CS 3307-01 – Operating System 2 – AY2023-T5  
Assignment Unit 2

Instructor: Rupali Memane

University of the People

Jun 2023

Parallel Computing

## Concurrent computing

Concurrent computing is a computation type in that we split a computation work into (often dependent) parts and then distribute them to processing units for simultaneous execution. Processing units in this context are ***threads in the same processor core***. Therefore, this also is called ***multi-threaded*** programming, where a running program (i.e., a process) runs on more than one thread. A thread can be physical (hardware threads, often having two per core) or logical (software-managed). Threading libraries often provide Thread API, and we rarely care which type of thread we are using unless we use a low-level programming language. The purpose of concurrent computing is virtualization as if having more simultaneously running processing units than actual.

## Concurrency - Shared resources, critical sections, race conditions, indeterminate, deterministic, and mutual exclusion

In a multi-threaded program, different threads often access or change the same variables, the ***shared resources***. Unfortunately, this can cause bad behavior, the data inconsistency; a typical example is to increase an integer variable from two or more threads.

Suppose the value of variable i is 1, Thread #1 will increase i by 1, and Thread #2 will increase i by 2. We expect the value of i will be four after they finish.

|  |  |
| --- | --- |
| **Thread #1** | **Thread #2** |
| load var i (1) | load var i (1) |
| increase by 1 | increase by 2 |
| write 2 to var i | write 3 to var i |

Either Thread #1 or Thread #2, who completes first, the value of i is incorrect.

So, the code section, where multiple threads manipulate the share resources, is called a ***critical section***. The context of more than one thread concurrently running through one critical section is called a ***race condition***. A race condition leads to an ***indeterminate*** result that is different from our expectation, the ***deterministic*** result (Arpaci-Dusseau, 2018). Thus, to prevent indeterminate results, we only let one run through a critical section at a time, called ***mutual exclusion***, the concern of concurrency.

## Parallel computing

Parallel computing is a computation type in that we split a computation work into independent parts and then distribute them to processing units for execution in parallel. Note that processing units in this context are ***threads in different cores*** either in a ***multi-core processor*** or a ***multi-processor*** machine. Since parts are not dependent on each other, no race condition occurs. In parallel computing, the purpose is to reduce the time to complete the work, and the main concern is how to split and distribute parts efficiently.

Nowadays, most computers are multi-core or even multi-processor. Thus, we use parallel computing every time we work with our computers. Indeed, parallel computing doesn't limit to a specific field. We can see them in space science, biology science, AIs, and everywhere else.

Although all popular OSes support parallel computing, some Linux distros, such as RHEL or Debian, are the most suitable for works that heavily depend on parallel computing since they concentrate on this instead of GUI and others functionality (Red Hat, n.d.).

References

Arpaci-Dusseau, R. H., & Arpaci-Dusseau, A. C. (2018). *Operating Systems: Three Easy Pieces*. Arpaci-Dusseau Books. https://pages.cs.wisc.edu/~remzi/OSTEP/

Red Hat, Inc. (n.d.). *Red Hat Enterprise Linux operating system*. https://www.redhat.com/en/technologies/linux-platforms/enterprise-linux