*

Parallel Computers

*Material based on B. Wilkinson et al., "PARALLEL PROGRAMMING. Techniques and Applications Using Networked Workstations and Parallel Computers"

Why Parallel Computing?

- Many areas require great computational speed: ie. numerical modelling, and simulation of scientific and engineering problems .

- Require repetitive computations on large amounts of data.

- Must complete in a "reasonable" time.

  - For manufacturing, engineering calculations and simulation must only take seconds or minutes.

  - A simulation that takes two weeks is too long. A designer requires a quick answer, so they can try different ideas and fix errors.

  - Some problems have a specific deadline. ie. weather forecasting

- Grand challenge problems, like global weather forcasting and modelling large DNA structures, are problems that can not be handled in a "reasonable" time by today's computers.

- Such problems are always pushing the envelope.

N-body Problem

- Predicting the motion of astronomical bodies in space requires a large number of calculations.

- Each body attracted to each other body by gravitational forces.

- These forces can be calculated and the movement of each body predicted. Requires calculating total force acting on each body.

- For N bodies, there will be $N-1$ forces to calculate. Approx N^2 calculations.

- A galaxy might have 10^{11} stars. That's 10^{22} calculations!

- Assuming each calculation took 10^{-6} seconds, even an efficient $N \log_2 N$ approximate algorithm would take almost a year!

- Split the computation across 1000 processors, and that time could reduce to about 9 hours.

- A lot easier to get 1000 processors than build one processor 1000 times as fast.

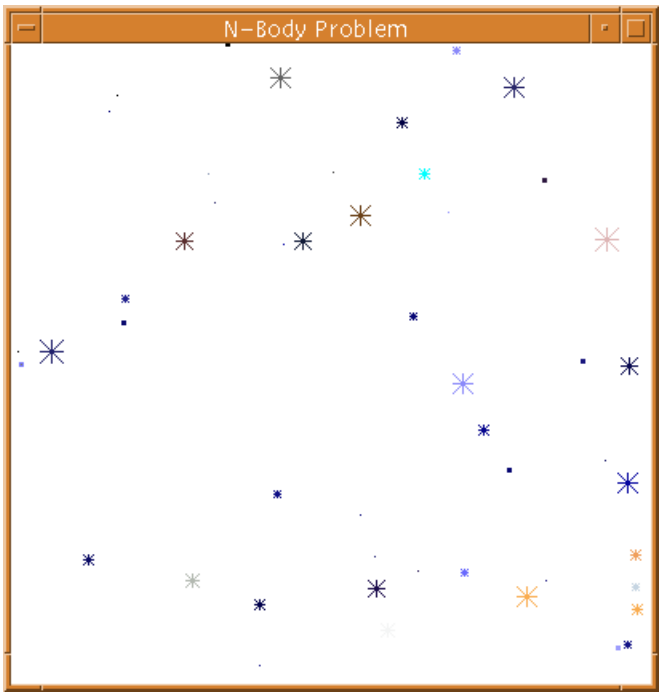

Figure 1.1 Astrophysical *N*-body simulation by Scott Linssen (undergraduate University of North Carolina at Charlotte [UNCC] student).

Parallel Computers

- A parallel computer consists of multiple processors operating together to solve a single problem. This provides an effective and relatively inexpensive means to solve problems requiring large computation speed.

- To use a parallel computer, one must split the problem into parts, each to be performed on a separate processor in parallel.

- Parallel programming is the art of writing programs of this form.

- The idea is that n processors can provide up to n times the speed.

- Ideal situation. Rarely achieved in practice.

  - Problems can't always be divided perfectly into independent parts.

  - Interaction required for data transfer and synchronization (overhead).

- Parallel computers offer the advantage of more memory: The aggregate memory is larger than the memory for a single processor.

- Because of speed and memory increase, parallel computers often allow larger or more precise solutions to be solved.

- Multi-processor computers are becoming the norm. IBM, HP, AMD, and Intel are designing processors that can execute multiple threads/programs in parallel on a single chip.

Types of Parallel Computers

Parallel computers are either specially designed computer systems containing multiple processors or multiple independent computers interconnected.

We will discuss three types of parallel computers:

- Shared memory multiprocessor systems
- Message-Passing multicomputers
- Distributed shared memory systems

Shared Memory Multiprocessor Systems

Conventional Computer consists of a single processor executing program stored in memory.

Each memory location has an address from 0 to $2^n - 1$ where the address has n bits.

See Figure 1.2.

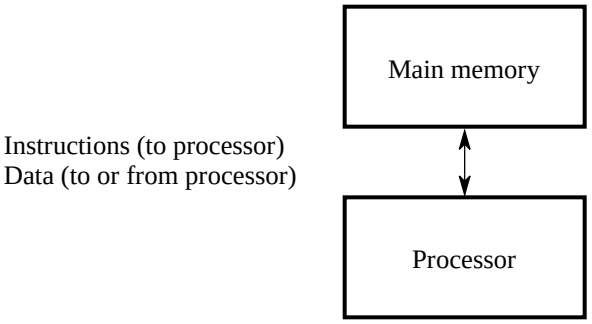

Figure 1.2 Conventional computer having a single processor and memory.

Multiprocessor system extends this by having multiple processors and multiple memory modules connected through an interconnection network.

See Figure 1.3.

Each processor can access each module. Called "shared memory" configuration.

Employs a single address space. Each memory location has unique address, and processors all use same address.

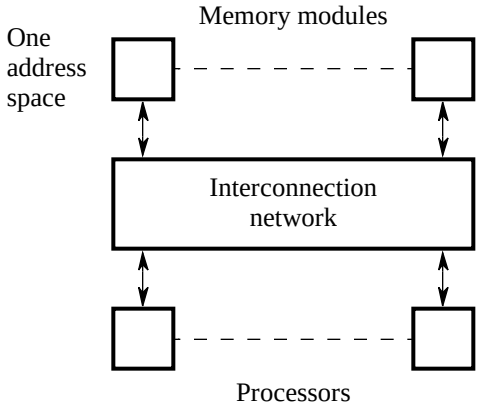

Figure 1.3 Traditional shared memory multiprocessor model.

Programming Shared Memory Multiprocessor Systems

Each processor has its own executable code stored in memory to execute.

Data for each processor is stored in memory, and thus accessible to all.

Can use a "parallel programming language" with special constructs and statements. ie. "FORTRAN 90" or "high performance FORTRAN." See chapter 13 of High Performance Computing. Rely on compilers.

Can also use "threads." A multi-threaded program has regular code sequences for each processor. Communicate through shared memory locations. We will examine the POSIX standard "Pthreads."

Types of Shared Memory Systems

Two main types of shared memory multiprocessor systems:

- Uniform Memory Access (UMA)
- Nonuniform memory access (NUMA) systems.

In a UMA system, each processor can access each memory module in the same amount of time.

A common example is a Symmetric multiprocessing (SMP) system such as a duo processor Pentium III computer.

Does not scale well above 64 processors. Expensive to have same access time to multiple memory modules and processors due to physical distance and number of interconnects.

NUMA Systems

NUMA systems solve this by having hierarchical or distributed memory structure.

Processors can access physically nearby memory locations faster than distant locations.

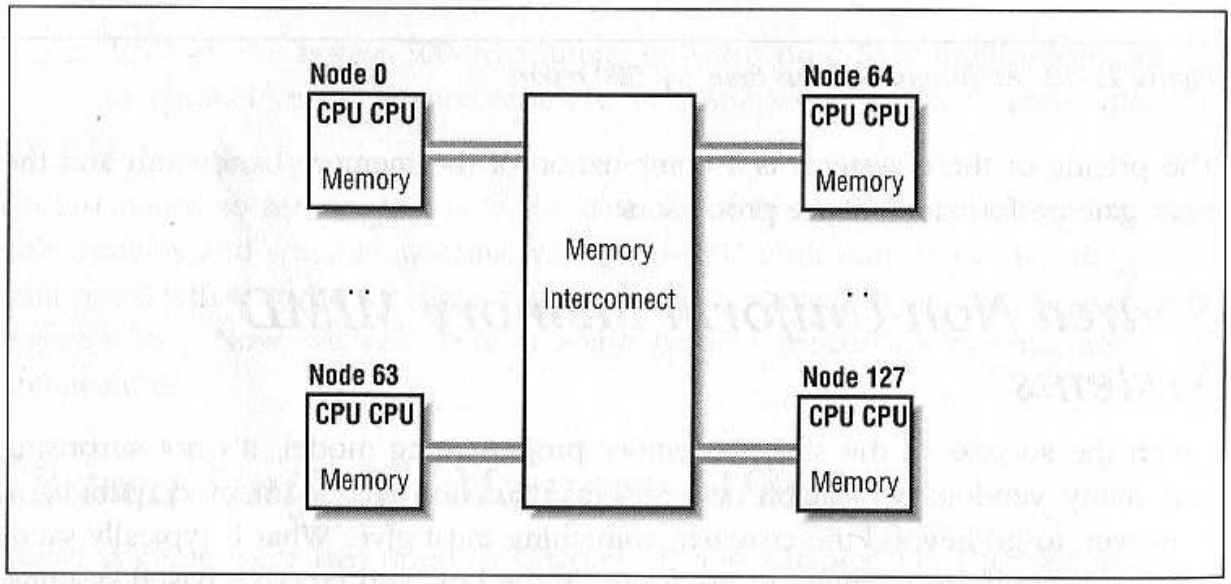

Figure 12-14: Nodes connected using a memory interconnect *

Can scale to 100's and 1000's of processors.

*K. Dowd and C. Severance, *High Performance Computing, 2nd Ed.*, O'reilly, 1998.

Message-Passing Multicomputers

A shared memory multiprocessor is a special designed computer system.

Alternately, can can create multiprocessor by connecting complete computers through interconnection network. See Fig 1.4.

- Each computer has a processor and local memory not accessible to other processors

- Each computer has its own address space.

- A processor can only access a location in own memory.

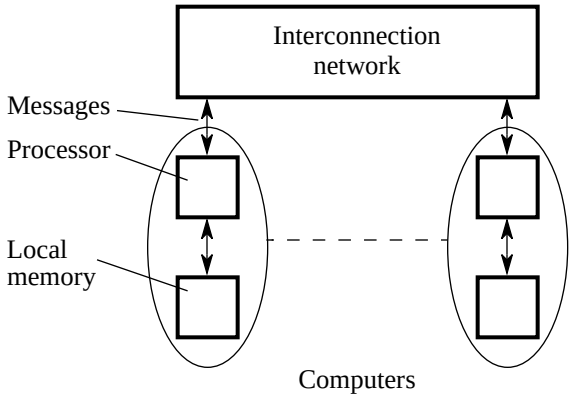

Figure 1.4 Message-passing multiprocessor model (multicomputer).

Message-Passing Multicomputers Cont.

Interconnection network is used to send messages between processors.

Messages may be instructions, synchronization info, as well as data other processors need for computations.

Systems of this type are called message-passing multiprocessors or multicomputers.

Examples: network of workstations (NOW), beowulf clusters.

Message-passing multiprocessors scale better than shared memory multiprocessor systems. Cheaper and more flexible to construct. Design more open. Easy to extend.

Programming Multicomputers

Problem divided into parts intended to be executed simultaneously on each processor.

Typically, we have multiple independent processes running in parallel to solve problem. Can be on same processor or not.

Messages carry data between processes as dictated by program.

Use message-passing library routines linked to sequential programs. We will examine the message-passing interface (MPI) libraries.

Pros/cons of Message-Passing Model

Advantages:

Universality: Can be used with multiple processors connected by a (fast/slow) communication network. ie. either a multiprocessor or network of workstations.

Ease of Debugging: Prevents accidental overwriting of memory. Model only allows one process to directly access to a specific memory location.

The fact that no special mechanisms are required to control concurrent access to data can greatly decrease execution time.

Performance: Associates data with specific processor and memory. Makes cache-management and compilers work better. Applications can exhibit superlinear speedup.

Pros/cons Cont.

Disadvantages:

Requires programmers to use explicit program calls to pass messages. Error prone and has been compared to low-level assembly language programming.

Data can not be shared. It must be copied. Problem if need to do many tasks using a lot of data.

Distributed Shared Memory

Gives the programming flexibility of shared memory with the hardware flexibility of Message-Passing Multicomputers.

Each processor has access to entire memory using a single common address space.

Memory access to a location not local to a processor is done using message passing in an automated fashion. Called *shared virtual memory.*

See Figure 1.5.

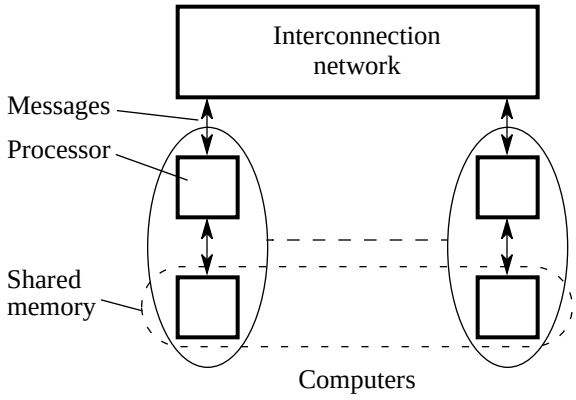

Figure 1.5 Shared memory multiprocessor implementation.

Flynn Computer Classifications

SISD: For single processor computer, have a single stream of instructions operating on a single stream of data. Called a *single instruction stream - single data stream (SISD)* computer.

MIMD: In a multiprocessor system, each processor has a stream of instructions acting upon a separate set of data. Called a *multiple instruction stream - multiple data stream (MIMD)* computer. See Figure 1.6.

SIMD: This is when a single program generates a single stream of instructions which are broadcast to multiple processors who execute the same instruction in synchronism, but on different data. Called a *single instruction stream - multiple data stream (SIMD)* computer.

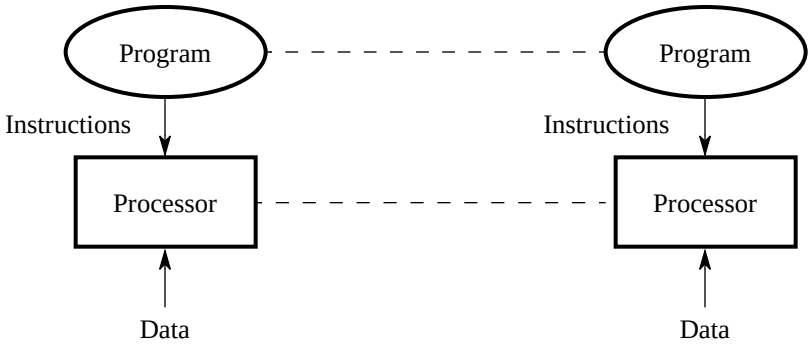

Figure 1.6 MPMD structure.