CS267 / E233
Applications of Parallel Computers

Lecture 1: Introduction

1/18/99

James Demmel

demmel@cs.berkeley.edu

http://www.cs.berkeley.edu/~demmel/cs267_Spr99

Outline

- Introductions
- Why large important problems require the capabilities of powerful computers
- Why powerful computers must be parallel processors
- Structure of the course

Administrative

- Instructors
  - Prof. Jim Demmel, 737 Soda, demmel@cs.berkeley.edu
  - TA: Fred Wong, 533 Soda, fredwong@cs.berkeley.edu
- Office hours
  - T Th 2:15 - 3:30, and by appointment
- Accounts and others -- fill out online registration!
- Class survey -- fill out online!
- Discussion section: TBD, based on survey
- Most class material will be on class home page (including these notes):
  - www.cs.berkeley.edu/~demmel/cs267_Spr99

Why we need powerful computers

Units of High Performance Computing

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

Why we need powerful computers

° **Traditional scientific and engineering paradigm**
  - **Do theory or paper design**
  - **Perform experiments or build system**

° **Replacing both by numerical experiments**
  - **Real phenomena are too complicated to model by hand**
  - **Real experiments are:**
    - **too hard, e.g., build large wind tunnels**
    - **too expensive, e.g., build a throw-away passenger jet**
    - **too slow, e.g., wait for climate or galactic evolution**
    - **too dangerous, e.g., weapons, drug design**

° **Why parallel computers for this? Serial Computers too slow**

Some Challenge Computations

- Global Climate Modeling
- Dyna3D- crash simulation
- Astrophysical modeling
- Earthquake (structures) modeling
- Heart simulation
- Web search
- Transaction processing
- Drug design
- Phylogeny -- History of species
- Nuclear Weapons
- now.cs.berkeley.edu/Millennium

Global Climate Modeling

- **Climate is a function of 4 arguments**
  Climate(longitude, latitude, elevation, time)
- **Which returns a vector of 6 values**
  Temperature, pressure, humidity, and wind velocity
- **To model this on a computer we**
  - discretize the domain using a finite grid, e.g., points 1 kilometer apart
    - roughly .1 TB of data
  - devise and algorithm to predict weather at time t+1 from weather at time t
    - e.g., solving Navier-Stokes equations for fluid flow of gasses in the atmosphere
    - say this is roughly 100 Flops per grid point with a timestep of 1 minute
  - to at least match real time (bare minimum)
    - 5*10^11 flops / 60 secs = 8 Gflop/s
  - weather prediction (7 days in 24 hours) => 7x faster => 56 Gflop/s
  - climate prediction (50 years in 30 days) => 50*12=600x faster => 4.8 Tflops
- **Current models use much coarser grids**
  - www-fp.mcs.anl.gov/chammp

Heart Simulation

° Many biological structures can be modeled as an elastic structure in an incompressible fluid.

° Using the "immersed boundary method" this involves solving Navier-Stokes equations plus some feature-specific computation on the bodies [Peskin&McQueen]

° 20 years of development in model, used to design artificial valves

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

° Done on a Cray C90 -- could use 100x faster and 100x more memory

More computing power => more accurate (usable) model

Parallel Computing in Web Search

° Functional parallelism
  - crawling, indexing, sorting

° Parallelism between queries
  - multiple users

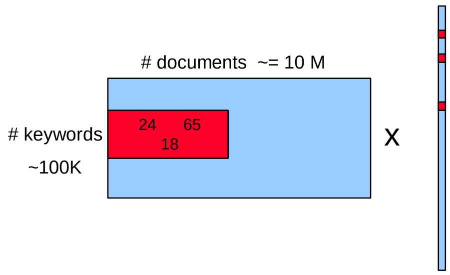
° Finding information amidst junk

° 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 kind of junk mail should you receive
  - finding medical problems in a community

Application: Document Retrieval

- Finding useful documents on the Web
- One algorithm, Latent Semantic Indexing (LSI), needs large sparse matrix-vector multiply



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

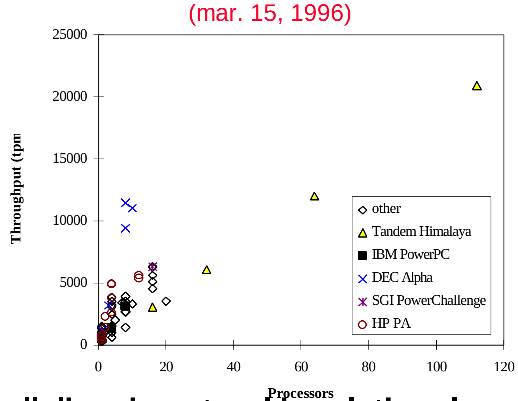
- 10 Million documents in typical matrix.
- Web storage increasing 2x every 5 months.
- Similar ideas may apply to image retrieval.

LSI Challenges

- On conventional microprocessor node
  - UltraSparc 166 MHz, 330 Mflops peak, Cache miss is 300 ns
  - Matrix-vector multiply, does roughly 3 loads and 2 flops, with 1.37 cache misses on average
  - ~4.5 Mflops (2-5 Mflops measured)
  - Memory accesses are irregular
- On T3E
  - Osni Marques at LBNL parallelized for the T3E
- Implementation is also I/O intensive

Transaction Processing - it's all parallel at some scale

(mar. 15, 1996)

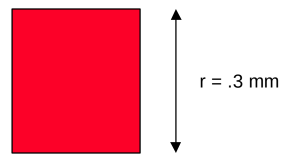


- Parallelism is natural in relational operators
  - select, join, ...
- Many difficult issues
  - data partitioning, locking, threading

Why powerful computers are parallel

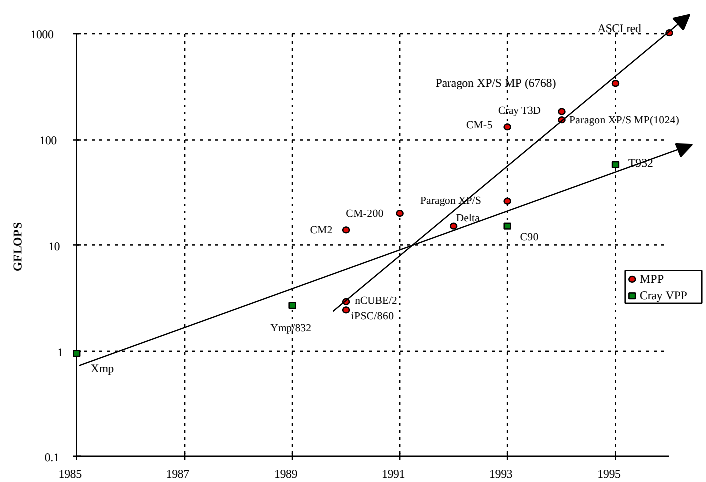
How fast can a serial computer be?

1 Tflop 1 TB
sequential
machine



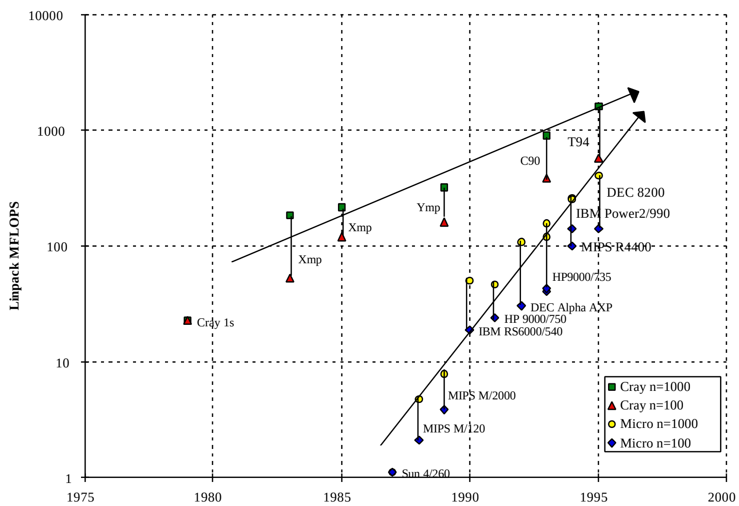
- Consider the 1 Tflop sequential machine
  - data must travel some distance, r, to get from memory to CPU
  - to get 1 data element per cycle, this means 10^12 times per second at the speed of light, c = 3e8 m/s
  - so r < c/10^12 = .3 mm
- Now put 1 TB of storage in a .3 mm^2 area
  - each word occupies about 3 Angstroms^2, the size of a small atom

Trends in Parallel Computing Performance

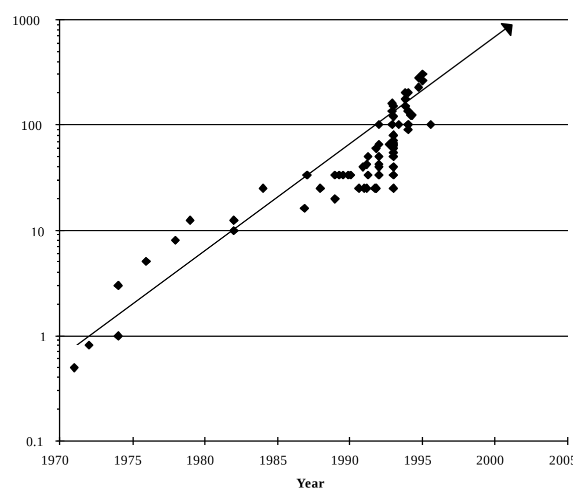


- 1 TFLOPS on Linpack, 12/16/96, ASCI Red (7264 Intel PPros)
- Up to 1.6 Tflops by 1/99, on ASCI Blue (5040 SGI R10ks)
- performance.netlib.org/performance/html/PDStop.html

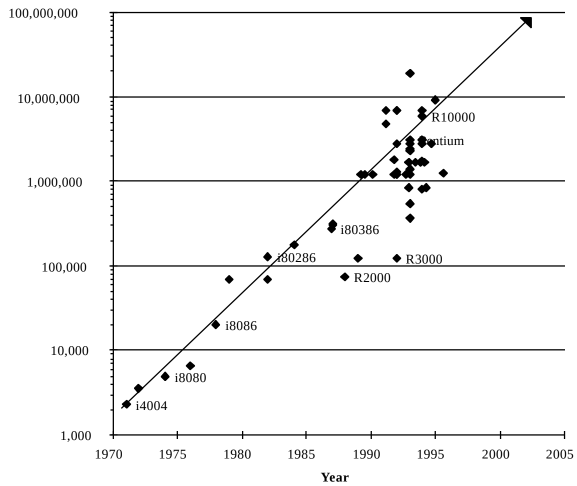
Empirical Trends: Microprocessor Performance



Microprocessor Clock Rate



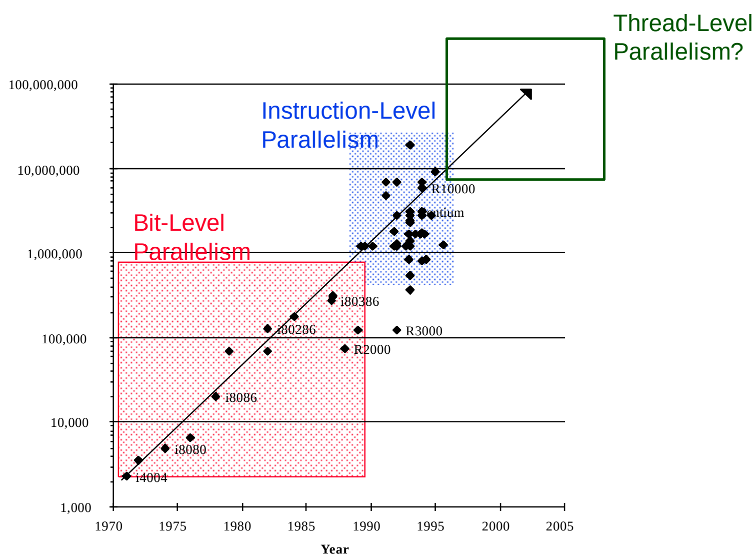
Microprocessor Transistors



CS267 L1 Intro.19

Demmel Sp 1999

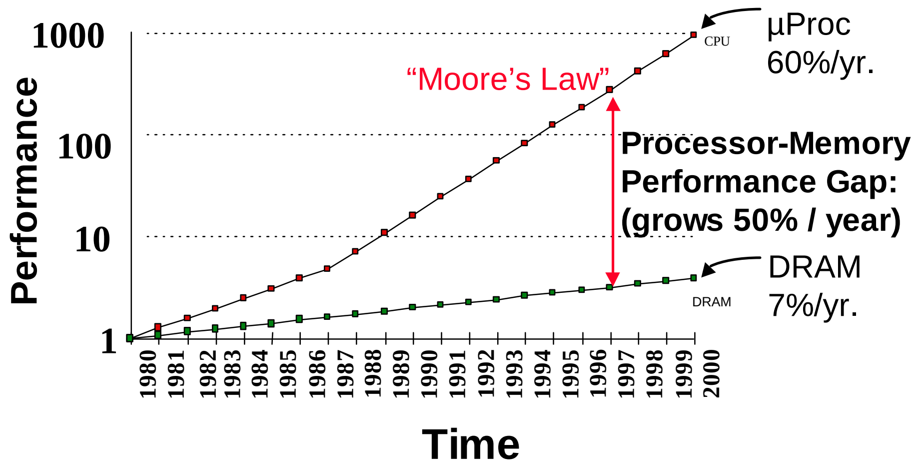
Microprocessor Transistors & Parallelism



CS267 L1 Intro.20

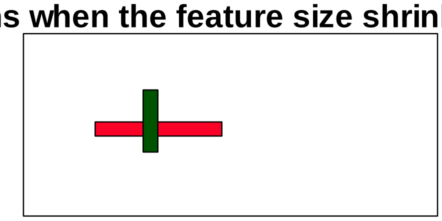
Demmel Sp 1999

Processor-DRAM Gap (latency)



1st Principles

- What happens when the feature size shrinks by a factor of x ?



- Clock rate goes up by x
  - 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 !
  - of which x^3 is devoted either to parallelism or locality

Principles of Parallel Computing

- Parallelism and 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

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

Speedup(P) = Time(1)/Time(P)

<= 1/(s + (1-s)/P)

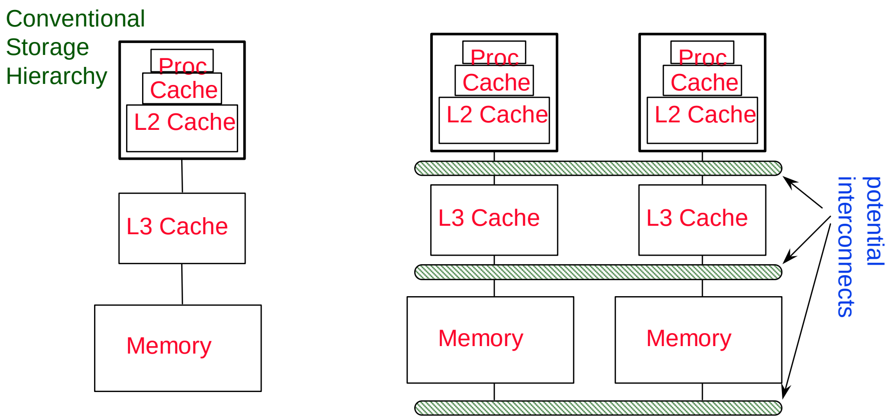
<= 1/s

- Even if the parallel part speeds up perfectly may be limited by the sequential part

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



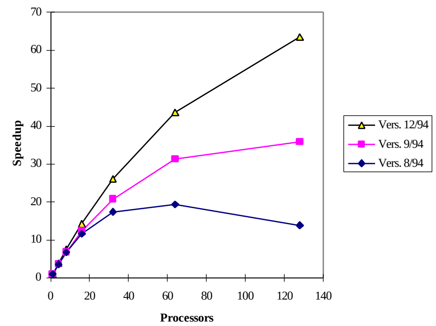
- Large memories are slow, fast memories are small
- Storage hierarchies are large and fast <u>on average</u>
- Parallel processors, collectively, have large, fast $
  - the slow accesses to "remote" data we call "communication"
- Algorithm should do most work on local data

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

Parallel Programming for Performance is Challenging

Amber (chemical modeling)



° Speedup(P) = Time(1) / Time(P)

° Applications have "learning curves"

Course Organization

Schedule of Topics

- Introduction
- Parallel Programming Models and Machines
  - Shared Memory and Multithreading
  - Distributed Memory and Message Passing
  - Data parallelism
- Sources of Parallelism in Simulation
- Algorithms and Software Tools (depends on student interest)
  - Dense Linear Algebra
  - Partial Differential Equations (PDEs)
  - Particle methods
  - Load balancing, synchronization techniques
  - Sparse matrices
  - Visualization (field trip to NERSC)
  - Sorting and data management
  - Metacomputing
- Applications (including guest lectures)
- Project Reports

Reading Materials

- 3 on-line texts
  - JD's notes from CS267 Spring 1996
  - Culler and Singh's, Parallel Computer Architecture (CS258 text, first chapter on-line)
  - Ian Fosters, "Designing and Building Parallel Programming"
- Papers, books to be on reserve
- the web (see class homepage for some pointers)

Computing Resources

° NOW
  • 100 Sun Ultrasparcs with a fast network

° 4 clustered Sun Enterprise 5000 8-proc SMPs

° Millennium prototype: clustered Intel SMPs

° Assorted other SMPs from IBM, DEC

° Possibly Cray T3E at NERSC for some projects of mutual interest

Requirements

° Fill out on-line account registration

° Fill out on-line survey, including available times for discussion section

° Weekly reading
  • be ready to discuss in class (10 %)

° ~4 programming assignments (25 %)
  • hands-on experience, interdisciplinary teams
  • if you don't do it yourself, you'll drop when the project gets interesting

° Midterm (20 %)

° Final Project (45 %)
  • teams of 3 - interdisciplinary is best
  • interesting applications or advance of systems

Projects

° Challenging team programming effort on a problem worth solving

° Conference quality publication

° Required presentation at end of semester

° Interdisciplinary (usually)

What you should get out of the course

In depth understanding of:

(1) how to apply parallel computers to demanding problems

(2) requirements of parallel applications (and their programmers)

(3) hardware, software, theory and practice of parallel computing

First Assignment

- See home page for details
- Find an application of parallel computing and build a web page describing it.
  - Choose something from your research area
  - Or from the web or elsewhere
- Evaluate the project. Was parallelism successful?
- Due one week from today (1/26)