Advanced Computer Architecture (0630561)

Lecture 14

Design Issues for Parallel Computers

Prof. Kasim M. Al-Aubidy

Computer Eng. Dept.

Introduction:

- When approaching a new parallel computer system, three fundamental questions to ask are:
- Q1: what are the nature, size, and number of processing elements?
- Q2: what are the nature, size, and number of memory modules?
- Q3: what are the processing and memory elements interconnected?

Processing Element (PE): ALUs to complete CPUs size; small portion of a chip.

Memory System: split up into modules that operate independently in parallel to allow access by many CPUs at the same time. Memory models may be small or large, closely integrated with CPUs or located on a different board.

Connection: a collection of chips that are connected in one manner or other.

- Some parallel computers are designed to run multiple independent jobs simultaneously (do no communicate).
- Parallel computer used to run a single job consisting of many parallel processors.

- **Loosely Coupled:** systems with a small number of large, independent CPUs that have low-speed connections between CPUs.
- **Tightly Coupled:** systems with smaller components, closer together and interact with each other frequently over a high bandwidth communication networks.

Communication Models:

Two designs have been proposed and implemented;

- Multiprocessors, and
- Multicomputers.

Multiprocessors:

- All CPUs share a common physical memory.
- All processes working together on a multiprocessor can share a single virtual address space mapped onto the common memory.
- Any process can read or write a word of memory by just executing LOAD or STORE instruction.

Multicomputers:

- Each CPU has its own local memory, accessible only to itself and not to any other CPU. Access achieved by LOAD and STORE instructions.
- Multicomputers have one physical address space per CPU, while multiprocessors have a single physical address space shared by all CPUs.
- CPUs on multicomputers can not communicate by just reading and writing the common memory, they need a different communication mechanism (using the interconnection network).

Interconnection Networks:

- Multiprocessors and multicomputers require both similar interconnection networks, this is mainly due to:
  1. Multiprocessors have multiple memory modules that must be interconnected with one another and with the CPUs.
  2. Both of them use message passing.
  3. In large multiprocessors, communication between CPUs and remote memory consists of the CPU sending an explicit message (called packet) to memory requesting some data, and the memory sending back a reply packet.
- Interconnection networks consist of FIVE components;
  1. CPU
  2. Memory modules
  3. Interfaces
  4. Links
  5. Switches

Interface: is a chip or a board that is attached to each CPU's local bus and can talk to CPU and to local memory. It has a programmable processor and RAM private to it alone.

Links: are physical channels over which the bits move. They can be electrical or optical fiber, serial or parallel. Each link has a maximum bandwidth (no of bits/sec). Links can be;

1. Simplex (unidirectional),
2. Half-duplex (one way at a time),
3. Full-duplex (both ways at a time)

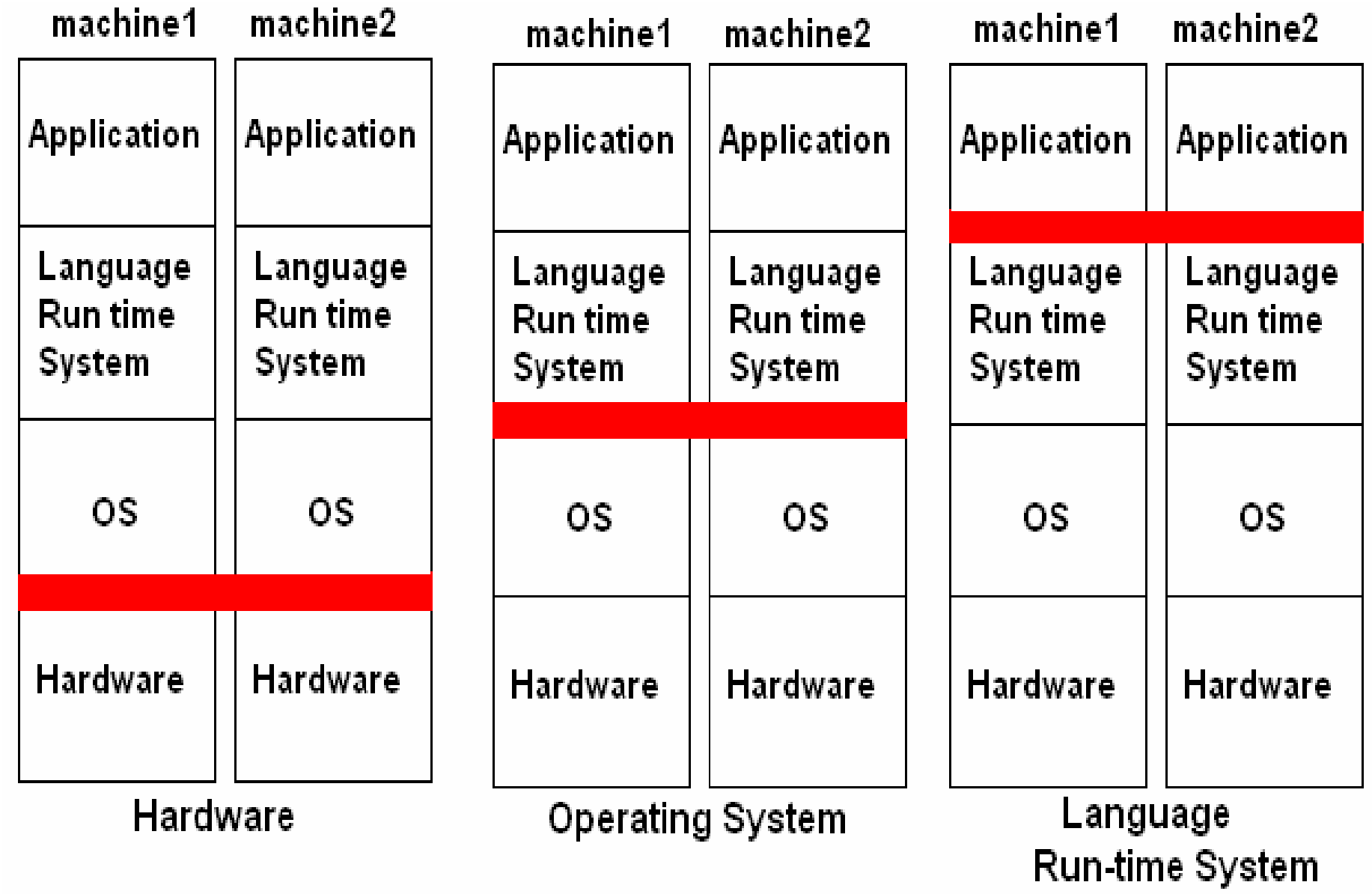
Switches: are devices with multiple input ports and multiple output ports. When a packet arrives at a switch on an input port, some bits in the packet are used to select the output port to which the packet is sent. A packet might be a short (2 or 4 bytes) or longer (e.g. 8KBs).

Q: Why would anyone build multicomputers, when multiprocessors are easier to program?

- Large multicomputers are much simpler and cheaper to build than multiprocessors with the same number of CPUs. Implementing a memory shared by even a few hundred CPUs is a big problem, whereas building a multicomputer with 10000 CPUs or more is straightforward.

- **Multiprocessors are hard to build but easy to program, whereas multicomputers are easy to build but hard to program.**
- One approach to building hybrid systems is based on the fact that modern computer systems are constructed as a series of layers. This opens the possibility of implementing the shared memory at any one of several layers.

- The 2nd approach is to use multicomputer hardware and have the OS simulate shared memory by providing a single system-wide paged shared virtual address space. This approach is called **Distributed Shared Memory (DSM).**
- Each page is located in one of the system memories.
- Each CPU has its own virtual memory and its own page tables.
- When a CPU does a LOAD or STORE on a page it does not have, a trap to the OS occurs. The OS then locates the page and asks the CPU currently holding it to unmap the page and send it over the interconnection network. When it arrives, the page is mapped in and the faulting instruction is restarted.
- The 3rd approach is to have a user-level runtime system implement a form of shared memory. The programming language provides some kind of shared memory abstraction, which is then implemented by the compiler and runtime system.


machine1
machine2
Application
Application
Language
Run time
System
Language
Run time
System
OS
OS
Hardware
Hardware
Hardware
machine1
machine2
Application
Application
Language
Run time
System
Language
Run time
System
OS
OS
Hardware
Hardware
Operating System
machine1
machine2
Application
Application
Language
Run time
System
Language
Run time
System
OS
OS
Hardware
Hardware
Language
Run-time System

Interconnection Networks:

- **Mode of Operation:**

– **Synchronous**: a single global clock is used by all components in the system.

– **Asynchronous**: No global clock required. Hand shaking signals are used to coordinate the operation of asynchronous systems.

- **Control Strategy:**

– **Centralized**: one central control unit is used to control the operations of the components of the system.

– **Decentralized**: the control function is distributed among different components in the system.

Interconnection Networks:

- **Switching Techniques:**

– **Circuit switching**: a complete path has to be established prior to the start of communication between a source and a destination.

– **Packet switching**: communication between a source and a destination takes place via messages divided into smaller entities, called packets.

- **Topology:**

– Describes how to connect processors and memories to other processors and memories.

– **Static**: direct fixed links are established among nodes to form a fixed network.

– **Dynamic**: connections are established when needed.