Matrix Addition and Multiplication in AWS

Cloud Computing

Computational and Software Techniques in Engineering MSc

Authors:
Ayush Maria - s349256

Module Leader:
Dr. Stuart Barnes

Cranfield University
School of Aerospace, Transport and Manufacturing

September 7, 2023

Contents

List of Figures

List of Tables

Abstract

Cloud computing is the on-demand use of computer resources housed in a distant data centre and controlled by a cloud services provider, such as programs, servers (both physical and virtual), data storage, development resources, networking facilities. [10] Through this study a distributed cloud environment was created using Amazon Web Services (AWS) EC@ instances. Matrix Multiplication and Addition of dense matrices were carried out using the Message Passing Interface (MPI) and the results were noted. It was observed that the solution created is cost effective but lacks the computing power to carry out Matrix Multiplication, while it efficiently handles Matrix Addition.

Introduction

1.1 Matrix Addition

Matrix addition is one of the most fundamental operations involving matrices first introduced by author Cayley. [3]

It has uses in multiple fields like solving linear equations for complex problems in engineering or calculation of the Gross Domestic Product in economics.[9]

1.2 Matrix Multiplication

The French scientist Jacques Philippe Marie Binet devised matrix multiplication in 1812 to show the synthesis of linear plots represented by matrices. Matrix multiplication is thus a fundamental tool of linear algebra, having many applications in engineering, statistics, math, finance, and medicine. Matrix product calculation is an essential operation in all linear algebra related computational applications. [7]

If M is a i × j matrix and N is a j × i matrix, the matrix product P = MN (unaccompanied by multiplication symbols) is described to be i × i matrix. Figure **??** gives birth to a little understanding of the procedure.

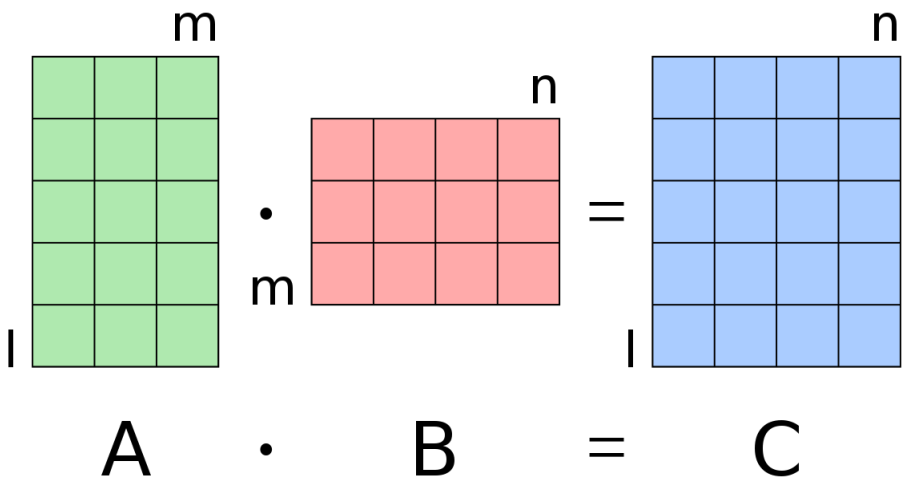


Figure 1.1: Matrix Multiplication

Since it is considered as the key tool for numerous other computational processes in different areas, such as earthquake analytics, data processing, interstellar simu-

lations, aerodynamic calculations, electronic signal calculations, and so on, matrix multiplication is indeed very significant in many science disciplines. [8]

1.3 MPI

Message Passing Interface (MPI) is a portable message carrying standard that allows the creation of parallel softwares, libraries and programs. [6]

The standard defines the semantics and syntax of a core set of library routines that will be beneficial to a wide variety of users who are creating portable message passing applications in programming languages like Fortran, C, C++, Java and Python. MPI is also used as a target by compilers for languages such as High Performance Fortran. MPI implementations in the corporate and open domain also exist. These run on both closely connected hugely parallel machines and the cloud. [11]

MPI is designed to be the existing norm for message passing interface for parallel processing and library development. MPI's fundamental substance is point-to-point interaction between processes and collective interaction among groups of processes. MPI also has more complex message passing features that allow users to modify process groups, offer topological architecture for process groups, and aid in the development and usage of parallel libraries. [4]

1.4 Amazon Web Services(AWS) - Cloud

Figure 1.2: Matrix Multiplication

Most businesses are embracing the freshly developed paradigm of cloud computing. It encompasses the idea of on-demand services, which entails employing cloud resources whenever needed and scalability of resources in response to demand. Without a doubt, cloud computing offers countless advantages and is a practical approach. AWS

is amongst the most reputable cloud computing suppliers, offering superior cloud security in addition to sublime cloud services. [12]

"Elastic Compute Cloud" (EC2), "Simple Storage Service" (S3), and "CloudFront", the Content Delivery Network (CDN), are arguably most popular Amazon cloud services. [2]

Scalable computing power is offered by Amazon Elastic Compute Cloud (Amazon EC2) in the Amazon Web Services (AWS) Cloud. By using Amazon EC2, a user may create and deploy apps more quickly because they won't need to make an upfront hardware investment. They may deploy as virtual instances as they require, set up networking and security, and control storage using Amazon EC2. The user no longer has to predict traffic since Amazon EC2 allows them to scale up or down to address changes in demand or spikes in demand. [2]

Methodology

2.1 Numerical Methods

2.1.1 Matrix Addition

Matrix addition can be performed by adding the relevant matrix elements. Two or even more matrices that have the same dimensions can always be added.

If P = $[p_{ij}]$ and Q = $[q_{ij}]$ are two matrices of the same size, that is, they contain the same amount of rows and columns, then

$$P + Q = [p_{ij}] + [q_{ij}] = [p_{ij} + q_{ij}] \tag{2.1}$$

This is highlighted in 2.1 below.

$$\begin{aligned}\mathbf{A} + \mathbf{B} &= \begin{bmatrix} a_{11} & a_{12} & \cdots & a_{1n} \\ a_{21} & a_{22} & \cdots & a_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} & a_{m2} & \cdots & a_{mn} \end{bmatrix} + \begin{bmatrix} b_{11} & b_{12} & \cdots & b_{1n} \\ b_{21} & b_{22} & \cdots & b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ b_{m1} & b_{m2} & \cdots & b_{mn} \end{bmatrix} \\ &= \begin{bmatrix} a_{11} + b_{11} & a_{12} + b_{12} & \cdots & a_{1n} + b_{1n} \\ a_{21} + b_{21} & a_{22} + b_{22} & \cdots & a_{2n} + b_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ a_{m1} + b_{m1} & a_{m2} + b_{m2} & \cdots & a_{mn} + b_{mn} \end{bmatrix}\end{aligned}$$

Figure 2.1: Matrix Addition

2.1.2 Matrix Multiplication

A few notions from vector algebra can be used to describe the idea of the multiplication of two matrices as well as many other operations on matrices. In three-dimensional Euclidean space, the scalar multiplication of both the (position) vectors i and j is a crucial and practical concept.[8] It has the following characteristics:

$$\vec{i} \cdot \vec{j} \doteq \left|\vec{i}\right| \left|\vec{j}\right| \cos\alpha \tag{2.2}$$

where α is the angle between the above mentioned vectors. It is obvious that if and only if i and j are orthogonal, the scalar multiplication of nonzero vectors i and j is equal to zero. Possessing the coordinates of the vectors i and j, i1, i2, i3, and j1, j2, j3, makes finding the scalar product simple.

$$\vec{i} \cdot \vec{j} = i_1 j_1 + i_2 j_2 + i_3 j_3 \tag{2.3}$$

The concept of identifying the multiplication of row i and column j (in such an order) as the sum on the right in a matrix is the foundation of matrix multiplication as referred in equation 2.3. Thus,

$$\mathbf{i}^\top \mathbf{j} = \begin{pmatrix} i_1 & i_2 & i_3 \end{pmatrix} \begin{pmatrix} j_1 \\ j_2 \\ j_3 \end{pmatrix} \doteq i_1 j_1 + i_2 j_2 + i_3 j_3 \tag{2.4}$$

This is recognised as "row x column" multiplication.[8]

In contrast to equation 2.2, equation 2.4 lends itself readily to substantial additions that will be highly advantageous.

- It is not necessary for the elements of i and j to be actual numbers; they might originate from any field.
- For each positive integer n, the vectors can each contain n items.

Therefore, if i and j are both n-membered vectors (the components could be real or complex integers),

$$\mathbf{i}^\top \mathbf{j} = \begin{bmatrix} i_1 & i_2 & i_3 \end{bmatrix} \begin{bmatrix} j_1 \\ j_2 \\ j_3 \end{bmatrix} \doteq \sum_{k=1}^{n} i_k j_k \tag{2.5}$$

Let I be a matrix containing p rows $I_a^\top$ $(l \leq a \leq p)$ with length l (in a way that I is p x l) and J be a matrix containing q columns $J_b^\top$ $(l \leq b \leq q)$ with equal length l (in a way that Q is l x q). The product of I and J is therefore construed as the p x q matrix $K = [K_{ab}]$, where

$$\mathbf{k}_{ab} = {i_a}^\top \mathbf{j_b} \qquad (1 \leq a \leq p, \quad l \leq b \leq q) \tag{2.6}$$

Hence,

$$\mathbf{I}J = \begin{bmatrix} i_1^\top \\ i_2^\top \\ . \\ . \\ . \\ i_m^\top \end{bmatrix} \begin{bmatrix} j_1 & j_2 & . & . & . & j_m \end{bmatrix} \doteq \begin{bmatrix} i_1^\top j_1 & i_1^\top j_2 & . & . & . & i_1^\top j_m \\ i_2^\top j_1 & i_2^\top j_2 & . & . & . & i_2^\top j_m \\ . & . & & & & . \\ . & . & & & & . \\ . & . & & & & . \\ i_m^\top j_1 & i_m^\top j_2 & . & . & . & i_m^\top j_3 \end{bmatrix} \equiv [i_a^\top j_b]_{a,b=1}^{p,q} \quad (2.7)$$

Using 2.5, the following formula is obtained,[8]

$$IJ = [\sum_{c=1}^{l} i_{ac} j_{cb}]_{a,b=1}^{p,q} \quad (2.8)$$

2.2 Computational Architecture - Cloud (Windows)

2.2.1 Instance Type Pricing

The following instances described by figure 2.2 were considered for the distributed system created.

Viewing 536 of 536 available instances

t2 1

Instance name	On-Demand hourly rate	vCPU	Memory	Storage	Network performance
t2.xlarge	$0.1856	4	16 GiB	EBS Only	Moderate
t2.2xlarge	$0.3712	8	32 GiB	EBS Only	Moderate
t2.large	$0.0928	2	8 GiB	EBS Only	Low to Moderate
t2.micro	$0.0116	1	1 GiB	EBS Only	Low to Moderate
t2.medium	$0.0464	2	4 GiB	EBS Only	Low to Moderate
t2.small	$0.023	1	2 GiB	EBS Only	Low to Moderate
t2.nano	$0.0058	1	0.5 GiB	EBS Only	Low

Figure 2.2: AWS Pricing

Two separate instances were considered for the master node,

- t2.small: $0.023
- t2.micro $0.0116

The t2.nano was discarded because it provided little computation power while the instance types like t2.medium and above were discarded because of cost related reasons. The difference between the cost of deploying a t2.small and a t2.micro instance was found out to be a mere $0.0007/hr, while providing double the computational power of 2 GiB RAM. Ultimately, the t2.micro instance was chosen because 1 GiB of RAM was deemed enough to perform the operations required by this study. The t2.micro instance was further used for each of the worker node because of the amount of computational power available at the price mentioned above. The overall costs incurred will further be presented in the results section.

2.2.2 EC2 Instance Creation

The following steps were followed in order to create the Master node and the subsequent worker nodes. [1]

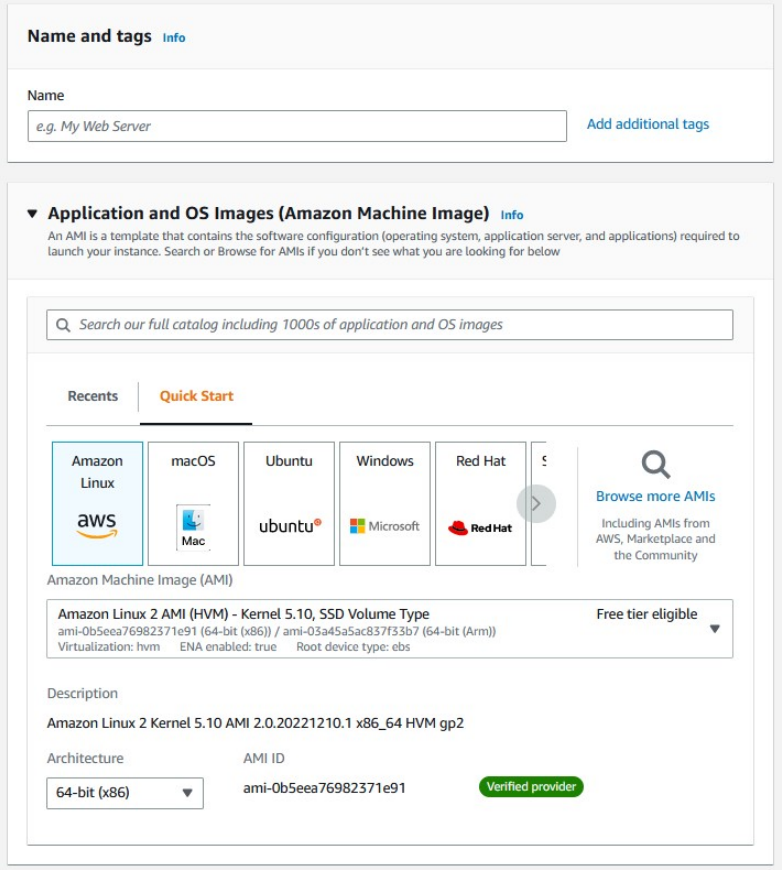


Figure 2.3: Selecting the AMI

- Select the 'Launch Instance' button
- Create a name for the Instance, for example, node1
- Select the Amazon Machine Image (AMI), for example, Amazon Linux 2

- Select the Instance type, for example, t2.micro, t1.micro etc
- Select the Security Group Rules

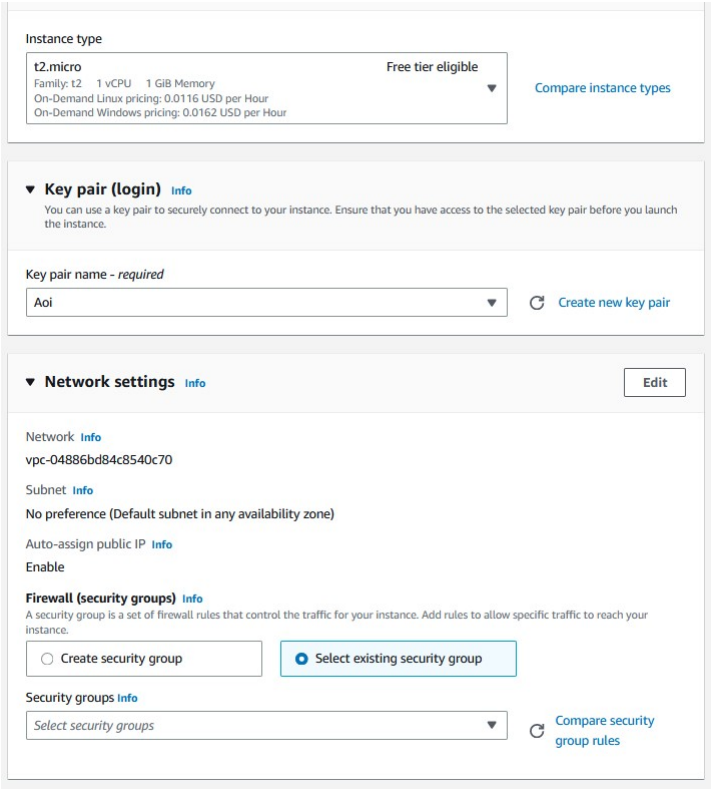


Figure 2.4: Selecting the Instance Type and Security Rules

- Select the storage required for the Instance
- Launch the Instance

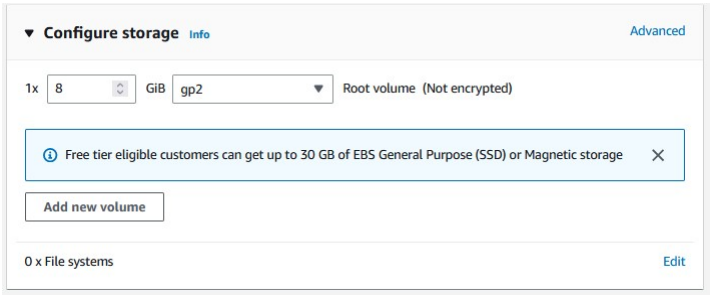


Figure 2.5: Selecting Storage

A total of 5 nodes were created with 1 master and 4 worker nodes, all of which were of type - t2.micro

2.2.3 Configuring the Virtual Cluster - Windows

The following steps were followed in order to connect the Master Node to the subsequent Worker Nodes. The steps followed on the Master node were distinct to that of the worker nodes.[5]

Steps followed on the Master Node

- Navigate to the .ssh folder in the ec2-user directory and create a new key pair using,

  ssh-keygen -t rsa

- Enter the filename, for example, id_rsa
- Enter the passphrase
- Copy the contents of the id_rsa.pub file created

Steps followed on the Worker Nodes

- Edit the sshd_config file using,

  sudo nano /etc/ssh/sshd_config

- Add the following lines or uncomment them if they exist in the file,

  RSAAuthentication yes

  PubkeyAuthentication yes

- Navigate to the following path and append the copied contents of id_rsa.pub to authorized_keys

  /home/ec2-user/.ssh/authorized_keys

To automate the access to each node a shell script was created. This script uses simple looping procedures to loop over the internal-ip address of each node while connecting to them using the **scp** command. These steps establish the inter-connectivity of the master node to the worker nodes involved.

Implementation

In order to run MPI on the cluster, each node required the MPI library installed on them along with the appropriate files added to their respective paths. The MPI programs were implemented in C language, with a minimum matrix size of 1000 x 1000.

3.1 Programming

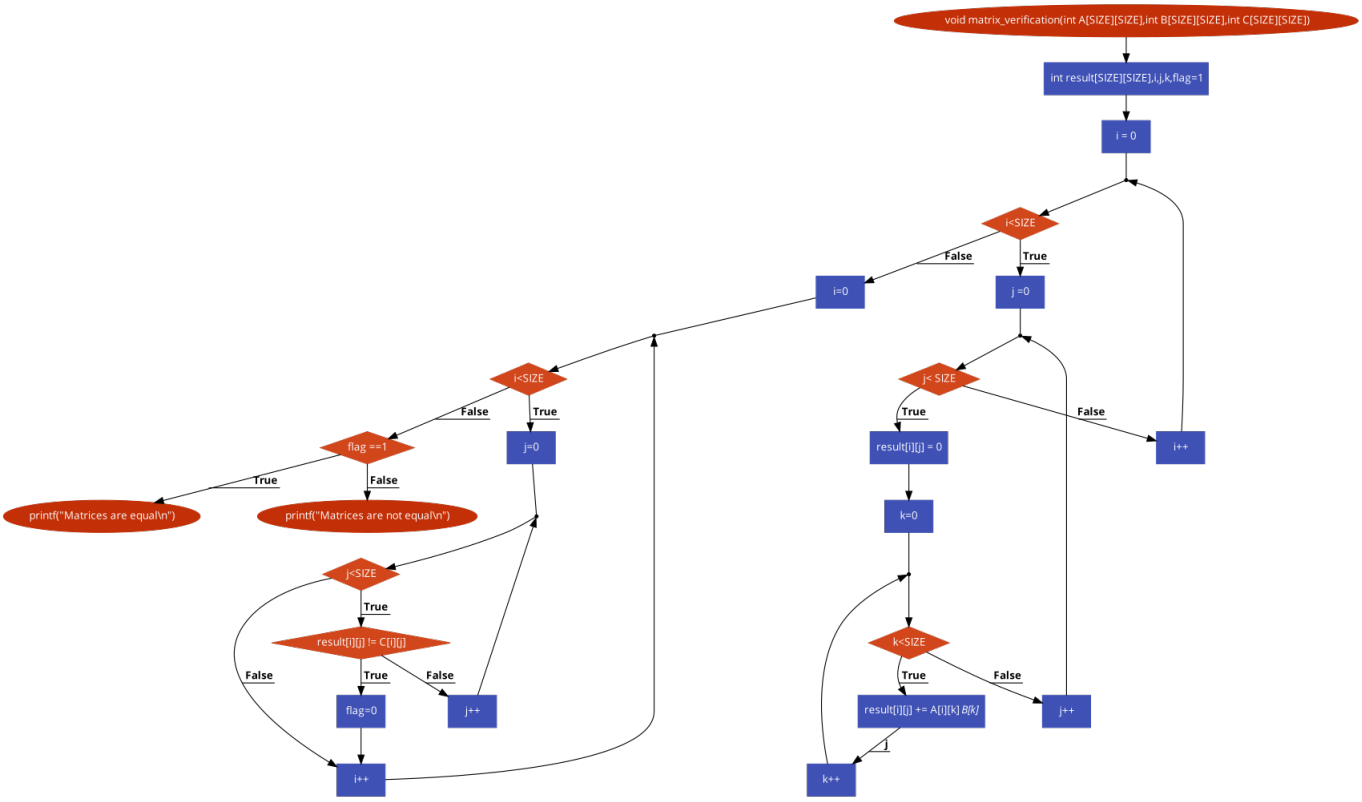


Figure 3.1: Matrix Multiplication Flowchart

The matrix addition and matrix multiplication programs are implemented with the help of three distinct functions, main, create matrix, verify result. The main function is responsible for sending and gathering the data from the worker nodes along with invoking the other two functions. The create matrix function is responsible for creating the matrix. The verify matrix function verifies whether the output of the parallel multiplication matches that of the single node multiplication.

3.1.1 Matrix Verification and Calculation Monitoring

The flowchart of the function used to verify the output of the parallel matrix multiplication and addition programs are provided by figures 3.1 and 3.2 respectively. By comparing simple single node calculations these functions ensure that the solution is correct by printing, "Output Verified", on the screen as exhibited by figure 3.3.

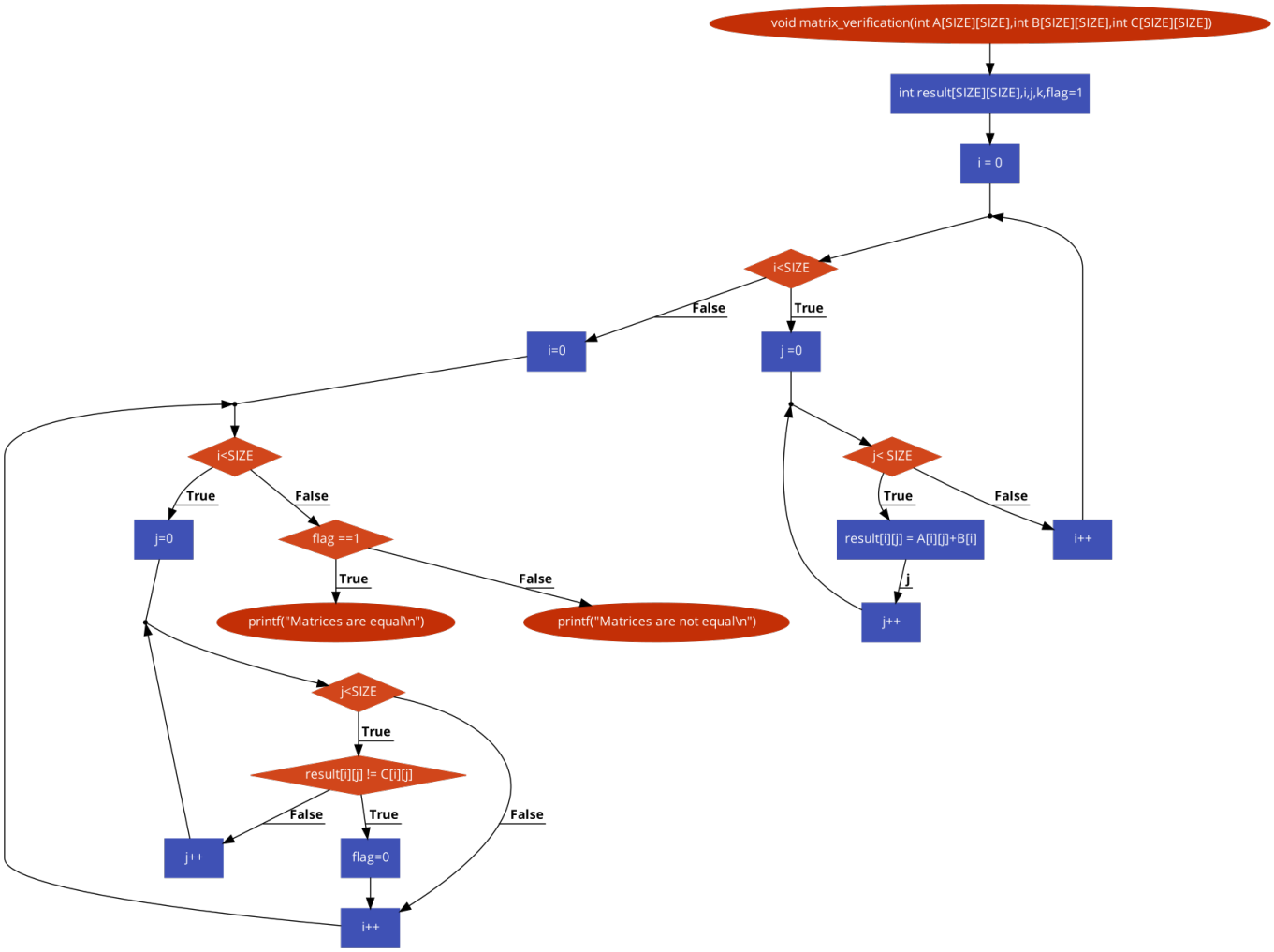


Figure 3.2: Matrix Addition Flowchart

Furthermore, there is also a provision in code that allows the user to monitor the progress of the calculation by printing out the location of each package sent to the distributed worker nodes as depicted in figure 3.3. As it can be seen in the figure, the IP address of the node, along with the contents of package on that node can be seen by the user when the calculation is being carried out.

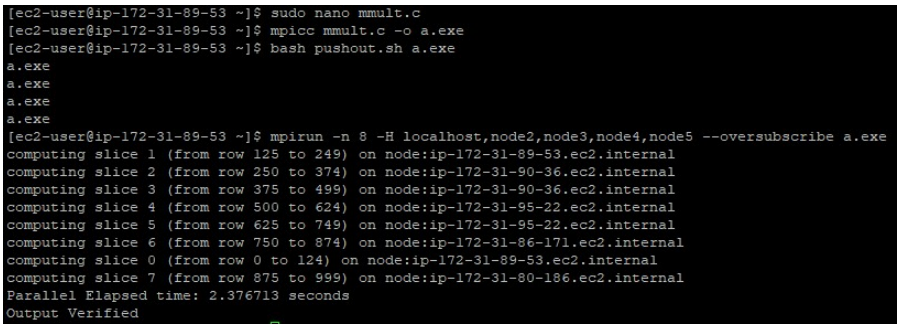
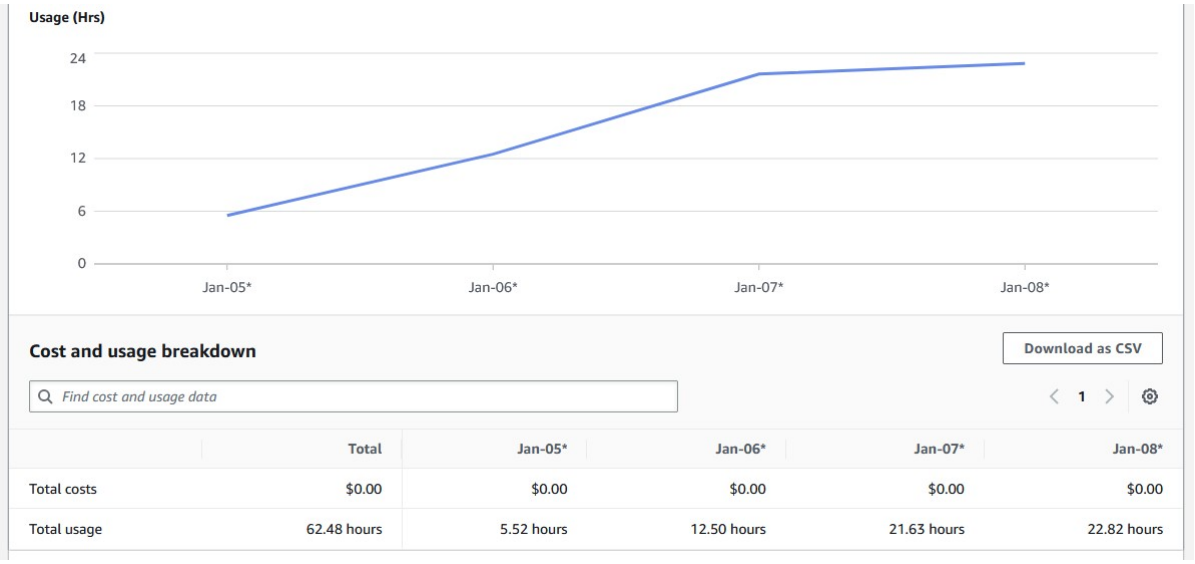


Figure 3.3: Calculation Monitoring

Results and Testing

4.1 Cost Analysis

The CPU usage by the cluster was analysed from the 5^{th} of January to the 8^{th} of January. A total of 62.48 hours of usage was reported. This is highlighted in figure 4.1. The cost incurred for this usage by the EC2 cluster was $ 1.05 in total shown in figure 4.2. This exhibits the cost effectiveness of the virtual environment created for this study.



	Total	Jan-05*	Jan-06*	Jan-07*	Jan-08*
Total costs	$0.00	$0.00	$0.00	$0.00	$0.00
Total usage	62.48 hours	5.52 hours	12.50 hours	21.63 hours	22.82 hours

Figure 4.1: AWS Usage

The amount of CPU utilized on an average per day is also highlighted through figure 4.3. With a maximum usage of an estimated 14% on a daily average, the system handles both Matrix Addition and Multiplication well. Though it does seem to require more computational power for multiplication. This will be discussed in the conclusions.

Figure 4.4 displays that the the Master Node sends the most data out while the worker nodes input more data on an average. It should be noted that on an average all the worker nodes input data in similar quantities.

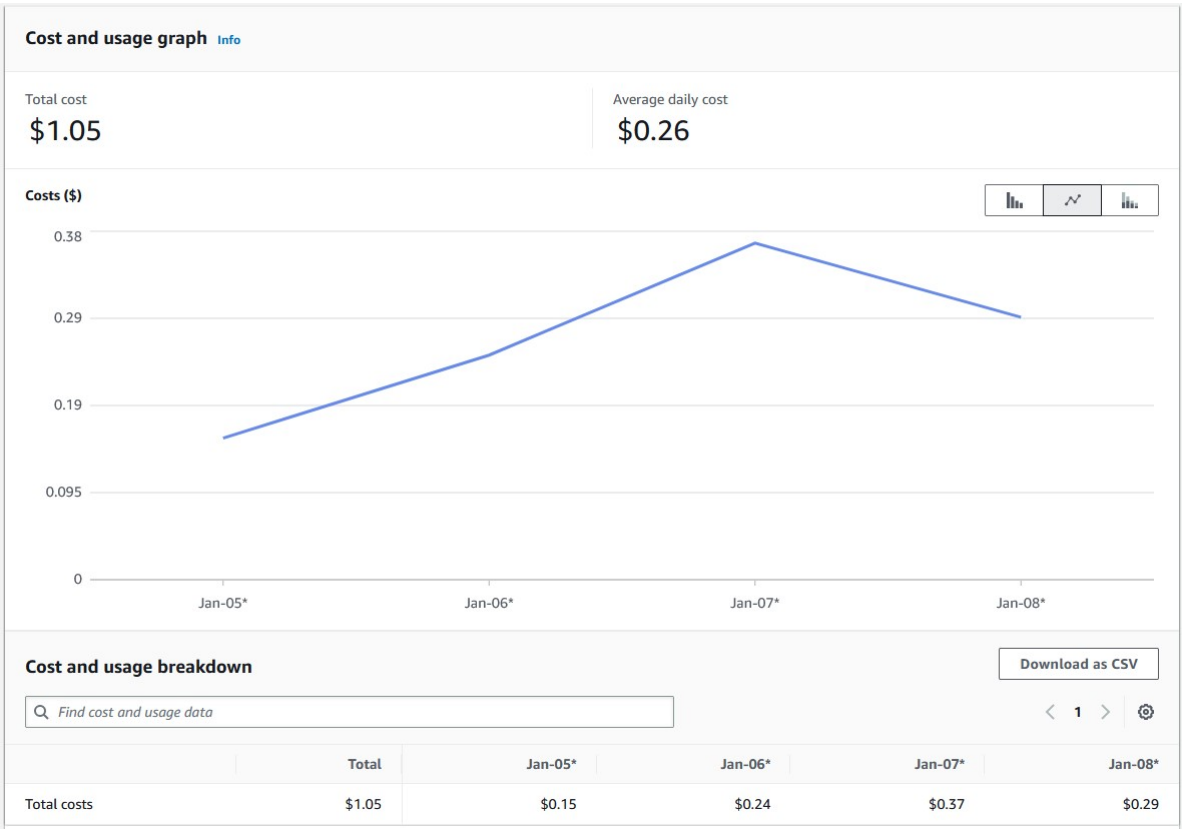


Figure 4.2: AWS Cost

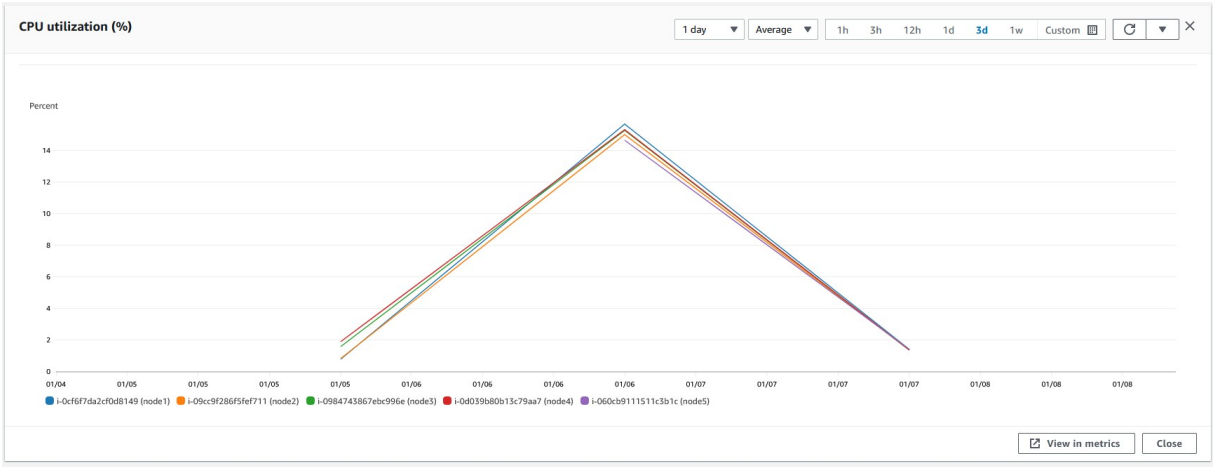


Figure 4.3: CPU Usage

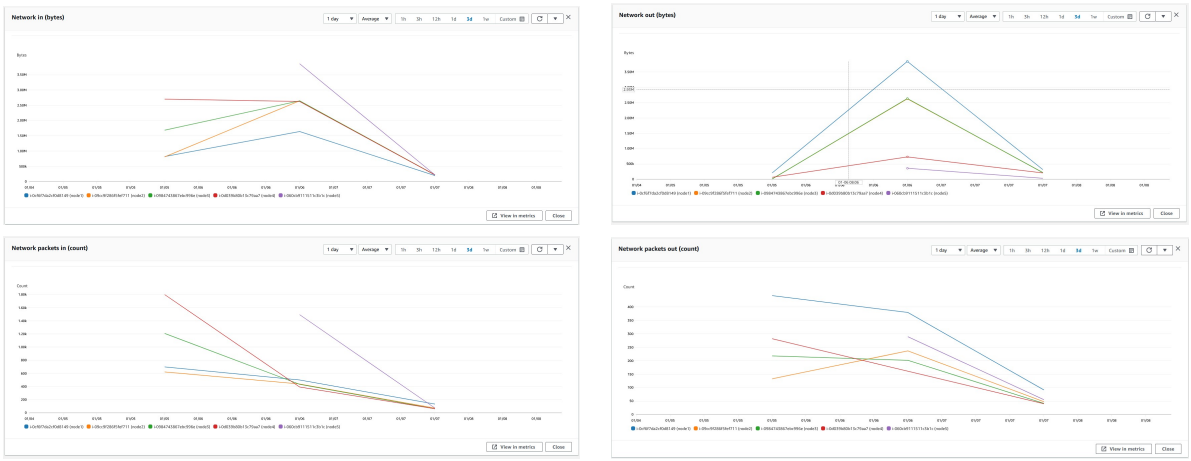


Figure 4.4: Network and Network Packets Nature

4.2 Quantitative Analysis

The system performance was noted for matrix multiplication, on a matrix of the dimensions 2000 x 2000. It was noted that (table 4.1) increasing the size of the work packages reduced the calculation time and but increasing the number of calculating nodes increased the calculation time. This was not the case with matrix addition, however.

Note: Each work package contains n number of rows of matrix A.

Multiplication	Package Size: 125	Package Size: 250	Package Size: 500
3 Worker Nodes	32.048153	30.329995	29.366279
4 Worker Nodes	31.245398	32.625560	29.551941

Table 4.1: System performance on 2000 x 2000 Matrix

In the case of addition, though increasing the size of the work packages reduced calculation time, increasing the number of nodes further reduced the calculation time as observed in tables 4.2 and 4.3.

Addition	Package Size: 125	Package Size: 250	Package Size: 500
3 Worker Nodes	2.999510	1.921346	0.920227
4 Worker Nodes	2.498468	1.887441	0.885208

Table 4.2: System performance on 2000 x 2000 Matrix for Addition

Addition	Package Size: 125	Package Size: 250	Package Size: 500
3 Worker Nodes	10.336744	5.597057	5.633039
4 Worker Nodes	8.200667	5.577550	5.567542

Table 4.3: System performance on 4000 x 4000 Matrix for Addition

Conclusion and Furture Work

The objective of this report was to study matrix multiplication and addition on a distributed cloud environment. It is observed that the current system cannot handle Matrix Multiplication as efficiently as Matrix Addition. An alternative approach to improve the efficiency of this system to handle multiplication would be to inculcate t2.small instances in it. As far as the CPU utilization and cost effectiveness goes, this system is highly cost effective, while CPU usage is moderate. This is observed in the results section where after a usage of about 63 hours, only $1.05 were spent in total. The CPU touched a maximum of 14% in daily average usage. Furthermore, a difference in impact on the environment due to the two operations i.e., addition and multiplication for matrices was also noticed. While increasing the number of nodes and size of the work packages benefits the time consumed addition of two dense matrices, that is not the case for matrix multiplication. Increasing the amount of nodes in the system leads to an increase in time taken for the multiplication of two matrices. Although, increasing the size of the work packages does benefit the multiplication operation as well. While the system takes a mere few seconds to calculate matrix addition, it takes a lot more time to handle matrix multiplication as noticed in the results section.

The future work for this study should be focused on testing cost effective instance types that are able to handle matrix multiplication as well as matrix addition.

Bibliography

[1] AWS AWS. *EFS*. Jan. 2000. URL: https://docs.aws.amazon.com/efs/latest/ug/gs-step-one-create-ec2-resources.html.

[2] Ignacio Bermudez et al. "Exploring the cloud from passive measurements: The Amazon AWS case". In: *2013 Proceedings IEEE INFOCOM*. IEEE. 2013, pp. 230–234.

[3] Arthur Cayley. "II. A memoir on the theory of matrices". In: *Philosophical transactions of the Royal society of London* 148 (1858), pp. 17–37.

[4] Lyndon Clarke, Ian Glendinning, and Rolfclarke1994mpi Hempel. "The MPI message passing interface standard". In: *Programming environments for massively parallel distributed systems*. Springer, 1994, pp. 213–218.

[5] Donald J. Daly and Donald J. Daly. *Economics 2: EC2*. Jan. 1987. URL: https://docs.aws.amazon.com/ec2/index.html.

[6] Jack J Dongarra et al. "An introduction to the MPI standard". In: *Communications of the ACM* 18 (1995).

[7] Ivor Grattan-Guinness. *Convolutions in French mathematics, 1800–1840: from the calculus and mechanics to mathematical analysis and mathematical physics*. Vol. 4. Birkhäuser, 2017.

[8] Brian D. Hahn and Daniel T. Valentine. "Chapter 6 - Matrices and Arrays". In: *Essential MATLAB for Engineers and Scientists (Seventh Edition)*. Ed. by Brian D. Hahn and Daniel T. Valentine. Seventh Edition. Academic Press, 2019, pp. 127–161. ISBN: 978-0-08-102997-8. DOI: https://doi.org/10.1016/B978-0-08-102997-8.00012-9. URL: https://www.sciencedirect.com/science/article/pii/B9780081029978000129.

[9] Frank E. Harris. "Chapter 4 - Vectors and Matrices". In: *Mathematics for Physical Science and Engineering*. Ed. by Frank E. Harris. Boston: Academic Press, 2014, pp. 111–162. ISBN: 978-0-12-801000-6. DOI: https://doi.org/10.1016/B978-0-12-801000-6.00004-3. URL: https://www.sciencedirect.com/science/article/pii/B9780128010006000043.

[10] IBM IBM. *What is cloud computing?* Jan. 2010. URL: https://www.ibm.com/topics/cloud-computing.

[11] Charles H Koelbel et al. *The high performance Fortran handbook*. MIT press, 1994.

[12] Saakshi Narula, Arushi Jain, et al. "Cloud computing security: Amazon web service". In: *2015 Fifth International Conference on Advanced Computing & Communication Technologies*. ieee. 2015, pp. 501–505.