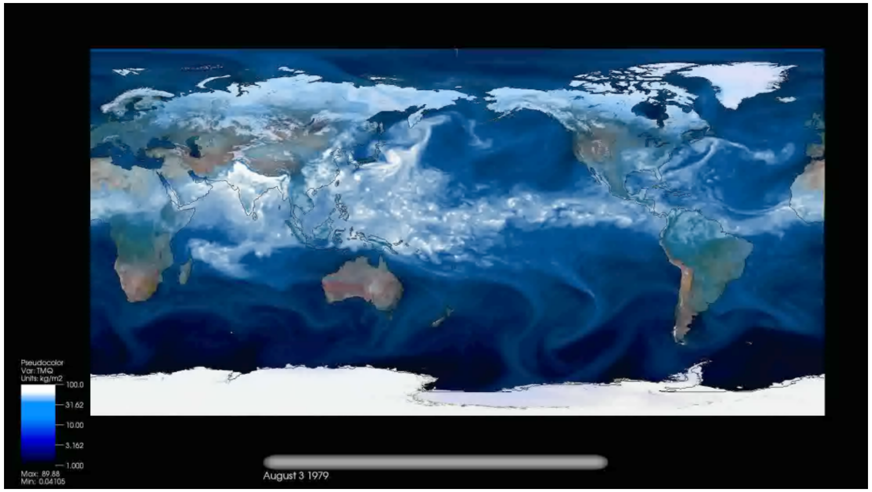
- Problem is to compute:

  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+δt given t

- Uses:
  - Predict major events, e.g., hurricane, El Nino
  - Use in setting air emissions standards
  - Evaluate global warming scenarios



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

Scalable Web Service/Processing Infrastructure

- Infrastructure scalability:

  Bigdata: Tens of billions of documents in web search

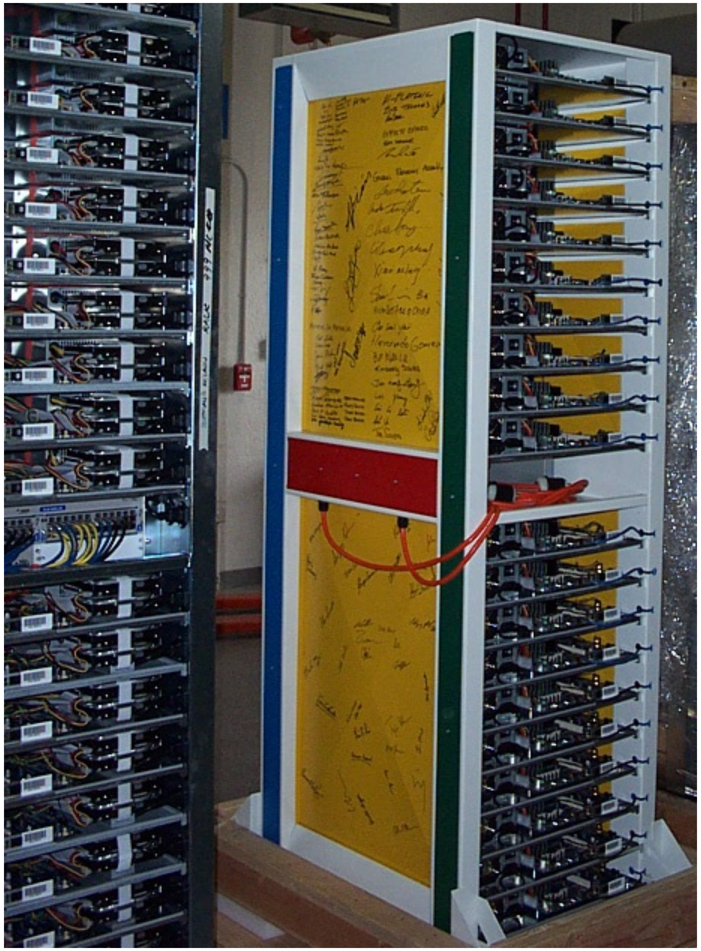
  Tens/hundreds of thousands of machines.

  Tens/hundreds of Millions of users

  Impact on response time, throughput, &availability,

- Platform software
  - Google GFS, MapReduce and Bigtable .
  - fundamental building blocks for fast data update/access and development cycles

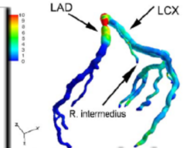
  …



What do commercial and CSE applications have in common?

Motif/Dwarf: Common Computational Methods

(Red Hot → Blue Cool)

	Embed	SPEC	DB	Games	ML	HPC	 Health	 Image	 Speech	 Music	 Browser
1 Finite State Mach.	🟥	🟥	🟥	🟨	🟨	🟦	🟦	🟦	🟦	🟦	🟥
2 Combinational	🟥	🟦	🟩	🟦	🟩	🟦	🟦	🟦	🟦	🟦	🟥
3 Graph Traversal	🟥	🟨	🟨	🟨	🟥	🟦	🟥	🟦	🟥	🟩	🟩
4 Structured Grid	🟥	🟥	🟦	🟨	🟦	🟥	🟦	🟥	🟦	🟦	🟦
5 Dense Matrix	🟥	🟥	🟨	🟥	🟥	🟥	🟦	🟥	🟥	🟥	🟦
6 Sparse Matrix	🟨	🟨	🟦	🟥	🟥	🟥	🟥	🟦	🟦	🟥	🟦
7 Spectral (FFT)	🟨	🟦	🟦	🟨	🟨	🟥	🟦	🟩	🟥	🟥	🟥
8 Dynamic Prog	🟨	🟦	🟥	🟦	🟥	🟦	🟦	🟦	🟨	🟦	🟥
9 N-Body	🟦	🟨	🟦	🟨	🟦	🟥	🟩	🟦	🟦	🟦	🟦
10 MapReduce	🟦	🟩	🟥	🟦	🟥	🟥	🟥	🟥	🟨	🟥	🟨
11 Backtrack/ B&B	🟦	🟦	🟨	🟦	🟥	🟦	🟦	🟦	🟦	🟨	🟦
12 Graphical Models	🟦	🟦	🟨	🟦	🟥	🟦	🟦	🟦	🟦	🟥	🟦
13 Unstructured Grid	🟦	🟦	🟦	🟨	🟨	🟥	🟥	🟦	🟦	🟥	🟦

Outline

- Why ~~powerful~~ all computers must be parallel processors
  - Including your laptops and handhelds
- Large CSE/commerical problems require powerful computers
- Performance models
- Why writing (fast) parallel programs is hard

Principles of Parallel Computing

- Finding enough parallelism (Amdahl's Law)
- Granularity
- Locality
- Load balance
- Coordination and synchronization
- Performance modeling

⇨ 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
- I/O parallelism in storage level

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 x be the fraction of work done sequentially, so (1-x) is fraction parallelizable
  - P = number of processors

$$
\begin{aligned}
\text{Speedup}(P) &= \text{Time}(1)/\text{Time}(P) \\
&<= 1/(x + (1-x)/P) \\
&<= 1/s
\end{aligned}
$$

- Even if the parallel part speeds up perfectly performance is limited by the sequential part

Caveat: Amdahl's Law

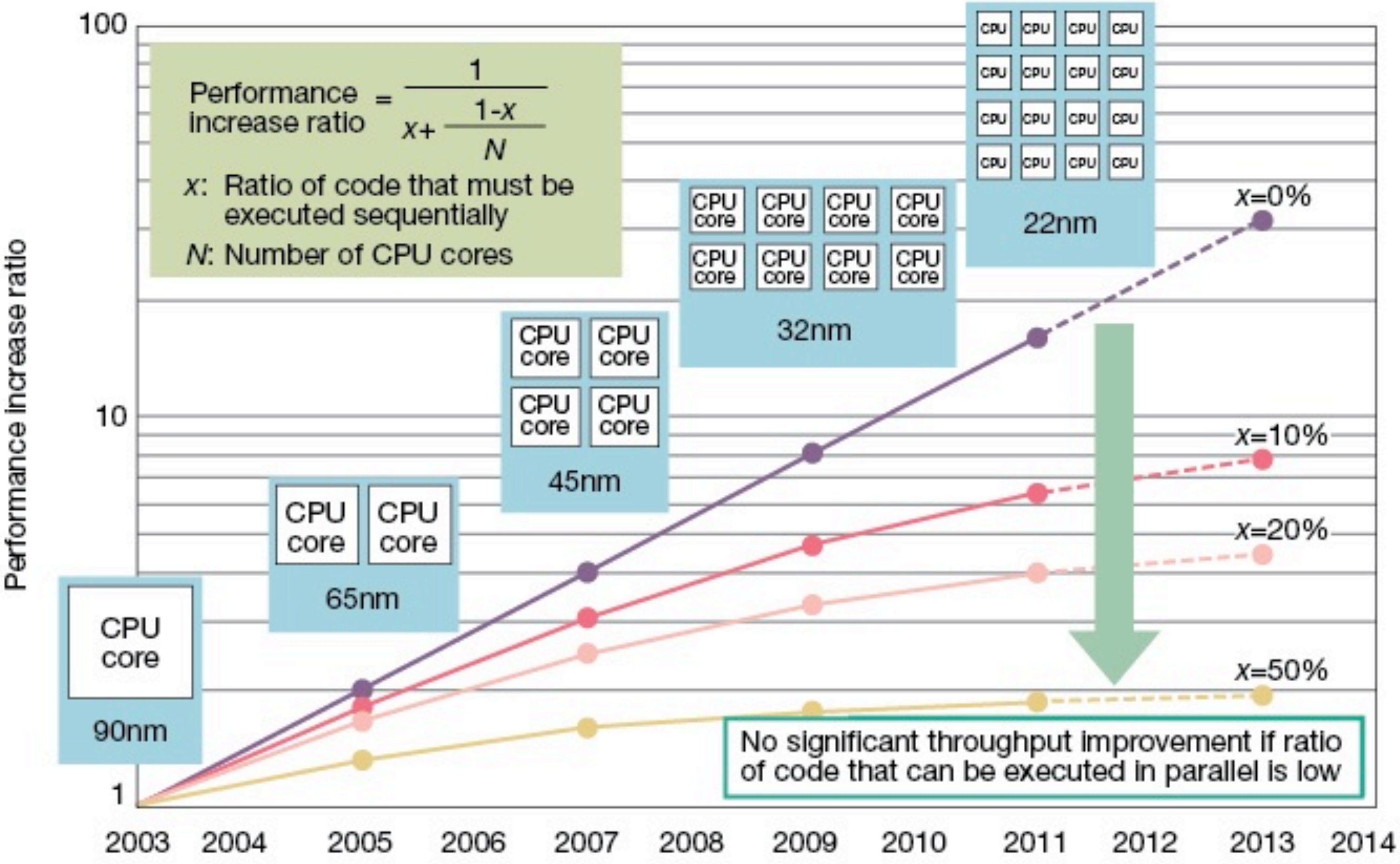


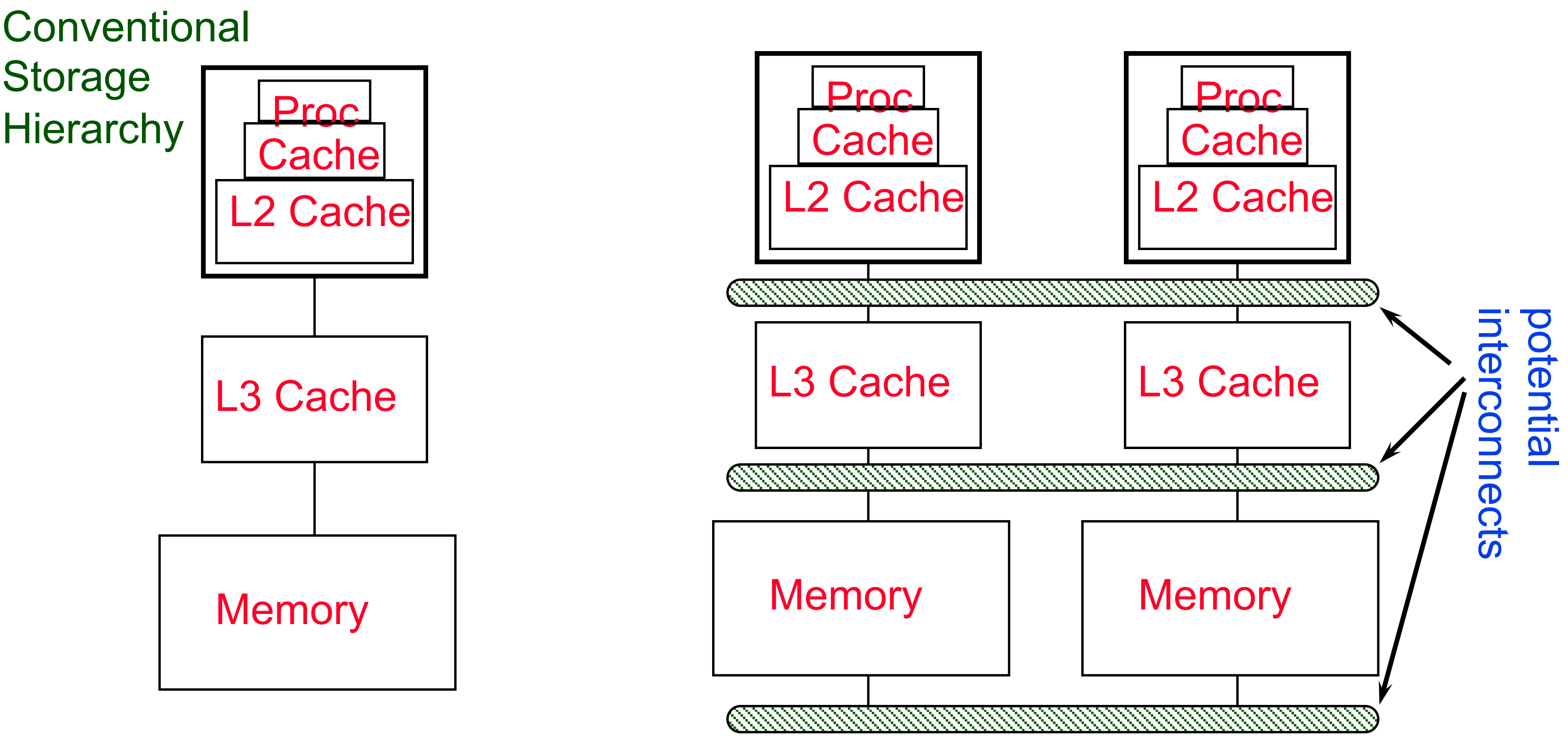
Fig 3 Amdahl's Law an Obstacle to Improved Performance Performance will not rise in the same proportion as the increase in CPU cores. Performance gains are limited by the ratio of software processing that must be executed sequentially. Amdahl's Law is a major obstacle in boosting multicore microprocessor performance. Diagram assumes no overhead in parallel processing. Years shown for design rules based on Intel planned and actual technology. Core count assumed to double for each rule generation.

Gene Amdahl
Computer Pionee

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 accessing data, 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

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”

Improving Real Performance

Peak Performance grows exponentially, a la Moore's Law

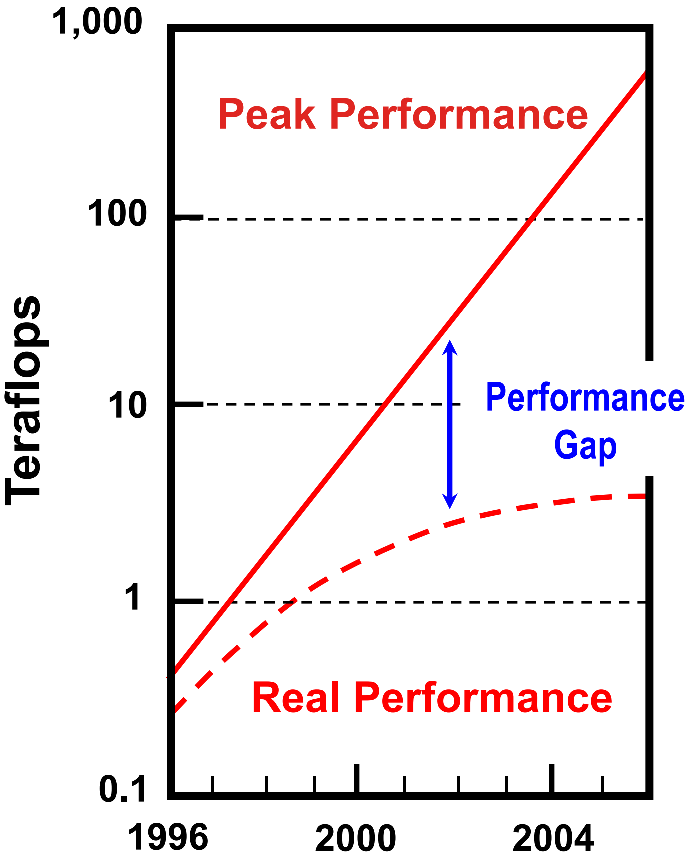
- In 1990's, peak performance increased 100x; in 2000's, it will increase 1000x

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)

Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  a) 1 / month
  b) 1 / week
  c) 1 / day
  d) 1 / hour

Coping with Failures

- 4 disks/server, 50,000 servers
- Failure rate of disks: 2% to 10% / year
  - Assume 4% annual failure rate
- On average, how often does a disk fail?
  - a) 1 / month
  - b) 1 / week
  - c) 1 / day
  - d) 1 / hour

50,000 x 4 = 200,000 disks

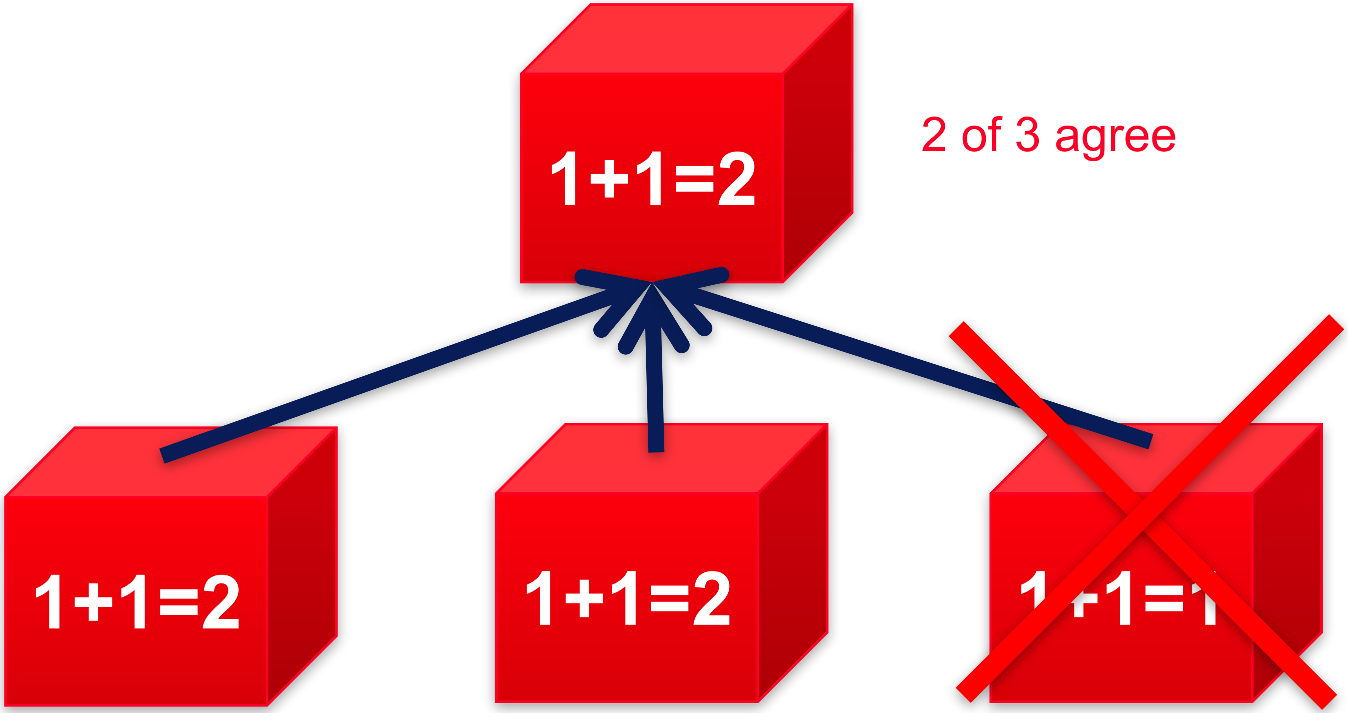
200,000 x 4% = 8000 disks fail

365 days x 24 hours = 8760 hours

d

Dependability via Redundancy

- Redundancy so that a failing piece doesn't make the whole system fail



2 of 3 agree

FAIL!

Increasing transistor density reduces the cost of redundancy

Dependability via Redundancy

- Applies to everything from datacenters to storage to memory to instructors
  - Redundant datacenters so that can lose 1 datacenter but Internet service stays online
  - Redundant disks so that can lose 1 disk but not lose data (Redundant Arrays of Independent Disks/RAID)
  - Redundant memory bits of so that can lose 1 bit but no data (Error Correcting Code/ECC Memory)

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.
- Some important parallel applications and the algorithms
- Performance analysis and tuning
- Exposure to various open research questions

Course Deadlines (Tentative)

- Week 2: join Google discussion group. Open Comet cluster account.
- End of Oct
  - HW1 (C programming)
  - 1-page project proposal.
    The content includes: Problem description, challenges (what is new?), what to deliver, how to test and what to measure, milestones, and references
  - Meet with me
- Nov 20 week: Paper review presentation and project progress.
- End of Nov
  - . HW2 due. (Python or Java programming)
- Final Week. Take-home exam. Final project presentation/report.
- Weight distribution:
  - CS/CE Track: Project 40%. Exam 40%. HW 20%
  - Others: HW is not required, more Project Load (60%)