CS140: Parallel Scientific Computing

Class Introduction

Tao Yang, UCSB

Tuesday/Thursday. 11:00-12:15 GIRV 1115

CS 140 Course Information

- **Instructor: Tao Yang (tyang@cs).**
  - Office Hours: T/Th 10-11(or email me for appointments or just stop by my office). HFH building, Room 5113
- **Supercomputing consultant**: Kadir Diri and Stefan Boeriu
- **TA:** Xin Jin [xin_jin@cs]. Steven Bluen [sbluen153@yahoo]
- **Text book**
  - An Introduction to Parallel Programming" by Peter Pacheco, 2011, Morgan Kaufmann Publisher
- **Class slides/online references:**
  - http://www.cs.ucsb.edu/~tyang/class/140s14
- **Discussion group:** registered students are invited to join a google group

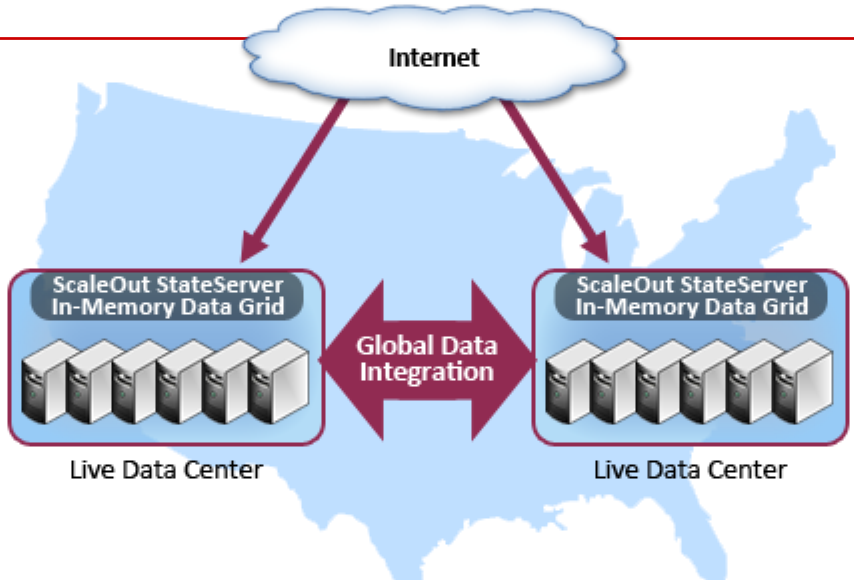
Introduction

- Why all computers must be parallel computing
- Why parallel processing?
  - Large Computational Science and Engineering (CSE) problems require powerful computers
  - Commercial data-oriented computing also needs.
- Why writing (fast) parallel programs is hard
- Class Information

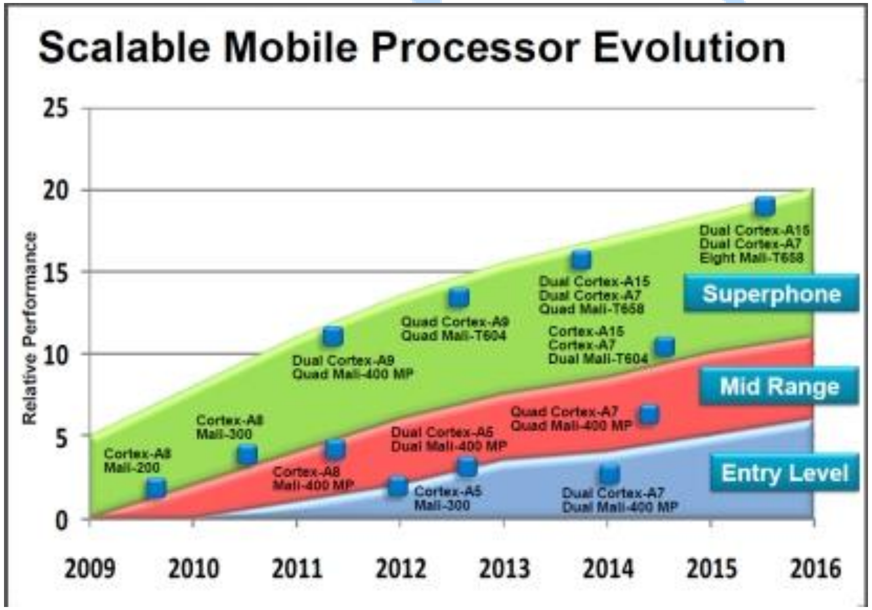
All computers use parallel computing

- **Web+cloud computing**

Big corporate computing



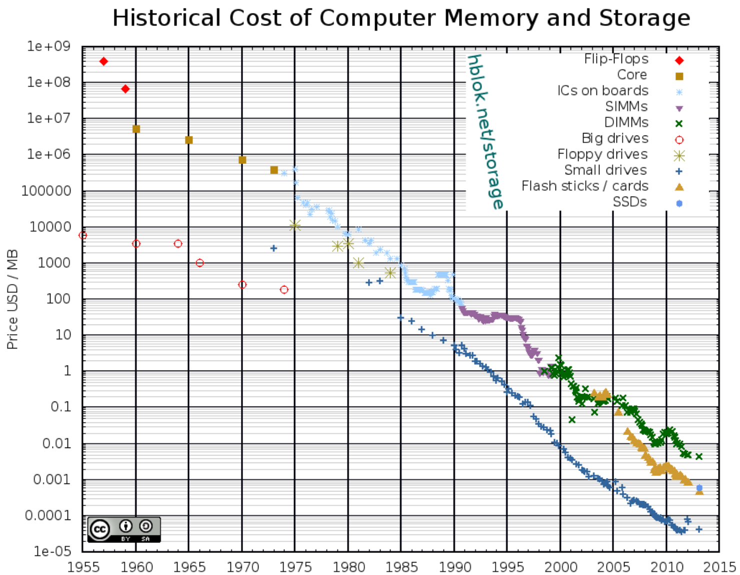
- **Enterprise computing**

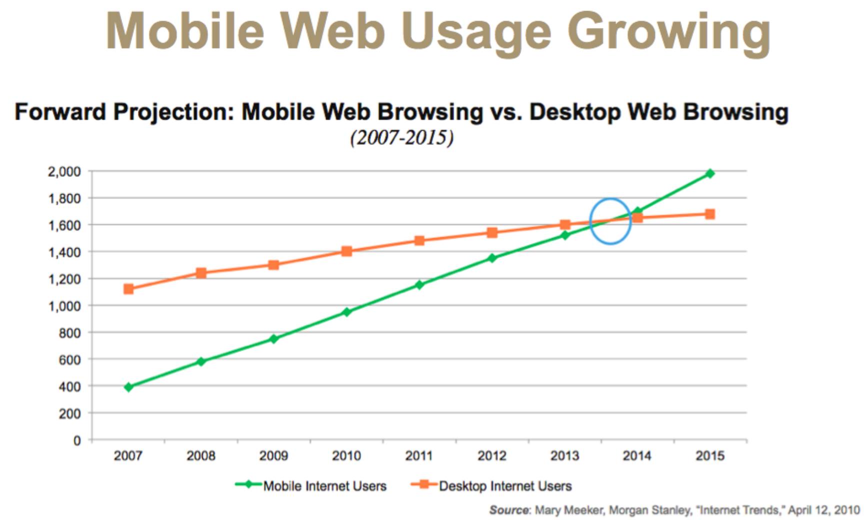


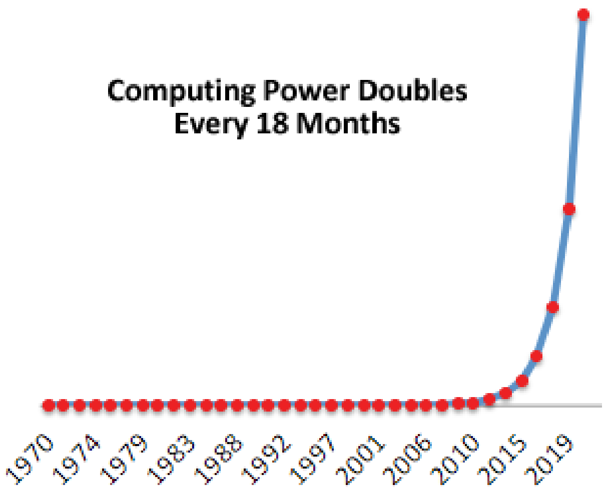
- **Home computing**

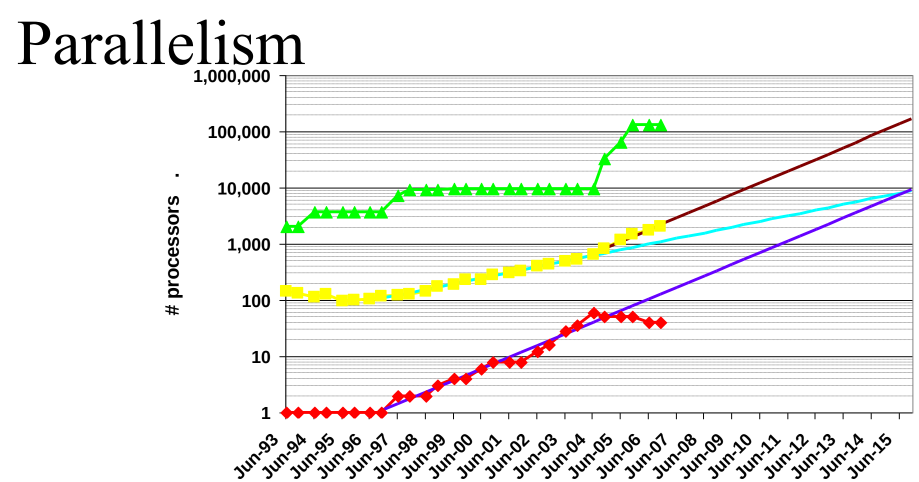
Desktops, laptops, handhelds & phones

Drivers behind high performance computing



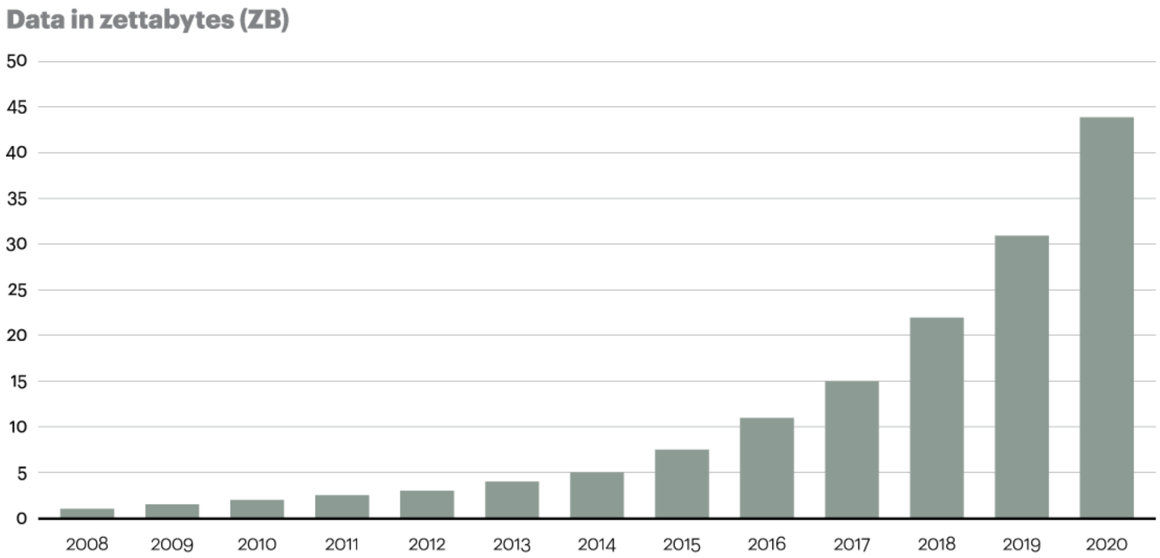




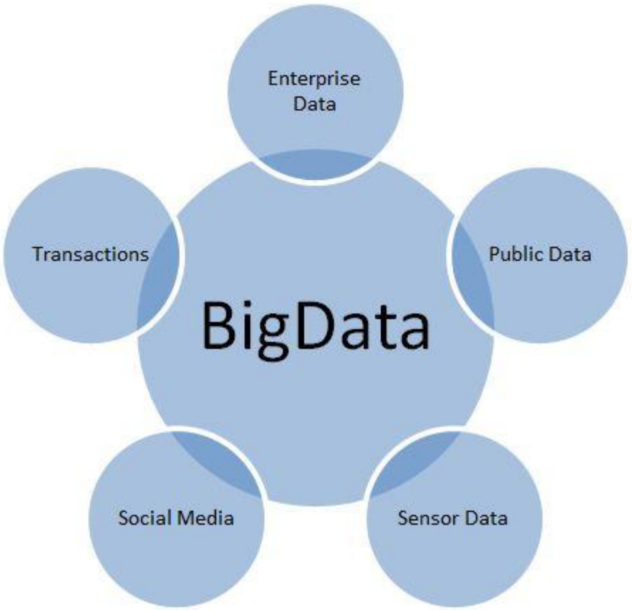


Big Data Drives Computing Need Too

Figure 1
Data is growing at a 40 percent compound annual rate, reaching nearly 45 ZB by 2020



Source: Oracle, 2012



Zettabyte = 2^{70} ~ 1 billion Terabytes
Exabyte = 1 million Terabytes

Examples of Big Data

- **Web search/ads** (Google, Bing, Yahoo, Ask)
  - 10B+ pages crawled -> indexing 500-1000TB /day
  - 10B+ queries+pageviews /day → 100+ TB log
- **Social media**
  - Facebook: 3B content items shared. 3B- “like”. 300M photo upload. 500TB data ingested/day
  - Youtube: A few billion views/day. Millions of TB.
- **NASA**
  - 12 data centers, 25,000 datasets. Climate weather data: 32PB → 350PB
  - NASA missions stream 24TB/day. Future space data demand: 700 TB/second

Metrics in Scientific Computing World

- **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…**
- **Current fastest (public) machines in the world**
  - Up-to-date list at www.top500.org
  - Top one has 33.86 Pflop/s using 3.12 millions of cores

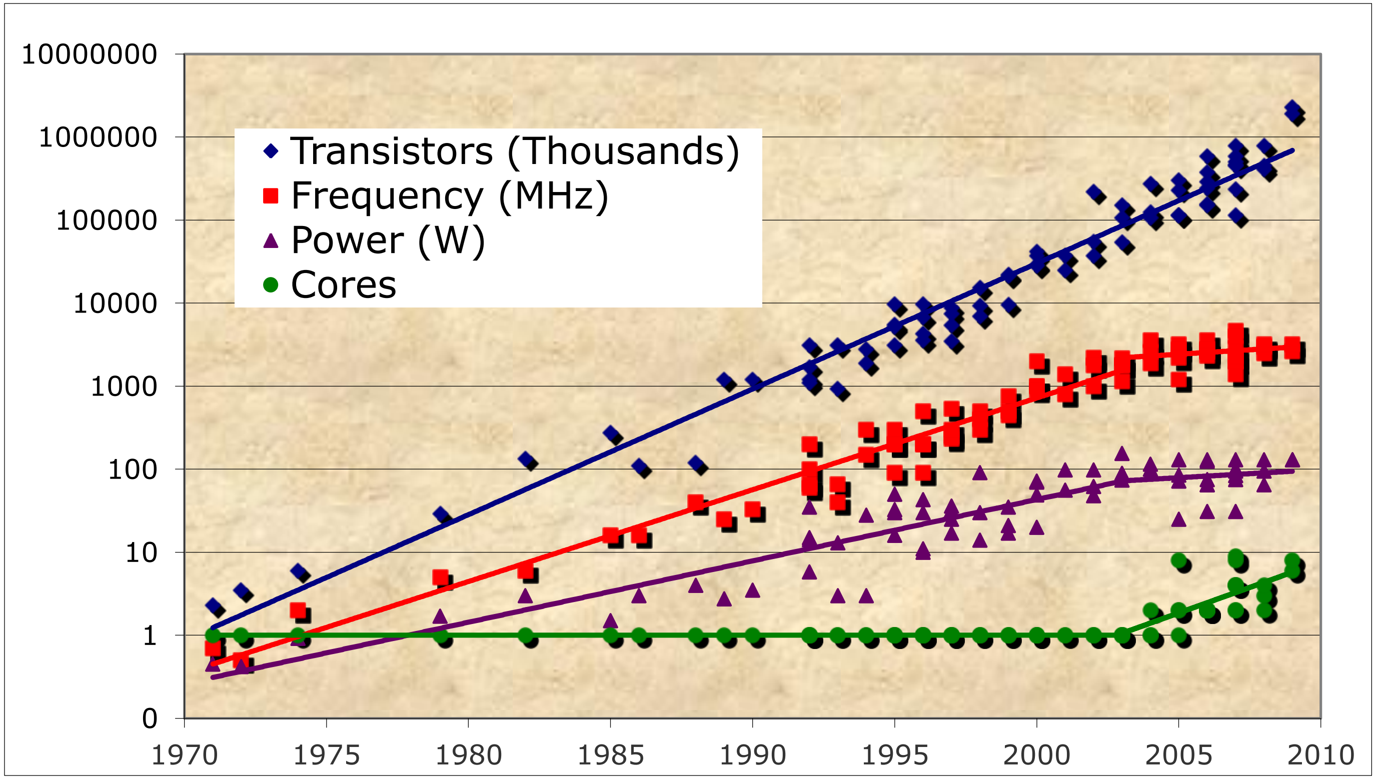
Typical sizes are millions, billions, trillions…

Mega	Mflop/s = 10^6 flop/sec	Mbyte = 2^{20} ~ 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

From www.top500.org (Nov 2013)

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	NSCC China	**MilkyWay-2** - Intel Xeon E5 2.2GHz NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** AMD Opteron, 2.2GHz NVIDIA K20x Cray Inc.	560640	17590.0	27112.5	8209
3	DOE/NNSA/LLNL United States	**Sequoia** - BlueGene/Q, Power BQC 16C 1.60 GHz, Custom IBM	1572864	16324.8	20132.7	7890

Why parallel computing? Can a single high speed core be used?

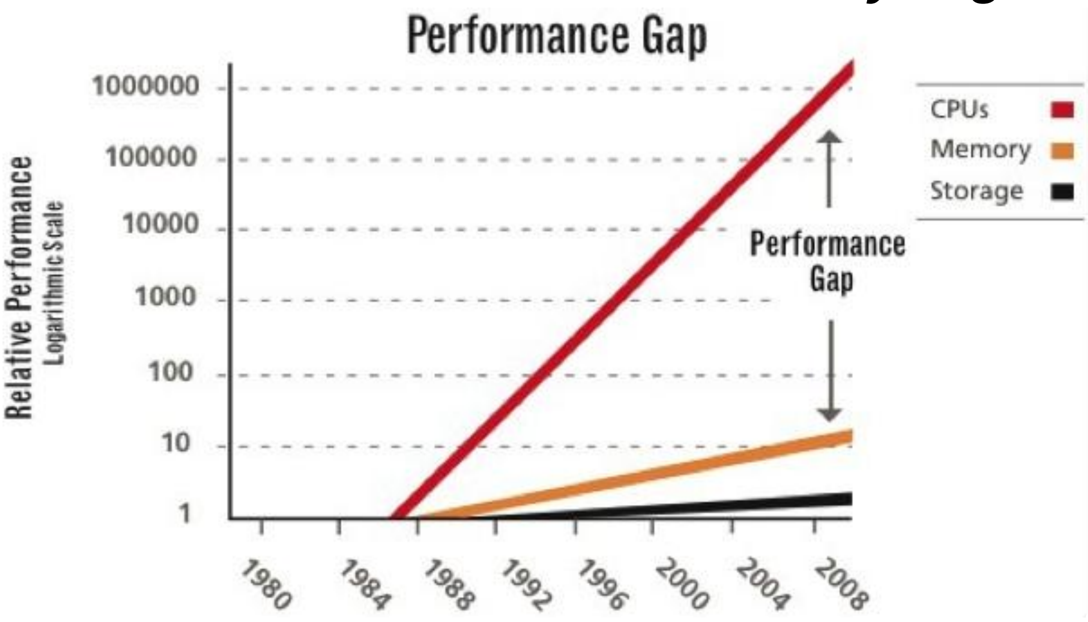


- Chip density is continuing increase ~2x every 2 years
- Clock speed is not
- Number of processor cores may double instead
- Power is under control, no longer growing

Can we just use one machine with many cores and big memory/storage?

Technology trends against increasing memory per core

- Memory performance is not keeping pace, even
    - Memory density is doubling every three years
    - Storage costs (dollars/Mbyte) are dropping gradually
- *have to use a distributed architecture for many highend computing*



Impact of Parallelism

- **All major processor vendors are producing *multicore* chips**
  - Every machine is 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**

Slide source: Demmel/Yelick

Roadmap

- Why all computers must be parallel computing
- Why parallel processing?
  - Large Computational Science and Engineering (CSE) problems require powerful computers
  - Commercial data-oriented computing also needs.
- Why writing (fast) parallel programs is hard
- Class Information

Examples of Challenging Computations That Need High Performance Computing

- **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

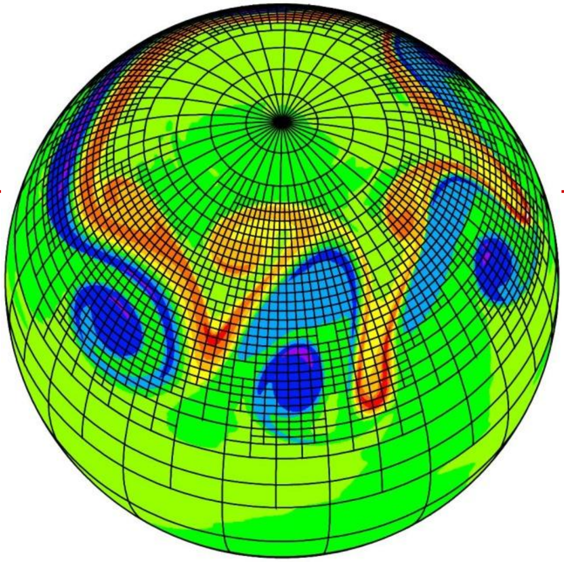
Slide source: Demmel/Yelick

Economic Impact of High Performance Computing

- **Airlines:**
  - System-wide logistics optimization on parallel systems.
  - Savings: approx. $100 million per airline per year.
- **Automotive design:**
  - Major automotive companies use 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.

Slide source: Demmel/Yelick

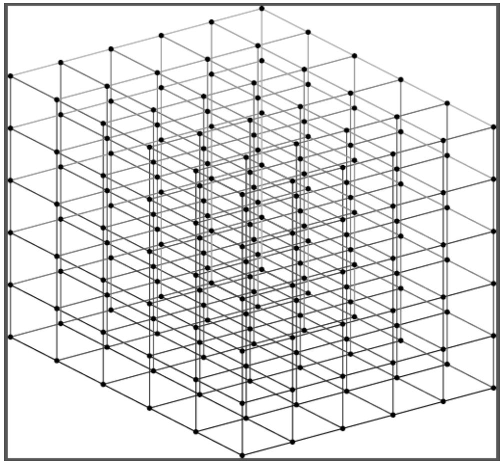
Global Climate Modeling



- **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 step



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

Slide source: Demmel/Yelick

Global Climate Modeling: Computational Requirements

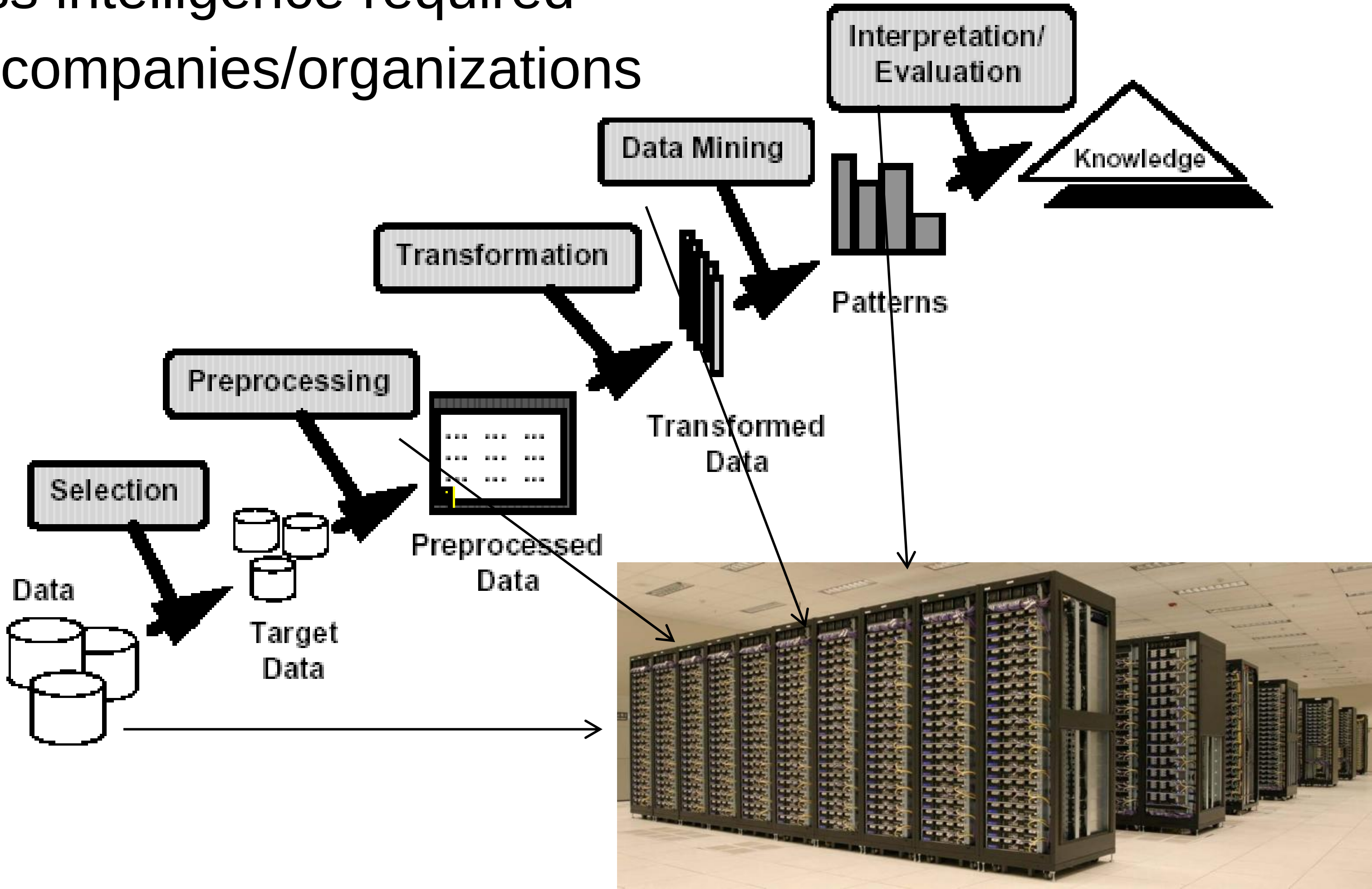
- **One piece is modeling the fluid flow in the atmosphere**
  - Solve numerical equations
    - Roughly 100 Flops per grid point with 1 minute timestep
- **Computational requirements:**
  - To match real-time, need 5×10^{11} flops in 60 seconds = 8 Gflop/s
  - Weather prediction (7 days in 24 hours) → 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**

Slide source: Demmel/Yelick

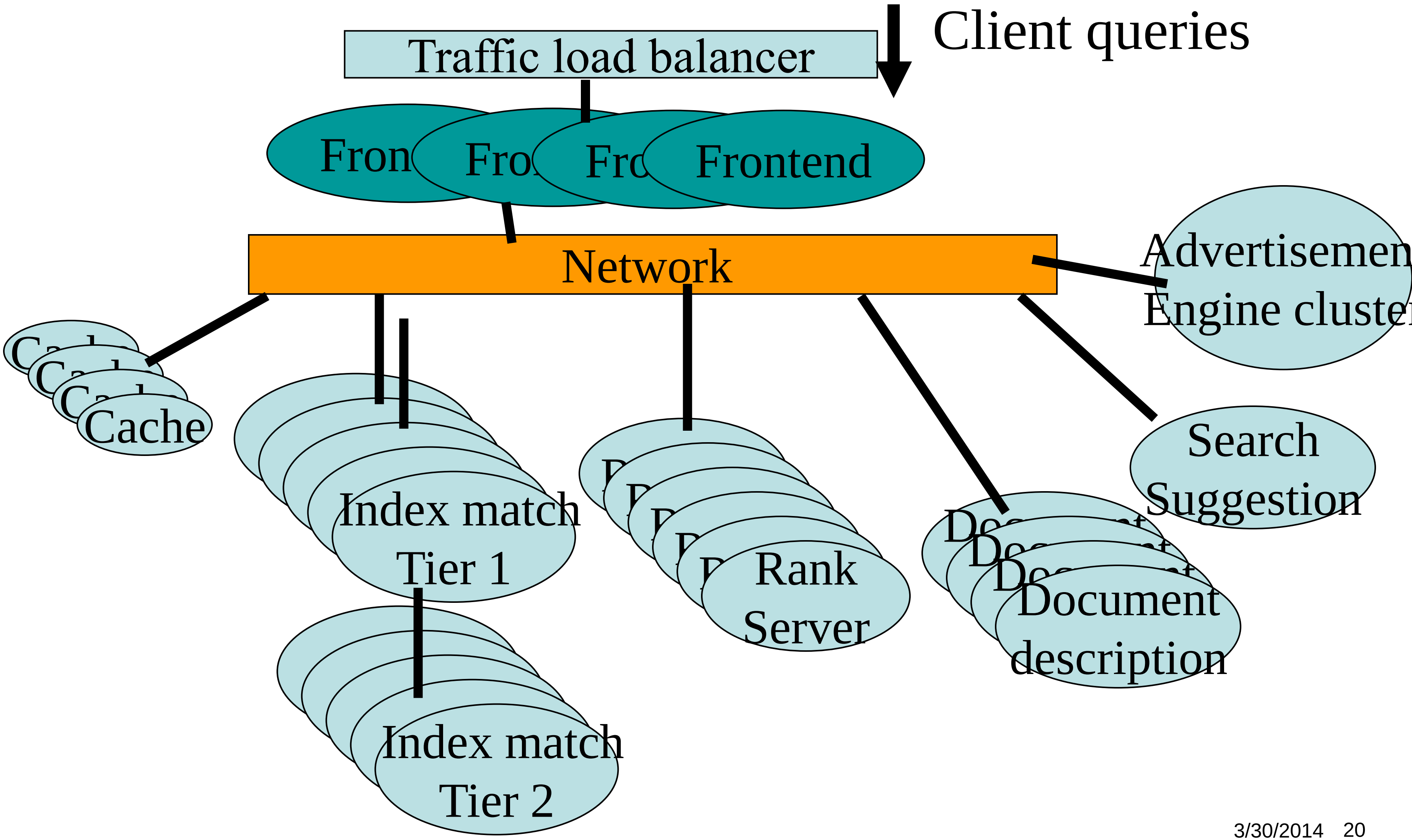
Mining and Search for Big Data

- Identify and discover information from a massive amount of data
- Business intelligence required

by many companies/organizations

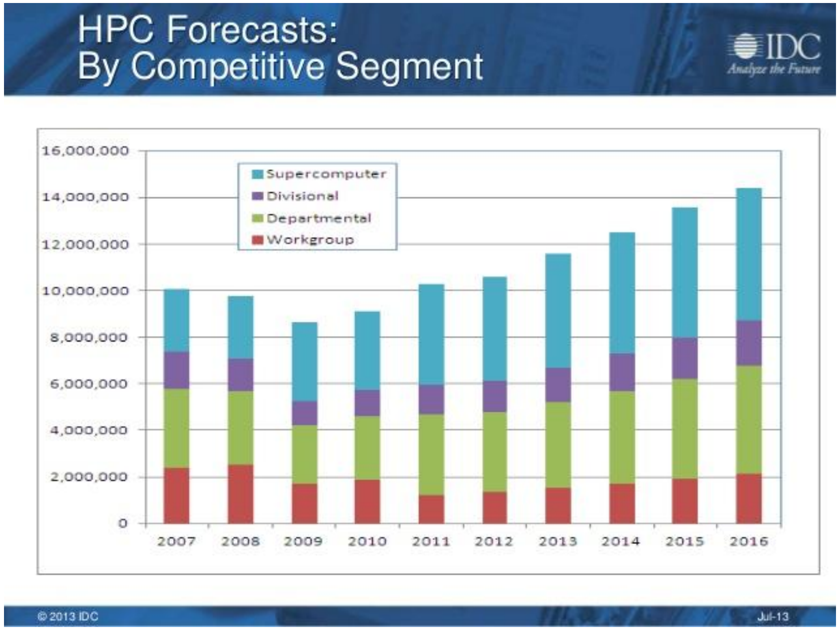


Multi-tier Web Services: Search Engine



IDC HPC Market Study

- International Data Corporation (**IDC**) is an American market research, analysis and advisory firm
- **HPC covers all servers that are used for highly computational or data intensive tasks**
  - **HPC revenue for 2014 exceeded $12B**
  - **forecasting ~7% growth over the next 5 years**

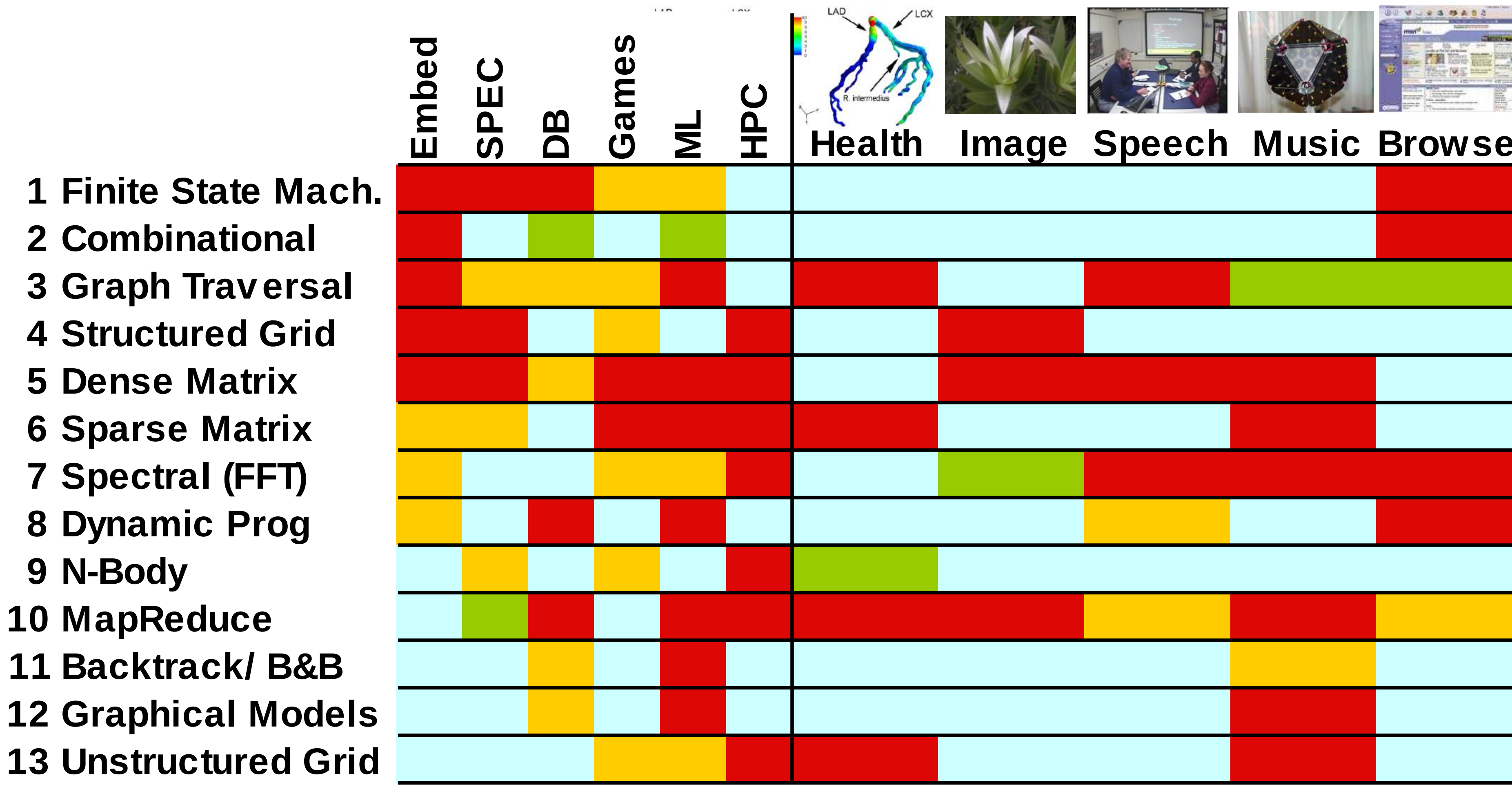


Source: IDC July 2013 Supercomputer segment: IDC defines as systems $500,000 and up.

What do compute-intensive applications have in common?

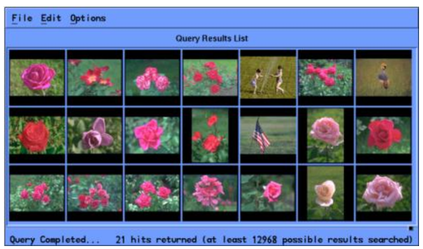
Motif/Dwarf: Common Computational Methods

(Red Hot → Blue Cool)

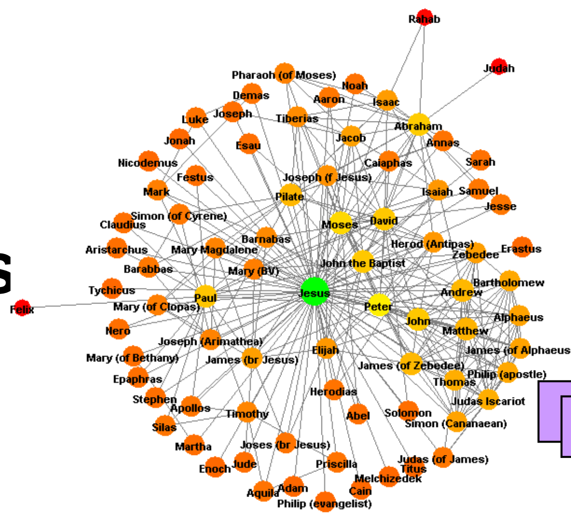


Types of Big Data Representation

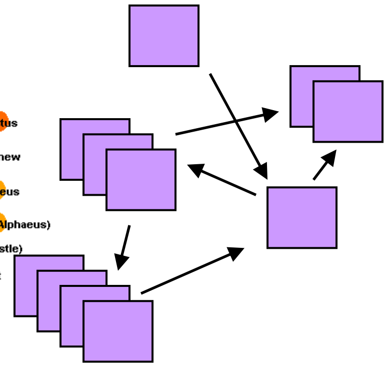
- Text, multi-media, social/graph data



- Represented by weighted feature vectors matrices, graphs



Social graph



The Web

Basic Scientific Computing Algortihms

- **Matrix-vector multiplication.**
- **Matrix-matrix multiplication.**
- **Direct method for solving a linear equation.**
  - Gaussian Elimination.
- **Iterative method for solving a linear equation.**
  - Jacobi, Gauss-Seidel.
- **Sparse linear systems and differential equations.**

Roadmap

- Why all computers must be parallel computing
- Why parallel processing?
  - Large Computational Science and Engineering (CSE) problems require powerful computers
  - Commercial data-oriented computing also needs.
- Why writing (fast) parallel programs is hard
- Class Information

Principles of Parallel Computing

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

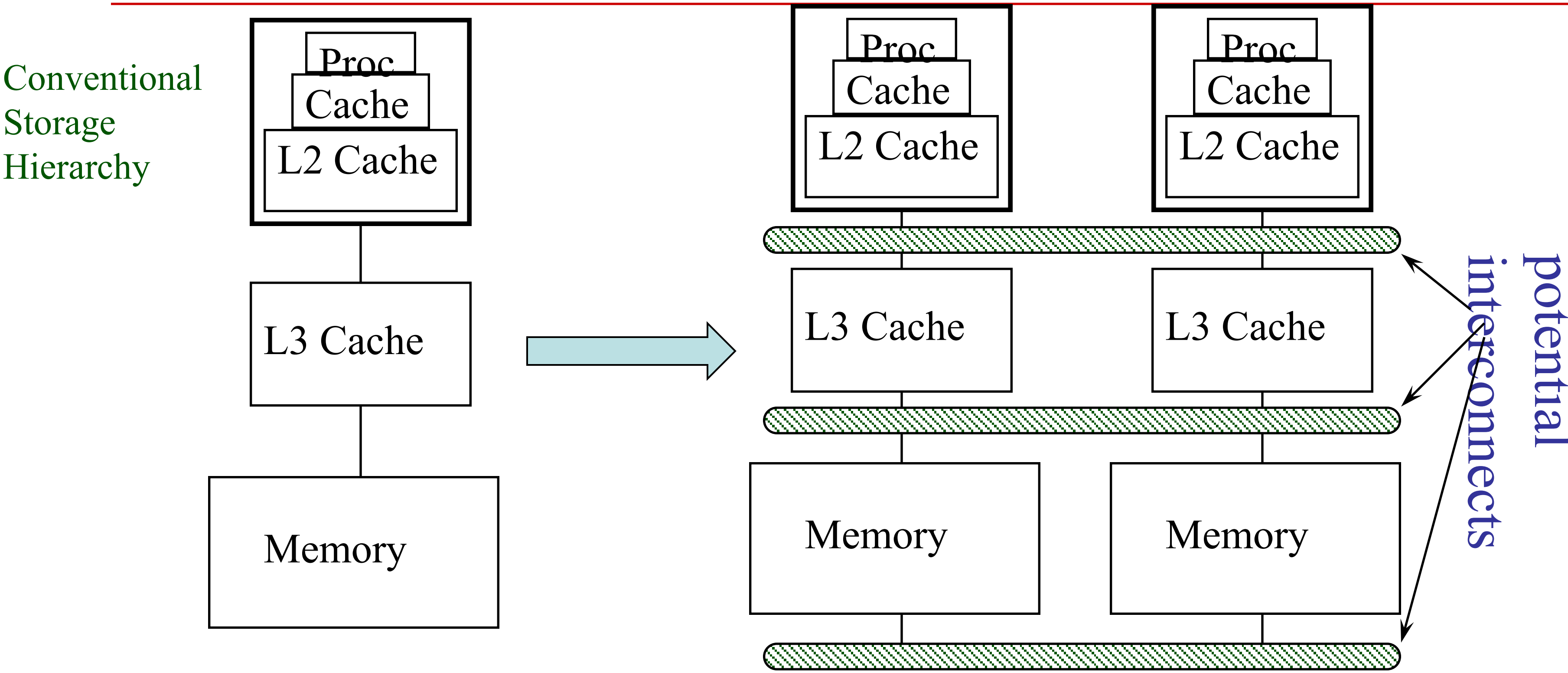
⇨ All of these things makes parallel programming even harder than sequential programming.

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

Slide source: Demmel/Yelick

Locality and Parallelism



- **Large memories are slow, fast memories are small**
- **Slow accesses to "remote" data or communicate with other machines**
- **Algorithm should do most work on local data, and minimize communication overhead**

Slide source: Demmel/Yelick

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:** tree-structured computations. 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"

Slide source: Demmel/Yelick

Improving Real Performance

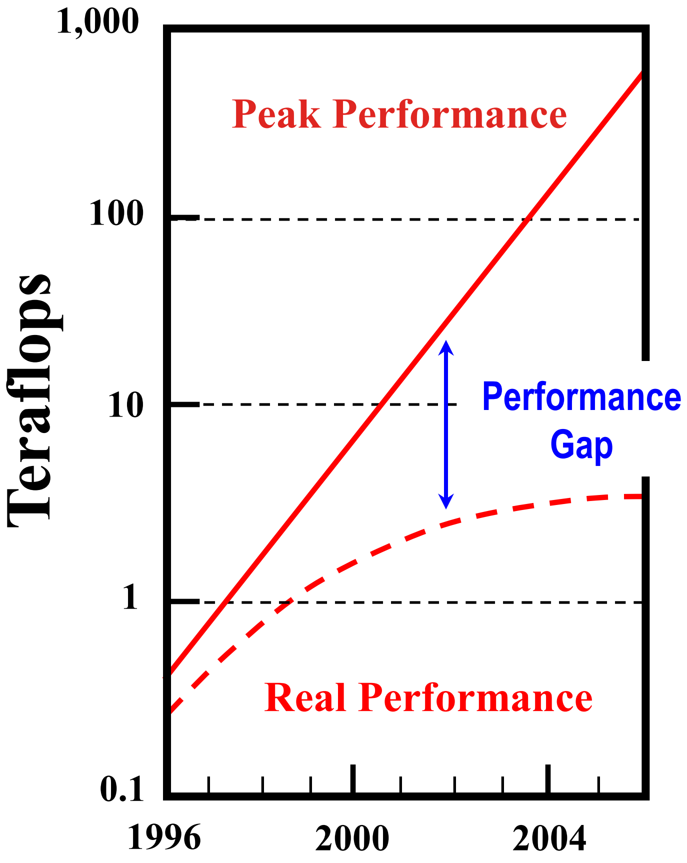
Peak Performance grows exponentially

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 ...

- Computing 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



Slide source: Demmel/Yelick

Roadmap

- Why all computers must be parallel computing
- Why parallel processing?
  - Large Computational Science and Engineering (CSE) problems require powerful computers
  - Commercial data-oriented computing also needs.
- Why writing (fast) parallel programs is hard
- Class Information

Course Objective

In depth understanding of:

- When is parallel computing useful?
- Understanding of parallel computing hardware options.
- Overview of programming models (software) and tools and performance analysis
- Some important parallel applications and the algorithms for scientific/data-intensive computing

Course Topics

- **High performance computing**
  - Basics of computer architecture, clusters&cloud systems. Storage.
- **Parallel programming models, software/libraries**
  - Task graph computation. Embarrassingly parallel, divide-and-conquer, and pipelining.
  - Partitioning and mapping of program/data for shared memory vs distributed memory
  - Threads, MPI, MapReduce/Hadoop, and openMP if time permits
- **Patterns of parallelism. Optimization techniques for parallelization and performance**
- **Core computing algorithms in scientific and data-intensive web applications**

Class Computing Resource

TSCC Cluster at San Diego Supercomputer Center

Computing: Up to 512 cores.
Node architecture

- 16 cores/machine, 2.6GHz Intel Xeon E5-2670 (Sandy Bridge)
- Memory: 64GB per machine

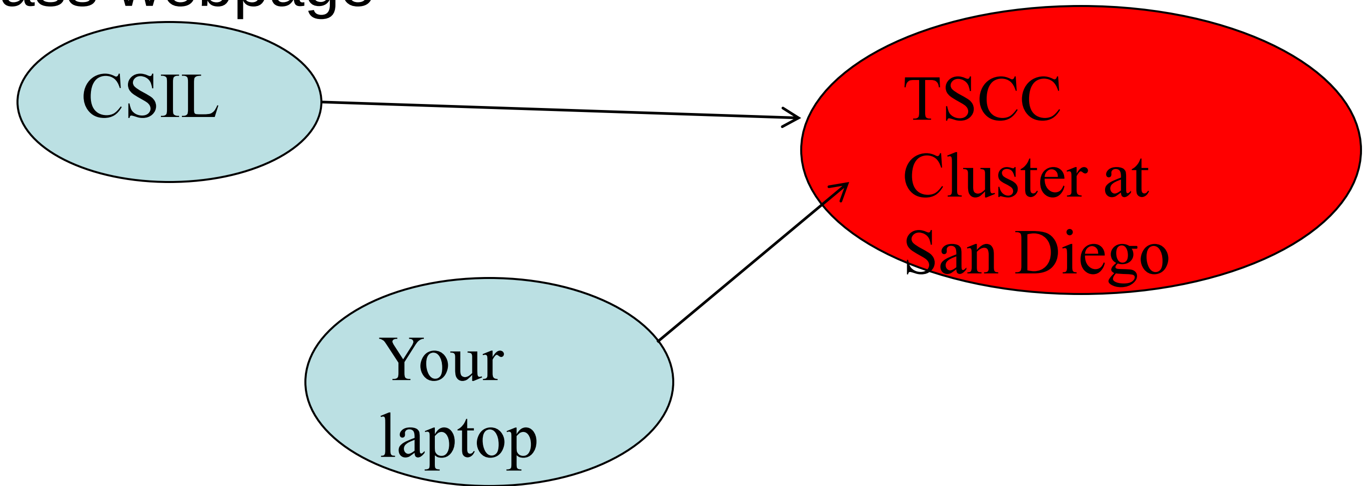
Network: 10GbE (QDR InfiniBand optional)

Storage: 100GB/user with a backup.
200TB shared scratch space
available to all users.

Class Computing Resource

- **Triton Shared Computing Cluster (TSCC) accounts:**
  - Apply in week 1

    Get a class account in Triton by emailing your name, UCSB email, and ssh public key with subject "CS140 ssh key" to scc@oit.ucsb.edu .
  - Instructions on generating ssh keys can be found in class webpage



Prerequisites and Misc Info

- **Prerequisites**
  - *Data structure and algorithms* (CS 130A).
    - Graph, tree, stack, queue data structures
    - Sorting. Shortest path algorithms. Algorithm complexity
  - Programming experience with *C and Java on Linux*.
    - OS and programming experience!
  - Linear algebra (e.g. Math 5A or 4A)
    - Vectors, matrix. Linear equation solving.
  - Basic computer architecture (CPUs, cache, memory)
- **Class material is updated in** http://www.cs.ucsb.edu/~tyang/class/140s14
- **Text book source code:** http://www.cs.usfca.edu/~peter/ipp/
- **CS140 class discussion group** at Google

Course Workload and Challenges

- **Workload and weighting**

  2-person group homework (55%). Exams (45%).

  - 4-5 homework and programming assignments. One group interview.
  - Midterm (May 6) Final (June 11?)

- **Challenges**
  - Textbook/documents may not represent the latest development:
    - Parallel system is complex. Big data/large scale computing is hard
    - Parallel computing technology evolves fast in last ten years.
    - Documentation is weak (e.g. Hadoop Mapreduce)
  - Reading with self-searching of web material is needed.