Harvey Mudd College
CS 156

# Parallel and Real-Time Computation

Spring 2000

Prof. Bob Keller
keller@cs.hmc.edu
x 18483

## Materials

- Slides will be posted to course web page:

  http://www.cs.hmc.edu/courses/cs156

- They will be available in two forms:
  - for web browser
  - downloadable as Microsoft Powerpoint presentation

## Course Emphasis Plan

- Parallel computation: about 80%
- Real-time computation: about 20%

## Textbook for the parallel part

**Barry Wilkinson and Michael Allen,**
**Parallel Programming: Techniques and Applications Using Networked Workstations and Parallel Computers,**
**Prentice-Hall, 1999,**
**ISBN 0-13-671710-1.**


**I will label this "PP".**

## Other sources for the parallel part

- Various papers, books, web pages

## Other sources for the real-time part

- Various books and papers

## Outline for Parallel Part
(consider this approximate)

(1 day) Basic ideas and vocabulary of parallel computing (PP chapter 1)
(2 days) Message-passing computation, MPI (PP chapter 2)
(1 day) Embarassingly-parallel problems (PP chapter 3)
(2 days) Partitioning (PP chapter 4)
(1 day) Pipelining (PP chapter 5)
(1 day) Synchronous computations (PP chapter 6)
(1 day) Load balancing and termination detection (PP chapter 7)
(2 days) Programming with shared memory (PP chapter 8)
(2 days) Sorting algorithms (PP chapter 9)
(2 days) Numerical algorithms (PP chapter 10)
(1 day) Image processing (PP chapter 11)
(2 days) Searching and optimization, genetic algorithms (PP chapter 12)
(1 day) ZPL language, Fortran-based languages
(1 day) Functional language approach; Linda, Javaspaces
(1 day) PRAM model (PP appendix D)
(1 day) NESL model and language

## Outline for Real-time Part

(1 day) Scheduling algorithms
(1 day) Rate-Monotonic scheduling
(1 day) Scheduling algorithms
(1 day) Rate-Monotonic scheduling
(1 day) Real-time languages
(1 day) Real-time operating systems
(1 day) Temporal logic and specifications day) Real-time languages
(1 day) Temporal logic and specifications

## Definitions

- **Parallel computation**: Computation in which there are multiple computing activities going on simultaneously.
- **Real-time computation**: Computation in which it is essential that *time* be taken into account in defining correctness.

## Integrating parallel and real-time

- Parallel computation can be used in the context of real-time computation.

## Purposes of Parallel Computation

- Speedup
- Fault-tolerance
- Geographical Conformance
- Natural Structuring of a Computation

## Why Parallel Computing will become essential in our lifetimes

- The processor performance-improvement curve of doubling every two years cannot continue forever.
- There are physical limitations on switching and propagation speeds.
- Saturated device capabilities leaves two alternatives:
  - Better sequential algorithms
  - Parallel algorithms

## Why Parallel Computing will become essential in our lifetimes

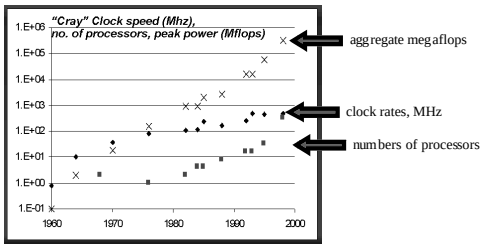
- The processor performance-improvement curve of doubling every two years cannot continue forever.
- There are physical limitations on switching and propagation speeds.
- Saturated device capabilities leaves two alternatives:
  - Better sequential algorithms
  - Parallel algorithms


## Why Parallel Computing will become essential in our lifetimes

- The processor performance-improvement curve of doubling every two years cannot continue forever.
- There are physical limitations on switching and propagation speeds.
- Saturated device capabilities leaves two alternatives:
  - Better sequential algorithms
  - Parallel algorithms


## Cray Supercomputing What do these curves tell you?

From Gordon Bell: http://research.microsoft.com/users/gbell/craytalk/sld013.htm




## Parallel Processor: Cray T3e

http://www.cray.com/products/systems/crayt3e/1200/

- 32-2048 processing elements (PEs)
- 8 Million words (64 MB) per PE
- PE is DEC Alpha, 600 Mflops
- 3-D torus interconnect
- interprocessor rate: 480 MB/s
- bisection bandwidth > 122 GB/s

## Parallel Processor: IBM SP3

http://www.rs6000.ibm.com/hardware/largescale/index.html

- 1-8 processors, shared memory
- up to 16 GB memory
- networkable to 512 processors
- Processor is IBM RS/6000
- 2-level coherent cache
- 14.2 GB/s crosspoint switch

## SGI Origin 2800

http://www.sgi.com/origin/2000/

- 64-512 processors
- up to 1 TB shared memory
- R12000 processors, 300 MHz
- NUMA architecture
- 38.4 Gflops for 64 processors

## *Today's* fastest computer: ASCI Red

http://www.sandia.gov/ASCI/Red/RedFacts.htm

- Intel
- 4640 nodes
- 666 MOPs peak per node
- 2 TOPS peak Linpack performance
- 606 GB memory

## For the latest score ...

TOP500 SUPERCOMPUTER SITES

http://www.top500.org/

Currently dominated by: Cray/SGI, IBM, Hitachi, Fujitsu, NEC, HP, Sun,
although Intel holds the #1 slot.
The first Beowulf listed is #144, 140 processors.

## Parallel Computing has something for everyone

- Application developers
- Language developers
- Algorithm developers
- Computer architects
- System developers
- Entrepreneurs
- Philosophers
- Engineers
- Physicists

For any topic of investigation X, one can investigate "parallel X".

## Human-Parallel Examples

- Supermarket checkout lanes
- Multi-screen movie theatres
- Multi-lane highways
- Assembly lines
- Excavations
- Orchestra

## Thought Experiments

- Each person in this room is a processor.
- Decide how to make efficient use of parallelism to do the following tasks.

## Thought Experiment 1

- One person is handed a white-page phone book and given a number.
- The problem is to find the individual who has that number, or indicate that none exists.

## Thought Experiment 2

- One person is handed a white-page phone book.
- The problem is to produce an ordering of all of the telephone numbers in increasing numeric order.

## Thought Experiment 3

- Reconsider how to do the problem in Experiment 1, given that Experiment 2 has been done.

## Thought Experiment 4

- Everyone is given a phone-book size list of coefficients for a set of simultaneous linear equations.
- Find a reasonably accurate solution to the equations, assuming one exists.

## Requirements for Parallelism

- **Applications** that are amenable to parallel processing
- **Architecture** capable of parallel execution
- **Language** in which parallel execution can be expressed
- **Analysis** of the applications to produce parallel programs that run on parallel architectures

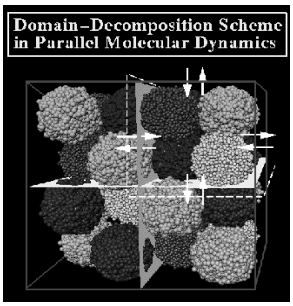
## Parallel Application Areas

- Most applications with high volumes of data
  - Searching (AI, Databases)
  - Sorting
  - Computations involving large matrices
  - Graph computations
  - Mathematical programming (linear programming, etc.)
- Speculative computation (multiple “what-if”s)

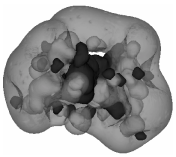
## Specific Applications

- **Physical simulations (e.g. numerical weather prediction)**
- **Biological and chemical simulations**
- **Logistic simulations, planning, and decision support tools (traffic, markets, robotics, battle, etc.)**
- **VLSI wire routing and other CAD problems**
- **Image processing, computer vision**
- **Graphic rendering, visualization**
- **Financial forecasting**
- **Game playing (e.g. Deep Blue)**
- **Evolutionary programming**
- **Neural networks**
- **Theorem proving, automated reasoning**
- **Natural language understanding**
- **Combinatorial optimization**

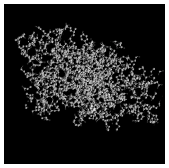
Partitioning of a molecular subsystem for parallel execution
(http://www.cclms.lsu.edu/cclms/research/research-parallel.html)




Solution of the Poission-Boltzmann equation
(from http://compbio.caltech.edu/applications/pmg/sod.html)



Isosurfaces of electrostatic potentials in the SOD (SuperOxide Dismutase) enzyme obtained by solving the nonlinear Poisson-Boltzmann equation numerically. The SOD enzyme is an antiradical or antioxident, meaning that it moves around the body binding to and then deactivating free radicals in the body, preventing them from causing cancer or other cell damage in the body.



SOD enzyme molecular structure

## Application Links

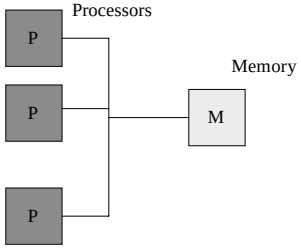
For a list of 17 parallel applications, ca 1993, see

http://www.npac.syr.edu/pub/by_module/presentations/fox/1993/93QQ-ProgramParadigm/normal/0260.html

## Parallel Architecture Varieties

- Shared-Memory
- Distributed Memory
- Processing-in-Memory
- Distributed Shared-Memory

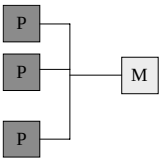
## Shared-Memory




## Shared-Memory

The diagram *suggests* a bus-connection and a single memory module.

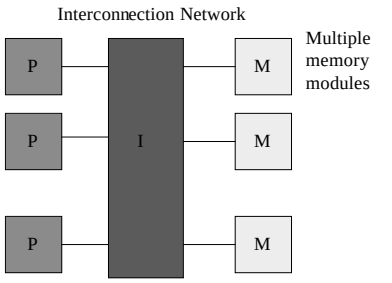
This may not be an ideal configuration.



## Parallel Communication

- Parallel processing capability alone is not sufficient.
- Parallel communication (e.g. processor-to-memory) is also necessary, to prevent the communication path from becoming a bottleneck.
- Typically memory speeds have lagged processor speeds. Multiple processors sharing a single communication channel may simply aggravate the problem.

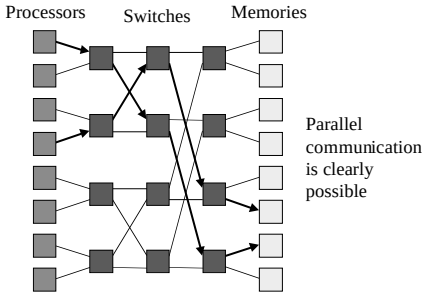
## Shared-Memory (2)




## Examples of Shared-Memory

- turing: 4-processors (SPARC) on a shared bus.
- muddcs: 6-processors (older SPARC) on a shared bus.

## Example of an Interconnection Network: Butterfly Switch



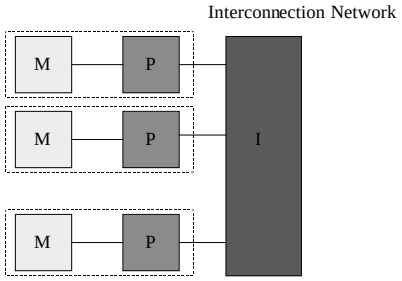

## Advantages of Shared Memory

- Uniformity of access to memory
- Programmer doesn't have to be very concerned about communication structure (since addressing mechanism takes care of routing)

## Disadvantages of Shared Memory

- Lacks scalability
  - As number of processor grows, latency through switch increases, creating a slow-down in memory access
- Requires exclusion protocol to insure atomicity of certain operations
- Cache coherence mechanism necessary

## Distributed-Memory




## Examples of Distributed Memory

- Beowulf cluster (hrothgar at HMC, 16 Pentium II's)
- henry (16 Intel 80286's)
- NOW (Network Of Workstations)

## John Koza's 1000 node Beowulf (used for genetic programming)

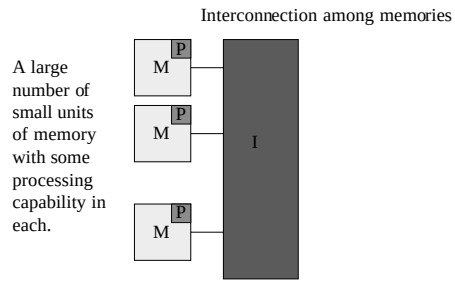
## Advantages of Distributed Memory

- Local memory access is fast
- Exclusive access to memory eliminates need for mutual exclusion protocol
- May be able to use off-shelf processor-memory (multi-computer)

## Disadvantages of Distributed Memory

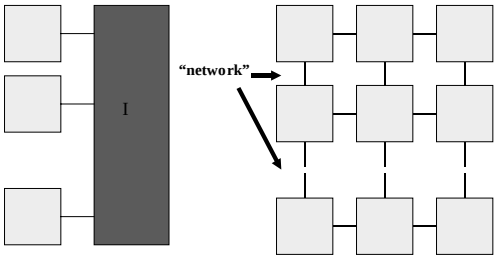
- Remote memory access is slow
- May need to plan communication structure carefully to attain speedup

## Processing-in-Memory (Example: Cellular Automaton)



## Note on Interconnection "Networks"

- A network can be monolithic or partitionable.




## Distributed Shared-Memory

- Attempts to make distributed memory behave as if shared memory
- Use sophisticated caching techniques
- Can extend to distributed *virtual* memory (e.g. with paging)

## Flynn's Classification for Architectures

instruction stream, data stream basis

- **SISD** (single-instruction stream, single data stream) = ordinary sequential processor
- **SIMD** (single-instruction stream, multiple data stream) = array of processors controlled by a master, usually in lock-step fashion
- **MIMD** (multiple-instruction stream, multiple data stream) = collection of processors, each with its own instruction stream (no master)

## Usual Connotations

- SIMD is usually associated with synchronous (or lock-step) computation
  - The PRAM family of theoretical models is SIMD
  - SIMD are sometimes linked to "Data Parallel" architectures
  - Cellular automata are a form of SIMD architecture, with no master control
  - Systolic arrays are a similar form
- MIMD is usually associated with asynchronous computation

## Add-ons to Flynn

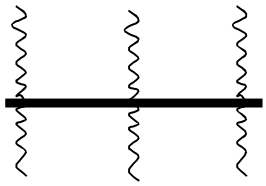
- SPMD (single-program, multiple data stream) Each processor runs its own copy of the *same* program (synchronized only by barriers and the like).

  This is similar to SIMD, except that between barriers, the processes proceed asynchronously.

## Barriers

One way to prevent data from being used prematurely

threads/processes on different processors



**Barrier**:
All threads wait for each other here.

## Example of Barrier Use

- Many numeric computations involve repeated steps of the assignment

```
M = f(M);
```

where M is a matrix. Typically all elements of f(M) can be computed in parallel. However, we don't want to "install" the new elements into M until all elements are computed. We actually compute

```
T = f(M);
barrier
M = T;
```

where T is a temporary matrix holding the new values of M.

## An Advantage of SIMD

- Barriers and similar synchronization mechanisms entail overhead, e.g. "handshaking" between processes.
- SIMD computers do not have this overhead; they rely on the common clock to keep processes synchronized.
- In effect, each step is followed by an implied barrier.

## A Disadvantage of SIMD

- Not all applications are commensurate with the SIMD style.
- Examples are irregularly-structured computations, where different parallel threads require different numbers of steps.

## SIMD lessons

- Thinking Machines Inc. Connection Machine series:
  - CM-1 and CM-2 were SIMD
  - (There was no CM-3 or 4.)
  - CM-5 was MIMD
  - (TMI is no longer in the computer business.)