Electrical Engineering and Computer Science

EECS 358 - INTRODUCTION TO PARALLEL COMPUTING

Lecture 1

Parallel Processing

Outline

- What is parallel processing
- Why parallel processing
- Parallel processing options
- Aspects of parallel processing
  - Architectures
  - Software
  - Algorithms
- Summary
- READING : Kumar - ch 1,2
- Web: www.intel.com, www.ibm.com

Course Administration

- Electrical Engineering and Computer Science 358: Introduction to Parallel Computing
- Class meets Tuesdays and Thursdays 2:00-3:20PM
- Instructor:
  - Gokhan Memik
  - Office: L475 Tech, Phone: (847) 467-1168,
  - Email: memik@eecs.northwestern.edu,
  - Web: http://www.eecs.northwestern.edu/~memik
  - Office Hours: Thursdays, 3:30-4:30pm (or by appointment)
- Teaching Assistant:
  - Majed Beigi
  - Room: Tech L458
  - Email: majed.beigi@northwestern.edu

  - Office Hours: TBD

- Grader:
  - Wenting Zhou
  - Room: Tech L458
  - Email: wentingzhou2017@u.northwestern.edu

- Class Material:
  - Everthing on Canvas one hour before the class

Outline of course

- Introduction to Parallel Programming (2 lectures)
- Shared Memory Parallel Programming (4 lectures)
- Distributed memory message-passing parallel programming (5 lectures)
- Data parallel and SIMD parallel programming (3 lectures)
- Parallel programming tools (1 lectures)
- Parallel algorithms and applications (3 lectures)

Books and Reading Material

Textbook:

- A. Grama, A. Gupta, G. Karypis, and Vipin Kumar, *Introduction to Parallel Computing*, Addison Wesley, 2nd edition, 2003

Other Material:

- S. Brawer, "Introduction to Parallel Programming", Academic Press, 2014.
- I. Foster, "Designing and Building Parallel Programs," Addison Wesley, 1994. Available online at: http://www.mcs.anl.gov/dbpp.
- B. Bauer, "Practical Parallel Programming," Academic Press, 1992.
- W. Gropp et al, "Using MPI: Portable Parallel Programming with the Message Passing Interface," MIT Press, 1994. Available online at:

  http://www.mcs.anl.gov/mpi/index.html.
- I will be using Canvas for course material

Grades for Class

- Three or four homeworks and programming assignments - 50 % total
- Two exams (25 % each) - 50 % total

  Midterm (Late April/Early May)

  Endterm (Last week of classes)
- There is a late penalty of 10% per calendar day and no homeworks/assignments will be accepted once the solutions are posted to the class web site.

What is Parallel Processing ?

- Parallel processing is the process by which a problem is solved used multiple resources working concurrently and collaboratively.
- Daily life examples:
  - House construction
  - Car manufacturing
  - Grocery store operation
- Effective parallel processing requires:
  - Appropriate division of labor
  - Proper coordination

Why Parallel Processing ?

- Parallel Processing is an effective answer for the tremendous future computing requirements:
  - Very complex applications
  - Real-time requirements of applications
- Existing algorithms for many applications running on uniprocessors inadequate for the future:
  - Need more variables for *better quality* results
  - Need to solve *larger problem sizes*
  - For many applications *fast-response is required*
- Large set of computational problems are inherently parallel in nature

Why Parallel Processing ?

- Computational Science Grand Challenges:
  - Science today: Experimentation, Theory, Simulation
  - Simulation relies heavily on parallel processing
- Aerodynamics: design of new aircraft
- Biology: modeling of genetic compounds
- Computer Science: VLSI CAD
- Physics: simulation of fusion reactions
- Civil Engineering: structural analysis
- Electrical Engineering: image and signal processing

Why Parallel Processing ?

- Rapidly growing parallel processing technology has recently become commercially available
  - No "uniprocessor" left
  - Most improvements in the future will come from parallel programming, in other words, everyone relies on parallel computing
- It is relatively easy to build hardware for systems with hundreds or thousands of processors which can theoretically achieve Teraflops/Petaflops
- Parallel processing has become more economical in recent years
  - 1 CRAY T90 processor - 2 GFlops ($2,500,000)
  - 8 Node IBM SP2 using PowerPC processors - 2 GFlops ($200,000)
  - 8 Node SGI Origin 2000 using R10000 processors - 10 GFlops ($100,000)
  - Single chip IBM Cell with 8 cores - 20 to 200 GFlops ($1,000)
  - Nvidia GeForce Titan - 1 to 10 TFlops ($1,000)

Parallel Processing Options

- Chip multiprocessors (IBM Cell, Sun Niagara, Intel N-Core)
- Network of workstations (lowest cost)
- Multiprocessor workstations ($15,000)
  - SUN Ultra 10, HP J5000, SGI
- Shared memory multiprocessors servers ($10,000-$200,000)
  - SGI Origin 2000, SUN Enterprise 6500, HP K580
- Distributed memory multicomputers ($200,000-$800,000)
  - IBM SP2
- Massively Parallel Processors ($10,000,000)
  - SGI ASCI Blue, Tera Corporation MTA, IBM Blue Gene, NEC Earth Simulator, K Computer, IBM Sequoia, Cray Titan, Tianhe

Aspects of Parallel Processing

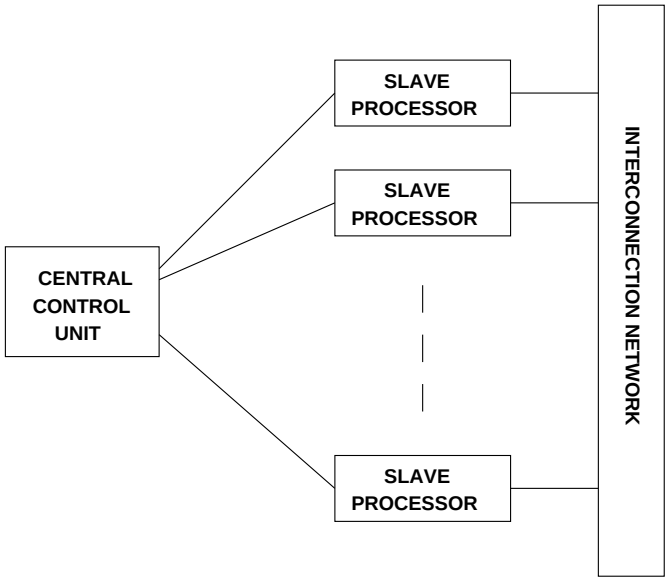
- Architectures:
  - Processors and memories connected together
- Software:
  - Operating systems
  - Compilers
  - Libraries
  - Tools - debuggers, performance analyzers
- Algorithms:
  - Designing software that best fits underlying architecture

Architecture

- Basic components of any architecture:
  - Processors and Memory (Processing Units)
  - Interconnect
- Logical classification based on:
  - Control mechanism - SIMD and MIMD
    - SIMD (Single Instruction Multiple Data stream)
    - MIMD (Multiple Instruction Multiple Data stream)
  - Address space organization - Shared and Distributed

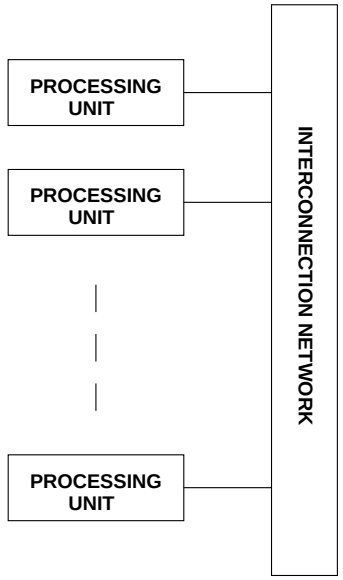
SIMD Architectures

- Centralized control processor defines the sequence of instructions executed by slaves
- Slave processors execute same instruction streams on distinct data sets



MIMD Architectures

- Each processor executes program independent of other processors

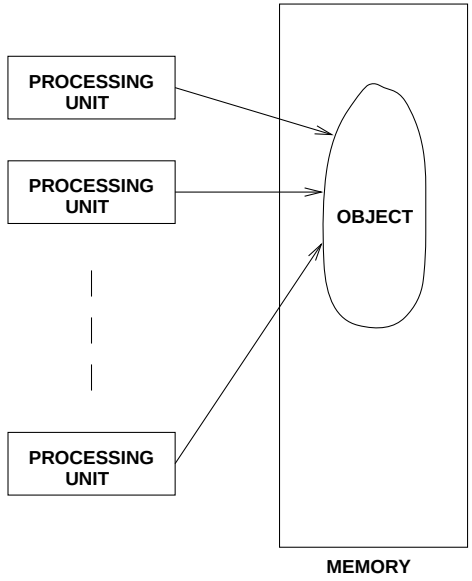


SIMD vs. MIMD

- SIMD:
  - Need custom hardware for processors
  - Slave processors are simple and therefore low cost
  - Good for programs with lot of synchronization due to lock step operation
- MIMD:
  - Easy to build out of commodity parts
  - Processors are complex, but availability of low cost commodity microprocessors offsets the SIMD advantage
  - Can handle a more general class of problems with reasonable efficiency

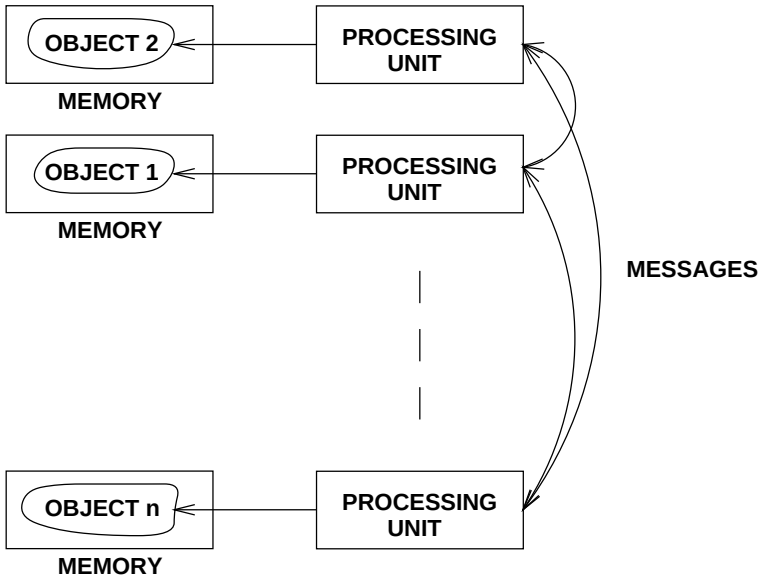
Shared Address Space

- Shared address space:
  - Processors can directly access all the data in the system
  - Interaction via modification of objects in shared space



Distributed Address Space

- Distributed Address space:
  - Processors can directly access only local data
  - Interaction via explicit message passing

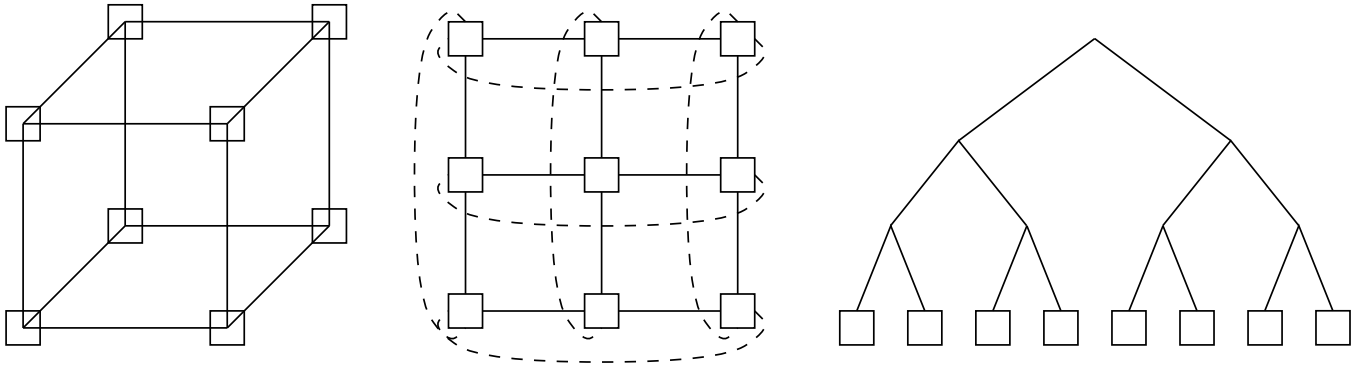


Shared Address vs. Distributed Address

- Shared Address:
  - Transparent access to all program data makes them easier to program
  - On large shared address machines, performance issues similar to distributed address space machines
  - Debugging difficult due to "race" conditions
- Distributed Address:
  - Explicit fetching of non-local data makes them difficult to program
  - Model most suitable for constructing massively parallel machines

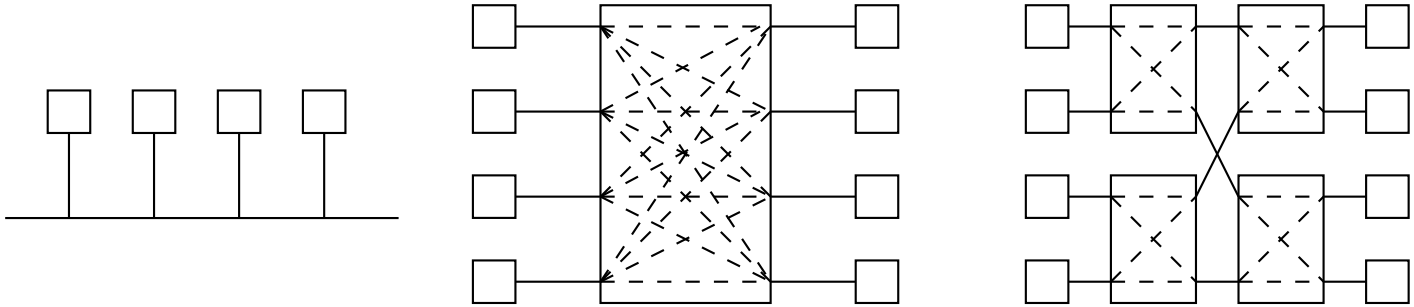
Static Interconnects

- Consist of point-to-point links between processors
- Can make parallel system expansion easy
- Some processors may be "closer" than others
- Examples : Hypercube, Mesh/Torus, Fat tree



Dynamic Interconnects

- Paths are established as needed between processors
- System expansion difficult
- Processors are usually equidistant
- Examples : Bus based, Crossbar, Multistage networks



Examples

- SGI Power Challenge
  - MIMD, Shared address space
  - Dynamic interconnect (bus)
- IBM SP-2
  - MIMD, Distributed address space
  - Dynamic interconnect (multistage)
- IBM Sequoia
  - MIMD, Distributed address space
  - Custom interconnect (Mesh/Torus)

Operating Systems

- Need to support tasks similar to serial OS like UNIX
  - Memory and process management, File systems, Security
- Additional support needed
  - Job scheduling - Time sharing, Space sharing
  - Parallel programming support - message passing, synchronization

Shared Address Space MIMD

- Typically time shared - many user programs co-exist in the system and compete for time slices
- Access to job queue can be centralized or decentralized
- Parallel programming support via primitives for creation, destruction and synchronization of processes

Distributed Address Space MIMD

- Usually access to parallel machine is via a host computer running a serial OS
- Nodes of parallel machine have a simpler version of OS
- Typically space shared - user programs have processor sub-sets allocated for exclusive use
- Parallel programming support via primitives for message passing and network management

Compilers

- Automatic parallelization:
  - Compiler detects and exploits concurrency optimally
  - Requires complex analysis and transformations
- Implicit parallel programming:
  - Programmer directives to help compiler analysis
  - Compiler performs all the "grunt" work
- Explicit parallel programming : easiest to support

Libraries

- Make using parallel machines easier
- Library implementations are usually done by skilled and experienced programmers working closely with machine designers resulting in high levels of performance.
- Library routines can be used as building blocks for complex applications
- Usually cover certain specialized application domains
- Examples : SCALAPACK, MPI, PVM

Tools

- Essential due to high degree of complexity in parallel machines
- Performance analyzers:
  - Help in identifying bottlenecks
  - Can identify relative importance of different parts of program with respect to possible performance gains
- Debugging:
  - Need to capture the state of multiple processes
  - Bugs commonly caused by synchronization errors are difficult to capture
- Examples : PARAGRAPH/PICL, Prism, Pablo, Upshot

Algorithms

- Divide a given problem into sub-problems
- Sub-problems must have a high degree of concurrency
- Must minimize synchronization among sub-problems
- Performance is a key aspect of algorithm design

Summary

- What is parallel processing
- Why parallel processing
- Parallel processing options
- Aspects of parallel processing
  - Architectures
  - Software
  - Algorithms
- NEXT CLASS : Introduction to Parallel Programming
- READING : Kumar - ch 1,2