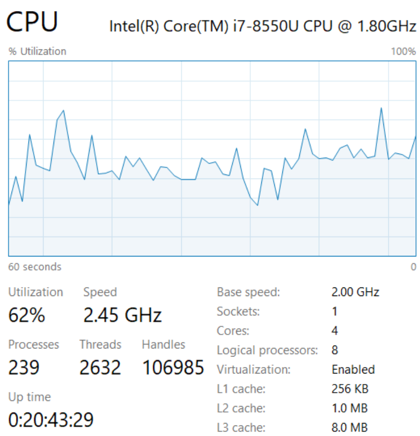
Assignment1: Report

Tasks:

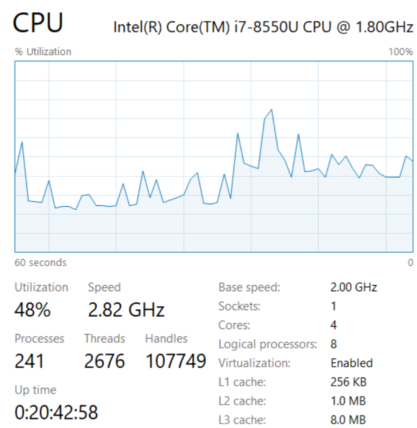
1. ‘Connect’ the check_prime function to the Pool processing function. Generate sets of work (numbers to be checked) to be processed by the pool. Quantify the speedup achieved with multiple cores (at least 2). **What lessons can be learned from these results?**

➔

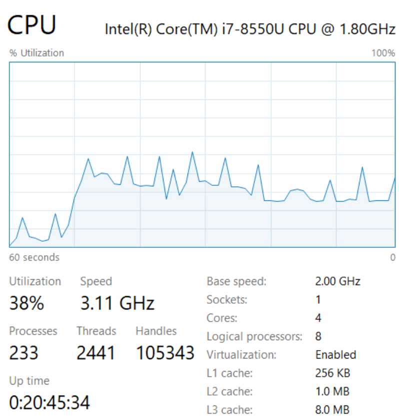
Multiprocessing is a suite that supports spawning processes using an API similar to the threading. Due to this, the multiprocessing module allows the programmer to fully leverage multiple processors on a given machine. Spawning extra processes introduces I/O overhead as data is having to be shuffled around between processors. This can add to the overall run-time. This object has a function called **map**, which takes the function we want to multi process and the list as arguments and then **iterates** through the list for that function.
In this scenario when the check_prime function (f) is connected to the pool processing function, the **check_prime** calculates whether input number in the data range provided is a prime or not. Then the time elapsed for the operation to execute is captured by the function. In Addition to this the **pool_proccesing** function allows the user to **allocate** the pool size or the number of processors against the operation. pool_process (f, data, pool_size):
Here the function must accept the external function for which the cores should run, the data (range of numbers) and number of physical cores allowed.



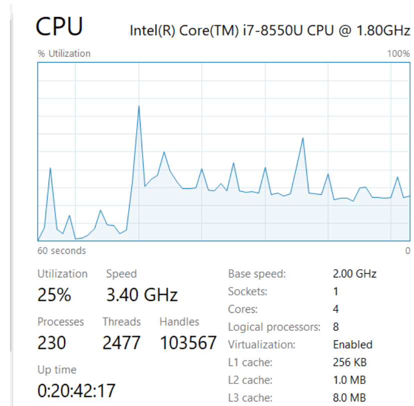
1 core



2 cores



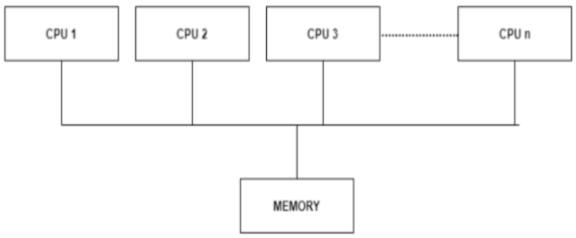
3 cores



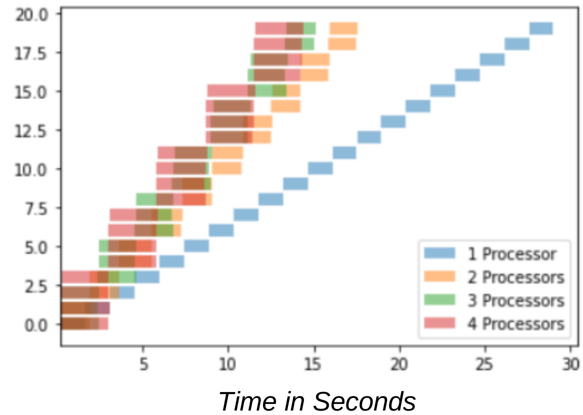
4 cores

Pool is most useful for **large** amounts of processes where each process can execute quickly, while **Process** is most useful for a **small** number of processes where each process would take a longer time to execute. Also, to improve performance of the PC **overclocking** may help to **decrease** the execution time from say **1.8GHz to 2.5GHz**. I see about a **40% decrease** in running time from **1 core to 2 cores**, but from **1 core to 4 cores** it is only **about 50%** so, if there are two cores being used for the multiprocessing, then chrome and other applications would use the other two free cores.

Parallel processing of tasks

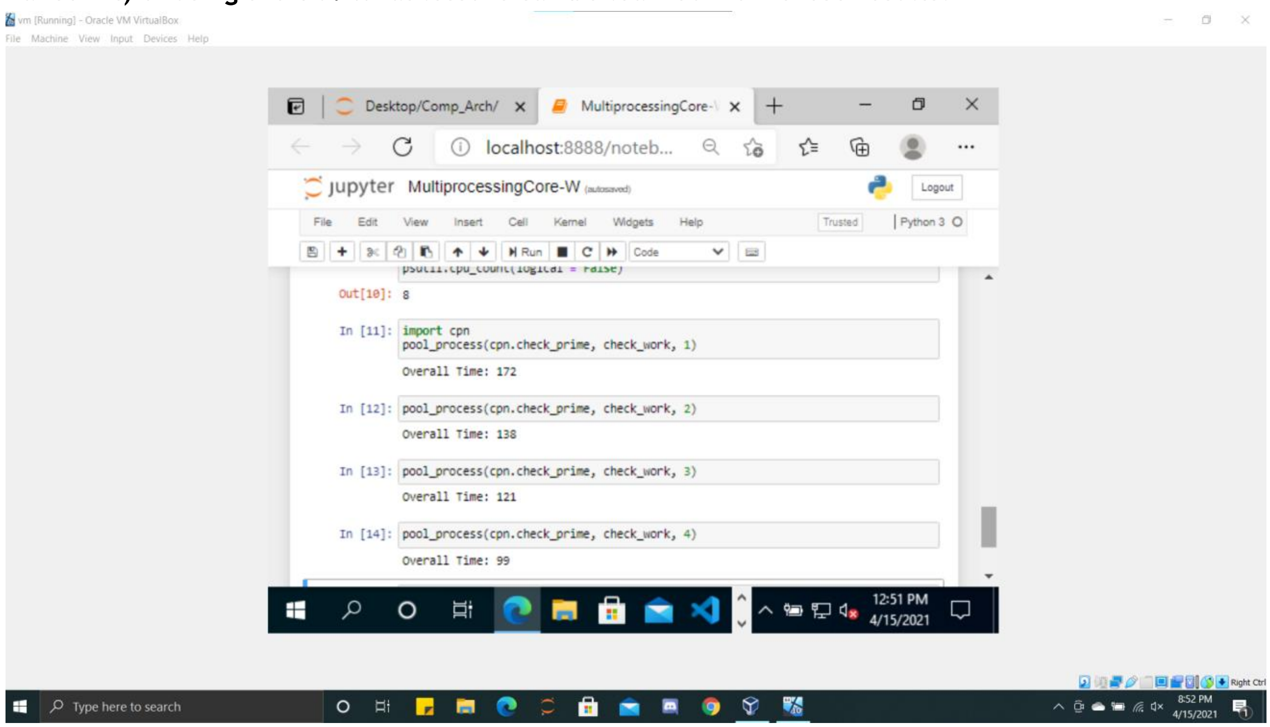


Multiprocessing (1 up to 4 cores)



- <u>Advantages:</u>
  i. More reliable
  ii. Enhanced Throughput
  iii. More Economic
- <u>Disadvantages:</u>
  i. Increased Expense
  ii. Can create deadlock.
  iii. Large Main Memory Required

Task 1: B. **Repeat the exercise in 1 running on a VM through VirtualBox and assess the impact (performance hit) of using the VM. What lessons can be learned from these results?**



In this scenario we have a configuration where 8 core processors, 16 GB of Random-access memory and a 500GB of SSD disk, running **64-bit** Windows 10 **Operating system**. On the other hand, just 1 virtual machine allowed to use all resources available. It depends on the **version** and memory along with processing capacity of the VMware. I personally configured it to **14GB** of RAM, activated all the **8** logical processors and keeping **64-bit** Windows OS. CPU virtualization adds varying amounts of **overhead** depending on the **workload** and the type of virtualization used. This overhead takes CPU processing time that the application itself can use. CPU virtualization overhead usually translates into a **reduction** in overall **performance**. **Emulation** adds an additional **60-70** microseconds to the I/O path, the bare metal SSD I/O **latency** is about 40 microseconds, so virtualization doubles or triples the **storage** I/O latency. So **Yes**, a virtualized environment is **slower** than a **native** system and that may be in a range of **50 to 80%.**