**Laboratory Report**

Contents

[Cluster and MPI Programs Deployment: 2](#_Toc194278159)

[1) Creating Virtual Machines in Azure 2](#_Toc194278160)

[1.1) Creating the Headnode VM 2](#_Toc194278161)

[1.2) Creating compute-node1 & 2 VM 5](#_Toc194278162)

[2) Establish Connection 8](#_Toc194278163)

[3) Open Ports in Azure Network Security Group (NSG) 10](#_Toc194278164)

[4) Setting Up a Beowulf Cluster on 3 Azure VMs Using Ubuntu 22.04 11](#_Toc194278165)

[4.1) Prepare the Head Node 12](#_Toc194278166)

[4.2) Install and Test MPI 16](#_Toc194278167)

[4.3) Create an MPI Programs 17](#_Toc194278168)

[Run on other clusters in different VMs: 30](#_Toc194278169)

[1) Add new user 30](#_Toc194278170)

[Data Analysis and Visualization: 34](#_Toc194278171)

# Cluster and MPI Programs Deployment:

1. Creating Virtual Machines in Azure

The aim of this session is to create Virtual Machines in Azure for a Virtual Computer Cluster (VCC).

Two types of VMs will be created: **Headnode** VM and **Worker** node(s) VM.

The VCC should have one Headnode and two Compute-node nodes.

The Headnode VM will require the following resources:

Resource group HPC

Operating system: Ubuntu 22.04

CPU (Size): Standard\_B1s 1 **vcpu** with 1GiB **Memory**

Disk: Standard SSD

Network interface: Virtual network HPC-vnet

The Compute-node nodes (Compute-node-one and Compute-node-two) VMs will require the following resources:

Resource group HPC

Operating system: Ubuntu 22.04

CPU (Size): Standard\_B1s 1 **vcpu** with 1GiB **Memory**

Disk: Standard SSD

Network interface: Virtual network HPC-vnet

All VMs must be on the same Virtual Network.

This process of configuring virtual Headnode and Compute-node nodes and their virtual network is equivalent to preparing laboratory hardware PCs with:

Ubuntu 22.04,

CPU (possible 4 cores), Memory (4 or 8 Gbytes)

Disk storage,

1 Gbit Ethernet Network Interface Card,

Network switch for private and public network interconnect.

* 1. Creating the Headnode VM

**Basics** set up:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Disk** set up:

A screenshot of a computer

Description automatically generated

**Networking** set up:

A screenshot of a computer

Description automatically generated

* 1. Creating compute-node1 & 2 VM

**Basics** set up:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Networking** set up:

Make sure that all VMs are on the same Virtual Network

A screenshot of a computer

Description automatically generated

***Repeat the same steps to create compute-node2:***

**Basics** set up:

A screenshot of a computer

Description automatically generated

A screenshot of a computer

Description automatically generated

**Networking** set up:

A screenshot of a computer

Description automatically generated

After successful completion of Tasks 1 and 2, VM Dashboard displays the following:

A screenshot of a computer

Description automatically generated

1. Establish Connection

In order to find out the ssh login details for the Headnode VM select Headnode VM from the list of the VMs and select **Connect and SSH** option:

A screenshot of a computer

Description automatically generated

Details of ssh path with the username provided in the Headnode configuration and IP address as below:

A screenshot of a computer

Description automatically generated

In the above screenshot, the ssh path of headnode would be *ssh* [*headnode@172.166.178.255*](mailto:headnode@172.166.178.255)*:*

A screenshot of a computer

Description automatically generated

1. Open Ports in Azure Network Security Group (NSG)

By default, Azure **blocks all inbound traffic** except for SSH (22). So, it need to manually open **6817 (Slurm Control)** and **6818 (Slurm Daemon)**.

**Modify NSG Rules in Azure Portal:**

1. **Go to the Azure Portal** → Navigate to **"Virtual Machines"**.
2. Click on the **head node VM** → Scroll down to **Networking**.
3. Under **"Inbound port rules"**, click **"Add inbound rule"**.
4. Configure the rule for Slurm ports **6817** and **6818**:
   * **Port:** 6817
   * **Protocol:** TCP
   * **Source:** Any
   * **Destination:** Any
   * **Action:** Allow
   * **Priority:** 101
   * **Name:** **Allow-Slurm-Control**
   * **Port:** 6818
   * **Protocol:** TCP
   * **Source:** Any
   * **Destination:** Any
   * **Action:** Allow
   * **Priority:** 102
   * **Name:** **Allow-Slurm-Daemon**
5. **Repeat steps 2-4 for each compute node**.

Headnode:  
A screenshot of a computer

Description automatically generated

Compute-node1:

A screenshot of a computer

Description automatically generated

Compute-node2:

A screenshot of a computer

Description automatically generated

1. Setting Up a Beowulf Cluster on 3 Azure VMs Using Ubuntu 22.04

The goal of this section is to **install and test cluster middleware** (Slurm and OpenMPI) on an **Azure-based Beowulf Cluster** with three VMs:

1. **Head Node (Master)** → Runs the Slurm Controller (slurmctld).
2. **Compute Node 1**
3. **Compute Node 2**

**🔹 Pre-requisites**

* Have **three Ubuntu 22.04 LTS VMs** in Azure:
  + **headnode**
  + **compute-node1**
  + **compute-node2**
* All VMs are in the **same Virtual Network (VNet)**.
* **SSH access** to all nodes.

### 4.1) Prepare the Head Node

**Connect to the Head Node**

*ssh headnode@172.166.178.255*

**Disable the Firewall and SELinux**

*sudo systemctl stop ufw*

*sudo systemctl disable ufw*

**Edit the Hosts File on All Nodes**

To enable **hostname resolution**, add all nodes to /etc/hosts.

Run:

*sudo nano /etc/hosts*

Add:

*10.0.0.4 headnode*

*10.0.0.6 compute-node1*

*10.0.0.5 compute-node2*

*A screenshot of a computer

AI-generated content may be incorrect.*

Save (CTRL+X, Y, ENTER).

**Open the SSH configuration file**:

*nano ~/.ssh/config*

**Add the following configuration**:

Host compute-node1

HostName compute-node1

User compute-node1

IdentityFile ~/.ssh/id\_rsa

Host compute-node2

HostName compute-node2

User compute-node2

IdentityFile ~/.ssh/id\_rsa

A screenshot of a computer

AI-generated content may be incorrect.

**Test Connectivity**

Ensure all nodes can **ping each other**:

*ping compute-node1*

*ping compute-node2*

*A computer screen shot of a program

AI-generated content may be incorrect.*

**Enable SSH Passwordless Login**

On the head node, generate an SSH key:

*ssh-keygen -t rsa -b 4096*

*A screen shot of a computer

AI-generated content may be incorrect.*

Copy the key to compute nodes:

*ssh-copy-id compute-node1*

*ssh-copy-id compute-node2*

*A screenshot of a computer program

AI-generated content may be incorrect.*

### 4.2) Install and Test MPI

**Install OpenMPI on All the 3 Nodes**

*sudo apt install -y openmpi-bin openmpi-common openmpi-doc libopenmpi-dev*

*A screen shot of a computer

AI-generated content may be incorrect.*

**MPI User (mpiuser - Recommended)**

**For running MPI jobs, it is best to create a non-root user with SSH access across nodes.**

**Create an MPI user (all nodes):**

*sudo useradd -m -s /bin/bash mpiuser*

*sudo passwd mpiuser*

**Give sudo privileges if needed:**

*sudo usermod -aG sudo mpiuser*

**Allow passwordless SSH for mpiuser (only Headnode):**

*sudo su - mpiuser*

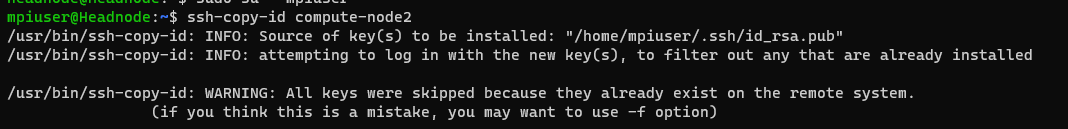
*ssh-keygen -t rsa -b 4096*

*ssh-copy-id compute-node1*

*A screen shot of a computer

AI-generated content may be incorrect.*

*ssh-copy-id compute-node2*

**

### 4.3) Create MPI Programs

For this project, we will evaluate the cluster with 2 prepared MPI Programs (**pingpong.c** and **mat\_mat.c**):

+ Create and run **pingpong.c**:

The **pingpong.c** implements an MPI-based communication benchmark that measures network performance between processes using a round-robin pattern. The code systematically tests different message sizes (1 byte to 1MB) by having processes send and receive data in pairs, calculating both latency (microseconds) and bandwidth (MiB/s). After warm-up iterations, it performs timed repetitions of the send/receive operations, records performance metrics, and writes the data results to a CSV file for later visualization. This helps characterize the communication capabilities of an MPI implementation on specific hardware

*The* ***pingpong.c*** *source file can also be found here:* [*https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/pingpong.c*](https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/pingpong.c)

On **headnode**, create a new file **pingpong.c**:

*nano pingpong.c*

Paste:



A black screen with white text

AI-generated content may be incorrect.

*The* ***pingpong.c*** *source file can also be found here:* [*https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/pingpong.c*](https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/pingpong.c)

Save and exit.

**Compile the MPI Program**

*mpicc pingpong.c -o pingpong*

*chmod +x pingpong*

**Make sure the compiled file have the (`x*`*) excecute permission**

*ls -l*

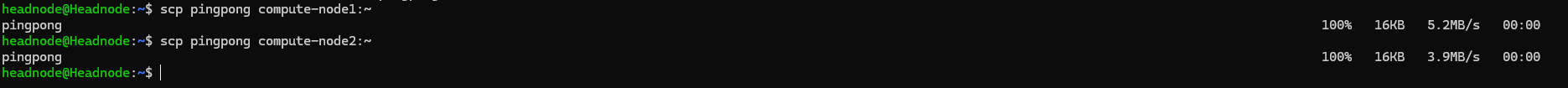
*A computer screen with white text and numbers

AI-generated content may be incorrect.*

**Copy pingpong to Compute Nodes**

*scp pingpong compute-node1:~*

*scp pingpong compute-node2:~*

**

**Create mpi\_hosts File**

On **headnode**, run:

*nano mpi\_hosts*

Add:

*compute-node1 slots=1*

*compute-node2 slots=1*

*A screenshot of a computer

AI-generated content may be incorrect.*

Save and exit.

**Run `pingpong` MPI Program**

- With **2 processes** and **1 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 2 --hostfile mpi\_hosts ./pingpong*

**Expected Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

- With **2 processes** and **2 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 2 --hostfile mpi\_hosts ./pingpong*

**Expected Output:**

A screenshot of a computer

AI-generated content may be incorrect.

- With **4 processes** and **2 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 4 --hostfile mpi\_hosts ./pingpong*

**Expected Output:**

A screenshot of a computer

AI-generated content may be incorrect.

+ Create and run **mat\_mat.c**:

The **mat\_mat.c** implements a parallel matrix-matrix multiplication benchmark using MPI to distribute computation across multiple processes. The program tests matrices of increasing sizes (2×2 to 1024×1024), dividing matrix B into portions that are scattered to different processes while broadcasting matrix C completely. Each process calculates its assigned portion of the result matrix A, which is then gathered back to the root process. The code measures execution time, calculates performance metrics (latency and bandwidth), and saves data results to a CSV file for later further analysis, providing insights into how matrix multiplication scales across distributed computing environments.

*The* ***mat\_mat.c*** *source file can also be found here:* [*https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/mat\_mat.c*](https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/mat_mat.c)

- Create and run **mat\_mat.c**:

On **headnode**, create a new file **mat\_mat.c**:

*nano mat\_mat.c*

Paste:



A screenshot of a computer

AI-generated content may be incorrect.

Save and exit.

**Compile the MPI Program**

*mpicc mat\_mat.c -o mat\_mat*

*chmod +x mat\_mat*

**Make sure the compiled file have the (`x*`*) excecute permission**

*ls -l*

*A screen shot of a computer

AI-generated content may be incorrect.*

**Copy mat\_mat to Compute Nodes**

*scp mat\_mat compute-node1:~*

*scp mat\_mat compute-node2:~*

**

**Run `mat\_mat` MPI Program**

- With **2 processes** and **1 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 2 --hostfile mpi\_hosts ./mat\_mat*

**Expected Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

- With **2 processes** and **2 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 2 --hostfile mpi\_hosts ./mat\_mat*

**Expected Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

- With **4 processes** and **2 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 4 --hostfile mpi\_hosts ./mat\_mat*

**Expected Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

# Run on other clusters in different VMs:

This section will try to run on **Hatim Moaiyadi** **(U2187171)**’s Azure VMs (with similar settings including *headnode*, *compute-node1* and *compute-node2*), then run and get results from those 2 above MPI Programs `pingpong.c` and `mat\_mat.c` for later benchmarks comparisons.

### 1) Add new user

**Connect to the Hatim’s Head Node**

*ssh headnode@****20.68.2.132***

**Create an new user “tom” (all nodes):**

*sudo useradd -m -s /bin/bash* ***tom***

*sudo passwd* ***tom***

**Check all users in the system:**

*cat /etc/passwd*

A screenshot of a computer program

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

A screen shot of a computer

AI-generated content may be incorrect.

**Give sudo privileges if needed:**

*sudo usermod -aG sudo* ***tom***

**Allow passwordless SSH for mpiuser (only Headnode):**

*sudo su -* ***tom***

*ssh-keygen -t rsa -b 4096*

A screenshot of a computer program

AI-generated content may be incorrect.

*ssh-copy-id compute-node1*

*ssh-copy-id compute-node2*

A screenshot of a computer program

AI-generated content may be incorrect.

### 2) Set up MPI Programs

For this project, we will evaluate the cluster with 2 prepared MPI Programs (**pingpong.c** and **mat\_mat.c**):

+ Create and run **pingpong.c**:

The **pingpong.c** implements an MPI-based communication benchmark that measures network performance between processes using a round-robin pattern. The code systematically tests different message sizes (1 byte to 1MB) by having processes send and receive data in pairs, calculating both latency (microseconds) and bandwidth (MiB/s). After warm-up iterations, it performs timed repetitions of the send/receive operations, records performance metrics, and writes the data results to a CSV file for later visualization. This helps characterize the communication capabilities of an MPI implementation on specific hardware

*The* ***pingpong.c*** *source file can also be found here:* [*https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/pingpong.c*](https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/pingpong.c)

On **headnode**, create a new file **pingpong.c**:

*nano pingpong.c*

Paste:



A black screen with white text

AI-generated content may be incorrect.

*The* ***pingpong.c*** *source file can also be found here:* [*https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/pingpong.c*](https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/pingpong.c)

Save and exit.

**Compile the MPI Program**

*mpicc pingpong.c -o pingpong*

*chmod +x pingpong*

**Make sure the compiled file have the (`x*`*) excecute permission**

*ls -l*

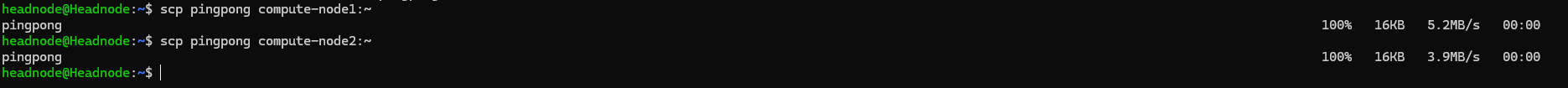
*A computer screen with white text and numbers

AI-generated content may be incorrect.*

**Copy pingpong to Compute Nodes**

*scp pingpong compute-node1:~*

*scp pingpong compute-node2:~*

**

**Create mpi\_hosts File**

On **headnode**, run:

*nano mpi\_hosts*

Add:

*compute-node1 slots=1*

*compute-node2 slots=1*

*A screenshot of a computer

AI-generated content may be incorrect.*

Save and exit.

**Run `pingpong` MPI Program**

- With **2 processes** and **1 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 2 --hostfile mpi\_hosts ./pingpong*

**Expected Output:**

**A screenshot of a computer

AI-generated content may be incorrect.**

- With **2 processes** and **2 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 2 --hostfile mpi\_hosts ./pingpong*

**Expected Output:**

A screenshot of a computer

AI-generated content may be incorrect.

- With **4 processes** and **2 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 4 --hostfile mpi\_hosts ./pingpong*

**Expected Output:**

A screenshot of a computer

AI-generated content may be incorrect.

+ Create and run **mat\_mat.c**:

The **mat\_mat.c** implements a parallel matrix-matrix multiplication benchmark using MPI to distribute computation across multiple processes. The program tests matrices of increasing sizes (2×2 to 1024×1024), dividing matrix B into portions that are scattered to different processes while broadcasting matrix C completely. Each process calculates its assigned portion of the result matrix A, which is then gathered back to the root process. The code measures execution time, calculates performance metrics (latency and bandwidth), and saves data results to a CSV file for later further analysis, providing insights into how matrix multiplication scales across distributed computing environments.

*The* ***mat\_mat.c*** *source file can also be found here:* [*https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/mat\_mat.c*](https://github.com/ThongLai/Cluster-and-Cloud-Benchmarking/blob/main/mat_mat.c)

- Create and run **mat\_mat.c**:

On **headnode**, create a new file **mat\_mat.c**:

*nano mat\_mat.c*

Paste:



A screenshot of a computer

AI-generated content may be incorrect.

Save and exit.

**Compile the MPI Program**

*mpicc mat\_mat.c -o mat\_mat*

*chmod +x mat\_mat*

**Make sure the compiled file have the (`x*`*) excecute permission**

*ls -l*

*A screen shot of a computer

AI-generated content may be incorrect.*

**Copy mat\_mat to Compute Nodes**

*scp mat\_mat compute-node1:~*

*scp mat\_mat compute-node2:~*

**

**Run `mat\_mat` MPI Program**

- With **2 processes** and **1 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 2 --hostfile mpi\_hosts ./mat\_mat*

**Expected Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

- With **2 processes** and **2 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 2 --hostfile mpi\_hosts ./mat\_mat*

**Expected Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

- With **4 processes** and **2 slots per node**:

`mpi\_hosts` configuration:



Running:

*mpirun -np 4 --hostfile mpi\_hosts ./mat\_mat*

**Expected Output:**

**A screen shot of a computer

AI-generated content may be incorrect.**

# Data Analysis and Visualization:

A screenshot of a computer program

AI-generated content may be incorrect.