**Resume Assignment of Operating System**

****

By :

Muhammad Wildan Ainur Rachim L200164002

Ilham Idkha Syaifudin L200164010

A multiprocessor can be defined as the computer which uses two or more processing units under the integrated control. Multi-processing is also defined as the way of using two or more than two CPUs within a single computer. As we all know that there are processors inside the computers, the multi processors, as the name indicates, have the ability to support more than one processor at a same time. Usually in multi-processing the processors are organized in the parallel form and hence a large number of the executions can be brought at the same time i.e. multi-processing helps in executing the same instructions a number of time at a particular time. Some other related definition of the multi processors are that multi-processing is the sharing of the execution process by the interconnection of more than one microprocessor using tightly or loosely couples technology. Usually multi-processing tasks carries two simultaneous steps. One is the performing the task of editing and the other is the handling the data processing. A multi-processor device comprising, over a single semiconductor chip a plurality of processors including a first group of processors and a second group of processors; a first bus to which the first group of processors is coupled; a second bus to which the second group of processors is coupled; a first external bus interface to which the first bus is coupled; and a second external bus interface to which the second bus is coupled. The term multiprocessing is also used to refer to a computer that has many independent processing elements. The processing elements are almost full computers in their own right. The main difference is that they have been freed from the encumbrance of communication with peripherals.

Advantages of Multiprocessor Systems

Some advantages of multiprocessor system are as follows:

1. Reduced Cost: Multiple processors share the same resources. Separate power supply or mother board for each chip is not required. This reduces the cost.
2. Increased Reliability: The reliability of system is also increased. The failure of one processor does not affect the other processors though it will slow down the machine. Several mechanisms are required to achieve increased reliability. If a processor fails, a job running on that processor also fails. The system must be able to reschedule the failed job or to alert the user that the job was not successfully completed.
3. More work: As we increase the number of processors then it means that more work can be done in less time. Id more than one processor cooperates on a task then they will take less time to complete it.
4. If we divide functions among several processors, then if one processor fails then it will not affect the system or we can say it will not halt the system, but it will effect on the work speed. Suppose I have five processors and one of them fails due to some reasons then each of the remaining four processors will share the work of failed processor. So it means that system will not fail but definitely failed processor will effect on its speed.
5. If you pay attention on the matter of which save much money among multi-processor systems and multiple single-processor systems then you will know that multiprocessor systems save moremoneythan multiple single-processor systems because they can share power supplies, memory and peripherals.
6. Increased Throughput: An increase in the number of processes completes the work in less time. It is important to note that doubling the number of processors does not halve the time to complete a job. It is due to the overhead in communication between processors and contention for shared resources etc.

Implementation Issues

Despite the conceptual elegance of RPC, there are a few snakes hiding under the grass. A big one is the use of pointer parameters. Normally, passing a pointer to a procedure is not a problem. The called procedure can use the pointer the same way the caller can because the two procedures reside in the same virtual address space. With RPC, passing pointers is impossible because the client and server are in different address spaces.

Distribute Shared Memory

Although RPC has its attractions, many programmers still prefer a model of shared memory and would like to use it, even on a multicomputer. Surprisingly enough, it is possible to preserve the illusion of shared memory reasonably well, ev en when it does not actually exist, using a technique called DSM (Distributed Shared Memory).

Replication

One improvement to the basic system that can improve performance considerably is to replicate pages that are read only, for example, program text, read-only constants, or other read-only data structures.

For example, if page 10 in Fig. 8-22 is a section of program text, its use by CPU 0 can result in a copy being sent to CPU 0 without the original in CPU 1’s memory being invalidated or disturbed, as shown in Fig. 8-22(c). In this way, CPUs 0 and 1 can both reference page 10 as often as needed without causing traps to fetch missing memory.

False Sharing

DSM systems are similar to multiprocessors in certain key ways. In both systems, when a nonlocal memory word is referenced, a chunk of memory containing the word is fetched from its current location and put on the machine making the reference (main memory or cache, respectively).

An important design issue is how big the chunk should be? In multiprocessors, the cache block size is usually 32 or 64 bytes, to avoid tying up the bus with the transfer too long. In DSM systems, the unit has to be a multiple of the page size (because the MMU works with pages), but it can be 1, 2, 4, or more pages. In effect, doing this simulates a larger page size.

Achieving Sequential Consistency

If writable pages are not replicated, achieving consistency is not an issue. There is exactly one copy of each writable page, and it is moved back and forth dynamically as needed. Since it is not always possible to see in advance which pages are writable, in many DSM systems, when a process tries to read a remote page, a local copy is made and both the local and remote copies are set up in their respective MMUs as read only.

Multicomputer Scgeduling

On a multiprocessor, all processes reside in the same memory. When a CPU finishes its current task, it picks a process and runs it. In principle, all processes are potential candidates. On a multicomputer the situation is quite different. Each node has its own memory and its own set of processes.

Load Balancing

There is relatively little to say about multicomputer scheduling because once a process has been assigned to a node, any local scheduling algorithm will do, unless gang scheduling is being used.

Distributed System

Having now completed our study of multicores, multiprocessors, and multicomputers we are now ready to turn to the last type of multiple processor system, the distributed system. These systems are similar to multicomputers in that each node has its own private memory, with no shared physical memory in the system. However, distributed systems are even more loosely coupled than multicomputers.

Network Hardware

Distributed systems are built on top of computer networks, so a brief introduction to the subject is in order. Networks come in two major varieties, LANs (Local Area Networks), which cover a building or a campus, and WANs (Wide Area Networks), which can be citywide, countrywide, or worldwide.

Network Services

Computer networks provide services to the hosts and processes using them. Connection-oriented service is modeled after the telephone system. To talk to someone, you pick up the phone, dial the number, talk, and then hang up.

Network Protocols

All networks have highly specialized rules for what messages may be sent and what responses may be returned in response to these messages. For example, under certain circumstances (e.g., file transfer), when a message is sent from a source to a destination, the destination is required to send an acknowledgement back indicating correct receipt of the message.

Document-Based Middleware

Now that we have some background on networks and protocols, we can start looking at different middleware layers that can overlay the basic network to produce a consistent paradigm for applications and users.

The original paradigm behind the Web was quite simple: every computer can hold one or more documents, called Web pages. Each Web page contains text, images, icons, sounds, movies, and the like, as well as hyperlinks (pointers) to other Web pages.

File-System-Based Middleware

The basic idea behind the Web is to make a distributed system look like a giant collection of hyperlinked documents. A second approach is to make a distributed system look like a great big file system. In this section we will look at some of the issues involved in designing a worldwide file system.

Trasfer Model

The first issue is the choice between the upload/download model and the remote-access model. In the former, shown in Fig. 8-33(a), a process accesses a file by first copying it from the remote server where it lives. If the file is only to be read, the file is then read locally, for high performance. If the file is to be written, it is written locally. When the process is done with it, the updated file is put back on the server.