Unit 13 Multiprocessor Systems

Structure:

13.1 Introduction

In the previous unit, we have discussed a most important issue viz. security and protection of data in computers. Once we secure our data, the next concern will be to increase the speed and performance of computers and that is what we shall discuss in this unit. There are basically two ways of increasing the speed of computer hardware. They are by using:

- High speed components
- New architectural structures

In the first category, high-speed components rely on exotic technology and fabrication processes and such technologies and processes tend to be expensive and non-standardized. Multiprocessor systems fall into the second category and provide an alternative for improving performance of computer systems by coupling a number of low cost standard processors.

Objectives:

After studying this unit, you should be able to:

- explain the advantages of multiprocessors
- discuss the classification of multiprocessors
- describe the interconnections in multiprocessors
- list out the types of multiprocessor operating systems

- discuss multiprocessor operating system functions, requirements and implementation issues

13.2 Multiprocessors

Multiprocessing can be applied to provide:

- Increased throughput: by executing a number of different user processes on different processors in parallel.
- Application speedup: by executing some portion of the application in parallel.

Throughput can be improved by executing a number of unrelated user processes on different processors in parallel. System throughput is improved as a large number of tasks are completed in unit time.

Application speedup may be obtained by exploiting the hidden parallelism in the application by creating multiple threads / processes / tasks for execution on different processors.

Inter-processor communication and synchronization are an overhead in multiprocessor systems. Design goals aim to minimize inter-processor interaction and provide an efficient mechanism for carrying them out when necessary.

Advantages of multiprocessors are manifold, which are given below:

- *Performance and computing power:* Use of multiprocessor systems speeds up an application. Problems with high inter-processor interaction can be solved quickly.
- *Fault tolerance:* The inherent redundancy in multiprocessor systems can be used to increase availability and eliminate single point failures.
- *Flexibility:* A multiprocessor system can be made to dynamically reconfigure itself so as to optimize different objectives for different applications such as throughput, application speedup or fault tolerance.
- *Modular growth:* Use of a modular system design overcomes certain problems and can be accomplished say by adding exactly tailor made components such as processors, memories, I/O devices and the like.
- *Functional specialization:* Specialized processors can be added to improve performance in particular applications.

- *Cost / performance:* Cost performance ratio in case of multiprocessor systems is far below that of large computers of the same capacity and hence multiprocessor systems are cost effective. They may be structured similar to large computers for a wide range of applications.

13.3 Multiprocessor Classification

Flynn classified computer systems based on how the machine relates its instructions to the data being processed. Instructions may form either a single instruction stream or a multiple instruction stream. Similarly the data which the instructions use could be either single or multiple. Based on such instructions and data, Flynn classified computer systems as follows:

- *SISD:* Single Instruction Stream, Single Data Stream. Usually found in conventional serial computer systems.
- *SIMD:* Single Instruction Stream, Multiple Data Stream. These are vector processors / array processors where a single instruction operates on different data in different execution units at the same time.
- *MISD:* Multiple Instruction Streams, Single Data Stream. Multiple instructions operate on a single data stream. Not a practical viability.
- *MIMD:* Multiple Instruction Streams, Multiple Data Stream. This is the most general classification where multiple instructions operate on multiple data stream simultaneously. This is the class that contains multiprocessors of different types.

Based on the relationships between processes and memory, multiprocessor systems can be classified as:

- *Tightly coupled:* Individual processors within the multiprocessor system share global shared memory. Inter-processor communication (IPC) and synchronization in tightly coupled multiprocessor systems is through shared memory. High bandwidth and low delay in interconnection paths are the main characteristics of tightly coupled multiprocessor systems.
- *Loosely coupled:* Individual processors within the multiprocessor system access their own private / local memory. This classification may not be very rigid and multiprocessor systems have both shared memory as well as local memory. In loosely coupled multiprocessor systems, message passing is the primary mechanism for IPC. Distributed systems fit into this class of loosely coupled systems. Lower bandwidth and high delay

in the interconnection paths of the past have reduced drastically with the use of optical fiber links and high speed LANs.

- *Hybrid systems:* Hybrid systems have both local and global memories. Some loosely coupled systems also allow access of global memory in addition to local memory.

Based on memory and access delays, multiprocessor systems are classified as:

- *Uniform Memory Access (UMA):* Multiple processors can access all the available memory with the same speed.
- *Non Uniform Memory Access (NUMA):* Different areas of memory have different access times. This is based on the nearness of the memory to a given processor and also on the complexity of the switching logic between the processor and the memory.
- *NO Remote Memory Access (NORMA):* Systems have no shared memory.

Self-Assessment Questions

1. Use of multiprocessor systems speeds up an application. (True / False)
2. SIMD stands for _____.
3. Distributed systems fit into the class of _____ systems. (Pick the right option)
   a) Loosely coupled
   b) Tightly coupled
   c) Hybrid
   d) Non-hybrid

13.4 Multiprocessor Interconnections

The nature of multiprocessor interconnections has an effect on the bandwidth for communication. Complexity, cost, IPC and scalability are some features considered in interconnections. Basic architectures for multiprocessor interconnections are as follows:

- Bus-oriented systems
- Crossbar-connected systems
- Hyper cubes
- Multistage switch-based systems

Bus-oriented systems

A shared bus connects processors and memory in the multiprocessor system as shown in figure 13.1.

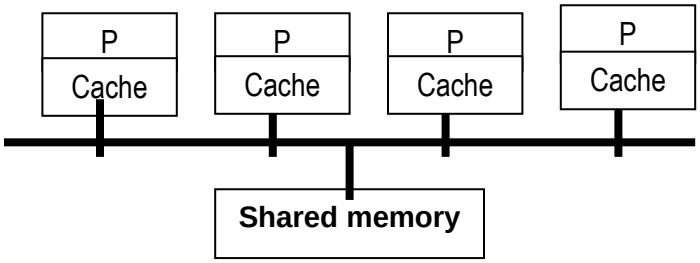

Fig. 13.1: Shared-bus multiprocessor organization

Processors communicate with each other and the shared memory through the shared bus. Variations of this basic scheme are possible where processors may or may not have local memory, I/O devices may be attached to individual processors or the shared bus and the shared memory itself can have multiple banks of memory.

The bus and the memory being shared resources there is always a possibility of contention. Cache memory is often used to release contention. Cache associated with individual processors provides a better performance. A 90% cache hit ratio improves the speed of the multiprocessor systems nearly 10 times as compared to systems without cache.

Existence of multiple caches in individual processors creates problems. Cache coherence is a problem to be addressed. Multiple physical copies of the same data must be consistent in case of an update. Maintaining cache coherence increases bus traffic and reduces the achieved speedup by some amount. Use of a parallel bus increases bandwidth.

The tightly coupled, shared bus organization usually supports 10 processors. Because of its simple implementation many commercial designs of multiprocessor systems are based on shared-bus concept.

Crossbar-connected systems

An interconnection of processors and memory in a multiprocessor system using crossbar approach is shown in figure 13.2:

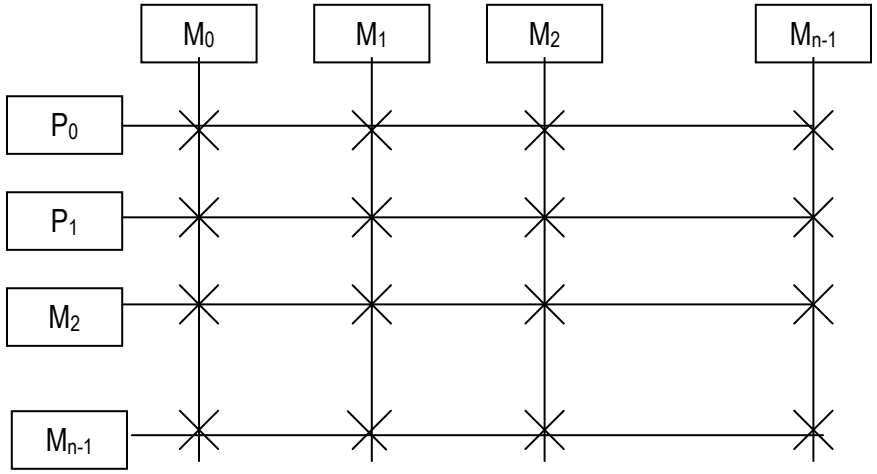

Fig. 13.2: Crossbar interconnection

Simultaneous access of 'n' processors and 'n' memories is possible if each of the processors accesses a different memory. The crossbar switch is the only cause of delay between processor and memory. If no local memory is available in the processors then the system is a UMA multiprocessor system. Contention occurs when more than one processor attempts to access the same memory at the same time. Careful distribution of data among the different memory locations can reduce or eliminate contention.

High degree of parallelism exists between unrelated tasks but contention is possible if inter-process and inter-processor communication and synchronization are based on shared memory, for example, semaphore.

Since 'n' processors and 'n' memory locations are fully connected, n^2 cross points exist. The quadratic growth of the system makes the system expensive and limits scalability.

Hyper cubes

A 3-dimensional hypercube can be visualized as shown in figure 13.3:

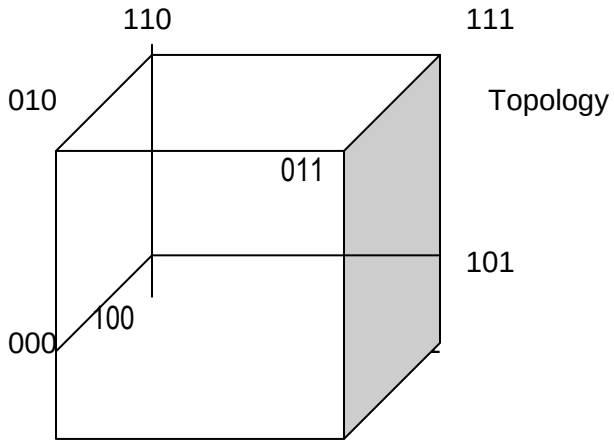

Fig. 13.3: 3-dimensional hypercube

Cube topology has one processor at each node / vertex. Given a 3-dimentional cube (a higher dimensional cube cannot be visualized), $2^3 = 8$ processors are interconnected. The result is a NORMA type multiprocessor and is a common hypercube implementation.

Each processor at a node has a direct link to $\log_2 N$ nodes where N is the total number of nodes in the hypercube. For example, in a 3-dimensional hypercube, N = 8 and each node is connected to $\log_2 8 = 3$ nodes. Hypercube can be recursive structures with high dimension cubes containing low dimension cubes as proper subsets. For example, a 3-dimensional cube has two 2-dimensional cubes as subsets. Hypercube have a good basis for scalability since complexity grows logarithmically whereas it is quadratic in the previous case. They are best suited for problems that map on to a cube structure, those that rely on recursion or exhibit locality of reference in the form of frequent communication with adjacent nodes. Hypercube form a promising basis for large-scale multiprocessors.

Message passing is used for inter-processor communication and synchronization. Increased bandwidth is sometimes provided through dedicated nodes that act as sources / repositories of data for clusters of nodes.

Multistage switch-based systems

Processors and memory in a multiprocessor system can be interconnected by use of a multistage switch. A generalized type of interconnection links N inputs and N outputs through $\log_2 N$ stages, each stage having N links to N / 2 interchange boxes. The structure of a multistage switch network is shown below in figure 13.4:

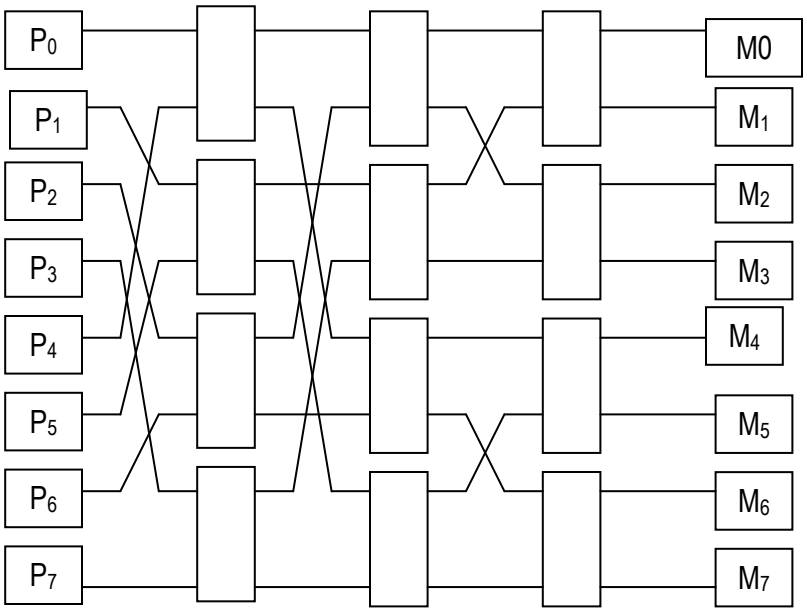

Fig. 13.4: Multistage switching network

The network has $\log_2 N = \log_2 2^3 = 3$ stages of N / 2 = 8 / 2 = 4 switches each. Each switch is a 2x2 crossbar that can do anyone of the following:

- Copy input to output
- Swap input and output
- Copy input to both output

Routing is fixed and is based on the destination address and the source address. In general to go from source S to destination D the i^{th} stage switch box in the path from S to D should be set to swap if $S_i \neq D_i$ and set to straight if $S_i = Di$.

Illustration: If S = 000 and D = 000 then Si = Di for all bits. Therefore all switches are straight.

If S = 010 and D = 100 then $S_1 \neq D_1$, $S_2 \neq D_2$ and $S_3 = D_3$. Therefore switches in the first two stages should be set to swap and the third to straight.

Multistage switching network provides a form of circuit switching where traffic can flow freely using full bandwidth when blocks of memory are requested at a time. All inputs can be connected to all outputs provided each processor is accessing a different memory. Contention at the memory module or within the interconnection network may occur. Buffering can relieve contention to some extent.

13.5 Types of Multiprocessor Operating Systems (MOS)

Three basic types of multiprocessor operating systems are:

- Separate supervisors
- Master / slave
- Symmetric

Separate supervisors

In separate supervisor systems, each node is a processor having a separate operating system with a memory and I/O resources. Addition of a few additional services and data structures will help to support aspects of multiprocessors.

A common example is the hypercube. Due to their regular repeating structure constructed of identical building blocks, they tend to replicate identical copies of a kernel in each node. Kernel provides services such as local process, memory management and message passing primitives. Parallelism is achieved by dividing an application into subtasks that execute on different nodes.

Master/Slave

In this approach, one processor – the master is dedicated to execute the operating system. The remaining processors are slaves and form a pool of computational processors. The master schedules and controls the slaves. This arrangement allows parallelism in an application by allocating to it many slaves.

Master / slave systems are easy to develop. A uniprocessor operating system can be adapted for master / slave multiprocessor operations with the

addition of slave scheduling. Such systems have limited scalability. Major disadvantages are that computational power of a whole processor is dedicated for control activity only and if the master fails, the entire system is down.

Symmetric

All processors are functionally identical. They form a pool of resources. Other resources such as memory and I/O devices are available to all processors. If they are available to only a few then the system becomes asymmetric.

The operating system is also symmetric. Any processor can execute it. In response to workload requirements and processor availability, different processors execute the operating system at different times. The processor which executes the operating system temporarily is the master (also called floating master).

An existing uniprocessor operating system such as UNIX can easily be ported to a shared memory UMA multiprocessor. Shared memory contains the resident operating system code and data structures. Any processor can execute the operating system. Parallel execution of applications is possible using a ready queue of processes in shared memory. The next ready process to the next available processor until either all processors are busy / queue is empty could be a possible allocation scheme. Concurrent access of shared data structures provides parallelism.

Self Assessment Questions

4. Complexity, cost, IPC and scalability are not considered in multiprocessor interconnections. (True / False)
5. In _____ approach, one processor is dedicated to execute the operating system and the remaining processors are form a pool of computational processors.
6. Simultaneous access of ‘n’ processors and ‘n’ memories is possible in the case of _____. (Pick the right option)
   a) Bus oriented systems
   b) Crossbar interconnection systems
   c) Hyper cubes
   d) Multistage switch based systems

13.6 Multiprocessor Operating System Functions and Requirements

Multiprocessor operating systems manage available resources to facilitate program execution and interaction with users. Resources to be managed include:

- Processors
- Memory
- I/O devices

Processor scheduling is crucial for efficient use of multiple processors. The scheduler has to:

- Allocate processors among applications
- Ensure efficient use of processors allocated to an application.

A tradeoff exists between the two. The former affects throughput while the latter affects speedup. Depending on an application if speedup is given priority then a large portion of the processors is dedicated to the application. On the other hand if throughput is to be increased then several applications are scheduled each availing fewer processors. The two main facts of operating system support for multiprocessor are:

- Flexible and efficient inter-process and inter-processor synchronization mechanisms.
- Efficient creation and management of a large number of threads / processes.

Memory management in multiprocessor systems is dependent on the architecture and interconnection scheme. In loosely coupled systems, memory management is usually independent. In shared memory system, operating system should provide access to shared data structures and synchronization variables in a safe and efficient way. A hardware independent unified model of a shared memory for easy portability is expected of a multiprocessor operating system. A unified memory model consisting of messages and shared memory provides a flexible tool.

Device management is of little importance in a multiprocessor operating system. This may be due to the importance attached to speedup in the so called compute-intensive applications with minimal I/O. As more general purpose applications are run on multiprocessor systems, I/O requirements will also be a matter of concern along with throughput and speed.

13.7 Operating System Design and Implementation Issues

Some major issues involved in processor and memory management in multiprocessor operating systems are described below:

Processor management and scheduling

The main issues in processor management include:

- Support for multiprocessing
- Allocation of processing resources
- Scheduling

The operating system can support multiprocessors by providing a mechanism for creating and maintaining a number of processes / threads. Each process has allocated resources, state and accesses I/O devices. An application having several co-operating processes can be viewed as a virtual multiprocessor. In a multiprocessor environment true multiprocessing is possible by allocating a physical processor to each virtual processor where as in a uniprocessor system an illusion of multiprocessing is created by multiplexing the processor among virtual processes.

When threads are used, each application is implemented as a process. Its concurrent portions are coded as separate threads within the enclosing process. Threads of a single process share memory and resources acquired by the process. Threads not only facilitate multiprocessors but also help the scheduling process. Related threads could be co-scheduled to reduce slowdown associated with the out-of-phase scheduling.

Processor allocation creates problems in massively parallel systems. One way of keeping track of processor resources is to organize them into a hierarchy. A processor at the highest level in the hierarchy notes state and activity of a group of processors. When an application is to be allocated them, then idle machines at the bottom level get allocated. The hierarchy can grow upwards. This is called wave scheduling.

If wanted number of resources is not available then a request to a higher level in the hierarchy is made. Fault tolerance is possible as a process higher up in the hierarchy could reallocate activities of a failed processor to another. But implementation has practical difficulties such as a note of wrong availability.

Processors are allocated to applications. These have to be scheduled. One main objective is to co-schedule processes that interact so that they run at the same time. Processes that can be co-scheduled include several threads of a single process, sender and receiver of a message, processes at the end of a pipe, etc. Individual processes may be uni-programmed or multi-programmed. Since multi-programming of individual processes creates problems for communication, co-scheduling process groups that communicate with each other are preferred.

In loosely coupled systems the scheduler should note affinity of some processes for certain processors that may be due to process state stored in local memory. Also placing interacting processes in the same processor or cluster with direct inter-processor links can reduce communication costs.

Memory management

In tightly coupled multiprocessor systems the operating system must provide access to shared memory through primitives for allocation and reallocation of shared memory segments. If shared virtual memory is supported then translation look aside buffers (TLBs) contain mappings to shared segments. Similarly open files could also be shared. Use of shared memory improves performance of message passing.

Self-Assessment Questions

7. Multiprocessor operating systems manage available resources to facilitate program execution and interaction with users. (True / False)
8. TLB stands for _____.
9. Processors are allocated to _____. (Pick the right option)
   a) Software
   b) Hardware
   c) Applications
   d) Programs

13.8 Summary

Let's recapitulate important concepts discussed in this unit:

- Multiprocessor systems provide an alternative for improving performance of computer systems by coupling a number of low cost standard processors.

- Flynn classified computer systems based on how the machine relates its instructions to the data being processed. Instructions may form either a single instruction stream or a multiple instruction stream.
- Complexity, cost, IPC and scalability are some features considered in interconnections.
- Memory management in multiprocessor systems is dependent on the architecture and interconnection scheme.
- In tightly coupled multiprocessor systems the operating system must provide access to shared memory through primitives for allocation and reallocation of shared memory segments.

13.9 Terminal Questions

1. List the advantages of multiprocessor systems.
2. How can multiprocessors be classified?
3. Explain the various multiprocessor interconnections.
4. Describe the basic types of multiprocessor operating systems.
5. Discuss the various issues for multiprocessor operating system design.

13.10 Answers

Self Assessment Questions

1. True
2. Single Instruction stream, Multiple Data stream
3. a) Loosely coupled
4. False
5. Master / Slave
6. b) Crossbar Interconnection Systems
7. True
8. Translation Look-aside Buffers
9. c) Applications

Terminal Questions

1. Use of multiprocessor systems speeds up an application. Problems with high inter-processor interaction can be solved quickly. The inherent redundancy in multiprocessor systems can be used to increase availability and eliminate single point failures. (Refer section 13.2 for detail)

2. Flynn classified computer systems based on how the machine relates its instructions to the data being processed. Instructions may form either a single instruction stream or a multiple instruction stream. Similarly the data which the instructions use could be either single or multiple. (Refer section 13.3 for details)
3. The nature of multiprocessor interconnections has an effect on the bandwidth for communication. Complexity, cost, IPC and scalability are some features considered in interconnections. (Refer section 13.4 for details)
4. Three basic types of multiprocessor operating systems are: Separate supervisors, Master / Slave and Symmetric. (Refer section 13.5 for details)
5. The operating system can support multiprocessors by providing a mechanism for creating and maintaining a number of processes / threads. Each process has allocated resources, state and accesses I/O devices. (Refer section 13.7 for details)