NHE2530 Computer Cluster and Campus Grid project

Name: Ruming Zheng

Student ID: U2180154

Date: 25/04/2024

Content

EQUIPMENT USED **3**

1 BRIEF OBJECTIVES **3**

1.1 Deployment of the Virtual Computer Cluster (VCC) 3
1.2 Execution of MPI Codes and Integration of Supplementary Applications 3
1.3 Performance Benchmarking and Comparative Analysis 3

2 SYSTEM SKETCHES **4**

3 PROCEDURE **4**

3.1 Create virtual machines in Azure 4
3.2 Installation of the cluster middleware 5
3.2.1 Installation of the cluster middleware in Headnode *5*
3.2.2 Installation of the cluster middleware in worker nodes *5*
3.3 Message Passing Interface (MPI) 6
3.4 Message Passing Interface with better GPU 6
3.5 Running MPI jobs on VCC using PBS scripts 8

4 TEST RESULTS **9**

4.1 Installation of the cluster middleware 9
4.2 Message Passing Interface (MPI) 10
4.3 Message Passing Interface with better GPU 11
4.4 Running MPI jobs on VCC using PBS scripts 15

5 ADDITIONAL APPLICATIONS WITH MPI CODES **17**

6 BENCHMARKING – COMPARISON VCC CLUSTER WITH A COLLEAGUE **22**

7 DISCUSSION **25**

8 CONCLUSION **26**

9 REFERENCE **27**

Equipment Used

PC: Desktop in the university with Windows 10 Professional operating system.

Microsoft Azure: A cloud platform to create virtual machines.

PuTTY: It is a remote access tool that allows users to securely connect and operate remote servers via the SSH protocol.

1 Brief Objectives

1.1 Deployment of the Virtual Computer Cluster (VCC)

Facilitate the comprehensive deployment process of my Virtual Computer Cluster (VCC), ensuring all necessary configurations and integrations are implemented for optimal functionality.

1.2 Execution of MPI Codes and Integration of Supplementary Applications

Execute MPI (Message Passing Interface) codes effectively within my VCC cluster to enhance parallel computing capabilities.

1.3 Performance Benchmarking and Comparative Analysis

Compare the performance of benchmarking of my VCC cluster with that of my colleagues.

2 System Sketches

Sequence diagram:

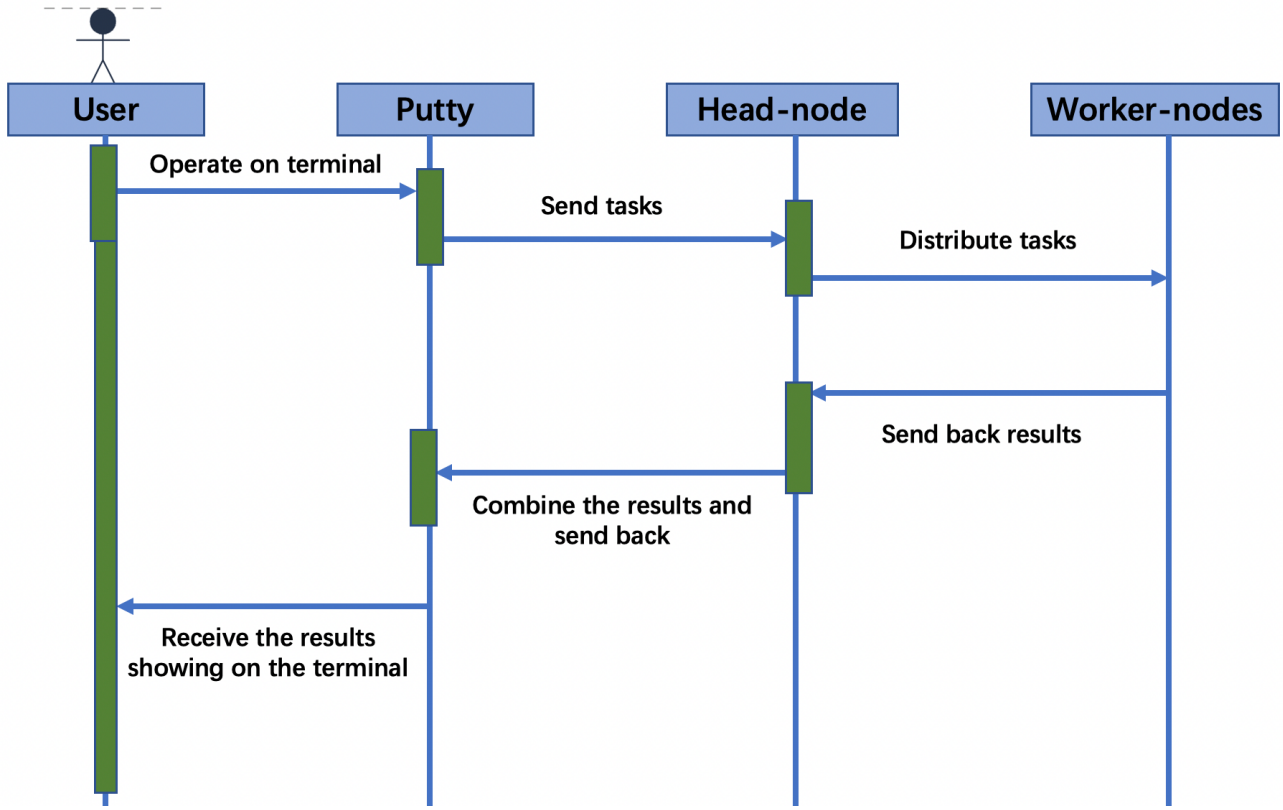

Figure 1 Sequnce Diagram

3 Procedure

3.1 Create virtual machines in Azure

In this section, 3 virtual machines, including one Headnode and two worker nodes, were created in Azure. Headnode and the worker nodes were using the same resource group, operating system (CentOS 7.4), disk and network interface. For the choice of CPU, in the early stage, Standard_B1s was chosen for both the headnode and the worker nodes.

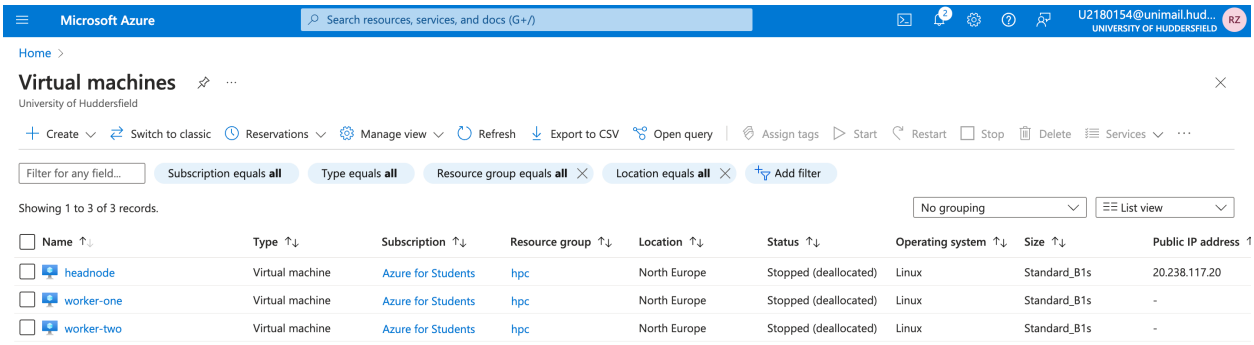

Figure 2 Virtual Machines

After three VMs were connected, the public IP of the headnode was used in Putty to operate the cluster in a terminal to run some basic commands in a Linux terminal.

3.2 Installation of the cluster middleware

3.2.1 Installation of the cluster middleware in Headnode

Login to the headnode in the Linux terminal and then SSH the private IP of worker-one and worker-two to check if the cluster is working well.

Installed the epel release and git and Ansible. After that, I modified the configuration files, including network settings and master node identifiers. Used the ssh-keygen to create the ssh key to realize passwordless in headnode.

Edited the host files to include private IPs of worker nodes so that the nodes can recognise and communicate with each other.

3.2.2 Installation of the cluster middleware in worker nodes

Firstly, the IP and MAC address of worker nodes were added to the inventory file on the headnode. ssh-keygen and other commands were used to create the private and public keys for two worker nodes for passwordless login.

Then, I collected all node public keys into one document and updated the ~/.ssh/authorized_keys file on each node to include all keys so that all nodes can communicate with each other without the password.

Returned to the headnode, installed rsync, and ran the Ansible playbook to install and update node definitions, verifying node status to ensure the middleware is correctly installed.

3.3 Message Passing Interface (MPI)

Firstly, the availability of the worker nodes was tested by the command “pbsnodes -a” on the headnode. Then, added a new user (e.g., hpcuser) using the “useradd” command, generated SSH keys for this user, and updated the cluster configuration to allow password-less SSH access to the worker nodes.

After verifying the available MPI modules, a C program named “hello.c” was created by the nano editor. Using “mpicc” and “mpirun” commands to compile and execute the MPI C program on worker-one and worker-two.

3.4 Message Passing Interface with better GPU

Resized worker-one and worker-two with 4 CPUs by allocating 2 vCPUs and 4GiB of RAM (B2s) to each worker node. However, B2s didn’t fit the worker-two for some reason.

Hence, worker-two was equipped with D2as_V4, which has 2 vCPUs and 8GiB of RAM at an acceptable price.

B2s	General purpose	2	4	4	1280	8 (SCSI)	Supported	US$32.85
B1s (free services eligible)	General purpose	1	1	2	320	4 (SCSI)	Supported	US$8.25
B2ms	General purpose	2	8	4	1920	16 (SCSI)	Supported	US$66.43
B1ls	General purpose	1	0.5	2	320	4 (SCSI)	Supported	US$4.16

Figure 3 B2s for Worker-one

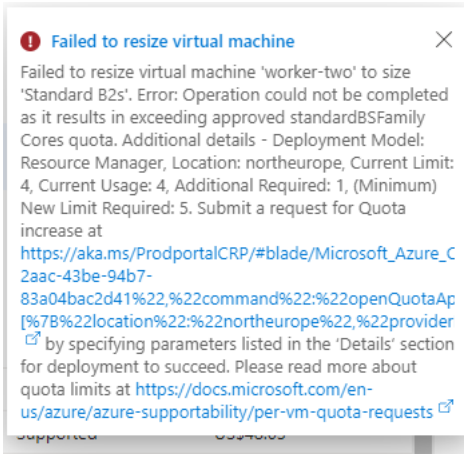

Figure 4 Error when resizing Worker-two

D2as_v4	General purpose	2	8	4	3200	16 (SCSI)	Supported	US$78.11

Figure 5 D2as_v4 for Worker-two

Complied and ran the MPI C program “hello1.c” on 2 and 4 cores, which means MPI distributes the tasks across two or four processes.

Identify the start and end functions of the MPI program:

MPI_Init(&argc, &argv) is the start function of the MPI program, which helps initiate the MPI execution environment.

Similarly, MPI_Finalize() is the end function of the MPI program, which helps terminate the MPI execution environment.

What is the meaning of rank and size?

Rank: the index of processors.

Size: the number of processors.

What would expect the values of rank and size to be for your cluster?

If run mpirun -np 4 my_mpi_program, the size will be 4, with ranks 0, 1, 2 and 3.

When compiling, $ mpicc -o hello1 hello1.c

Which is the output file and which is the source file in this command?

Source File: The source file is “hello1.c”.

Output File: The output file is “hello1”.

Why do we compile the program in this way?

Using mpicc -o hello1 hello1.c clearly defines both what to compile and what the output should be named, which gives easier management and execution of MPI programs.

How to control the distribution of processes among the cluster nodes?

When 4 cores, in machine file, we have:

```
node1 slots=2
node2 slots=2
```

This indicates that two processes should run on node1 and two on node2.

After that, compiled and ran some new programs about matrix-to-vector multiplication, matrix-to-matrix multiplication and pi calculation.

Eventually, inserted MPI function MPI_Wtime() to the pi calculation program to record the execution time.

After applications were successful, resized back the worker-one and worker-two to B1s to avoid running out of credit.

3.5 Running MPI jobs on VCC using PBS scripts

Firstly, created the pbs script “job.1pbs” by nano editor and submitted the pbs jobs to the work queue in the cluster by using the command “qsub” to give the job an ID number.

Using “qstat” to check the jobs’ status.

Using “qstat -x ID” and “qstat -H ID” to check the detail information of a specific job.

Recreated PBS job scripts to run the previous MPI programs, specifying different processes in each PBS script. After updating these MPI programs into the PBS script, observe and record the results of each job's execution.

Complied and ran the cpi_n.c MPI program by using different number N intervals for pi calculation, and different numbers of nodes and ppn.

4 Test Results

4.1 Installation of the cluster middleware

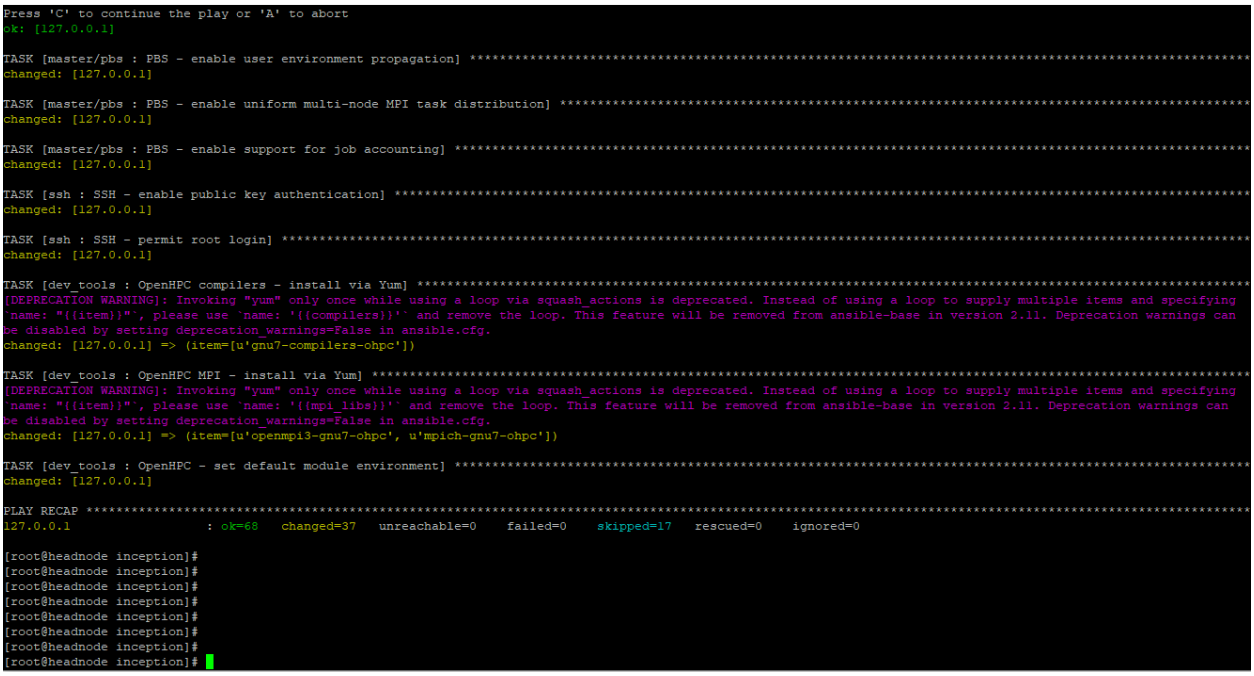

Figure 6 Installation of the cluster middleware in Headnode

The result of the install master playbook was perfect. Hence, the cluster middleware of the headnode was installed and configured correctly.

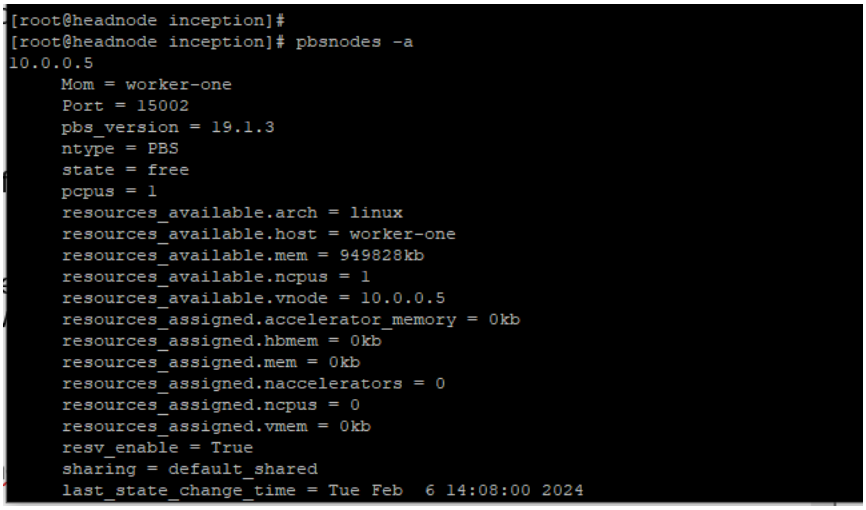

Figure 7 worker-one

```
10.0.0.6
     Mom = worker-two
     Port = 15002
     pbs_version = 19.1.3
     ntype = PBS
     state = free
     pcpus = 1
     resources_available.arch = linux
     resources_available.host = worker-two
     resources_available.mem = 949828kb
     resources_available.ncpus = 1
     resources_available.vnode = 10.0.0.6
     resources_assigned.accelerator_memory = 0kb
     resources_assigned.hbmem = 0kb
     resources_assigned.mem = 0kb
     resources_assigned.naccelerators = 0
     resources_assigned.ncpus = 0
     resources_assigned.vmem = 0kb
     resv_enable = True
     sharing = default_shared
     last_state_change_time = Tue Feb  6 14:08:00 2024

[root@headnode inception]#
```

Figure 8 worker-two

Used pbsnodes -a in headnode to check if worker-one and worker-two were working well. The results above show the state of two worker nodes was “free” as expected.

4.2 Message Passing Interface (MPI)

```
* an inability to create a connection back to mpirun due to a
  lack of common network interfaces and/or no route found between
  them. Please check network connectivity (including firewalls
  and network routing requirements).
--------------------------------------------------------------------------
--------------------------------------------------------------------------
ORTE does not know how to route a message to the specified daemon
located on the indicated node:

  my node:   headnode
  target node:  worker2

This is usually an internal programming error that should be
reported to the developers. In the meantime, a workaround may
be to set the MCA param routed=direct on the command line or
in your environment. We apologize for the problem.
--------------------------------------------------------------------------
[hpcuser@headnode jobs]$ nano hosts
[hpcuser@headnode jobs]$ mpirun -np 2 -machinefile hosts hello
[worker-two:06563] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-one:05709] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-two:06563] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-one:05709] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-one:05735] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-two:06589] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-one:05735] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-two:06589] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-two:06589] mca_base_component_repository_open: unable to open mca_btl_openib: librdmacm.so.1: cannot open sha
red object file: No such file or directory (ignored)
[worker-one:05735] mca_base_component_repository_open: unable to open mca_btl_openib: librdmacm.so.1: cannot open sha
red object file: No such file or directory (ignored)
[worker-two:06589] mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot ope
n shared object file: No such file or directory (ignored)
[worker-one:05735] mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot ope
n shared object file: No such file or directory (ignored)
[worker-two:06589] mca_base_component_repository_open: unable to open mca_mtl_psm2: libpsm2.so.2: cannot open shared
object file: No such file or directory (ignored)
[worker-one:05735] mca_base_component_repository_open: unable to open mca_mtl_psm2: libpsm2.so.2: cannot open shared
object file: No such file or directory (ignored)
Hello world from process 1 of 2
Hello world from process 0 of 2
```

Figure 9 MPI

4.3 Message Passing Interface with better GPU

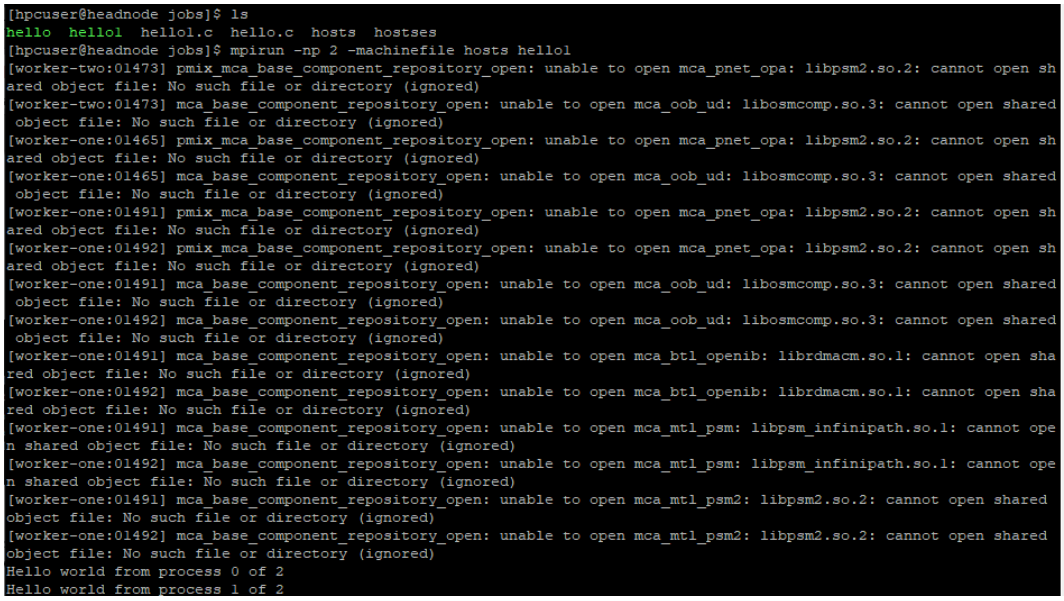

Figure 10 MPI result for 2 cores

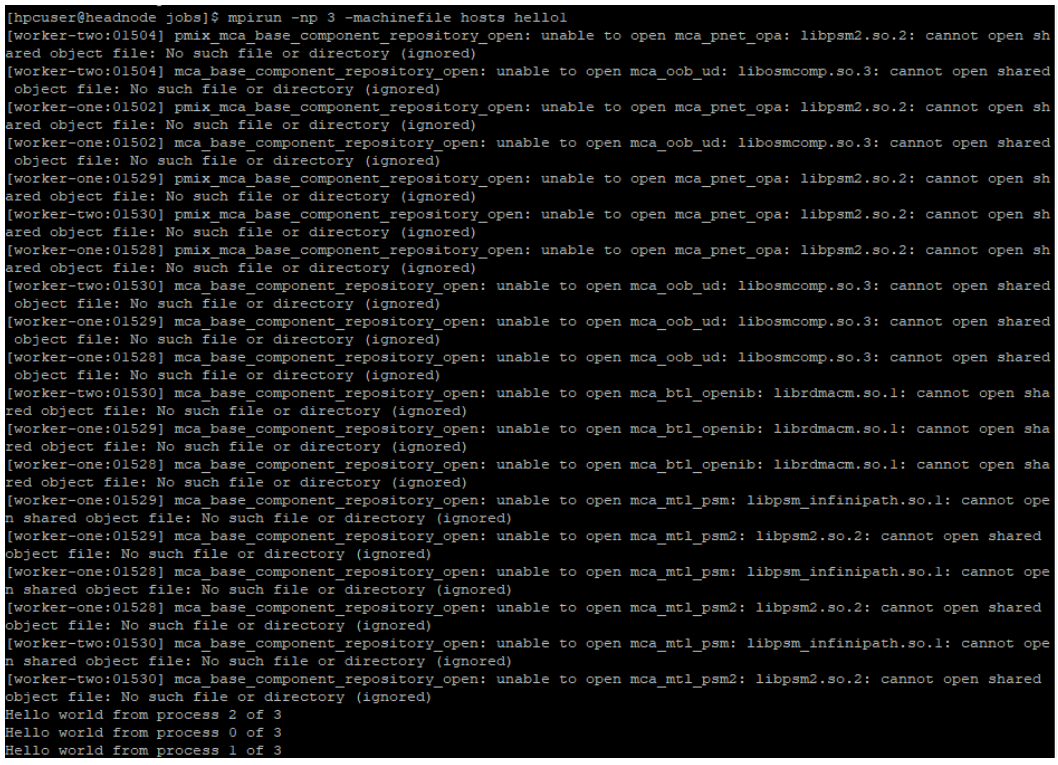

Figure 11 MPI result for 3 cores

```
[hpcuser@headnode jobs]$ mpirun -np 4 -machinefile hosts hello1
[worker-two:01538] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-two:01538] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-one:01539] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-one:01539] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
--------------------------------------------------------------------------
There are not enough slots available in the system to satisfy the 4 slots
that were requested by the application:
  hello1

Either request fewer slots for your application, or make more slots available
for use.
```

Figure 12 MPI result for 4 cores

The maximum number of cores the GPU could reach was 3 cores.

```
[hpcuser@headnode jobs]$ mpirun -np 2 -machinefile hosts mat-vecx
[worker-two:01314] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-two:01314] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-one:01309] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-one:01309] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-one:01335] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-two:01340] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-one:01335] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-two:01340] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-one:01335] mca_base_component_repository_open: unable to open mca_btl_openib: librdmacm.so.1: cannot open sha
red object file: No such file or directory (ignored)
[worker-two:01340] mca_base_component_repository_open: unable to open mca_btl_openib: librdmacm.so.1: cannot open sha
red object file: No such file or directory (ignored)
[worker-one:01335] mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot ope
n shared object file: No such file or directory (ignored)
[worker-one:01335] mca_base_component_repository_open: unable to open mca_mtl_psm2: libpsm2.so.2: cannot open shared
object file: No such file or directory (ignored)
[worker-two:01340] mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot ope
n shared object file: No such file or directory (ignored)
[worker-two:01340] mca_base_component_repository_open: unable to open mca_mtl_psm2: libpsm2.so.2: cannot open shared
object file: No such file or directory (ignored)

This is the result of the parallel computation:

A[0]=-15
A[1]=22
A[2]=-21
A[3]=-12

This is the result of the serial computation:

A_exact[0]=43
A_exact[1]=75
A_exact[2]=57
A_exact[3]=2
[hpcuser@headnode jobs]$
```

Figure 13 mat-vec

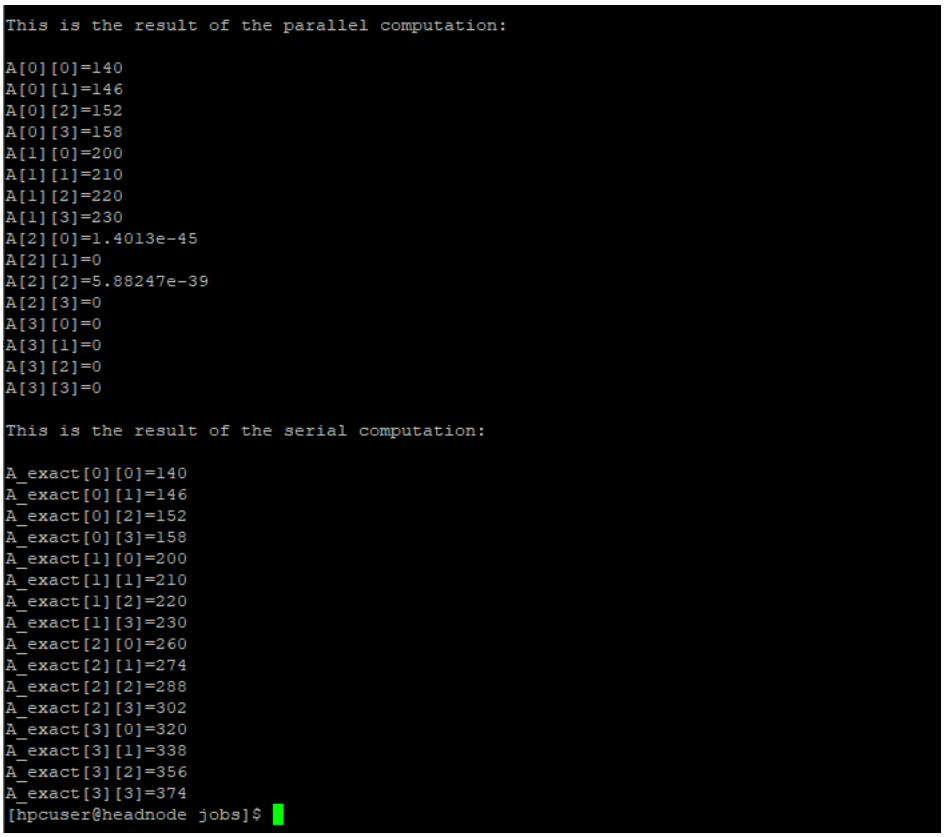

Figure 14 mat-mat

```
[hpcuser@headnode jobs]$ nano cpi.c
[hpcuser@headnode jobs]$ mpicc -o cpi cpi.c
[hpcuser@headnode jobs]$ mpirun -np 2 -machinefile hosts cpi
[worker-one:01426] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-one:01426] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-two:01427] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-two:01427] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-two:01453] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-one:01452] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open sh
ared object file: No such file or directory (ignored)
[worker-one:01452] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-two:01453] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared
 object file: No such file or directory (ignored)
[worker-one:01452] mca_base_component_repository_open: unable to open mca_btl_openib: librdmacm.so.1: cannot open sha
red object file: No such file or directory (ignored)
[worker-two:01453] mca_base_component_repository_open: unable to open mca_btl_openib: librdmacm.so.1: cannot open sha
red object file: No such file or directory (ignored)
[worker-one:01452] mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot ope
n shared object file: No such file or directory (ignored)
[worker-two:01453] mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot ope
n shared object file: No such file or directory (ignored)
[worker-one:01452] mca_base_component_repository_open: unable to open mca_mtl_psm2: libpsm2.so.2: cannot open shared
object file: No such file or directory (ignored)
[worker-two:01453] mca_base_component_repository_open: unable to open mca_mtl_psm2: libpsm2.so.2: cannot open shared
object file: No such file or directory (ignored)
Enter the number of intervals: (0 quits)
860
pi is approximately 3.1415927662633094, Error is 0.0000001126735163
Enter the number of intervals: (0 quits)
13
pi is approximately 3.1420857498385248, Error is 0.0004930962487317
Enter the number of intervals: (0 quits)
666
pi is approximately 3.1415928414653580, Error is 0.0000001878755649
Enter the number of intervals: (0 quits)
327
pi is approximately 3.1415934329231208, Error is 0.0000007793333276
Enter the number of intervals: (0 quits)
0
[hpcuser@headnode jobs]$
```

Figure 15 cpi

The results of matrix-to-vector multiplication, matrix-to-matrix multiplication and pi calculation were running successfully and these results had high accuracy.

```
[hpcuser@headnode jobs]$ mpicc -o cpi cpi.c
[hpcuser@headnode jobs]$ mpirun -np 2 -machinefile hosts cpi
[worker-two:01838] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open shared object file: No such file or directory (ignored)
[worker-one:01870] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open shared object file: No such file or directory (ignored)
[worker-two:01838] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared object file: No such file or directory (ignored)
[worker-one:01870] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared object file: No such file or directory (ignored)
[worker-two:01864] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open shared object file: No such file or directory (ignored)
[worker-one:01896] pmix_mca_base_component_repository_open: unable to open mca_pnet_opa: libpsm2.so.2: cannot open shared object file: No such file or directory (ignored)
[worker-two:01864] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared object file: No such file or directory (ignored)
[worker-one:01896] mca_base_component_repository_open: unable to open mca_oob_ud: libosmcomp.so.3: cannot open shared object file: No such file or directory (ignored)
[worker-two:01864] mca_base_component_repository_open: unable to open mca_btl_openib: librdmacm.so.1: cannot open shared object file: No such file or directory (ignored)
[worker-one:01896] mca_base_component_repository_open: unable to open mca_btl_openib: librdmacm.so.1: cannot open shared object file: No such file or directory (ignored)
[worker-two:01864] mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot open shared object file: No such file or directory (ignored)
[worker-two:01864] mca_base_component_repository_open: unable to open mca_mtl_psm2: libpsm2.so.2: cannot open shared object file: No such file or directory (ignored)
[worker-one:01896] mca_base_component_repository_open: unable to open mca_mtl_psm: libpsm_infinipath.so.1: cannot open shared object file: No such file or directory (ignored)
[worker-one:01896] mca_base_component_repository_open: unable to open mca_mtl_psm2: libpsm2.so.2: cannot open shared object file: No such file or directory (ignored)
Time1 = 6002.95
Enter the number of intervals: (0 quits)
Time1 = 6041.45
1
pi is approximately 3.2000000000000002, Error is 0.0584073464102071
Time2 = 6005.96
Enter the number of intervals: (0 quits)
840
pi is approximately 3.1415927716925909, Error is 0.0000001181027978
Time2 = 6017.49
Enter the number of intervals: (0 quits)
0
[hpcuser@headnode jobs]$
```

Figure 16 Execution time

By inserting the MPI_Wtime() at the beginning and the end of the cpi program, the result recorded the different execution time for different numbers of intervals.

4.4 Running MPI jobs on VCC using PBS scripts

```
[hpcuser@headnode jobs]$ qsub job1.pbs
0.headnode.local
[hpcuser@headnode jobs]$ qstat
Job id            Name             User              Time Use S Queue
----------------  ---------------- ----------------  -------- - -----
0.headnode        firsthello       hpcuser                 0 Q workq
[hpcuser@headnode jobs]$
```

Figure 17 qstat

```
[hpcuser@headnode jobs]$ qstat -H 0

headnode.local:
                                                                  Req'd  Req'd   Elap
Job ID          Username Queue    Jobname    SessID NDS TSK Memory Time  S Time
--------------- -------- -------- ---------- ------ --- --- ------ ----- - -----
0.headnode.loca hpcuser  workq    firsthello    --    2   2    --     --  Q   --
```

Figure 18 qstat -H ID

```
[hpcuser@headnode jobs]$ qstat -x 0
Job id            Name             User              Time Use S Queue
----------------  ---------------- ----------------  -------- - -----
0.headnode        firsthello       hpcuser                 0 Q workq
```

Figure 19 qstat -x ID

```
[hpcuser@headnode jobs]$ qstat
Job id            Name             User              Time Use S Queue
----------------  ---------------- ----------------  -------- - -----
0.headnode        firsthello       hpcuser                 0 Q workq
3.headnode        hello_hostname   hpcuser                 0 Q workq
4.headnode        mat-vecx         hpcuser                 0 Q workq
5.headnode        mat-mat          hpcuser                 0 Q workq
```

Figure 20 repeated process

The hello_hostname, mat-vecx, mat_mat were successfully updated in the PBS script.

RUN cpi_n.c
Mpirun -np 1 -machinefile hosts cpi_n

```
Process 0 on worker-one
pi is approximately 3.1415926535899006, Error is 0.0000000000001075
wall clock time = 0.015034
```

Figure 21 cpi_n for one core

Mpirun -np 2 -machinefile hosts cpi_n

```
Process 1 on worker-one
Process 0 on worker-one
pi is approximately 3.1415926535899179, Error is 0.0000000000001248
wall clock time = 0.007657
```

Figure 22 cpi_n for two core

Mpirun -np 3 -machinefile hosts cpi_n

```
Process 2 on worker-two
Process 1 on worker-one
Process 0 on worker-one
pi is approximately 3.1415926535899548, Error is 0.0000000000001616
wall clock time = 0.033012
```

Figure 23 cpi_n for three core

Mpirun -np 4 -machinefile hosts cpi_n

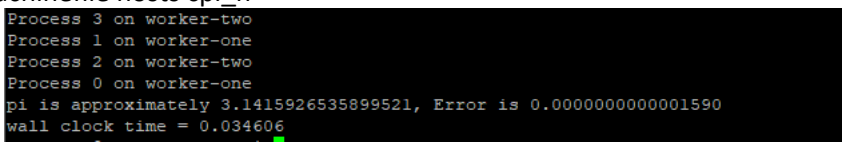

Figure 24 cpi_n 4 core

Nodes pi error time

Node	Pi calculation	Error	Time
1	3.1415926535899006	0.0000000000001075	0.015034
2	3.1415926535899179	0.0000000000001248	0.007657
3	3.1415926535899548	0.0000000000001616	0.033012
4	3.1415926535899521	0.0000000000001590	0.034606

Table 1 Results for cpi_n

Accuracy decreased with the increment of the node.

Also, the time for each process decreased in node 2 and increased in node 3 and node 4, which was not as expected.

5 Additional Applications with MPI codes

Based on the original program for OPENMP (According to Burkardt (n.d.), the SAXPY operation can be effectively parallelized using OpenMP), I changed the library to MPI for testing.

This program shows how a SAXPY operation could be marked up for MPI execution.

```
#include "mpi.h"
#include <stdio.h>
#include <math.h>
#include<stdlib.h>

double f( double );
double f( double a )
{
   return (4.0 / (1.0 + a*a));
}

int main ( int argc, char *argv[] )
{
  int i;
  int n = 10000000;
  float s = 1.5;
  float *x;
  float *y;
  float *z;

  //from mpi
          int  myid;
          int        numprocs;
      double startwtime = 0.0, endwtime;
          int k = 2;
          int  namelen;
   char processor_name[MPI_MAX_PROCESSOR_NAME];

   MPI_Init(&argc,&argv);

   MPI_Comm_size(MPI_COMM_WORLD,&numprocs);
   MPI_Comm_rank(MPI_COMM_WORLD,&myid);
   MPI_Get_processor_name(processor_name,&namelen);

   fprintf(stderr,"Process %d on %s\n",
             myid, processor_name);

 startwtime = MPI_Wtime();

          MPI_Bcast(&k, 1, MPI_INT, 0, MPI_COMM_WORLD);
```

```
  printf ( "\n" );
  printf ( "SAXPY\n" );
  printf ( "  C/OpenMP version\n" );
  printf ( "\n" );
  printf ( "  A program which adds a multiple of one vector to another.\n" );

  printf ( "\n" );
  // printf ( "  Number of processors available = %d\n",
MPI_Comm_size(MPI_COMM_WORLD,&numprocs));
  // printf ( "  Number of threads =            %d\n",  MPI_Comm_rank(MPI_COMM_WORLD,&myid));
/*
  Allocate the vector data.
*/
  x = ( float * ) malloc ( n * sizeof ( float ) );
  y = ( float * ) malloc ( n * sizeof ( float ) );
  z = ( float * ) malloc ( n * sizeof ( float ) );
/*
  Begin the parallel region.
*/
# pragma omp parallel private ( i )
{
/*
  Set the vector data.
*/
# pragma omp for
  for ( i = 0; i < n; i++ )
  {
    x[i] = ( float ) ( ( i + 1 ) % 7 );
    y[i] = ( float ) ( ( i + 1 ) % 3 );
  }
/*
  Perform the SAXPY operation.
*/
# pragma omp for
  for ( i = 0; i < n; i++ )
  {
    z[i] = x[i] + s * y[i];
  }
/*
  End the parallel region.
*/
}
/*
  Print a few entries.
*/
  printf ( "\n" );
  printf ( "   i       x[i]       y[i]      z[i]=x[i]+s*y[i]\n" );
```

```
  printf ( "\n" );
  for ( i = 0; i < n && i < 10; i++ )
  {
    printf ( "  %2d  %10.4f  %10.4f  %10.4f\n", i, x[i], y[i], z[i] );
  }
/*
  Free memory.
*/
  free ( x );
  free ( y );
/*
  Terminate.
*/
endwtime = MPI_Wtime();
                        printf("wall clock time = %f\n",
                             endwtime-startwtime);
  printf ( "\n" );
  printf ( "SAXPY\n" );
  printf ( "  Normal end of execution.\n" );

  MPI_Finalize();

  return 0;
}
```

Result:

```
[hpcuser@headnode jobs]$ mpirun -np 4 -machinefile hosts SAXPY
Process 0 on worker-one
Process 1 on worker-one
Process 3 on worker-two
Process 2 on worker-two
  A program which adds a multiple of one vector to another.


   i        x[i]        y[i]      z[i]=x[i]+s*y[i]

   0      1.0000      1.0000      2.5000
   1      2.0000      2.0000      5.0000
   2      3.0000      0.0000      3.0000
   3      4.0000      1.0000      5.5000
   4      5.0000      2.0000      8.0000
   5      6.0000      0.0000      6.0000
   6      0.0000      1.0000      1.5000
   7      1.0000      2.0000      4.0000
   8      2.0000      0.0000      2.0000
   9      3.0000      1.0000      4.5000
wall clock time = 0.116824
```

```
SAXPY
  Normal end of execution.

   i        x[i]        y[i]      z[i]=x[i]+s*y[i]

   0      1.0000      1.0000      2.5000
   1      2.0000      2.0000      5.0000
   2      3.0000      0.0000      3.0000
   3      4.0000      1.0000      5.5000
   4      5.0000      2.0000      8.0000
   5      6.0000      0.0000      6.0000
   6      0.0000      1.0000      1.5000
   7      1.0000      2.0000      4.0000
   8      2.0000      0.0000      2.0000
   9      3.0000      1.0000      4.5000
wall clock time = 0.120468
```

```
SAXPY
  Normal end of execution.

   i        x[i]        y[i]      z[i]=x[i]+s*y[i]

   0      1.0000      1.0000      2.5000
   1      2.0000      2.0000      5.0000
   2      3.0000      0.0000      3.0000
   3      4.0000      1.0000      5.5000
   4      5.0000      2.0000      8.0000
   5      6.0000      0.0000      6.0000
   6      0.0000      1.0000      1.5000
   7      1.0000      2.0000      4.0000
   8      2.0000      0.0000      2.0000
   9      3.0000      1.0000      4.5000
wall clock time = 0.122893
```

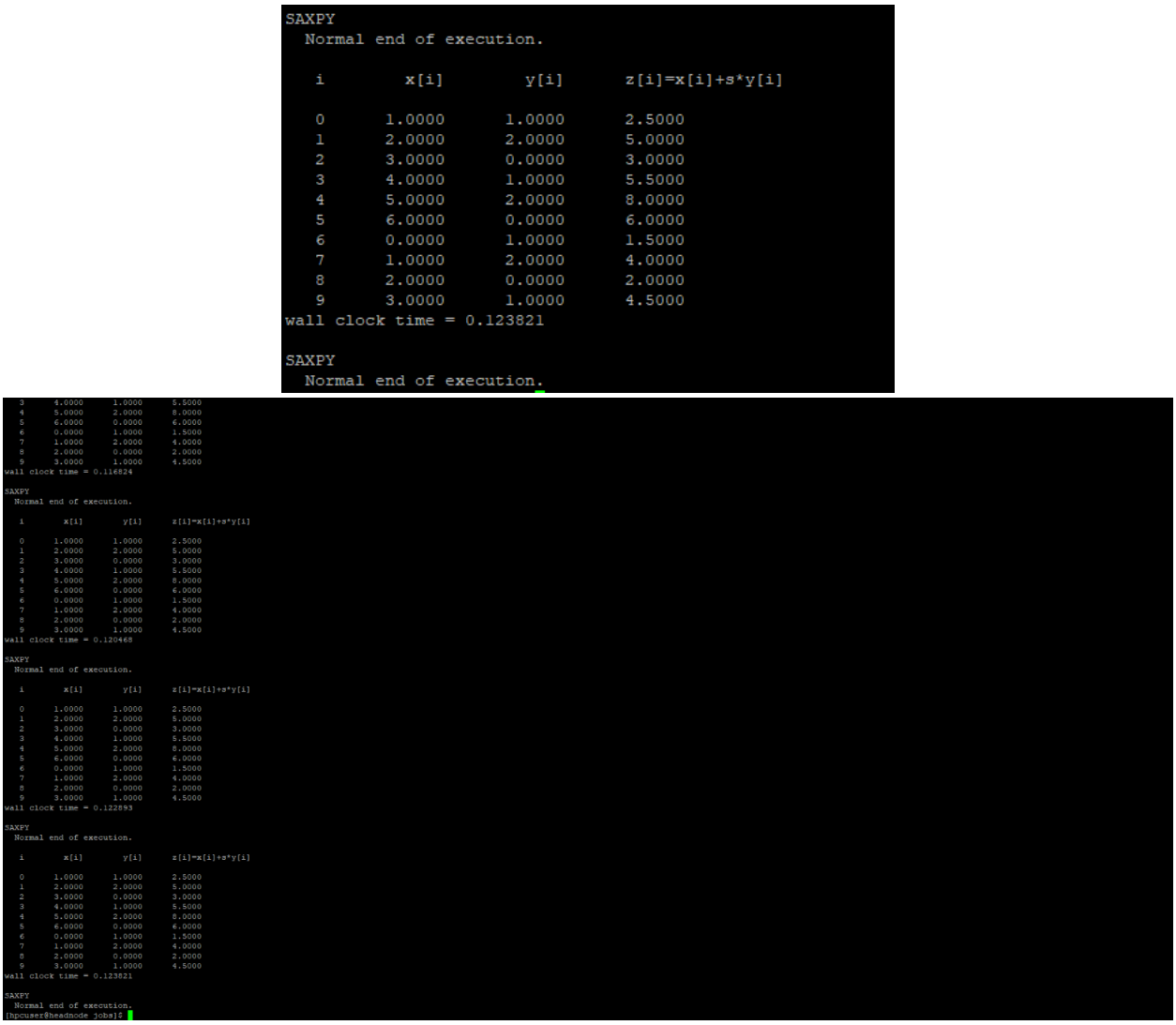

Figure 25 Results of SAXPY for 4 cores

In this program, every node was calculating the same task. That's why the result was the same in every node, but the execution time for every node increased.

6 Benchmarking – comparison VCC Cluster with a colleague

1. Created a new user in the headnode with username “kbly”

```
[root@headnode headnode]# useradd kbly
[root@headnode headnode]# su kbly
[kbly@headnode headnode]$ pbsnodes -a
```

Figure 26 My new user

2. Generate public and private keygen by Putty

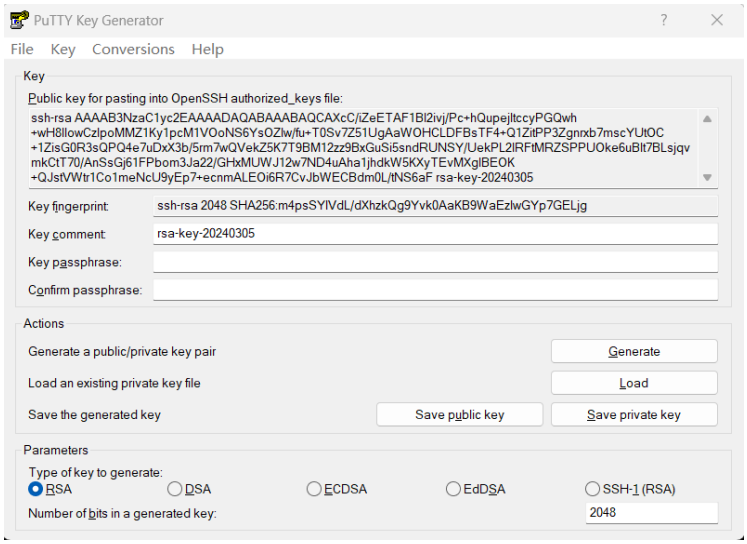

Figure 27 Generation public key in Putty

Put my private key in Putty.

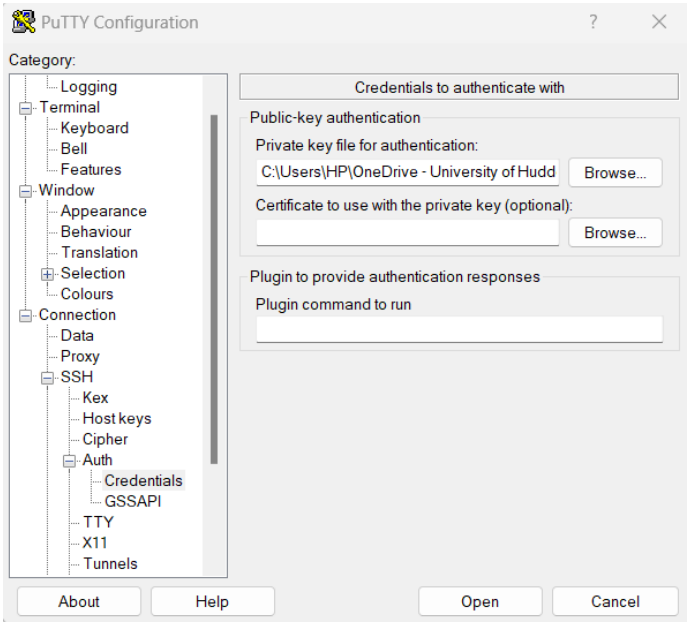

Figure 28 Private key

Added the colleague's public keygen to my new user.

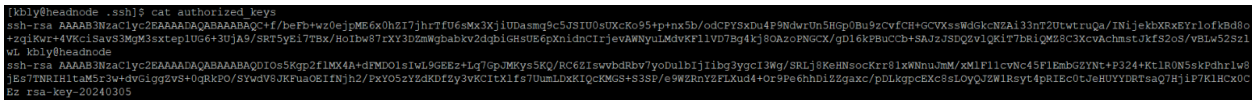

Figure 29 Public key from the colleague

Used the public IP from the colleague's cluster to login in Putty terminal.

Successfully connected to the colleague's cluster without password!

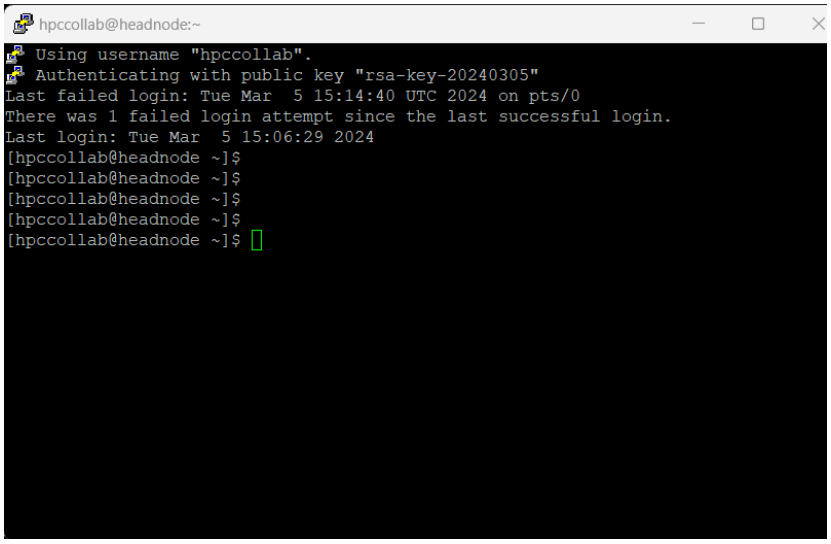

Figure 30 Connected successfully

3. Compiled and ran cpi_n.c program in the colleague's user

Testing with 1 core:

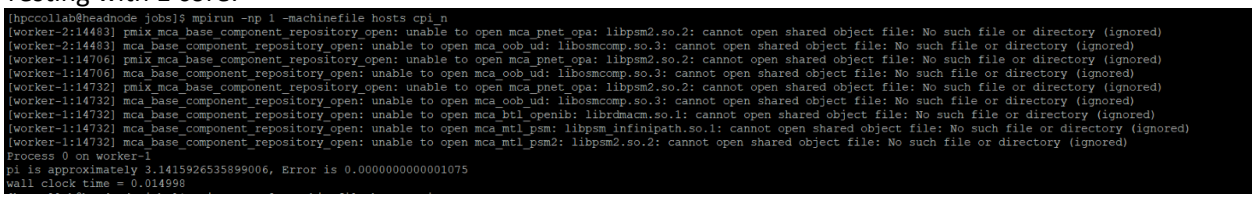

Figure 31 one core

Testing with 2 cores:

Figure 32 two cores

Testing with 3 cores:

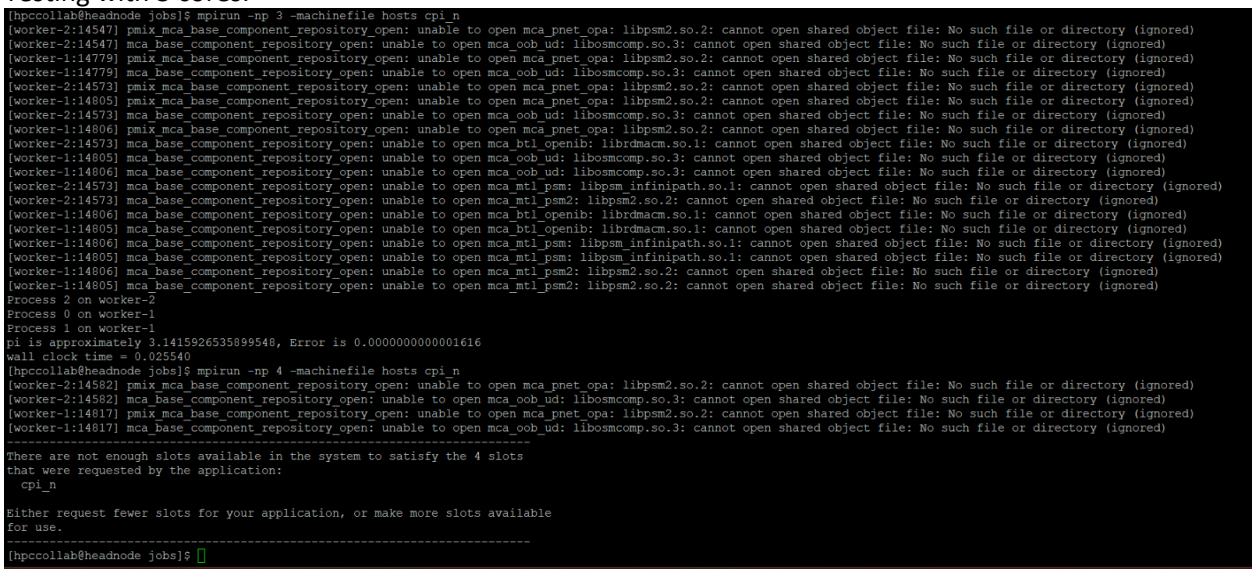

Figure 33 three cores

Couldn't run the 4 cores, which is the same as the colleague's result in his cluster.
The table below shows the comparison between the results in my cluster and the results in the colleague's cluster for the cpi_n program.

Node	Pi calculation	Error	Time
1	3.1415926535899006	0.0000000000001075	0.015034
2	3.1415926535899179	0.0000000000001248	0.007657
3	3.1415926535899548	0.0000000000001616	0.033012
4	3.1415926535899521	0.0000000000001590	0.034606

Table 2 Results in my cluster

Node	Pi calculation	Error	Time
1	3.1415926535899006	0.0000000000001075	0.014998
2	3.1415926535899179	0.0000000000001248	0.007608
3	3.1415926535899548	0.0000000000001616	0.025540

Table 3 Results in the colleague's cluster

Because of the limitation in the colleague's cluster, only compared the results from node 1 to node 3. Also, from the tables above, the only difference in our results is the execution time.

node	Execution time of my Cluster	Execution time of the colleague's cluster	Differences
1	0.015034	0.014998	+0.000036
2	0.007657	0.007608	+0.000049
3	0.033012	0.025540	+0.007472

Obviously, the execution time of my cluster was a bit longer than the colleague's cluster.

7 Discussion

From the results I learnt:

- Performance gains from parallel computing:

  When using different numbers of cores, the execution time changes. For example, in my results, the change from 1 core to 2 cores might bring a significant improvement, but the change from 3 to 4 cores might be a little. Also, when running the program in 4 cores, the execution time decreased in node 2 and increased in nodes 3 and 4, which was not as expected.

- Performance comparison between systems

  When using different types of GPU sizes, the accuracy, speed of the result and the maximum number of cores the cluster can handle might be different, which is most evident in complex programs.

Errors met and reasons for errors

The execution time of each node was not always the same and faster with the increase in nodes. when running the same MPI program. The reasons might be the network delay, Cache Effects and the performance of the MPI environment itself.

Could the accuracy of the experiment be increased?

The accuracy of the experiment could be increased by more repetition, controlling the same software and hardware environment, benchmarking and using more than 4 nodes in each experiment.

Reasons for different execution time in benchmarking comparison

It might be caused by variations in network latency, system load, or even subtle configuration differences within the Azure environment that impact the efficiency of parallel computations.

Future work

Extend the tests to more nodes and a broader range of processor configurations to understand the scalability limits of the MPI. Also, start exploring different types of libraries (e.g., OpenMP, CUDA for GPUs) based on the understanding of MPI.

8 Conclusion

The deployment of the Virtual Computer Cluster (VCC) was successfully completed with few errors, with MPI codes effectively executed and performance benchmarks indicating potential areas for improvement. Future work will concentrate on exploring the scalability of MPI and other practical libraries in clusters.

9 Reference

Burkardt, J. (n.d.). Source code for SAXPY operation using OpenMP. Retrieved April 21, 2024, from https://people.math.sc.edu/Burkardt/c_src/openmp/saxpy_openmp.c