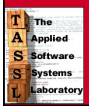
RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

ECE-451/ECE-566 - Introduction to Parallel and Distributed Programming

Lecture 1: Introduction

Department of Electrical & Computer Engineering
Rutgers University



Course Information

- Class: Mon/Wed 8:10-9:30pm, PH-111
- Instructors:
  - Manish Parashar

    CoRE 628
    parashar@rutgers.edu
    http://nsfcac.rutgers.edu/people/parashar
  - Ivan Rodero

    CoRE 625
    irodero@rutgers.edu
    http://nsfcac.rutgers.edu/people/irodero
- Class website:

  http://nsfcac.rutgers.edu/people/irodero/classes/13-14/ece451-566/
- Office hours: **by appointment**

Objectives

- Introduction to parallel/distributed programming
  - Parallel/distributed **programming**
    - Approaches, paradigms, tools, etc.
  - Issues and approaches in parallel/distributed **application development**
  - **Current state** of parallel/distributed architectures and systems
  - Current and **future trends**

Course Organization

- Introduction (today)
- 3 main units
  - **Parallel** Computing/Programming
    - What is parallelism? How are parallel systems?
    - How to evaluate performance?
    - Parallel programming with threads, OpenMP and MPI
  - Programming with **accelerators** (GPU)
    - Programming with CUDA and OpenCL
  - **Distributed** Computing/Programming
    - Distributed paradigms and
    - Cloud computing
    - MapReduce/Hadoop

Grading Policy

- Undergraduate (ECE 451)
  - Programming Assignments (3): 30%
  - Midterms (2): 30%
  - Final Project (**Groups of 2-3 students**): 30%
  - Quizzes/Class Participation: 10%
- Graduate (ECE 566)
  - Programming Assignments (3): 30%
  - Midterms (2): 30%
  - Final Project (**Individual or group of 2 students**): 30%
  - Quizzes/Class Participation: 10%

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Why we need powerful computers?

TASSL The Applied Software Systems Laboratory

COC

Demand for Computational Speed

- Continual demand for greater computational speed from a computer system than is currently possible
- Areas requiring great computational speed include numerical modeling and simulation of scientific and engineering problems.
- Computations must be completed within a "**reasonable**" time period.

Simulation: The Third Pillar of Science

- Traditional scientific and engineering paradigm:
  - Do theory or paper design.
  - 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 paradigm:
  - Use high performance computer systems to simulate the phenomenon
    - Base on known physical laws and efficient numerical methods.

Some Particularly Challenging Computations

- Science
  - Global climate modeling
  - Astrophysical modeling
  - Biology: Genome analysis; protein folding (drug design)
- Engineering
  - Crash simulation
  - Semiconductor design
  - Earthquake and structural modeling
- Business
  - Financial and economic modeling
  - Transaction processing, web services and search engines
- Defense
  - Nuclear weapons -- test by simulations
  - Cryptography

RUTGERS ECE 451/566, Fall'13

Units of Measure in HPC

- High Performance Computing (HPC) units are:
  - Flop/s: floating point operations
  - Bytes: size of data

- Typical sizes are millions, billions, trillions…

Mega	Mflop/s = 10^6 flop/sec	Mbyte = 10^6 byte (also 2^{20} = 1048576)
Giga	Gflop/s = 10^9 flop/sec	Gbyte = 10^9 byte (also 2^{30} = 1073741824)
Tera	Tflop/s = 10^{12} flop/sec	Tbyte = 10^{12} byte (also 2^{40} = 10995211627776)
Peta	Pflop/s = 10^{15} flop/sec	Pbyte = 10^{15} byte (also 2^{50} = 1125899906842624)
Exa	Eflop/s = 10^{18} flop/sec	Ebyte = 10^{18} byte
Yotta	Yfop/s = 10^{21} flop/sec	Ebyte = 10^{21} byte
Zetta	Zflop/s = 10^{24} flop/sec	Ebyte = 10^{24} byte

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.
- Securities industry:
  - Savings: approx. $15 billion per year for U.S. home mortgages.

RUTGERS ECE 451/566, Fall'13

Other impacts (e.g., Natural Hazards)

Hurricane Mitigation Background

- Computationally Intensive
- Improvement requires cross-disciplinary expertise
- High Performance Computing
- Resource Allocation
- Weather Modeling
- Weather Research and Forecasting (WRF)

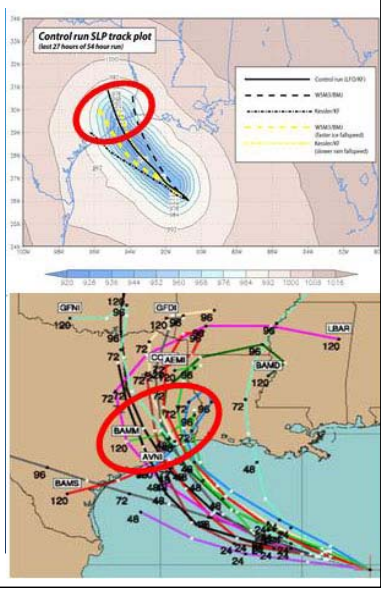
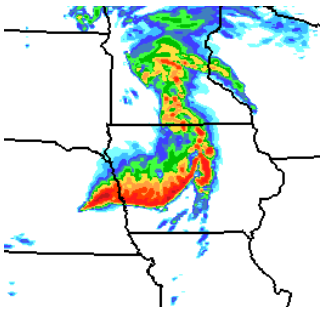


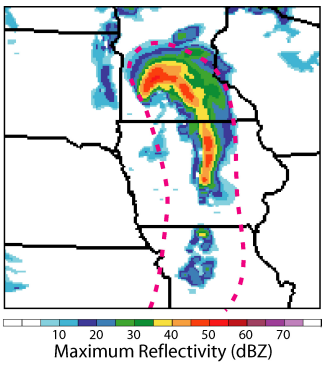
Image Source: http://mls.jpl.nasa.gov

Performance need

4-km WRF



10-km WRF



Parameterized convection (on the 10 km grid) cannot differentiate different mode of convection

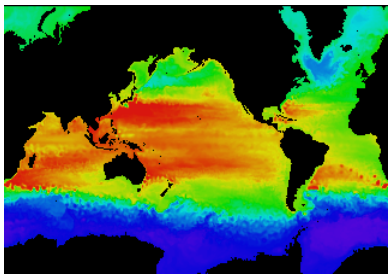
Dashed magenta indicates approximate area of rainfall Produced by convective parameterization

Source: The National Center for Atmospheric Research)

Global Climate Modeling Problem

- Problem is to compute:
  f(latitude, longitude, elevation, time) →
  temperature, pressure, humidity, wind velocity
- Approach:
  - Discretize the domain, e.g., a measurement point every 1km
  - Devise an algorithm to predict weather at time t+1 given t

- **Uses:**
  - **Predict major events, e.g., El Nino**
  - **Use in setting air emissions standards**



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

RUTGERS ECE 451/566, Fall'13

Heart Simulation

- Problem is to compute blood flow in the heart
- Approach:
  - Modeled as an elastic structure in an incompressible fluid.
    - The "immersed boundary method" due to Peskin and McQueen.
    - 20 years of development in model
    - Many applications other than the heart: blood clotting, inner ear, paper making, embryo growth, and others
  - Use a regularly spaced mesh (set of points) for evaluating the fluid
- Uses
  - Current model can be used to design artificial heart valves
  - Can help in understand effects of disease (leaky valves)
  - Related projects look at the behavior of the heart during a heart attack
  - Ultimately: real-time clinical work, design new drugs

Heart Simulation Calculation

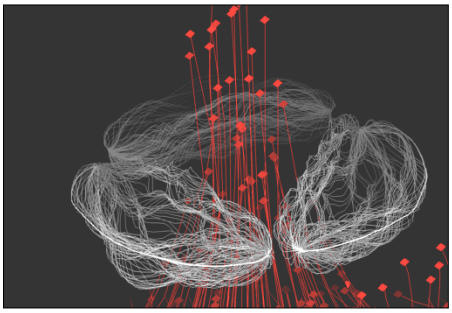
The involves solving Navier-Stokes equations

–64^3 was possible on Cray YMP, but 128^3 required for accurate model (would have taken 3 years).

–Done on a Cray C90 -- 100x faster and 100x more memory

–Until recently, limited to vector machines

- **Needs more features:**
  - **Electrical model of the heart, and details of muscles, E.g.,**
    - **Chris Johnson**
    - **Andrew McCulloch**
  - **Lungs, circulatory systems**



RUTGERS ECE 451/566, Fall'13

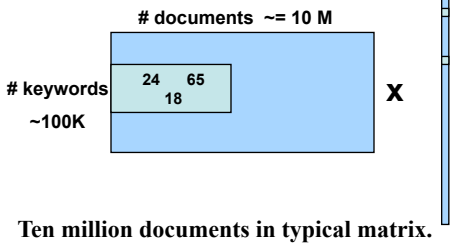
Parallel Computing in Web Search

- Functional parallelism: crawling, indexing, sorting
- Parallelism between queries: multiple users
- Finding information amidst junk
- Preprocessing of the web data set to help find information

- General themes of sifting through large, unstructured data sets:
  - when to put white socks on sale
  - what advertisements should you receive
  - finding medical problems in a community

Document Retrieval Computation

- Approach:
  - Store the documents in a large (sparse) matrix
  - Use Latent Semantic Indexing (LSI), or related algorithms to "partition"
  - Needs large sparse matrix-vector multiply



•Matrix is compressed
•"Random" memory access
•Scatter/gather vs. cache miss per 2Flops

Ten million documents in typical matrix.
Web storage increasing 2x every 5 months.
Similar ideas may apply to image retrieval.

RUTGERS
THE STATE UNIVERSITY OF NEW JERSEY

Why powerful (all ?) computers are parallel?

TASSL The Applied Software Systems Laboratory

COC

How to Run Applications Faster ?

- There are 3 ways to improve performance:
  1. Work Harder
  2. Work Smarter
  3. Get Help

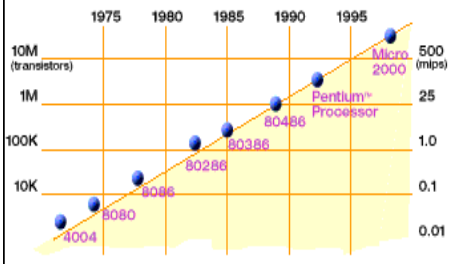
- Computer Analogy
  1. Use faster hardware: e.g. reduce the time per instruction (clock cycle).
  2. Optimized algorithms and techniques, customized hardware
  3. Multiple computers to solve problem: That is, increase no. of instructions executed per clock cycle.

What is Parallel Computing?

- **Parallel computing:** using multiple processors in parallel to solve problems more quickly than with a single processor
- Examples of parallel machines:
  - A **cluster computer** that contains multiple PCs combined together with a high speed network
  - A **shared memory multiprocessor** (SMP*) by connecting multiple processors to a single memory system
  - A **Chip Multi-Processor** (CMP) contains multiple processors (called cores) on a single chip
- Concurrent execution comes from desire for performance; unlike the inherent concurrency in a multi-user distributed system

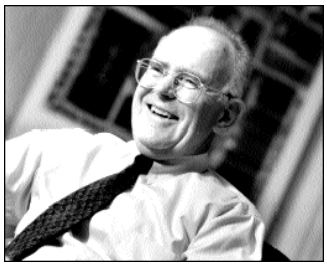
* Technically, SMP stands for "Symmetric Multi-Processor"

Technology Trends: Microprocessor Capacity



2X transistors/Chip Every 1.5 years
Called "Moore's Law"

Microprocessors have become smaller, denser, and more powerful.

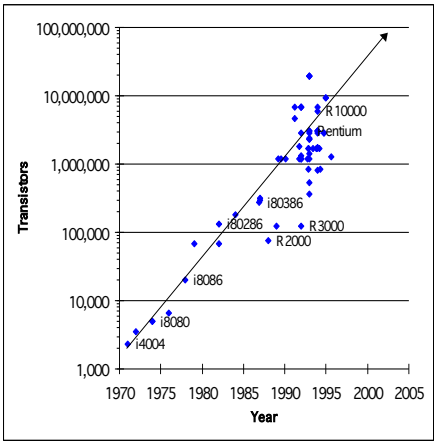


Gordon Moore (co-founder of Intel) predicted in 1965 that the transistor density of semiconductor chips would double roughly every 18 months.

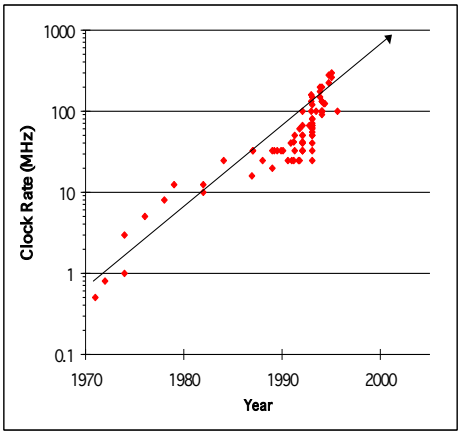
Slide source: Jack Dongarra

Microprocessor Transistors and Clock Rate

Growth in transistors per chip



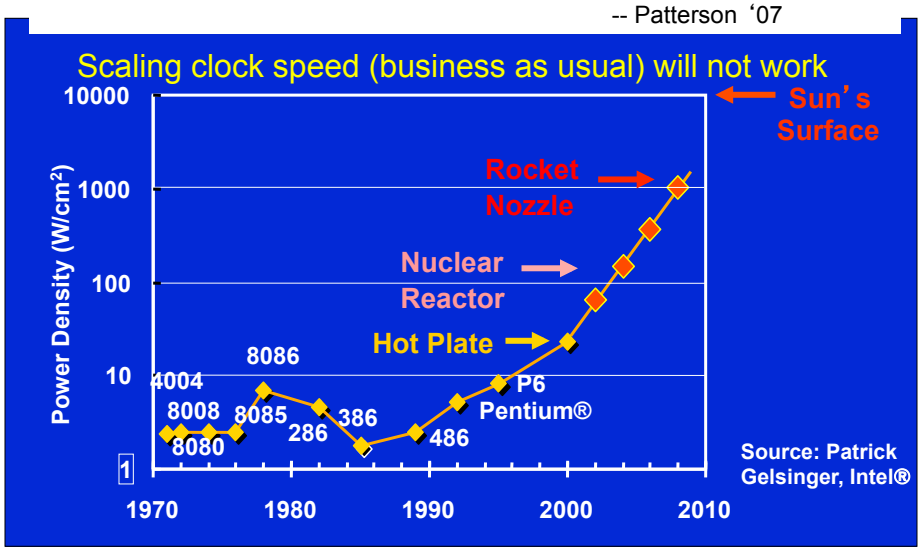
Increase in clock rate



Why bother with parallel programming? Just wait a year or two…

Power density limitations (heat becoming an unmanageable problem)

Can soon put more transistors on a chip than can afford to turn on.
-- Patterson '07



Sequential Architecture Limitations

- Sequential architectures reaching physical limitation
  - Speed of light, thermodynamics, etc.
- Hardware improvements like pipelining, superscalar, etc., are non-scalable and requires **sophisticated** compiler technology
- Vector processing works well for certain kind of problems

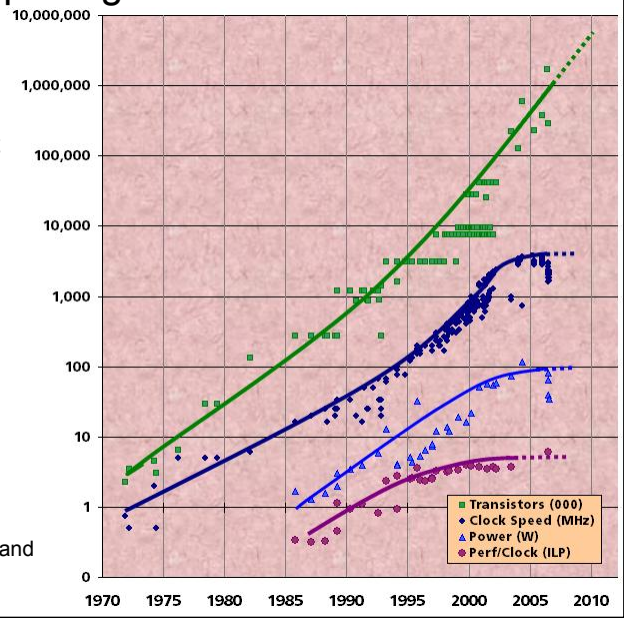
How fast can a serial computer be?

- Consider the 1 Tflop/s sequential machine:
  - Data must travel some distance, r, to get from memory to CPU.
  - To get 1 data element per cycle, this means 10^{12} times per second at the speed of light, $c = 3x10^8$ 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 word occupies about 3 square Angstroms, or the size of a small atom.

Revolution is Happening Now

- Chip density is continuing increase ~2x every 2 years
  - Clock speed is not
  - Number of processor cores may double instead
- There is little or no hidden parallelism (ILP) to be found
- Parallelism must be exposed to and managed by software

Source: Intel, Microsoft (Sutter) and Stanford (Olukotun, Hammond)



Parallelism Saves Power

- Exploit explicit parallelism for reducing power

Power = (C * V^2 * F)/4 Performance = (Cores * F)*1

Capacitance Voltage Frequency

- **Using additional cores**
  - Increase density (= more transistors = more capacitance)
  - Can increase cores (2x) and performance (2x)
  - Or increase cores (2x), but decrease frequency (1/2): same performance at ¼ the power
- **Additional benefits**
  - Small/simple cores → more predictable performance

RUTGERS ECE 451/566, Fall'13

What are Scalable Systems?

- Wikipedia Definition:
  - A scalable system is one that can easily be altered to accommodate changes in the number of users, resources and computing entities affected to it. Scalability can be measured in three different dimensions:
- Load scalability
  - A distributed system should make it easy for us to expand and contract its resource pool to accommodate heavier or lighter loads.
- Geographic scalability
  - A geographically scalable system is one that maintains its usefulness and usability, regardless of how far apart its users or resources are.
- Administrative scalability
  - No matter how many different organizations need to share a single distributed system, it should still be easy to use and manage.
- Some loss of performance may occur in a system that allows itself to scale in one or more of these dimensions.

Dimensions of Scalability

- Grassroots Scalability – Multicores, GPGPUs, etc.
- High-End HPC Systems
- Enterprise Systems (Datacenters)
- Wide-area Scalability

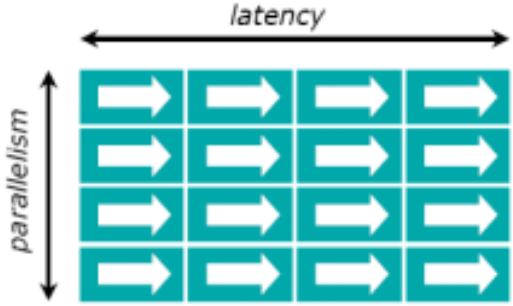
Grassroots Scalability – The Memory Bottleneck

Memory bottleneck:

- Cores share finite off-chip bandwidth
- Latency cannot be improved significantly
  ➔ *and is already 100's of cycles!*

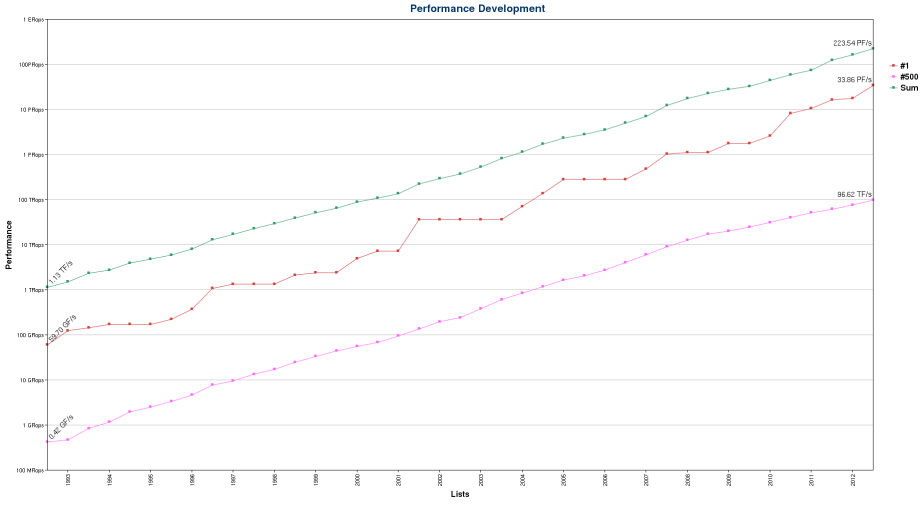
Grassroots Scalability – Hiding Latency

- To hide latency, need ***more*** parallelism than number of cores, ***schedule*** data:

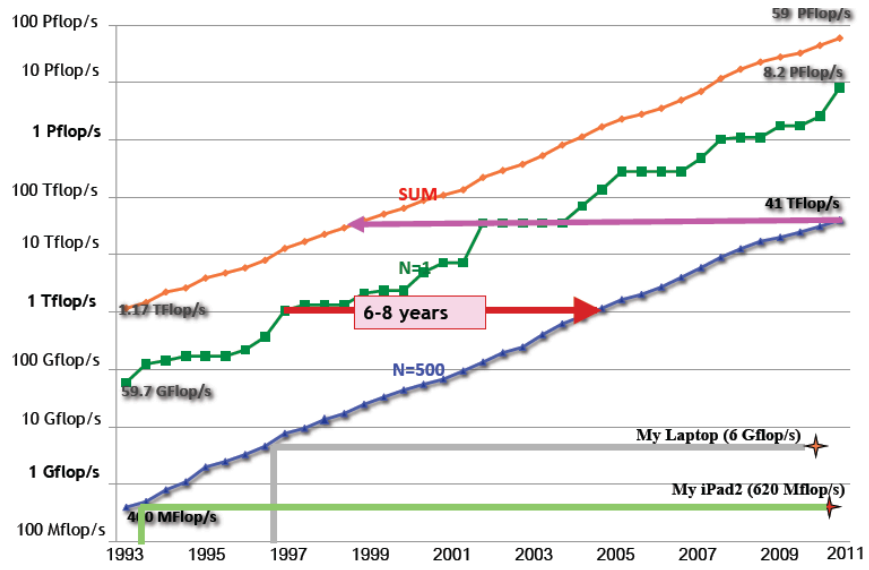


Concurrency = parallelism x latency

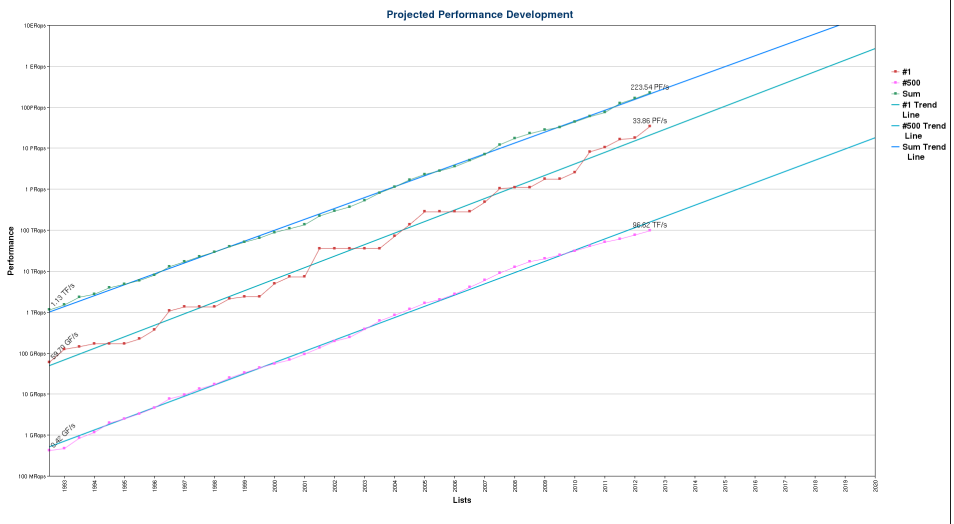
HPC Performance Trends (http://top500.org)



HPC Performance Trends



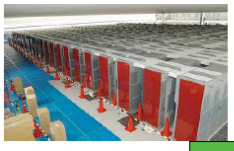
HPC Performance Trends



Top of the Top500

Rank	Site	System	Cores	Rmax (TFlop/s)	Rpeak (TFlop/s)	Power (kW)
1	National University of Defense Technology China	**Tianhe-2 (MilkyWay-2)** - TH-IVB-FEP Cluster, Intel Xeon E5-2692 12C 2.200GHz, TH Express-2, Intel Xeon Phi 31S1P NUDT	3120000	33862.7	54902.4	17808
2	DOE/SC/Oak Ridge National Laboratory United States	**Titan** - Cray XK7 , Opteron 6274 16C 2.200GHz, Cray Gemini interconnect, 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	17173.2	20132.7	7890
4	RIKEN Advanced Institute for Computational Science (AICS) Japan	K computer, SPARC64 VIIIfx 2.0GHz, Tofu interconnect Fujitsu	705024	10510.0	11280.4	12660
5	DOE/SC/Argonne National Laboratory United States	**Mira** - BlueGene/Q, Power BQC 16C 1.60GHz, Custom IBM	786432	8586.6	10066.3	3945
6	Texas Advanced Computing Center/Univ. of Texas United States	**Stampede** - PowerEdge C8220, Xeon E5-2680 8C 2.700GHz, Infiniband FDR, Intel Xeon Phi SE10P Dell	462462	5168.1	8520.1	4510
7	Forschungszentrum Juelich (FZJ) Germany	**JUQUEEN** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	458752	5008.9	5872.0	2301
8	DOE/NNSA/LLNL United States	**Vulcan** - BlueGene/Q, Power BQC 16C 1.600GHz, Custom Interconnect IBM	393216	4293.3	5033.2	1972
9	Leibniz Rechenzentrum Germany	**SuperMUC** - iDataPlex DX360M4, Xeon E5-2680 8C 2.70GHz, Infiniband FDR IBM	147456	2897.0	3185.1	3423
10	National Supercomputing Center in Tianjin China	**Tianhe-1A** - NUDT YH MPP, Xeon X5670 6C 2.93 GHz, NVIDIA 2050 NUDT	186368	2566.0	4701.0	4040

Architecture of a Typical High-End HPC System



HPC Scalability Issues

- Parallelism Model/Paradigm
  - SIMD, SPMD, MIMD, Vector
  - DAG/Event Driven
  - Hierarchical/Multi-level
- Runtime Management
  - Load-balancing
  - Adaptation
  - Fault-Tolerance
  - Data management
- Locality! Locality! Locality!
  - Memory, Cache
  - Communication

Principles of Parallel Computing

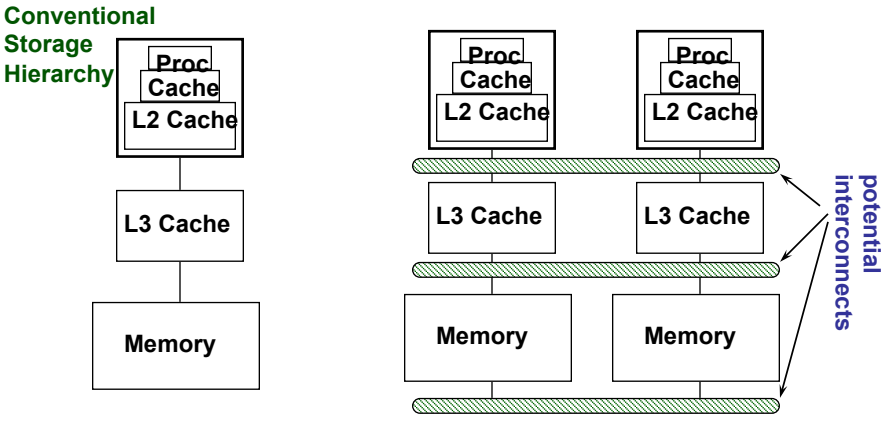
- Parallelism and Amdahl's Law
- Finding and exploiting granularity
- Preserving data locality
- Load balancing
- Coordination and synchronization
- Performance modeling

All of these things make parallel programming more difficult than sequential programming

Overhead of Parallelism

- Given enough parallel work, this is the most significant 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 memories -- 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
  - but techniques to balance load often reduce locality

"Automatic" Parallelism in Modern Machines

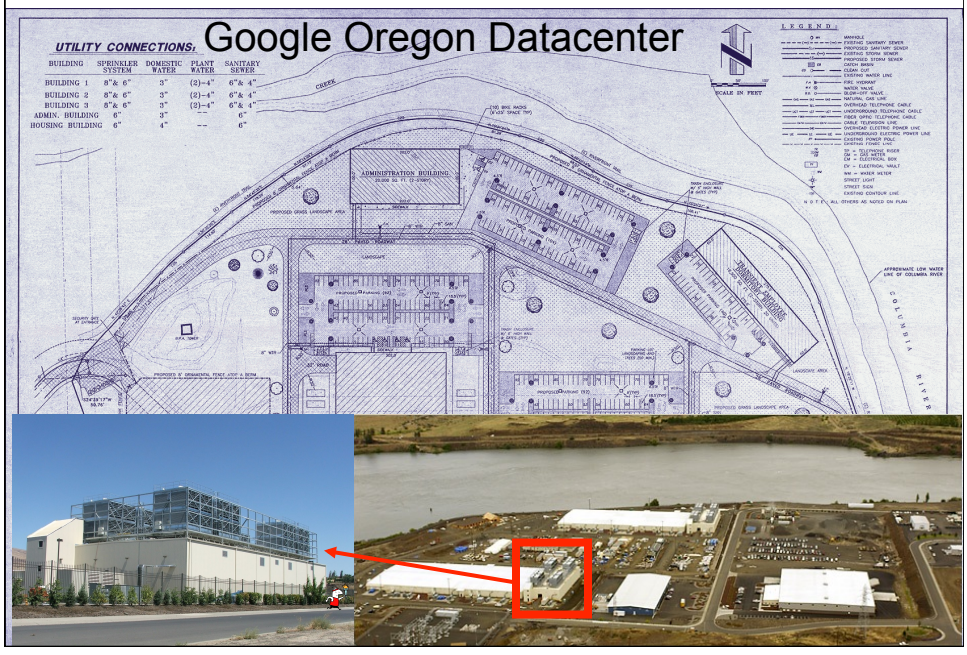
- Bit level parallelism: within floating point operations, etc.
- Instruction level parallelism (ILP): multiple instructions execute per clock cycle.
- Memory system parallelism: overlap of memory operations with computation.
- OS parallelism: multiple jobs run in parallel on commodity SMPs.

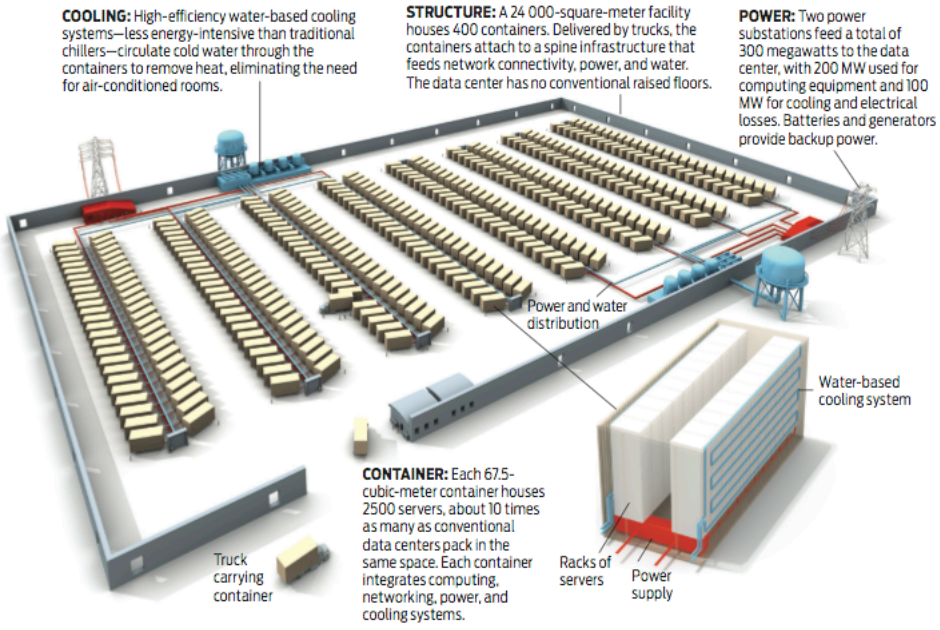
- There are limitations to all of these!
- Thus to achieve high performance, the programmer needs to identify, schedule and coordinate parallel tasks and data.

Enterprise Scalability - Datacenter is new "server"

- *"Program"* == Web search, email, map/GIS, …
- *"Computer"* == 1000's computers, storage, network
- Warehouse-sized facilities and workloads
- New datacenter ideas (2007-2008): truck container (Sun), floating (Google), datacenter-in-a-tent (Microsoft)
- ***How to enable innovation in new services without first building and capitalizing a large company?***

photos: Sun Microsystems & datacenterknowledge.com





Enterprise Datacenters – For Example

- A 50-rack datacenter
  - X 64 blades per rack
  - X 2 sockets per blade
  - X 32 cores per socket
  - X 10 VMs per core
  - X 10 Java objects per VM

20,480,000

managed objects

Enterprise Application Scalability - Problem

- How do you scale up applications?
  - Run jobs processing 100's of terabytes of data
  - Takes 11 days to read on 1 computer
- Need lots of cheap computers
  - Fixes speed problem (15 minutes on 1000 computers), but…
  - Reliability problems
    - In large clusters, computers fail every day
    - Cluster size is not fixed
- Need common infrastructure
  - Must be efficient and reliable

Enterprise Application Scalability - Solution

- Open Source Apache Project
- Hadoop Core includes:
  - Distributed File System - distributes data
  - Map/Reduce - distributes application
- Written in Java
- Runs on
  - Linux, Mac OS/X, Windows, and Solaris
  - Commodity hardware

Hadoop clusters

- ~20,000 machines running Hadoop @ Yahoo!
- Several petabytes of user data (compressed, unreplicated)
- Hundreds of thousands of jobs every month

Conclusion

- All processors will be multi-core
- All computer will be massively parallel
- All programmers will be parallel programmers
- All programs will be parallel programs

- The key challenge is how to enable mainstream developers to scale their applications on machines!
  - Algorithms – scalable, asynchronous, fault-tolerant
  - Programming models and systems
  - Adaptive runtime management
  - Data management
  - System support

Group Questions

- How should you study the final exam?
- How many assignments in this course?
- Do you need programming skills for the assignments?
- How long the final project will take?
- How many members in a project group?
- How important is the final project?
- Should you get involved in class?