Lab 1 - Cloud & Infrastructure Services

Alexander Stradnic - 119337263

Introduction

In this lab I took a matrix multiplication script written in Python and compared its execution time on my home computer running on Windows, on a VM running Ubuntu 20, and in a Docker container using the WSL Linux Kernel.

Concepts

Virtual Machine (VM)

A Virtual Machine is a piece of software that behaves like an independent computer. It has its own operating system, disk space, and resources that, while not real resources, appear as such to the virtual machine.

- A virtual machine allows you to run multiple compartmentalised “computers” on the same physical computer and host Operating System.
- It allows one to experiment with other operating systems and test pieces of software, without affecting the Host OS.
- They can be started and stopped at any time, without affecting other VMs or the host, and can be allocated a set amount of computing resources, which can be shared with other VMs.
- A piece of software known as a hypervisor is used to manage virtual machines and act as the interface between hardware and software, which can be run on bare metal, or on a Host OS.

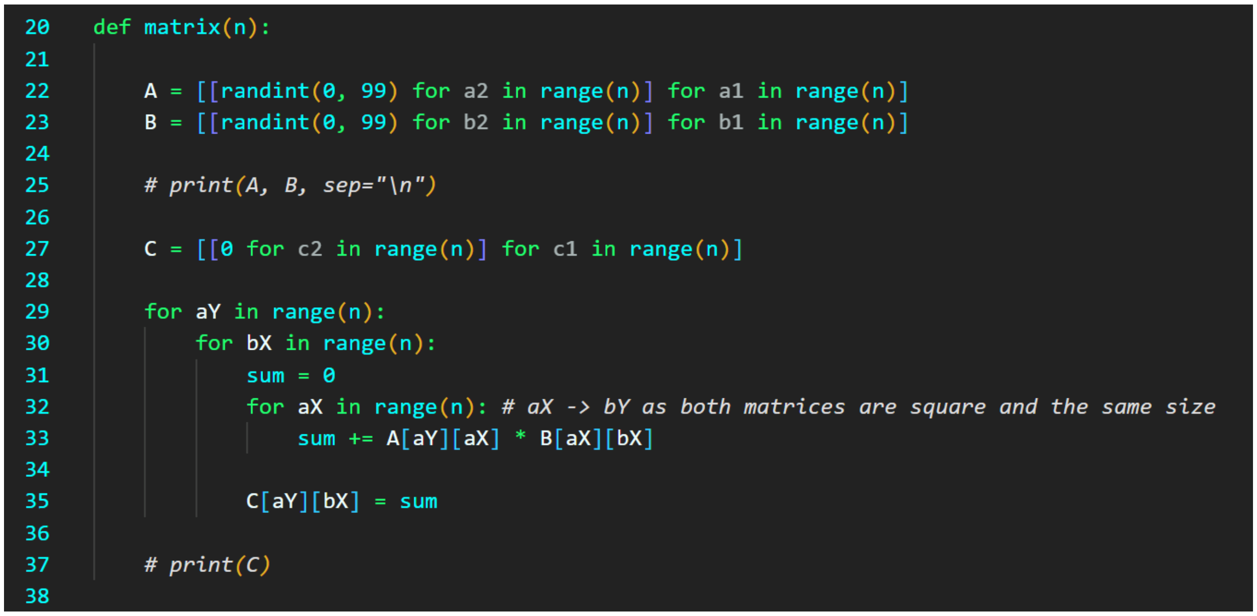
(Docker) Container

A Container is a packaged piece of software that can be executed in its own isolated environment. They typically house an application, such as a middleware.

- Containers run on a Host Kernel. They are assigned computing resources by the hypervisor and Docker, and are less independent than Virtual Machines
- To create a Container, an Image must first be created, containing the instructions on what must be put into the Docker Container, as well as passing configuration details such as assigning a Container set ports. A Docker Container can then be created from this Image.

- Multiple Containers can be managed and run together in clusters, where they can communicate with each other and load balance to parallelise high workloads
- Using a cluster management software such as Kubernetes, Containers can be spun up or shut down as needed. If an error occurs with a Container, an identical container can be started up from the base image.

The Algorithm



```python
def matrix(n):

    A = [[randint(0, 99) for a2 in range(n)] for a1 in range(n)]
    B = [[randint(0, 99) for b2 in range(n)] for b1 in range(n)]

    # print(A, B, sep="\n")

    C = [[0 for c2 in range(n)] for c1 in range(n)]

    for aY in range(n):
        for bX in range(n):
            sum = 0
            for aX in range(n): # aX -> bY as both matrices are square and the same size
                sum += A[aY][aX] * B[aX][bX]

            C[aY][bX] = sum

    # print(C)

```

My Performance Expectations

I would expect the program to run the fastest on bare metal, which in this case is the base Operating System. This is because it would be a direct command to the Kernel and there should be less overhead.

I would expect the Docker Container to be second fastest. This is because a container would have a bit of resource overhead from Docker. However, as it still goes through the same kernel as the host OS (or in this case through Windows Subsystem for Linux).

I would expect the VM to be last in place as it would have the most overhead, due to running a virtual file system and Guest Operating System, on top of the hypervisors (both hardware and software).

Results

These times were all calculated on the same machine running an Intel Core i5 4690k at 3.5GHz with 16GB of DDR3 RAM, a Radeon RX580 8GB, and off of a Sabrent M.2 SSD.
The times below show how much time it took for each to calculate 1-dimensional to 50-dimensional matrix multiplications, then 1-100 dim. multiplications, then for 1-150 dim. Multiplications.

The Host Operating System used was Windows 10 Version 10.0.19042 Build 19042.

Windows Results (Python ^3.9.2):
matrices(50): 0.228705s
matrices(100): 3.233922s
matrices(150): 15.332179s

VirtualBox was given 1 CPU core, with 2048MB of RAM.

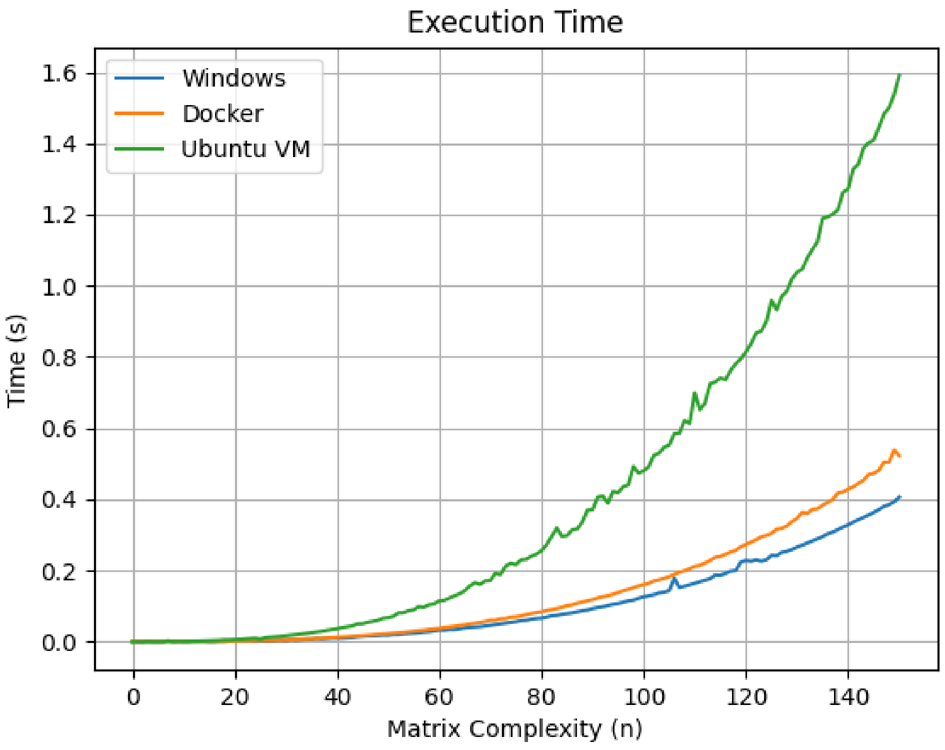
VirtualBox Ubuntu VM Results (Python ^3.8.10):
matrices(50): 0.900275s
matrices(100): 12.938741s
matrices(150): 61.35649s

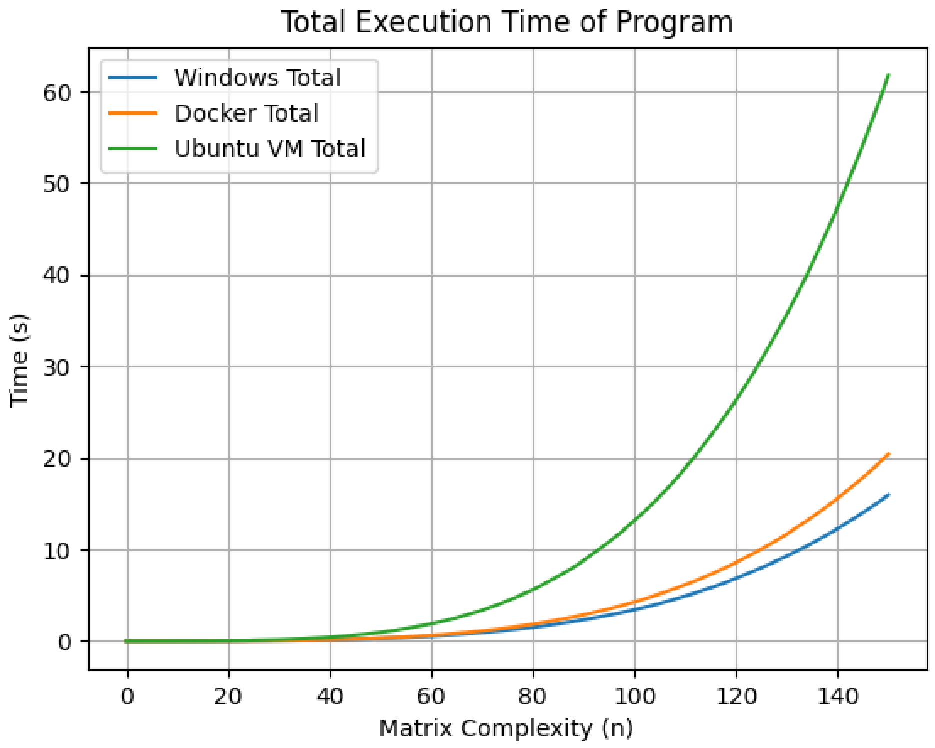
The Dockerised script used the *python:latest* image and ran through Windows Subsystem for Linux 2's Linux Kernel.

Docker on Windows using WSL2 (Python ^3.8.10):
matrices(50): 0.295389s
matrices(100): 4.117550s
matrices(150): 20.034180s

Graphs

The first graph shows the increase in time taken in each environment to calculate the multiplication of two matrices of size n, as n increases. The second graph shows the total time increase per iteration of matrix().


Execution Time
Windows
Docker
Ubuntu VM
Time (s)
1.6
1.4
1.2
1.0
0.8
0.6
0.4
0.2
0.0
0
20
40
60
80
100
120
140
Matrix Complexity (n)


Total Execution Time of Program
Windows Total
Docker Total
Ubuntu VM Total
Time (s)
60
50
40
30
20
10
0
0
20
40
60
80
100
120
140
Matrix Complexity (n)

Conclusion

In conclusion, from the graph I conclude that the overhead in the VM was significant enough to have a sizeable effect on the runtime of the program. It did indeed run the quickest on the base operating system.

However each solution should be used as seen fit, VMs offer a level of flexibility and freedom that one may not be able to achieve with containers or bare metal, for example.