OPERATING SYSTEM

Lecture 8

Synchronization

Dr Victoria Mukami

INTRODUCTION

This lecture continues with the concurrency topic but with a review of synchronization in computers. We focus on concurrent processes, parallel processing, multiprocessing configurations, and how multi-core processors work.

Learning objectives

By the end of this topic, you should be able to:

1. Review the role of parallel processing within an operating system.
2. Understand multi-core processing and how multi-core processors work

OVERVIEW

So far, we have concentrated on processing using one CPU. The last lecture was a review of deadlock and starvation. We mainly looked at the conditions that cause deadlock, ways in which to solve when a deadlock occurs and how to prevent deadlocks. The lecture was also a review of starvation, where processes or jobs do not get the resources that they require to move on to processing. This lecture focuses on processing and more so multi-processing. In today's world, computers are equipped with multiple processors mainly in the form of cores that work independently or as one. Before we begin, let us first define some terms.

Multiprogramming: This is the management of several processes within a single processor system [2].

Multiprocessing: This is the management of several processes within a system with multiple processors [2].

Distributed processing: This is the management of several processes that are executed on a multiple, distributed computer system [2].

Parallel Processing: This is where several processors within one system work on related or unrelated jobs [1].

PARALLEL PROCESSING

Let us first do a thorough review of parallel processing. So as seen from the definition above, parallel processing concerns itself with a scenario where a system has several processors either working on the same job or different jobs at the same time. It is the work of the process manager to ensure that jobs are coordinated between the different

CPUs to ensure that each processor is not idle. There are two main benefits of parallel processing [1].

Increased Reliability [1]: This comes mainly from the fact that more than one CPU can share the processes and can also serve as a backup [1]. Ideally, if a CPU failed, then the other CPUs can pick this CPUs workload and share it to ensure continuity. This is only made possible by the operating system, and it will restructure the available resources and allocation strategies to ensure that the remaining processors take over the work of the failing CPU.

Increased Processing Speed [1]: This is achieved when some instructions can be processed in parallel. This is when the system can allocate several CPUs to perform a set of instructions separately and later combine the results at the end of the job. Some systems, on the other hand, allocate a CPU to each job while other systems allocate a CPU to parts of a job/process.

Synchronization is key for any multiprocessing system's success as many things can go wrong [1].

Levels of Multiprocessing

Multiprocessing takes place at different levels and each of these levels requires different types of synchronization which include job level, process level and thread level. The job level requires little to no synchronization once the jobs are assigned to the processor. Within the job level, each job has its processor, and all the processes and threads are run by the same processor [1].

Within the process level, a moderate level of synchronization is required. Any unrelated processes no matter the job can be assigned to any available processor [1]. For the thread level, a high degree of synchronization is required to track each process and thread. This is because threads are allocated to any available processes regardless of the job or process [1].

MULTIPROCESSING CONFIGURATIONS

There are three types of multiprocessing configurations: master/slave, loosely coupled and symmetric [1]. These configurations are dependent on the configuration of the multiple processors.

Master/Slave Configuration

This is an asymmetric multiprocessing system. There is the core processor and the slaves. The core processor is known as the master and manages other sub-processors known as slaves. The master processor oversees the main memory and the input/output devices. The master processor receives the processes from the main memory and redirects them to the slave processors. It additionally, picks any i/o requests from the slave processors. It will check the status of the processes in the system, perform storage activities, schedules the other processor's work and executes all the processes [1]. The slave processors can send output back to the memory, but any allocation is handled by the master processor. The main advantage of the master-slave configuration is its simplicity. The following are the disadvantages of the master/slave configuration [1]

- It is not any more reliable than a single processor as everything is dependent on the master processor. If it fails, then none of the other processors can take over.
- It can lead to poor use of resources since the slaves can stay idle when the master-slave is busy.
- Several interrupts can lead to an overhead of resources as the interrupts from the slaves can lead to long queues within the processor.

Loosely Coupled Configuration

The loosely coupled configuration has a complete system where each processor has access to its memory and input/output devices. Each processor can communicate with the other processor. Each job is allocated to a specific processor that takes care of its processing till completion. Each processor has global tables indicating where each job is located [1]. The biggest drawback to this configuration is that it is hard to tell when a processor fails. On the other hand, in case a processor fails, the rest can take over a load of that processor.

Symmetric Configuration

This system features a decentralized processor scheduling [1]. Based on this system, a single copy of the O/S and a global listing of each job and its status are stored within memory. This is shared and accessed by all the processors. Additionally, all the

processors share the I/O devices. This configuration also known as the tightly coupled configuration has some advantages over the loosely coupled configuration [1].

- It is reliable
- Uses resources effectively
- Balances load well
- Degrades gracefully in case there is a failure.

The biggest drawback is that this configuration requires a high degree of synchronization to avoid deadlocks [1].

PROCESS SYNCHRONIZATION

Synchronization is necessary, especially in a bid to avoid starvation or jobs getting skipped due to issues during placement. Synchronization sometimes gets implemented as a lock and key arrangement before any keys can get into a critical region. A critical region ensures that a process does not get mixed and loses integrity in its operation. Before a process can work on a critical region the process must get the key. Once it has the key then it can work, and all other processes are locked out until it releases the key after completion [1]. Several locking mechanisms exist and are discussed.

Test and Set

This is a machine instruction that was introduced by IBM. During a single-machine cycle, the system will test to check the availability of a key. Basing it on binary language, a key can be a 0 representing free or a 1 representing busy. A process can only enter the critical region when it finds a 0 otherwise it waits until it successfully tests the condition and receives a 0. There are two major drawbacks to this configuration.

1. Starvation can occur if processes enter using a random order [1].
2. The waiting processes are unproductive and cause a major overhead due to the resources used when checking whether the key is available.

Wait and Signal

This is a modification of the test and sets configuration. In wait and signal, the wait is activated when a process encounters a busy condition [1]. The process is set into the

blocked status and links it to a queue of other waiting processes. Processes for execution are selected by the process scheduler. The signal is activated whenever a process exits the critical region. One process within the processes queue is selected and set into the ready state. The process scheduler chooses that process. There is a lower overhead in this method as the processes do not keep requesting for the free state.

Semaphores

This is a "non-negative integer variable that can be used as a binary signal - a flag [1]". A semaphore is either set as on or off that is a 1 or a 0 respectively. This is known as a binary semaphore. A semaphore is an algorithm where processes cooperate through signals. A process can send or receive a signal via semaphore.

SUMMARY

This lecture was a review of synchronization in computers. We focused on concurrent processes, parallel processing, multiprocessing configurations, and how multi-core processors work.

DISCUSSION TOPIC

Identify the different types of synchronization mechanisms that may work on a Windows 10 computer with quad-core processors. What might be different between that machine and a dual-core computer?

REFERENCES

[1] McHoes, A., & Flynn, I., Understanding Operating Systems. Boston: Cengage Learning, 2018

[2] Stallings, W., Operating Systems: Internals and Design Principles. Harlow: Pearson Education Limited, 2018.