CT5708701
Parallel and Distributed Computing

- Lecturer：Yo-Ming Hsieh
- Time: 09:30AM – 12:20PM (Tue.)
- Course website:

  On Blackboard System

  http://elearning.ntust.edu.tw

- Class room: IB-505
- Grading:
  - 30 pts: Mid/Final exam
  - 30 pts: Term project
  - 30 pts: Assignments
  - 10 pts: Class participation

Term Project

- The term project is about using parallel computing for solving a problem in engineering, science, mathematics, etc. – ask your advisor if you don't have good ideas…
- The project is to be delivered in 5 phases:
  1. 3/15/2011: A proposal for your term project.
  2. 4/12/2011: A complete description of the algorithm for solving your problem.
  3. 5/10/2011: Presenting the first stage of the development: the serial version of your final project
  4. 6/07/2011: Parallel version of your final project due
  5. 6/21/2011: Final presentation

Assignments

- Current plan (subject to changes) for assignments: **7 assignments** in total, and roughly one assignment for every two weeks. The first assignment is due on **Mar.1, 2011.**
- All assignments are to be submitted electronically. Either in Word document or PDF format. Your assignments will likely be posted publically so that the entire Internet will be able to access it!
- Policy:
  - Late assignments within one week will be accepted with a 10% discount.
  - Assignments late for more than one week will NOT be accepted.
  - **All references should be properly credited**. Fail to do so will either need to redo the assignment, or receive 0 on the assignment.
  - Copying you classmates' work is considered as cheating and you will get zero for that assignment.

Final Exam

- 6/21/2011(tentative)

Topics

- Prerequisite:
  - **C/C++ programming, Engineering Mathematics (linear algebra, vector, matrices)**
  - **Passionate about programming!**
- Introduction to Parallel Computing
- C/C^{++} Programming, advanced programming topics, and programming in Linux environment.
- MPI/OpenMP/CUDA Programming
- Parallel algorithms
  - Matrix operations
  - Sorting
  - Solving Ax=B
  - Finite difference method
  - …

References

- No textbook
- References
  - Parallel.And.Distributed.Programming.Using.C++, Cameron Hughes, et al., ISBN 0131013769 , Addison.Wesley
  - Reinders, James (2007, July). Intel Threading Building Blocks: Outfitting C++ for Multi-core Processor Parallelism (Paperback) Sebastopol: O'Reilly Media, ISBN 978-0-596-51480-8.
  - An Introduction to Parallel Computing: Design and Analysis of Algorithms, George Karypis, et al., ISBN 0201648652, Addison.Wesley
  - Beowulf Cluster Computing with Linux, Second Edition (Scientific and Engineering Computation), William Gropp et al., ISBN 0262692929, MIT Press
  - More references will be given during lectures in each topic.

References

- Related Course Sites
  - Berkeley, CS267
    - http://www.cs.berkeley.edu/~demmel/cs267_Spr10/
  - MIT, 18.337
    - http://beowulf.lcs.mit.edu/18.337/
- News website
  - http://insidehpc.com/
  - http://www.hpcwire.com/
  - http://www.linux-mag.com/topics/hpc/

Lecture 1

Introduction &
Computing Hardware

Topics

- Introduction
  - Definition
  - Applications
- Computing Hardware
  - Processor
  - Memory
- Parallel computer architecture
  - Shared Memory
  - Distributed Memory
- Modes of Parallel Processing

Introduction

What is “Parallel and Distributed Computing”

Definition

- Parallel Computing: the simultaneous execution of the same task on multiple processors in order to obtain faster results.
  - HPC: High Performance/Productivity Computing
  - Technical Computing
  - Cluster computing

- Distributed Computing: uses or coordinates physically separate computing resources
  - Grid computing / Cloud Computing

- Difference?

Applications

Some Particularly Challenging Computations (1/2)

- 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

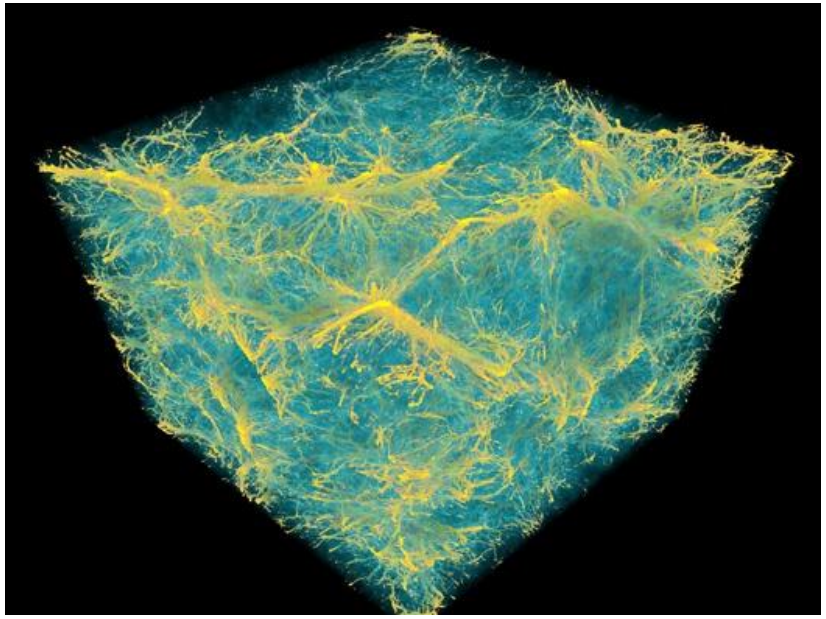
From: U.C.Berkeley CS267, Lecture 1 Notes

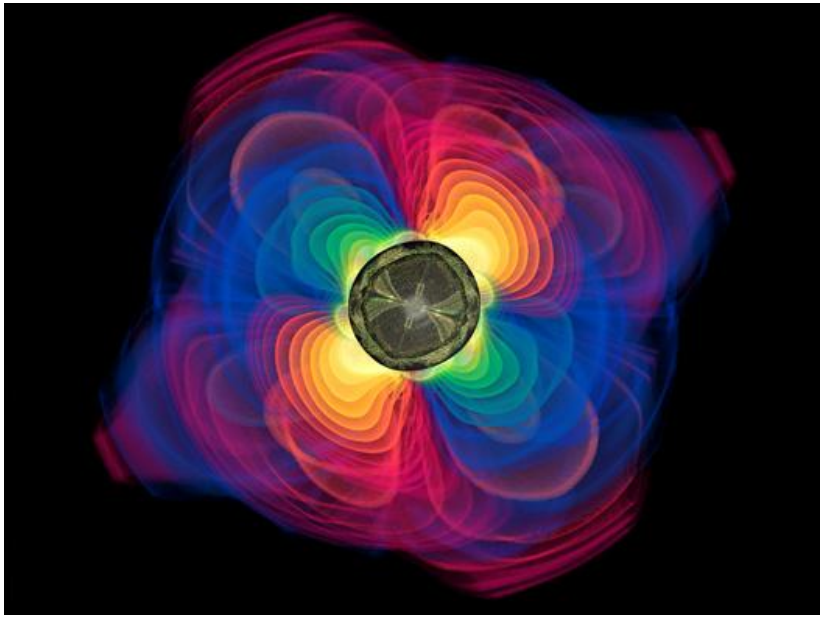
Some Particularly Challenging Computations (2/2)

- Business
  - Financial and economic modeling
  - Transaction processing, web services and search engines
- Defense
  - Nuclear weapons -- test by simulations
  - Cryptography

From: Berkeley CS267, Lecture 1 Notes

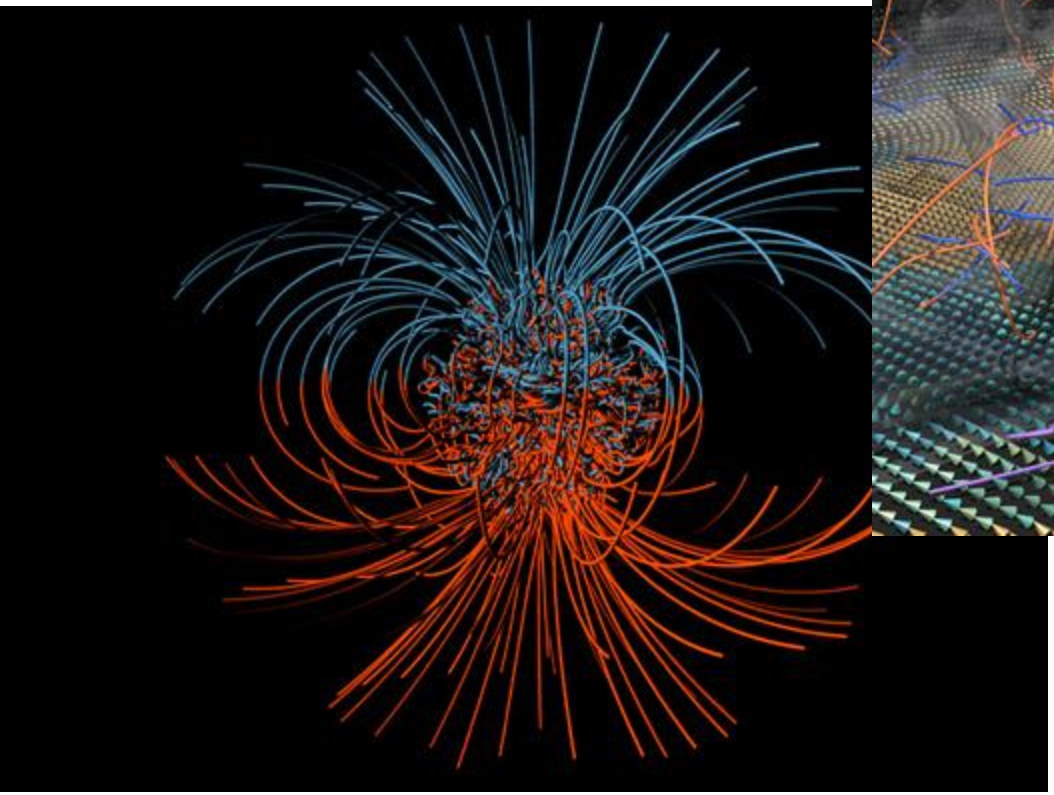
Astrophysics

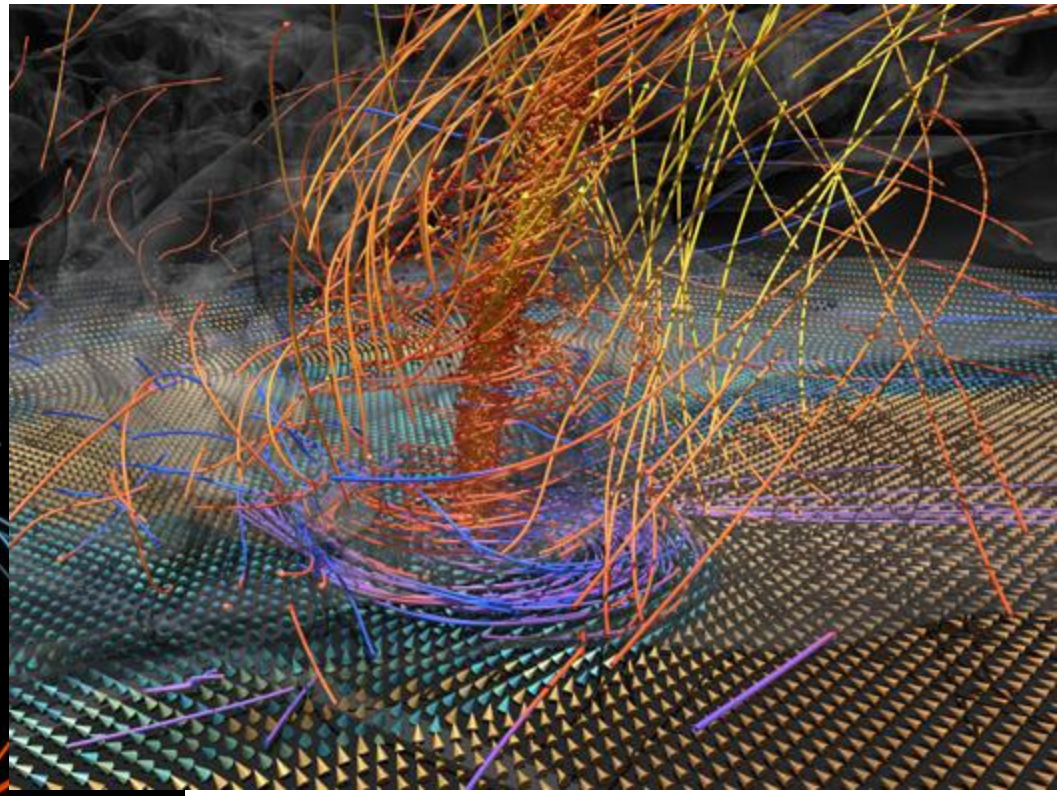




http://www.nsf.gov/news/overviews/computer/screensaver.jsp

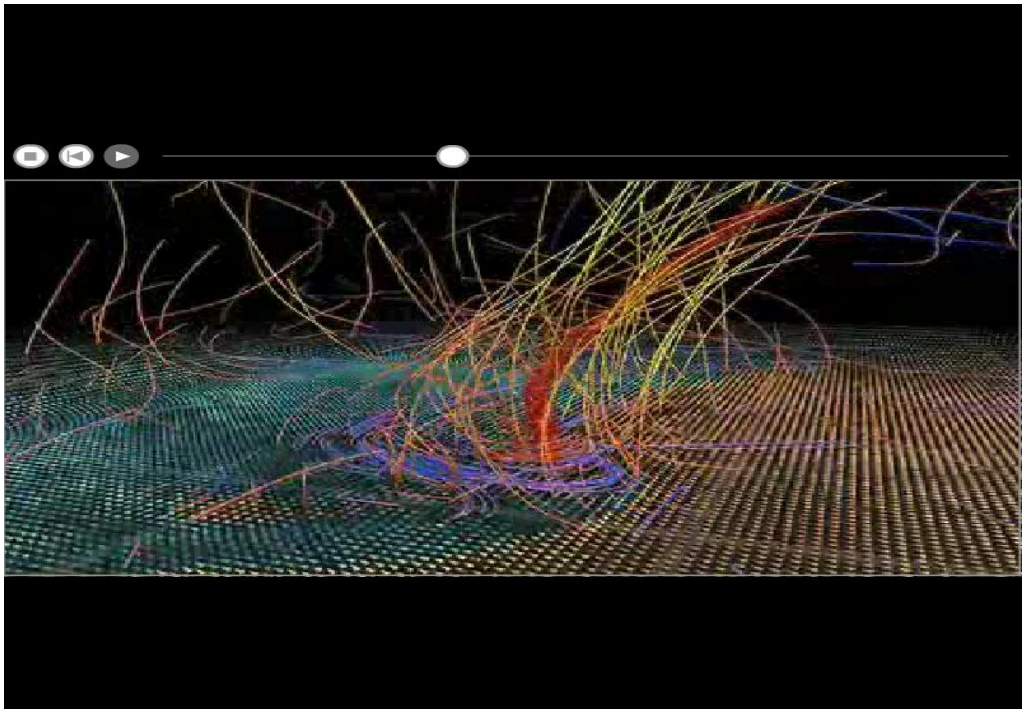
Earth Science





http://www.nsf.gov/news/overviews/computer/screensaver.jsp

In Nov. 2003, using 16 processors of a IBM P690 machine,
the simulation took 8 days to complete for a storm last for 2.5 hours.
Generated 650GB of data.



http://redrock.ncsa.uiuc.edu/CMG/gallery.html

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.

From: U.C.Berkeley CS267, Lecture 1 Notes

Applications of Distributed Computing

- P2P Network – eDonkey, BitTorrent, Skype, …
- Google
  - 1500^{+} (?) Linux machines behind Google search engine
- @home projects
  - SETI@home – Search for Extraterrestrial Intelligence
  - Folding@home – study protein folding

Computing Hardware

Why do we need parallel computing?

Computer Hardware

- Processor / CPU
  - Fetch instructions & their operands from memory
  - Decode the instruction
  - Dispatch instructions to appropriate functional units
    - Integer operations (e.g. 1+2*6)
    - Floating-point operations (e.g. 1.0 + 3.2* 4.5)
  - Execute the instruction
  - Store results
- Memory
  - Store instructions and data

Processor (1)

- Clock-speed
  - Working frequency → how many work cycles per second.
  - Each instruction usually takes one or more cycles to execute
- CISC vs. RISC
  - Complex Instruction Set Computer
  - Reduced Instruction Set Computer

Get food
Get fork
Eat

Go to kitchen
open fridge
get food
close fridge
open drawer
get fork
close drawer
open mouth
put food in mouth
close mouth
chew
swallow

Processor (2)

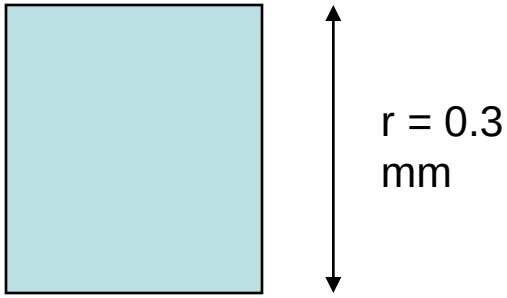
- FLOPS (Floating Operations per Second)
  - A measurement of computer performance
  - Modern PC processors: ~ 10GFLOPS
  - #1 (Tianhe-1A, NUDT), 186,368 cores, 2,566 TFLOPS, 229,376 GB RAM
- Performance-enhancing techniques for processors (see supplement)
  - Pipelining: overlaying different stages of execution: fetch→decode→dispatch→execute→write back
  - Out-of-order execution
  - Speculative execution / Branch prediction
  - Superscalar: multiple function units (multiplication, addition, …)
  - SIMD: MMX, SSE, 3DNOW, …
  - Hyperthreading

Processor (3)

- Limits of further processor development
  - Heat dissipation!
  - Physics
- Moore's Law
  - The number of transistors on a microprocessor would double every 18 months – Gordon Moore, Intel Inc.
- Raising performance problem: Memory latency!

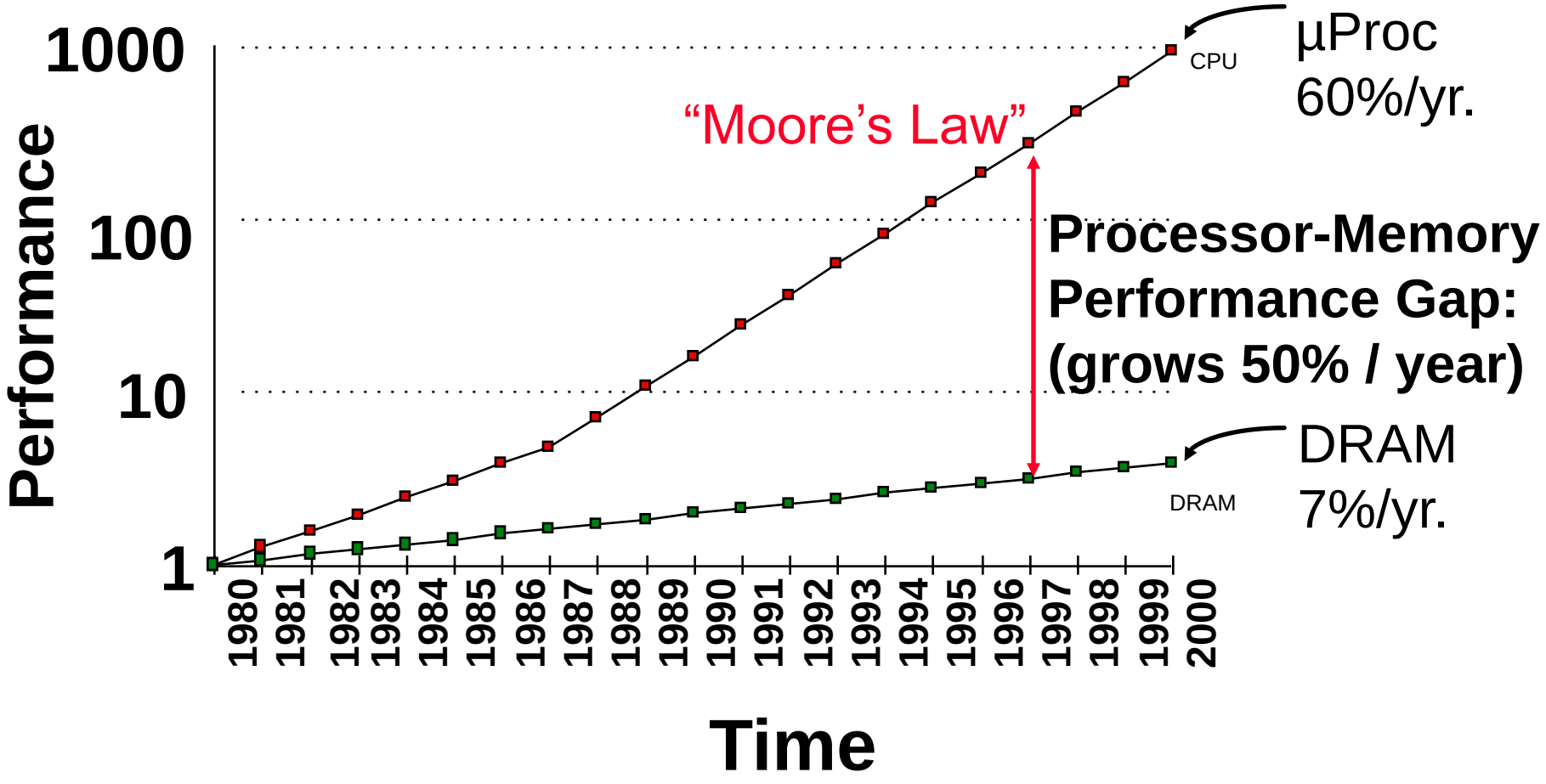
More Limits: How fast can a serial computer be?

1 Tflop/s, 1 Tbyte sequential machine



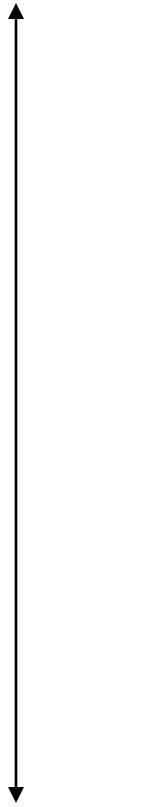
- Consider the 1 Tflop/s sequential machine:
  - Data must travel some distance, r, to get from memory to CPU.
  - To get 1 data element per cycle, this means 10^{12} times per second at the speed of light, $c = 3x10^8$ m/s. Thus $r < c/10^{12} = 0.3$ mm.
- Now put 1 Tbyte of storage in a 0.3 mm x 0.3 mm area:
  - Each word occupies about 3 square Angstroms, or the size of a small atom.
- No choice but parallelism

Processor-DRAM Gap (latency)

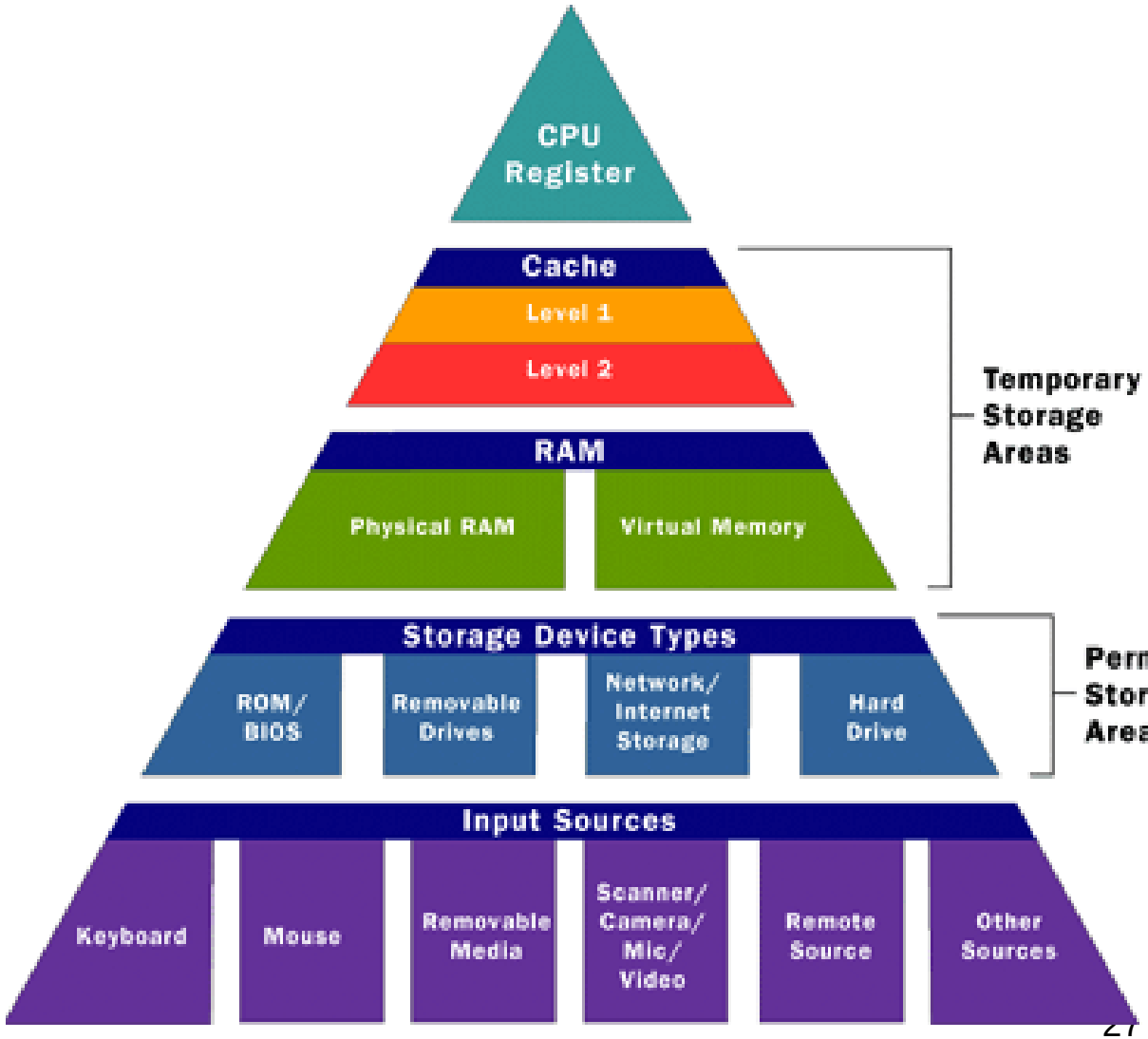


Memory Hierarchy

Fast, Expensive



Cheap, Slow



http://computer.howstuffworks.com/computer-memory7.htm

Effect of Memory Latency

- Effect of memory latency on performance
  - A single 1GHz processor with 2 multiply-add units that is capable of 4 floating-point calculations per cycle (1 ns) → 4GFLOPs
  - Assume memory latency of 100 ns for 1 word fetching/storing, the processor needs to wait 100 cycles before it can perform operations
  - Consider a program computing dot-product of two vectors (one addition and one multiplication). Two data fetches (200 ns) and two floating point operations (2ns) are needed for each element of the vector. The peak speed is therefore ~10MFLOPS. → 1/400 or 0.25% efficiency

Effect of Cache Memory

- Continue previous example, and introduce a 32KB cache memory between processor & memory
  - 1GHz processor, 4GFLOPS theoretical peak, 100ns memory latency
  - Assume 1ns cache latency (full-speed cache)
  - Multiply two matrices A & B of dimensions 32x32 (8KB or 1K words for each matrix)
  - Fetching two matrices into cache: 200000ns = 200 µs.
  - Multiplying 2 NxN matrices ~ $2N^3$ operations = 2^{16} operations, needs 16K cycles or 16 µs
  - Total computation time = 216 µs
  - FLOPS = $2^{16}/216$ ~ 303MFLOPS → 30 times improvement
  - But still < 10% CPU efficiency

Effect of Cache Line or Block Size

- Block size/cache line refers to the size of memory returned from a single memory request
- The same machine, with a block size of 4 words. Each memory request returns 4 words of data instead of one → 4 times more operations can be performed in 100ns: 40MFLOPS
- However, this technique only helps for continuous data layout (e.g. vectors)
- Programs need to be written such that the data access pattern is sequential to make use of cache line.
  - Fortran: column-wise storage for 2-D arrays
  - C: row-wise storage for 2-D arrays

Summary of Processor & Memory

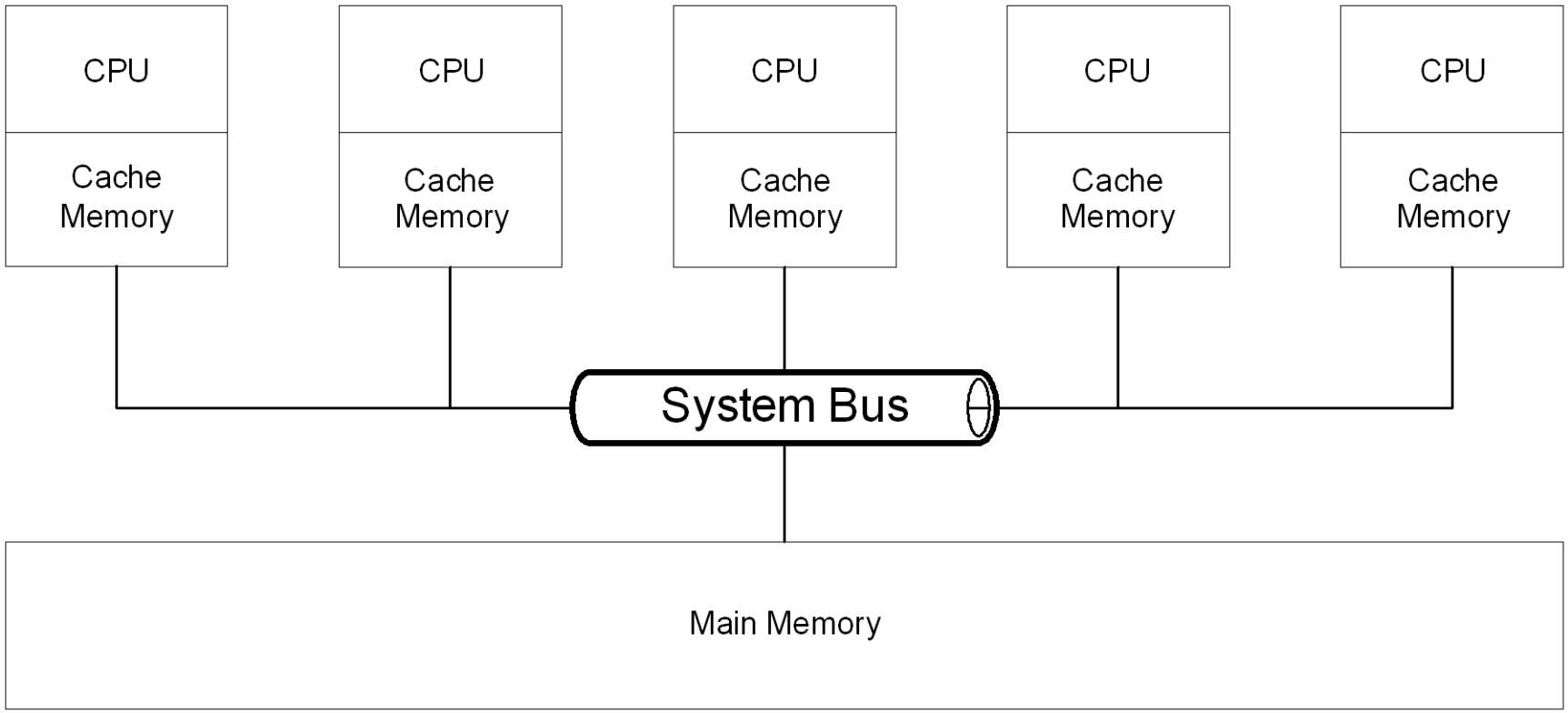
- Single processor capability is limited
  - Not many tricks (e.g. pipelining, superscalar) left for enhancing processor performances
  - Many tricks are related to parallelism already
    - Multi-core CPU design, superscalar, …
- Memory speed is much slower than CPU these days
  - Need to pay attention to data-access pattern (to be further discussed in the future lectures)

Parallel Computer Architecture

Shared-memory

Distributed-memory

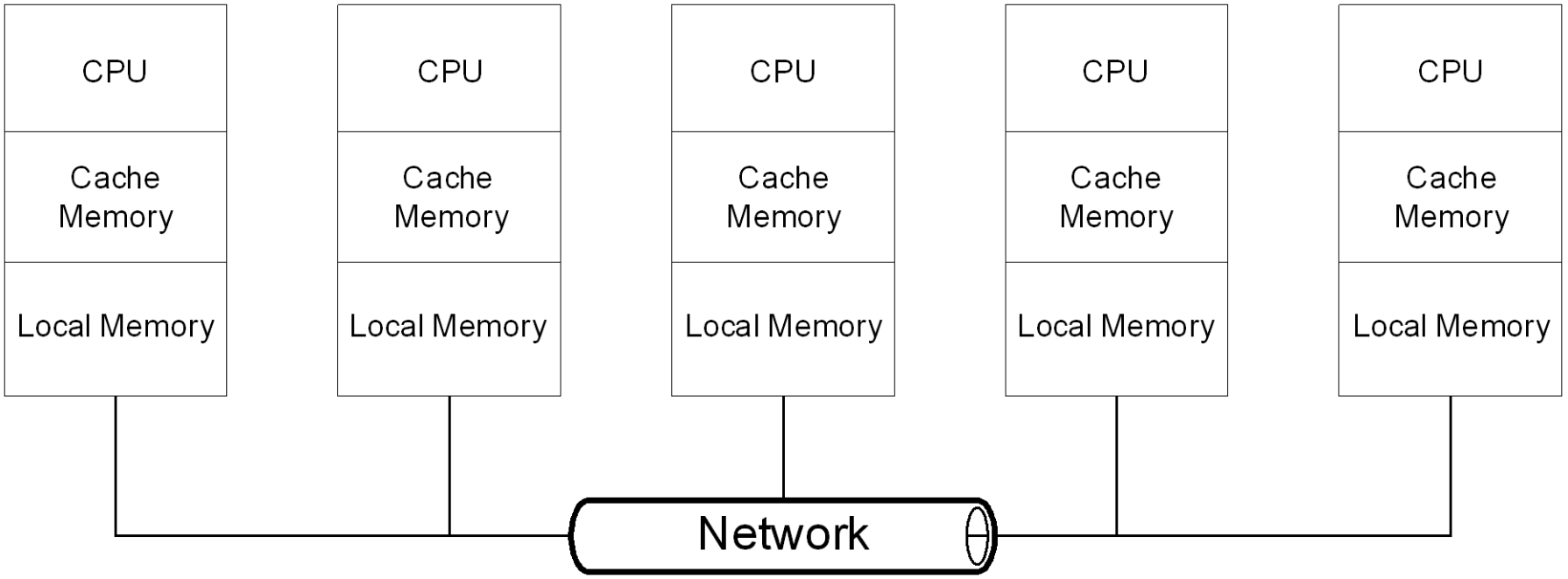
Shared-Memory Parallel Computer



Shared-Memory Parallel Computer

- Shared Memory
  - Programming through threading
  - Multiple processors share a pool of memory
  - Problems: *cache coherence*
  - UMA vs. NUMA architecture
- Pros:
  - Easier to program (probably)
- Cons:
  - Performance may surfer if the memory is located on distant machines
  - Limited scalability

Distributed-Memory Parallel Computer



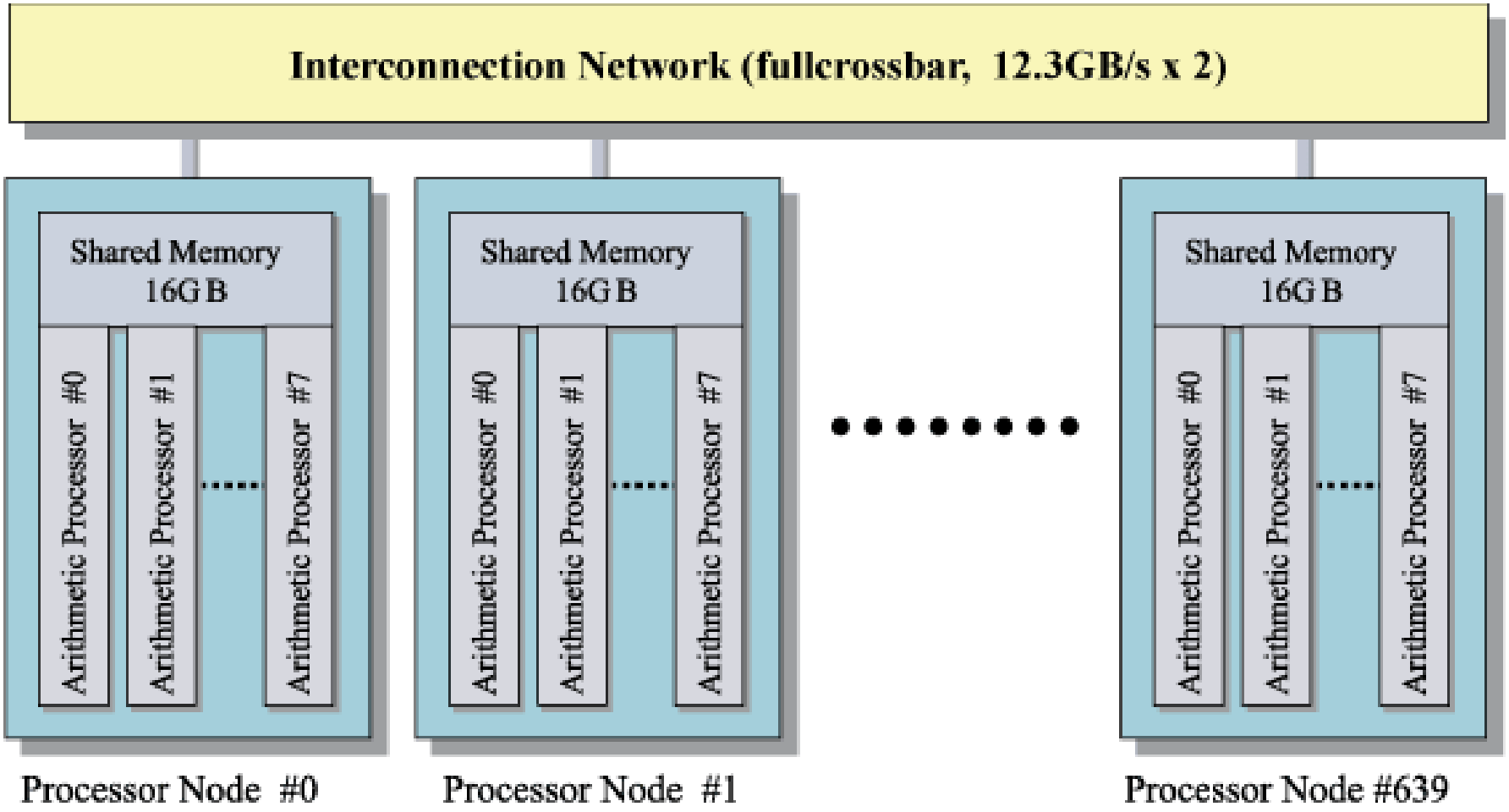
Distributed-Memory Parallel Computer

- Distributed Memory
  - Programming through processes
  - Explicit message passing
  - Networking
- Pros:
  - Tighter control on message passing
- Cons:
  - Harder to program

- Modern supercomputers are hybrids!

Earth Simulator

http://www.es.jamstec.go.jp/esc/eng/index.html
the fastest supercomputer in the world from 2002 to 2004



Beowulf Cluster

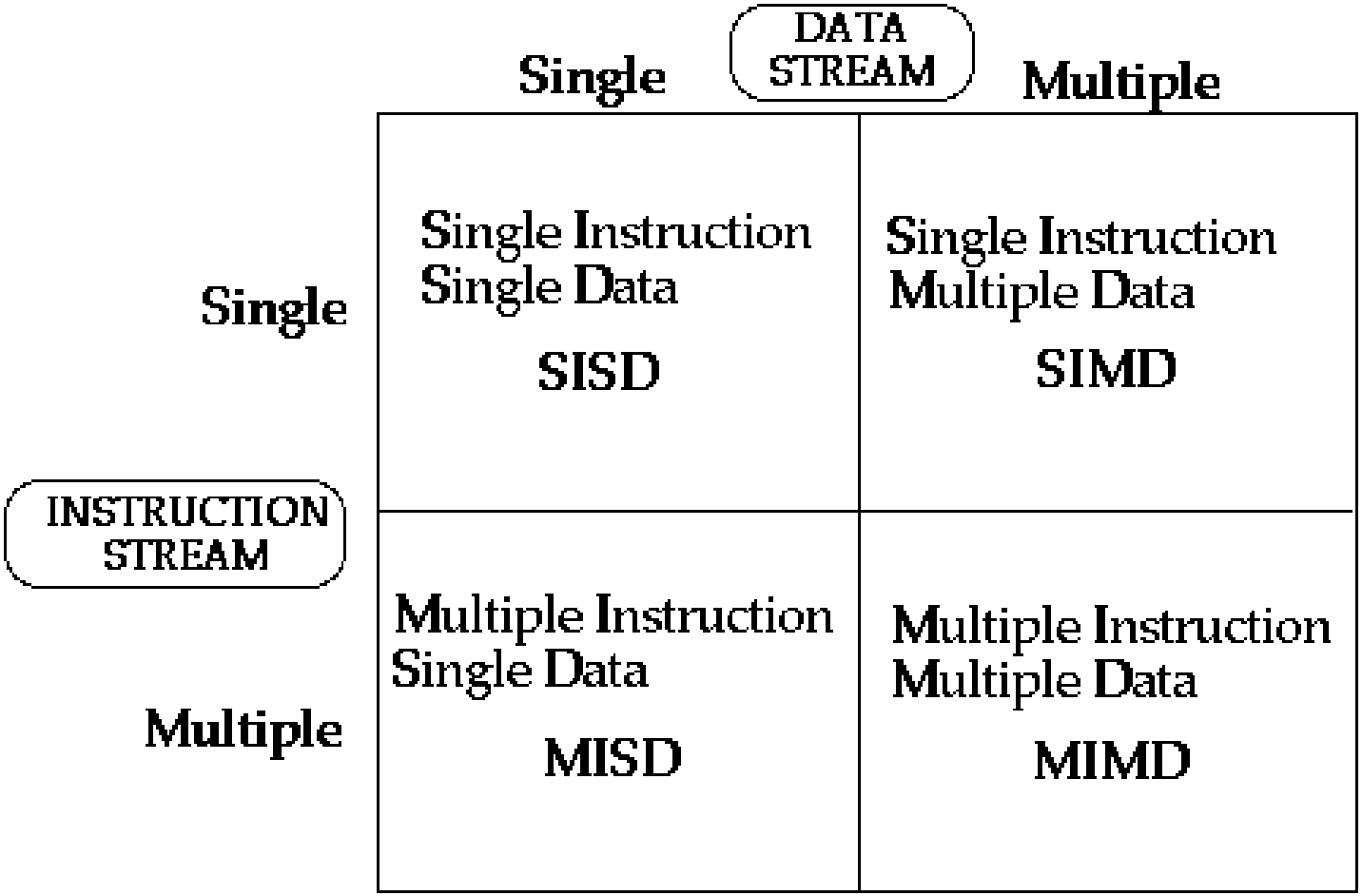
- Is a form of parallel computer
- Thomas Sterling, Donald Becker, … (1994)
- Emphasize the use of **COTS**
  - COTS: Components-Off-The-Shelf
  - Intel, AMD processors
  - Gigabit Ethernet (1000Mbps)
  - Linux
- A dedicated facility for parallel processing
  - Non-dedicated: NOW (Network Of Workstations)
- Performance/Price ratio is significantly higher than traditional supercomputers!

Models for Parallel Processing

Models for Parallel Processing

- Flynn (1966)
  - SISD (Single Instruction, Single Data)
    - Conventional uni-processor machines
    - Serial, Deterministic
  - SIMD (Single Instruction, Multiple Data)
    - SSE, data parallel
    - Synchronous, Deterministic
    - ILP: Instruction Level parallelism
  - MISD (Multiple Instruction, Single Data)
    - Rarely used
  - MIMD (Multiple Instruction, Multiple Data)
    - MPP (Massive Parallel Processing)
    - Synchronous or asynchronous
    - Deterministic or non-deterministic
    - Well-suited to block, loop, or subroutine level parallelism

Taxonomy of Computer Architectures (Flynn 1966)



Summary

- Introduction
  - Definition
  - Applications
- Computing Hardware
  - Processor
  - Memory
- Parallel computer architecture
  - Shared Memory
  - Distributed Memory
- Models of Parallel Processing

What you should know after today's class

- Definition: parallel computing, distributed computing
- Reasons for going parallel computing instead of waiting faster CPUs.
- Understand how cache-memory improves computer performance, and how memory speed affects performance.
- Architectures of parallel computers and advantages of each architecture.
- Why distributed memory / Beowulf clusters prevail today.

Preview

- Next week: Misc Topics in C/C++ Programming

- Review your C/C++ programming. Specifically,
  - Loop
  - Array
  - Memory management

Assignment #1

Supplement

CPU Performance Enhancing Techniques

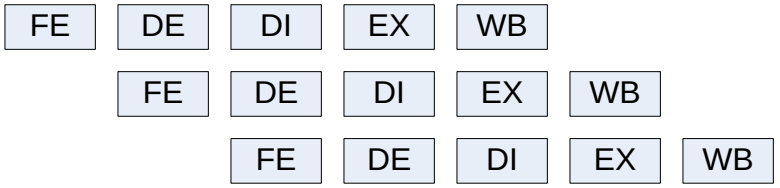
CPU Performance Enhancing Techniques

Pipelining

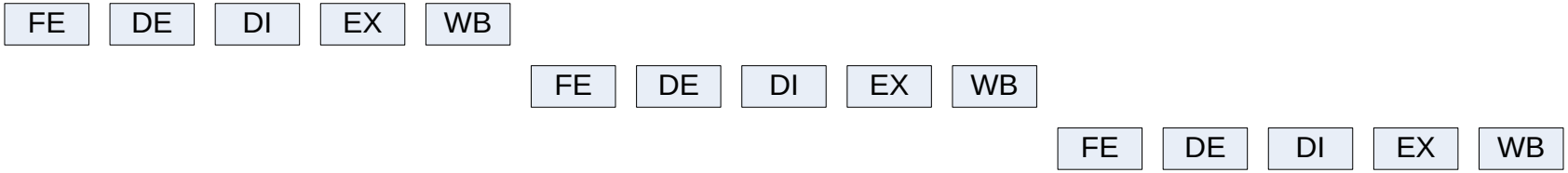
Execution of an instruction, 5 stages

Non-pipelined, 3 instructions, 15 stages, 15 cycles

Pipelined, 3 instructions, 15 stages, 7 cycles (we assume all stages are equally time consuming)



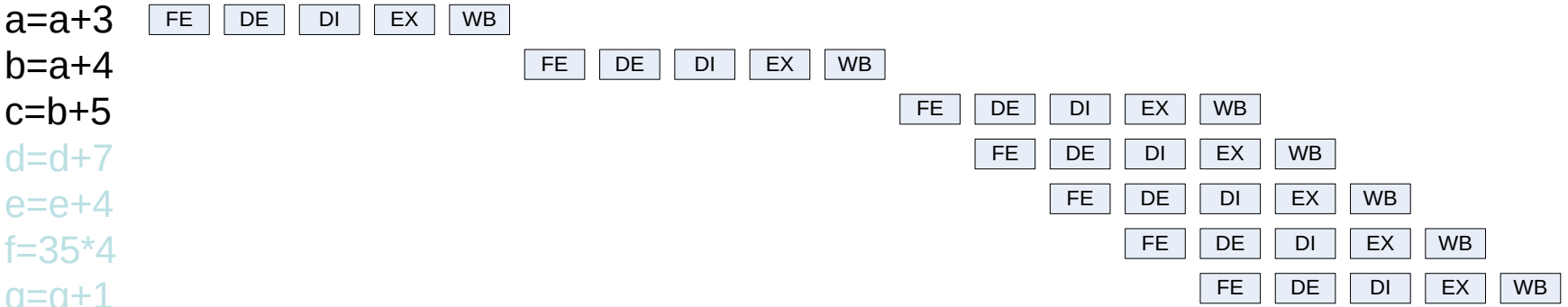
a=a+3
b=b+4
c=c+5



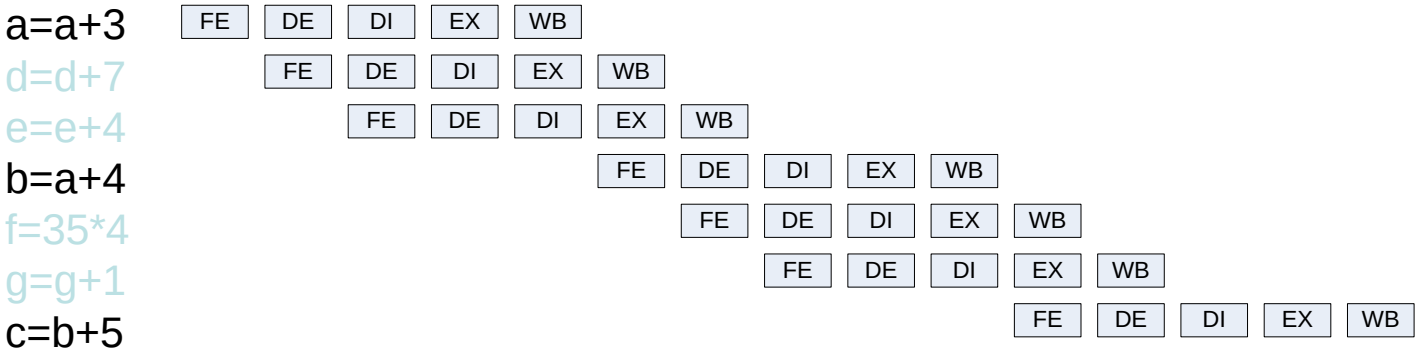
What if ?
a=a+3
b=a+4
c=b+5

CPU Performance Enhancing Techniques

Out-of-Order Execution



Reorder execution

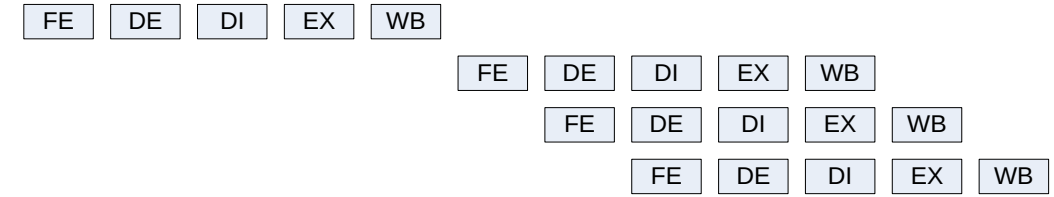


CPU Performance Enhancing Techniques

Speculative Execution / Branch Prediction

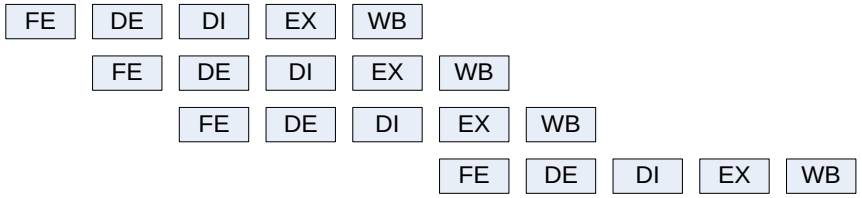
Without B.P.

```
If(a>5) {
        b=3
        c=c+10
}
Else {
        c=3
        b=b-10
}
f=f+1
```



With B.P. (correct)

```
If(a>5) {
        b=3
        c=c+10
}
Else {
        c=3
        b=b-10
}
f=f+1
```

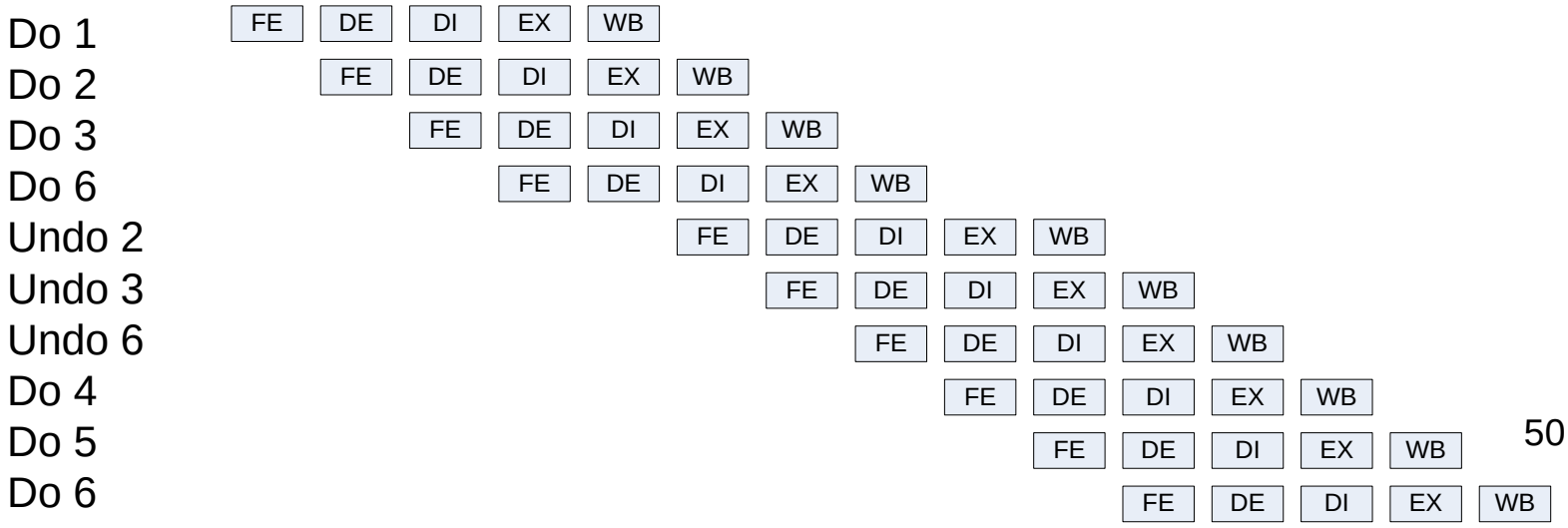


CPU Performance Enhancing Techniques

Speculative Execution / Branch Prediction

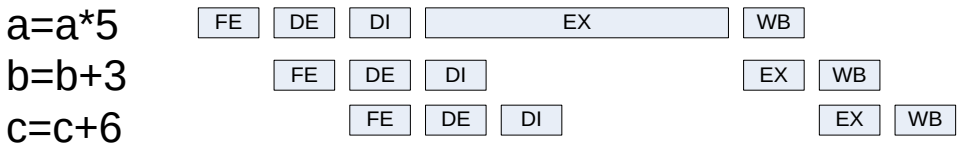
With B.P. (Incorrect prediction)

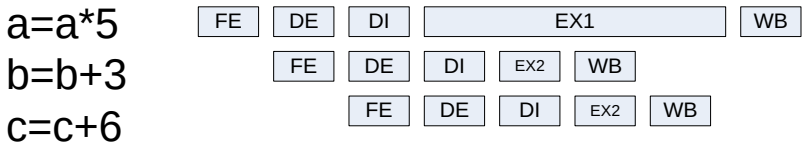
```
1   If(a>5) {
2           b=3
3           c=c+10
    }
    Else {
4           c=3
5           b=b-10
    }
6   f=f+1
```



CPU Performance Enhancing Techniques
Superscalar

- Use several function units (2 integer operations to execute operations in parallel)
  - Not all instructions take the same amount of time for execution (use different no. of cycles)





CPU Performance Enhancing Techniques
SIMD

- SIMD: Single Instruction Multiple Data
  - Applying the same operation on a range of data (array, vector, matrix…)
  - To efficiently use multiple functional units in CPU (superscalar CPUs)
  - MMX, 3DNow, SSE, SSE2, AltiVec, …
- Hyperthreading