Introduction to Parallel Programming:

Diagram

Description automatically generated

The diagram shows you a basic computer architecture diagram of **Von Neumann architecture** which any of our modern-day computers are more or less based on

Diagram

Description automatically generated

In any computer program, **I. instructions are stored in persistent memory or hard drive as executable file**. When you run it the instruction for the ***program*** will be *loaded* to CPU and ***basic data*** is *loaded* to the RAM. Then, the computing unit or the processor of our computer,

Diagram

Description automatically generated

will **start executing instruction** *while* **taking inputs at run time** and *the program* will **output the relevant results based on those inputs**.

***No matter how complex our application or program, it will go through the same process.*** Notice, according to this model ***we are executing instruction one after another (sequentially).***

But in modern day computers, we can run multiple application at once.

Table

Description automatically generated**Ex**: Open up my task manager windows, and you can see multiple applications like Visual Studio, multiple instances of Google Chrome browser, Microsoft word, and other application running at the same time.

To understand how a single core processor perform these simultaneous executions, we have to get to know the concept call ***process***.

***A process is an instance of a computer application***.

Chart

Description automatically generated with low confidenceWhen you **run an application**, ***instructions for that application will be fetched to the processor***

and

A picture containing arrow

Description automatically generated

***memory will be allocated in RAM to hold runtime resources***.

A picture containing arrow

Description automatically generated

This **running instance** refers to as a ***process***.

Other than the memory, an image of executable code or the instructions, there is an important thing called **Context** which is associated with the process.

**Context** is referring to **a collection of data about process**, which ***allows the processor to suspend or hold the execution of a process*** and ***restart the execution later***.

*Memory addresses and program counter states* are some of the **data contained** in the **Context**.

Arrow

Description automatically generated with medium confidenceWhen ***running a*** ***multiple application or processes simultaneously***, if we have only a *single core processor*, that processor *will execute instruction of these process* in a ***round-robin******fashion****.*

**Timeslots** will be allocated for **each process** depending on ***the readiness of particular***

A screenshot of a computer

Description automatically generated with low confidence***process memory*** and ***the priority of that process***. So, if we arrange the execution of the instruction in a processor in a timeline, one possible outcome would be look like this.

Here how colored each timeslot with same process color which is executing at that time slot.

So according to this convention the processor executes:

1. Instructions belong to ***process one*** in the ***first two time slots***.
2. Then the next ***time slot***, it executes the instructions belong to ***second process***
3. And on and on

This type of execution is possible due to the mechanism called **context switching**.

Using the context switching mechanism:

1. the processor can load the context of a process and
2. ***execute it from the state it hold the execution previously***, and then it can
3. ***suspend the execution of that process*** and
4. switch the context to another process, to
5. start the execution of that particular process.

We **call this type of execution** as ***round-robin fashion execution of instructions***.

Another important concept which is very similar to process is a thread.

Thread or thread of execution is the ***smallest sequence of a programmed instruction*** that ***can be managed independently***.

* **Thread is a component of a process**.
* ***Every*** ***process has at least one thread called main thread, which is the entry point for that process***.
* You can **assume thread as scale down version of process**.

One major distinction between threads and process is that within the same process *multiple threads* can ***co-exists*** and ***shared the memory between them*** (memory allocated for that particular process.)

But if we consider ***two processes***, there ***will not be any sharing of recourses between them***.

Diagram

Description automatically generatedEven in modern CPUs there are multiple cores available, which means multiple threads, or multiple process can execute simultaneously without context switching.

***Parallel processing is a type of simultaneous processing of multiple threads.***

**Let’s compare the previous to a real-world example:**

Table

Description automatically generated with medium confidenceImagine you are going to build a house and we have the following sub-tasks to complete it. We have to first plan and then we should start working. So, the plan is:

Graphical user interface, application, PowerPoint

Description automatically generatedNow let's arrange these tasks in a timeline.

Some tasks like planning and foundation laying may need to perform in sequential order or one after another. However, other tasks like *brick laying, plumbing, wiring, sewer pipe laying* can be done **simultaneously**.

**Ex**:

If the house has multiple rooms, some workers can work on plumbing some part of house while others can work on brick laying in another part of house and so on.

The idea is that in **almost every process** there are ***some parts*** which should ***perform sequentially*** and there is **some parts** which can ***perform simultaneously***. Hence, we will be able to finish our task quickly if we can perform task in parallel.

If we consider a computer application, it is also like our house building task.

**Some parts** of a program ***should be sequential*** and there are **some parts** which can be ***perform in parallel***.

But if we arrange our application code parallel and run it in ***single core processor*** there will not be much gain in performance. This is due to:

* In the **processor level** or as in **hardware level**, we still **execute** these instructions in ***sequential order*** with ***context switching***.

However, processors in modern PCs and laptops comes with multiple cores.

So, if **we know how to program our applications in parallel**, then ***we will be able to exploit the computation power of these modern hardware***.

There are two types of **parallelism**: **Task level parallelism** and **Data Level parallelism**.

**Ex**: In the house building task, we can **assign different workers** to ***perform different tasks*** **at the same time**. This type of **parallelism** is referred to as ***task level parallelism***.

**Ex**: We can **assign multiple people** for brick laying **at same time**.

Graphical user interface, application, Word, PowerPoint

Description automatically generatedWe can compare this to ***multiple threads*** which are ***executing same instruction*** but on ***different data***. This type of parallelism is referred to as ***data level parallelism***.

***Parallelism vs Concurrency***

Arrow

Description automatically generatedWe can run multiple applications simultaneously even we have single co-processor. In this case the **processor** will **accommodate** the **execution** with **context switching** and for us, this looked like a **parallel execution**. This type of simultaneous execution is referred to as **concurrency**. This is actually an **illusion of parallel execution** but in ***hardware level***, we ***execute instruction sequentially***.

**Parallelism** or **true parallelism** will happen if we have ***enough cores in our processor*** to ***run each of these process in different cores***. Then **all the process or threads will run in parallel**.

***HOWEVER, IN THIS COURSE WE WILL CONSIDER BOTH OF THESE ITEMS AS EQUAL.***