Implementation of Local Beowulf Cluster and Message Passing Interface

Table of Content

Abstract

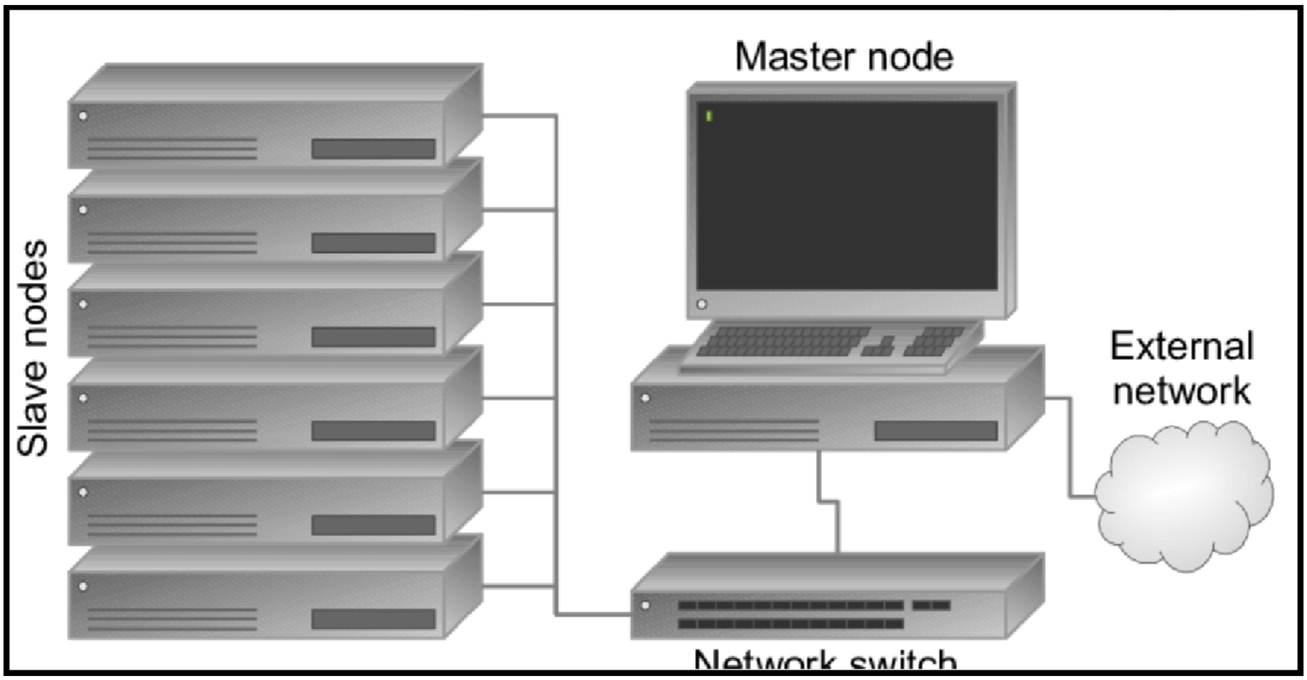
The project aims to establish a local Beowulf Cluster between a Laptop and a Desktop Computer running Ubuntu Desktop and further analysis of the Networking Components using Scapy (Python Library for Network Monitoring) and MPI (Message Passing Interface) for running Parallel Programmes.

For implementation of a high-complexity process, one can opt for the implementation of a high performance cluster architecture that is a set of computers interconnected to a local network. This set of computers try to give a unique behavior to solve complex problems using parallel computing techniques. The intention is to reduce the time directly proportional to the number of machines, giving a similarity of having a low-cost supercomputer.

We decided to use a laptop as Master and PC as Slave Node in the Local Beowulf Cluster.

Introduction

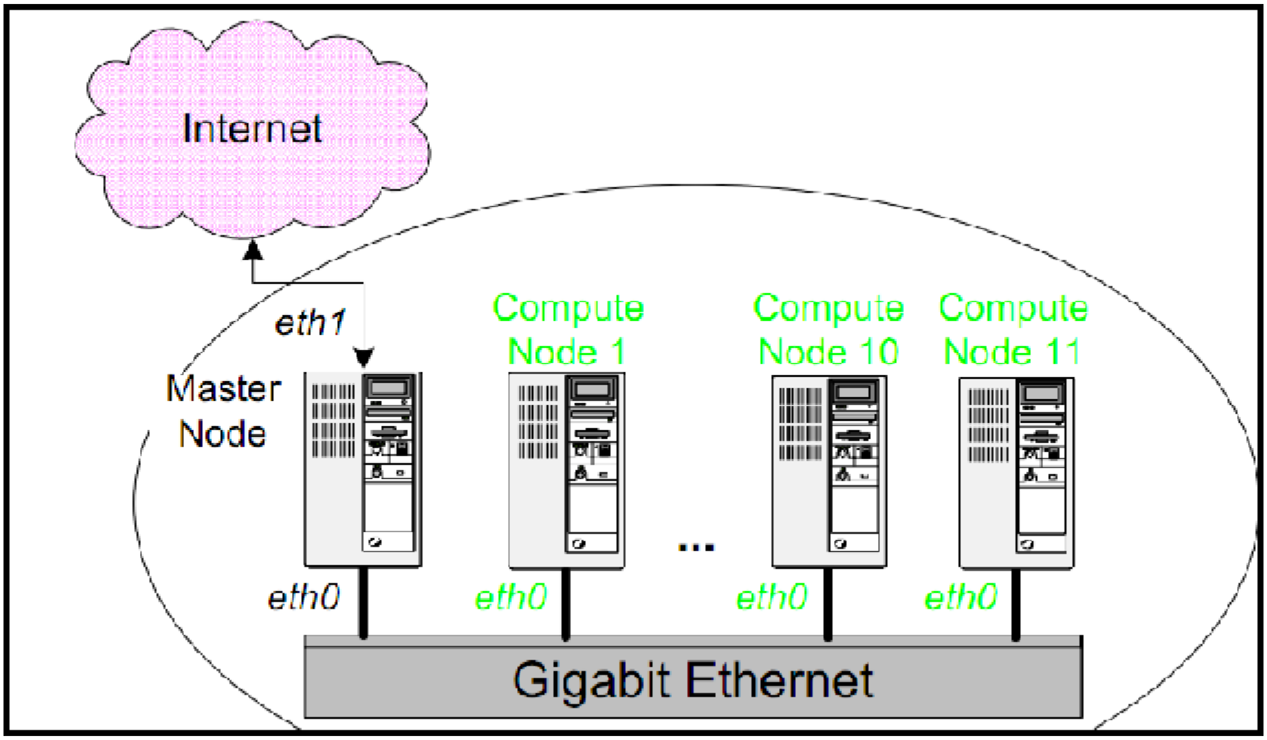
1. Beowulf Clusters are a group of Identical, commercially available computers, that run and operate on Free and Open Source Softwares generally like BSD, Solartis, Unix and in our case we have used GNU/Linux Debian derived Ubuntu Desktop. All the individual nodes are networked into a small TCP/IP LAN and have libraries and programs installed which allow processing to be shared among them.



2. The most popular ways of building these Beowulf Clusters are using Beowulf Toolkits like NPACI Rocks Toolkit and OSCAR Toolkit by Red Hat Linux Foundations, these toolkits are intuitive to use and deploy clusters and require minimal expertise from the end user.

3. So as part of this Innovative MTE Project evaluation, we decided to build one such cluster from scratch by manually setting up both the machines by having them run Ubuntu desktop and make them share the same Home Directory NFS and labelling Laptop as Master Node in the Local Cluster and PC as Slave Node.

4. In its simplest form, any Beowulf Cluster is a multi-computer architecture which can be used for parallel computations. The system thus formed consists of a pre-labeled Server/Master Node and at least one client node connected via ethernet or some other network components, like any simple PC running on Unix type OSes.



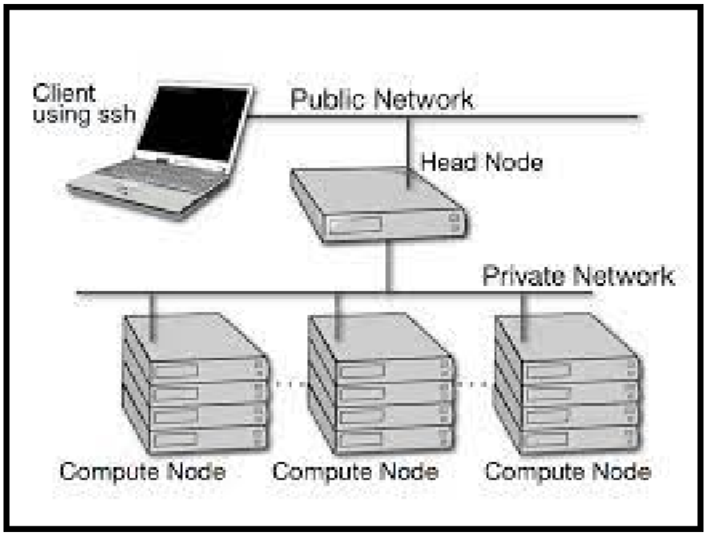
5. A term which is often confused with Beowulf Clusters is Cluster of Workstations (COW) is that Beowulf clusters together form a Single Machine rather than many different workstations, generally the client nodes do not need to have a full setup or dedicated PC components, the client nodes just need to be setup for running over the LAN using Ethernet Cable.

6. The Beowulf Nodes needs to have CPU + Memory Packages which can be plugged into the cluster, just like a CPU or memory module can be plugged into a motherboard.

7. There are several specific softwares for running Beowulf Clusters like Kernel Patches, PVM and MPI libraries, configuration tools which can make the architecture work really fast, and emulate behaviour of a low-cost Supercomputer. The idea here is to have simply two Networked Computers without any additional softwares, sharing /home file system via NFS and that trust each other to execute remote shells (rsh), that will complete our local Beowulf Cluster.

8. NFS stands for Network File System which is a distributed File System Protocol, allowing a user on a client computer to access files over a computer network much like local storage is accessed. NFS, like many other protocols, builds on the Open Network Computing Remote Procedure Call (ONC RPC) system. Remote Shells (rsh) are command line computer programs that can execute shell commands as another use and on another computer through the use of a computer network.

Methodology



1. The cluster consists of the following hardware parts : Network, Master/Server Node, Computer Node and a Gateway.

2. After getting all hardware components ready, we should be configuring the LAN and creation of Nodes, which should add nodes to the hosts files, so that we need not to type in IP addresses each time but only the username.

3. Then we should ping from the master nodes to all nodes to check whether the data and acknowledgement is being received or not.

4. Defining and setting up MPI Jobs.

5. Installation and setting up NFS (Network File System), so that all connected Slave Nodes have access to some part of the File System on the master node.

6. Next step is setting up Secure Remote Shells for communication between nodes and setting up Process Manager and Message Passing Interface Processes.

7. Running Jobs and analyzing the parallel computations using the clusters instead of a single computer.

Implementation

Computer (Slave Node)

Laptop (Master Node)

Network

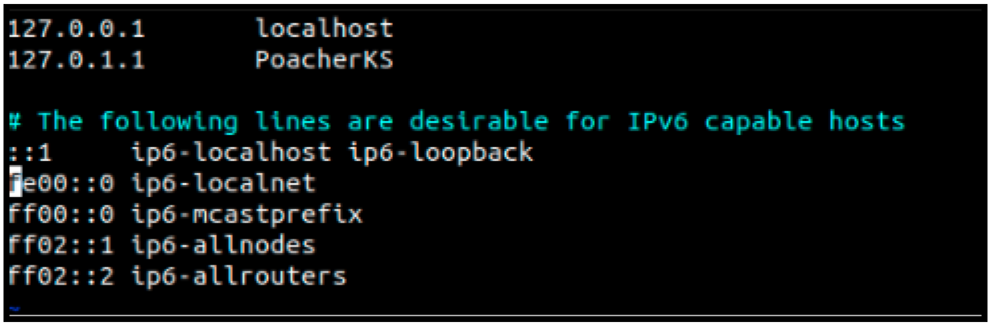
Routers (LAN)

1. All the required hardware components were set up, Laptop is running on Ubuntu Desktop Operating System and since the computer is running on Windows 10, we installed Windows Subsystem for Linux/Ubuntu to emulate the behaviour of GNU/Linux Operating System. For internet connectivity with the outer Public Network we used Wifi and for Private connection or LAN we used D-Link Router and Cat5 10/100 Mbps cables for connecting to the Public Network and for routing packets to Master and Slave Node.

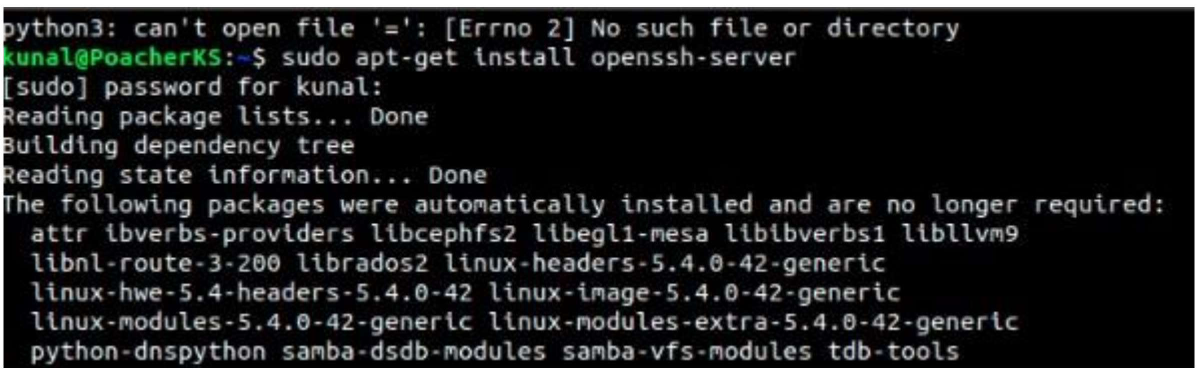
```
milk.txt                                xaa
music                                   ytrewq.txt
kunal@PoacherKS:~$ sudo -i
root@PoacherKS:~# sudo adduser clusterMaster
adduser: Please enter a username matching the regular expression configured
via the NAME_REGEX[_SYSTEM] configuration variable.  Use the `--force-badname'
option to relax this check or reconfigure NAME_REGEX.
root@PoacherKS:~# sudo adduser 'clusterMaster'
adduser: Please enter a username matching the regular expression configured
via the NAME_REGEX[_SYSTEM] configuration variable.  Use the `--force-badname'
option to relax this check or reconfigure NAME_REGEX.
root@PoacherKS:~# sudo adduser master
Adding user `master' ...
Adding new group `master' (1004) ...
Adding new user `master' (1004) with group `master' ...
Creating home directory `/home/master' ...
Copying files from `/etc/skel' ...
Enter new UNIX password:
Retype new UNIX password:
passwd: password updated successfully
Changing the user information for master
Enter the new value, or press ENTER for the default
        Full Name []: beowulfMasterNode
        Room Number []:
        Work Phone []:
        Home Phone []:
        Other []:
Is the information correct? [Y/n]
root@PoacherKS:~# ls /etc/shadow
/etc/shadow
```

```
kunal:$6$W.ZyNyjs$j0uxlUxEb8xRl5Zqic8oSe3/1r6v.1fVz6CTKXC
5kCop7AFcMcH0:18370:0:99999:7:::
flinuxmaster:!:18459:0:99999:7:::
kos:!:18462:0:99999:7:::
beowulfUser:$6$NV.AlXfk$//xYu6VOUF8K8OsLBMgzCZTGIcWqRdB66
BmJGhxbQ8AOtIsywMf0:18693:0:99999:7:::
sshd:*:18693:0:99999:7:::
master:$6$lFBFwx4c$3tuauV2gAMKcR3pxgJTndMppQ3icPTgKZ7.bec
/C4Iv2haOWJOA1:18731:0:99999:7:::
(END)
```

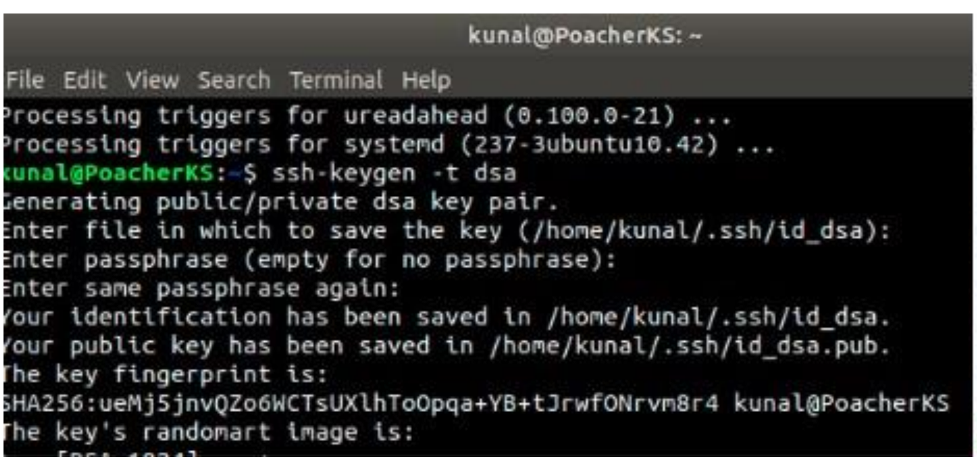
2. We added new users on both master and slave nodes as “master” and “slave” using [sudo useradd <username>] command and setting up the process for them, here you can see the snapshot of Master Node User creation and setup. Similarly another user as “slave” is created on Windows 10 PC using WSL for GNU/Linux Ubuntu. We issued ping commands like ping <ipv4 address>, and the PC (slave) returned us 52 bytes of data. This confirms that LAN is established and our nodes are interacting.

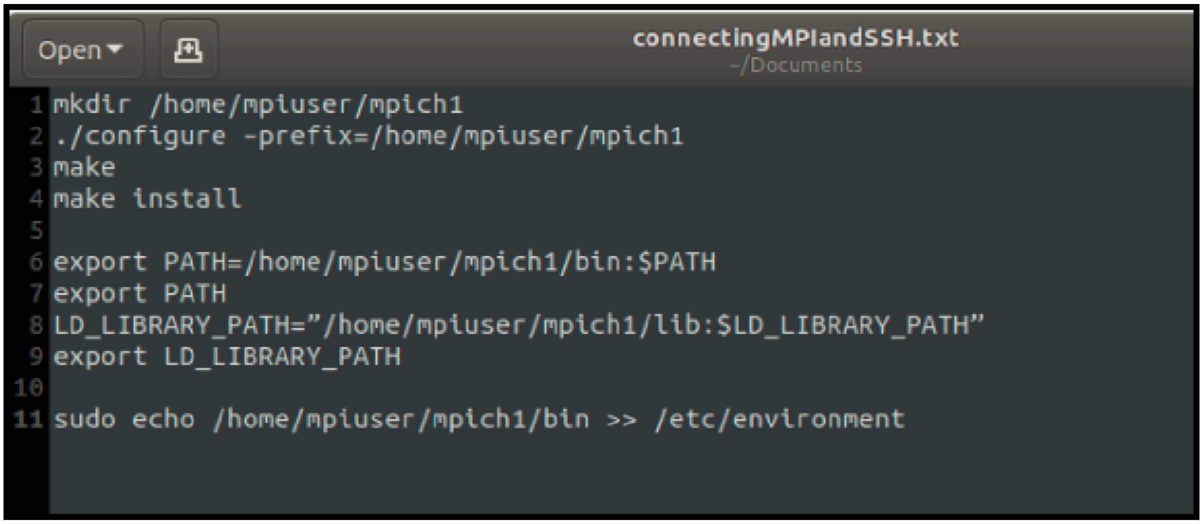


3. Then we configured the static IP addresses on both machines running Ubuntu Desktop OS, using YAML we configure the Network File and extract the file to know more information about our Ethernet cable link. This helped us to issue commands using the computer names and not the IP addresses since in the local architecture they are now behaving as Static IP so we can use node’s name instead of IP.



4. We installed Open-SSH Server in both nodes which is a tool for remote control and transfer of data over two networked nodes, since the Master and Slave nodes are in LAN connection, we can use SSH to pass on any message and transfer files from one node to another node (N2N) communication. Then we assigned both nodes Secure Key using the **ssh-key-t-dsa** commands, this key is used for authentication purposes.



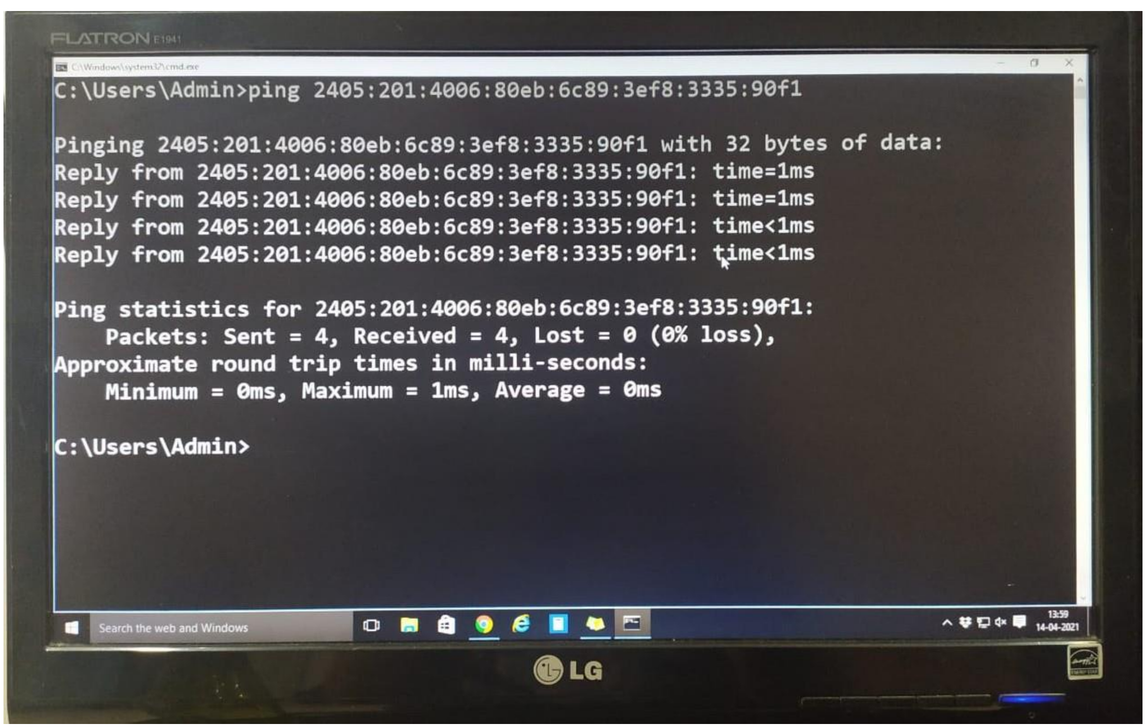



```
mkdir /home/mpiuser/mpich1
./configure -prefix=/home/mpiuser/mpich1
make
make install

export PATH=/home/mpiuser/mpich1/bin:$PATH
export PATH
LD_LIBRARY_PATH="/home/mpiuser/mpich1/lib:$LD_LIBRARY_PATH"
export LD_LIBRARY_PATH

sudo echo /home/mpiuser/mpich1/bin >> /etc/environment
```

5. Then we installed **MPICH2 which is the MPI-2 Implementation from UNIX labs**, the process involved downloading of C compiler (gcc) and the .tar file from UNIX labs website, we downloaded the files and using **mpich2-install commands**, installed the library and then we ran the script displayed above for connecting MPI and SSH, so that we can use this MPI generated to pass on the messages to other networked node which is using Secure Remote Shells Login we created above. Using VIM text editor in Terminal we went to file in FS in /etc/hosts dir and added the IP of the to be passed node, in our case it is Computer (Slave Node) and then in Computer we used command **mpd -h <master> -p 4000** & to login into our master node forms server node. Then any file can be sent over using the terminal command **mpiexec -n 4 /bin/hostname.**



6. When we tried issuing commands from MPI and SSH to our Slave Node from Master node data is received in packets of size 32 Bytes, to confirm whether the data being sent over is from Slave node, we tried sending data from Slave node and the result was Successful, indeed the nodes were connected as in LAN and MPI-SSH was working.

```
master:~$ sudo apt-get install nfs-kernel-server
$ sudo apt-get install nfs-common
master:~$ ls -l /home/ | grep mpiuser
master:~$ sudo chown mpiuser:mpiuser /path/to/shared/dir
/home/mpiuser *(rw,sync,no_subtree_check)
master:~$ sudo service nfs-kernel-server restart
master:~$ sudo ufw allow from 192.168.1.0/24
$ ls /home/mpiuser
```

7. Now to access files and not just send data we used the above commands as mentioned in the Blog Post on Building Cluster by Serrano Pereira in order to share the NFS /home directory.

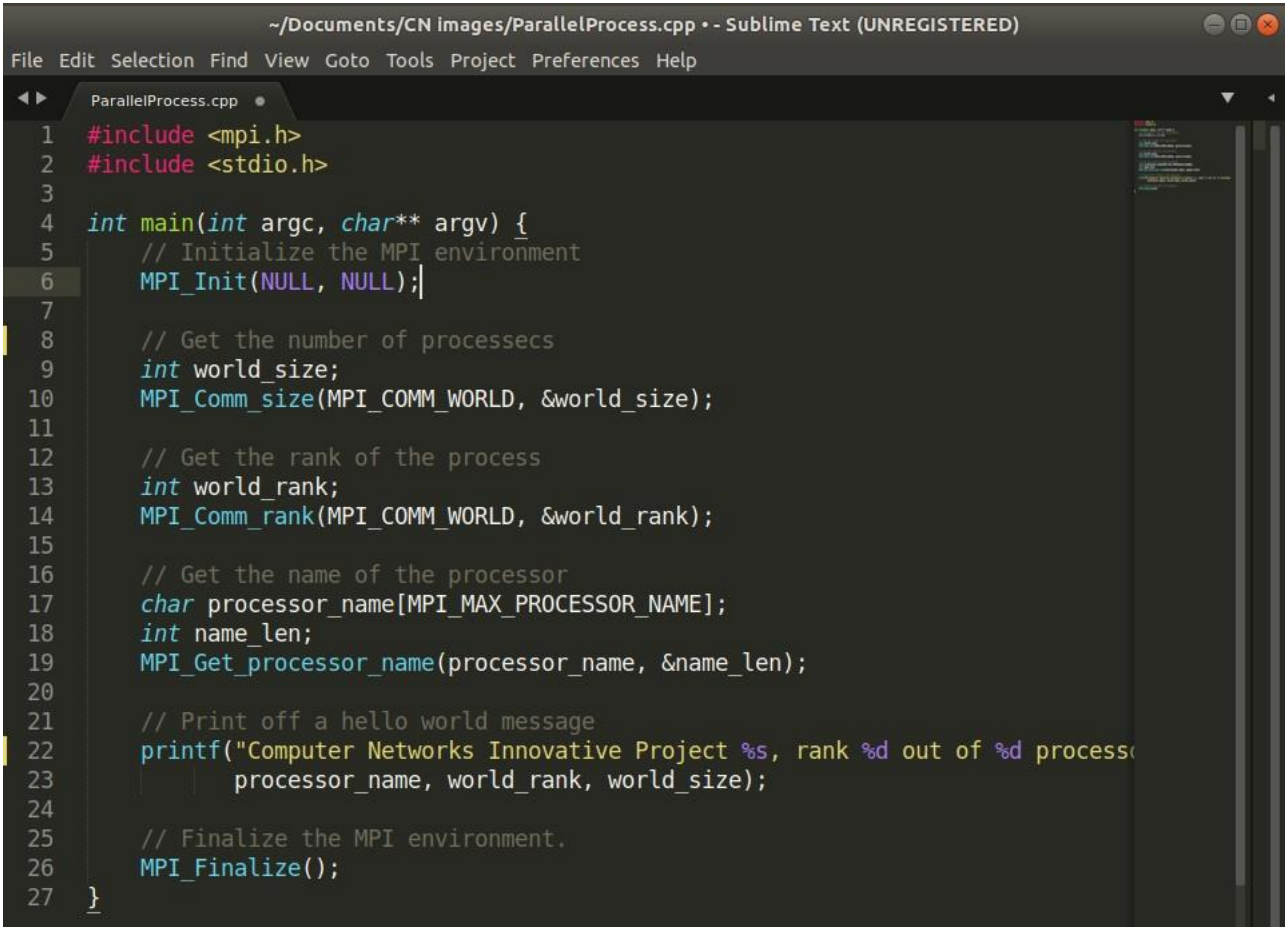
```
kunal@PoacherKS: ~/Documents/CN images
File  Edit  View  Search  Terminal  Help

  Other Hydra options:
    -verbose                         verbose mode
    -info                            build information
    -print-all-exitcodes             print exit codes of all processes
    -iface                           network interface to use
    -ppn                             processes per node
    -profile                         turn on internal profiling
    -prepend-rank                    prepend rank to output
    -prepend-pattern                 prepend pattern to output
    -outfile-pattern                 direct stdout to file
    -errfile-pattern                 direct stderr to file
    -nameserver                      name server information (host:port format)
    -disable-auto-cleanup            don't cleanup processes on error
    -disable-hostname-propagation    let MPICH auto-detect the hostname
    -order-nodes                     order nodes as ascending/descending cores
    -localhost                       local hostname for the launching node
    -usize                           universe size (SYSTEM, INFINITE, <value>)

Please see the intructions provided at
http://wiki.mpich.org/mpich/index.php/Using_the_Hydra_Process_Manager
for further details

kunal@PoacherKS:~/Documents/CN images$
```

8. Since both nodes are having their independent processes being run, we need some sort of process manager so we used Hydra Process manager which is configured for uses of MPICH.



```
~/Documents/CN images/ParallelProcess.cpp • - Sublime Text (UNREGISTERED)
File Edit Selection Find View Goto Tools Project Preferences Help
ParallelProcess.cpp

#include <mpi.h>
#include <stdio.h>

int main(int argc, char** argv) {
    // Initialize the MPI environment
    MPI_Init(NULL, NULL);

    // Get the number of processecs
    int world_size;
    MPI_Comm_size(MPI_COMM_WORLD, &world_size);

    // Get the rank of the process
    int world_rank;
    MPI_Comm_rank(MPI_COMM_WORLD, &world_rank);

    // Get the name of the processor
    char processor_name[MPI_MAX_PROCESSOR_NAME];
    int name_len;
    MPI_Get_processor_name(processor_name, &name_len);

    // Print off a hello world message
    printf("Computer Networks Innovative Project %s, rank %d out of %d process
            processor_name, world_rank, world_size);

    // Finalize the MPI environment.
    MPI_Finalize();
}
```

9. The penultimate step now is to run a parallel process and test whether the parallel processing has helped us to get better in computation and solve more challenging tasks by creating a low-cost supercomputer. For selecting of what process we needed, we wrote a small C++ Source Code which runs and does the same thing from different sources that finally prints the rank of the same process executed from different sources, we took 5 cases out of which the 3rd case was when we passed the file using MPI, that processes executed the fastest and took the least amount of time as compared to the time taken by the same process when executed solely on Master or Slave Node.

To run the above mentioned code, we installed from the help of package manager mpich code runner which builds and executes the C++ Code using Mingw g++ compiler and MPI.

> mpicc ParallelProcess.cpp -o ParallelProcess

> mpirun -np 5 ./ParallelProcess

```
$ s ud o apt  in s ta 11 npich
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following packages were automatically installed and are no longer required:
  attr ibverbs-providers libcephfs2 libegll-mesa libibverbsl libllvm9
  libnl-route-3-200 librados2 linux-headers-5.Q.0-66-generic
  fin u x - h we - 5. 4 - h ea der s - 5.  4.  0 - 42 1Tnu x - h we - 5.  4 - h ea d er s - 5.  4.  0 - 5 3
  fin  u x - h we - 5.  4 - hea der  s - 5. 4. 0 - 66 1Tnu x - Snag e - 5. 4. 0 - 66 - g ener be
  linux - nodu1es - 5.  4.  0 - 66 - gener Tc  linux - nodule s - ext ra - 5.  4.  0 - 66 - gene r be
  python - dnspython  sanba - dsdb - nodu1es  sanba - vfs   - nodu1es tdb - tools
Use  ' sudo apt  au tor enove '   to   r enove  then.
The f ollowing a ddLtd ona1 packages will be ins tailed :
  g fo r l r an g to r l r an - 7  hbloc - n ox libcr - dev libcr0  1Tbg fo r l r an - 7 - dev
  11bh bloc - plugins 1Tbh bloc 5 1 TbnpT ch - dev libnpTch 12 oc1 - dcd - tTbopen ct 1
Suggested  packages :
  gfo r t ran - nultTtTb  gfo r t r an - doc  gfo r t r an - 7 - nuttilTb  gfo r t r an - 7 - doc
  libgfort ran4 - dbg lTbcoa r rays - dev btcr - dkns 11bh toe - cont r lb  - plugins
  b1c r - utT1  npTch - doc  opencl - dcd
The following NEW packages will be Instaited:
  gfortran gfor Iran - 7 huloc - nox fiber - dev 11bcr0  lTbgfo rtran - 7 - dev
  11bhbloc - plugins  1Tbhbloc5  1 TbnpTch - dev  libnpTch12  npTch  oct - icd   - libopen ct 1
0 upgraded, i2 newly installed, 0 to remove and 83 not upgraded.
Need  to  get  12. 4  MB  of   a r chIves .
```

```
$ npTcc Pa r alIelProces s . cpp - o Para 11e1Proc
```

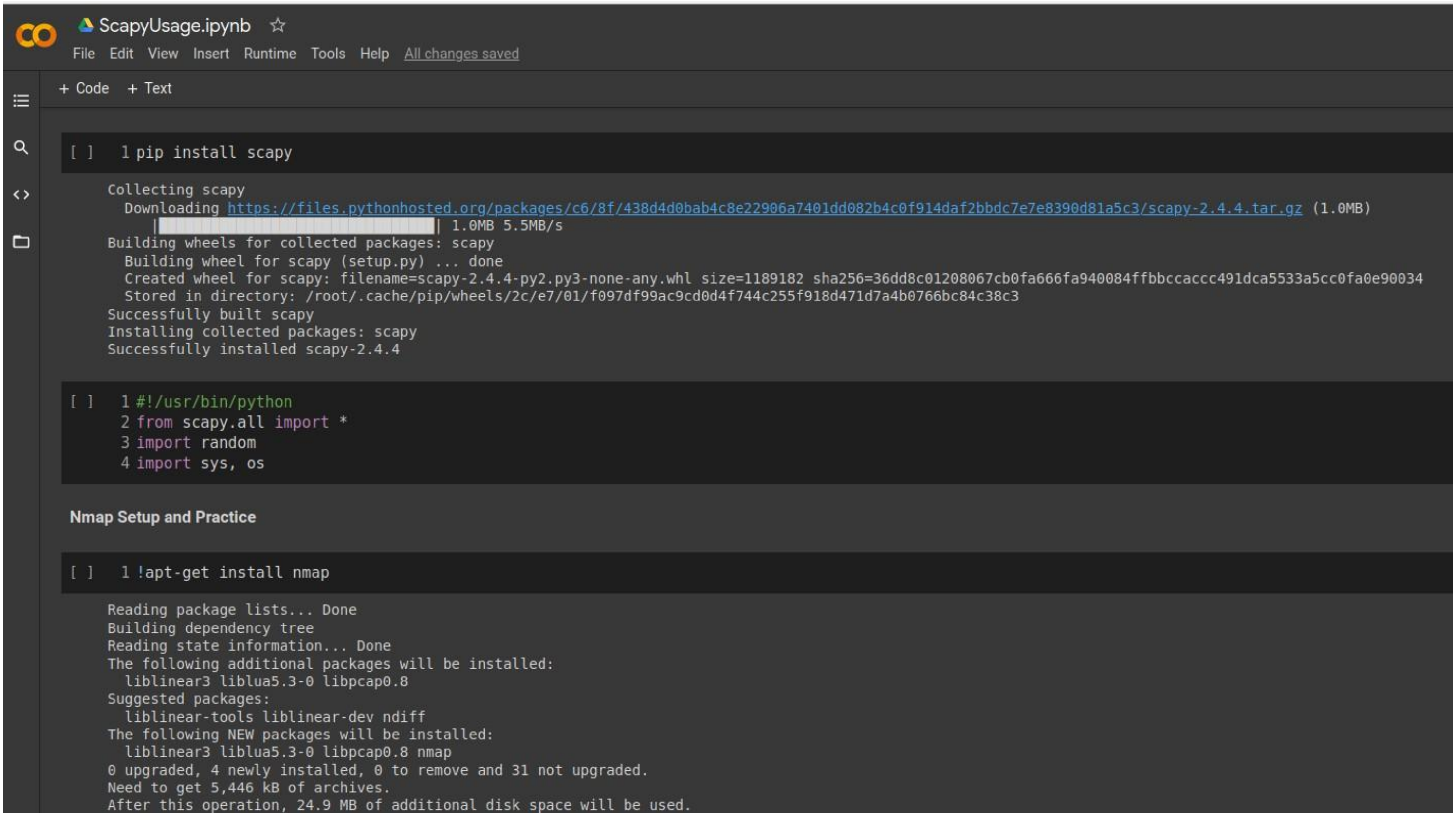
```
$ mpirun -np 5 ./ParallelProcess
Computer Networks Innovative Project PoacherKS, rank 3 out of 5 processors
Computer Networks Innovative Project PoacherKS, rank 4 out of 5 processors
Computer Networks Innovative Project PoacherKS, rank 0 out of 5 processors
Computer Networks Innovative Project PoacherKS, rank i out of 5 processors
Computer Networks Innovative Project PoacherKS, rank 2 out of 5 processors
$ npTexec   help
```

```
Usage: . /npTexec [globa1 opts] [toca1  opts      for   exec1]    exec1]    exec1  arg s]   :   [to
cat   opts for     exec2]   [exec2]   [exec2 a rgs ]

Global options (passed to all executables):

  Global environment options:
    - g en v (n ane} (va 1u e}                    env?ronnenl   va r Table n ane and va1ue
    - genv1?sl   (en v1, en v2, ...   }           env?ronnenl   va r la  b1e lts  I   to pa s s
    - genvnone                                    do not pas s any envlr onnent var tables
    - genva ll                                    pa ss  all    envy ronnenl   va r tables  n ot   nan aged
                                                        by th e launch er ( default)

  Other global option s :
```



ScapyUsage.ipynb
File Edit View Insert Runtime Tools Help All changes saved
+ Code + Text

```
pip install scapy
```

```
Collecting scapy
  Downloading https://files.pythonhosted.org/packages/c6/8f/438d4d0bab4c8e22906a7401dd082b4c0f914daf2bbdc7e7e8390d81a5c3/scapy-2.4.4.tar.gz (1.0MB)
     |████████████████████████████████| 1.0MB 5.5MB/s
Building wheels for collected packages: scapy
  Building wheel for scapy (setup.py) ... done
  Created wheel for scapy: filename=scapy-2.4.4-py2.py3-none-any.whl size=1189182 sha256=36dd8c01208067cb0fa666fa940084ffbbccaccc491dca5533a5cc0fa0e90034
  Stored in directory: /root/.cache/pip/wheels/2c/e7/01/f097df99ac9cd0d4f744c255f918d471d7a4b0766bc84c38c3
Successfully built scapy
Installing collected packages: scapy
Successfully installed scapy-2.4.4
```

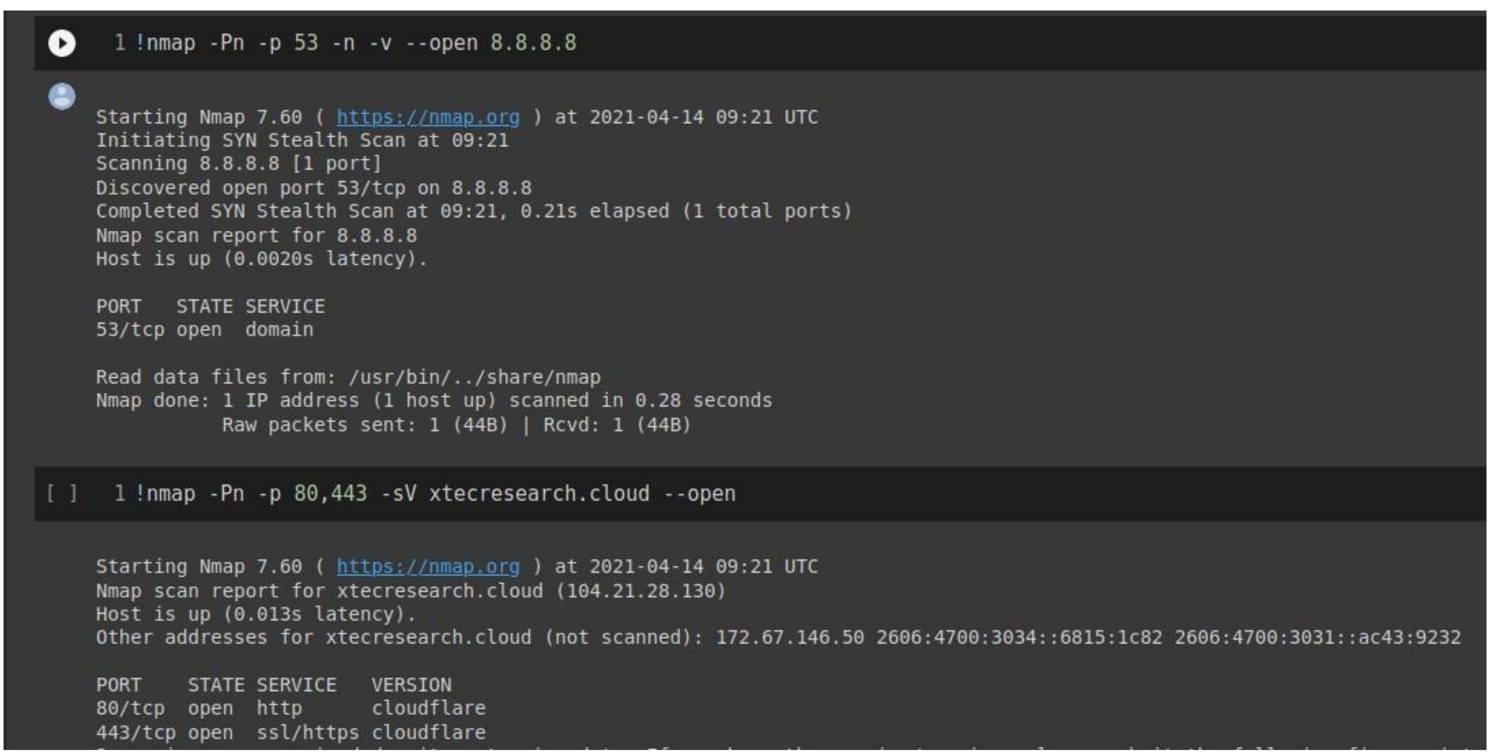
```
#!/usr/bin/python
from scapy.all import *
import random
import sys, os
```

Nmap Setup and Practice

```
!apt-get install nmap
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following additional packages will be installed:
  liblinear3 liblua5.3-0 libpcap0.8
Suggested packages:
  liblinear-tools liblinear-dev ndiff
The following NEW packages will be installed:
  liblinear3 liblua5.3-0 libpcap0.8 nmap
0 upgraded, 4 newly installed, 0 to remove and 31 not upgraded.
Need to get 5,446 kB of archives.
After this operation, 24.9 MB of additional disk space will be used.
```

10. The final step of Project is to use the Python Scapy library to inspect the network elements. In this step we worked using Google Colab Notebook to do the network tests. We used Wireshark which is a famous Network Protocol Analyzer Tool.



```
!nmap -Pn -p 53 -n -v --open 8.8.8.8
```

```
Starting Nmap 7.60 ( https://nmap.org ) at 2021-04-14 09:21 UTC
Initiating SYN Stealth Scan at 09:21
Scanning 8.8.8.8 [1 port]
Discovered open port 53/tcp on 8.8.8.8
Completed SYN Stealth Scan at 09:21, 0.21s elapsed (1 total ports)
Nmap scan report for 8.8.8.8
Host is up (0.0020s latency).

PORT   STATE SERVICE
53/tcp open  domain

Read data files from: /usr/bin/../share/nmap
Nmap done: 1 IP address (1 host up) scanned in 0.28 seconds
           Raw packets sent: 1 (44B) | Rcvd: 1 (44B)
```

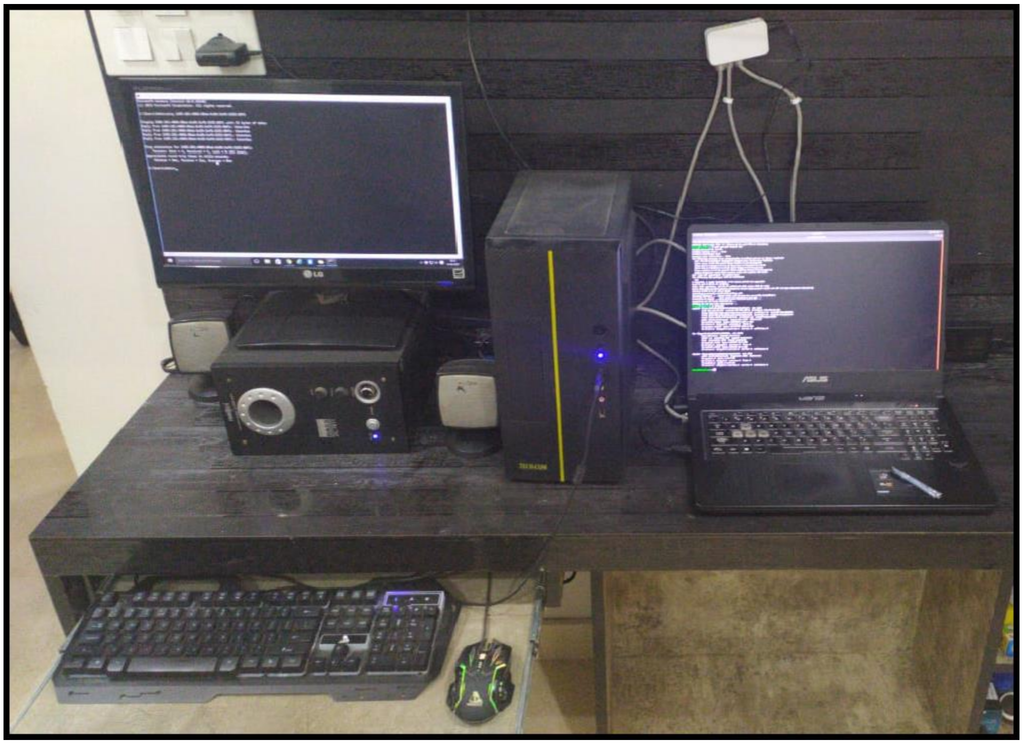
```
!nmap -Pn -p 80,443 -sV xtecresearch.cloud --open
```

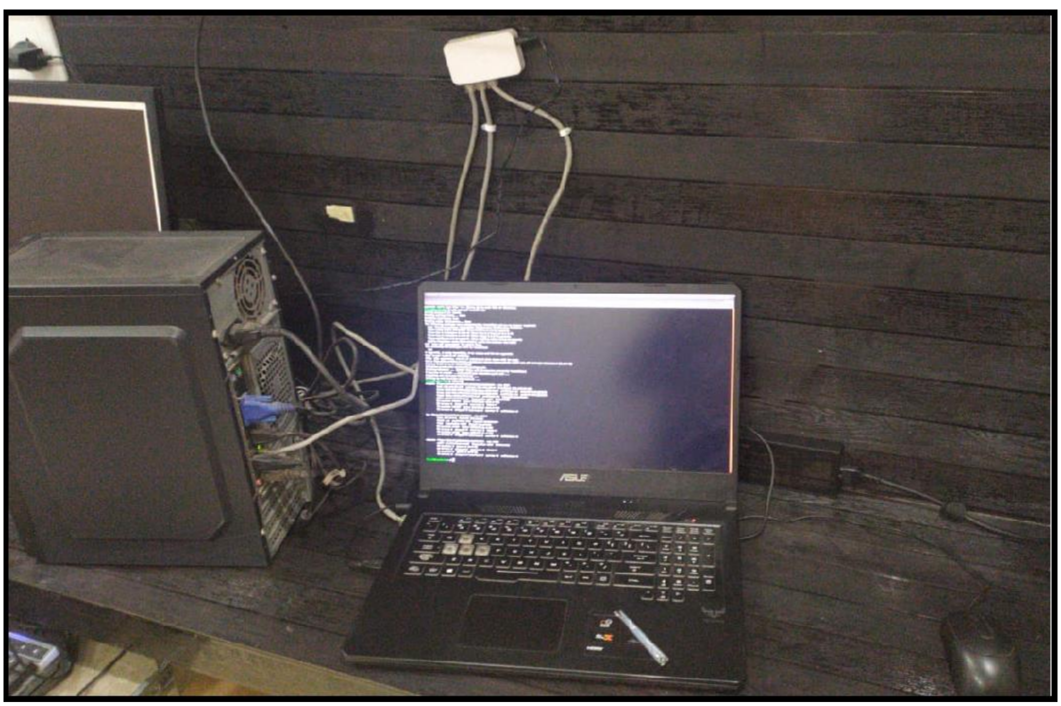
```
Starting Nmap 7.60 ( https://nmap.org ) at 2021-04-14 09:21 UTC
Nmap scan report for xtecresearch.cloud (104.21.28.130)
Host is up (0.013s latency).
Other addresses for xtecresearch.cloud (not scanned): 172.67.146.50 2606:4700:3034::6815:1c82 2606:4700:3031::ac43:9232

PORT    STATE SERVICE   VERSION
80/tcp  open  http      cloudflare
443/tcp open  ssl/https cloudflare
```

Results

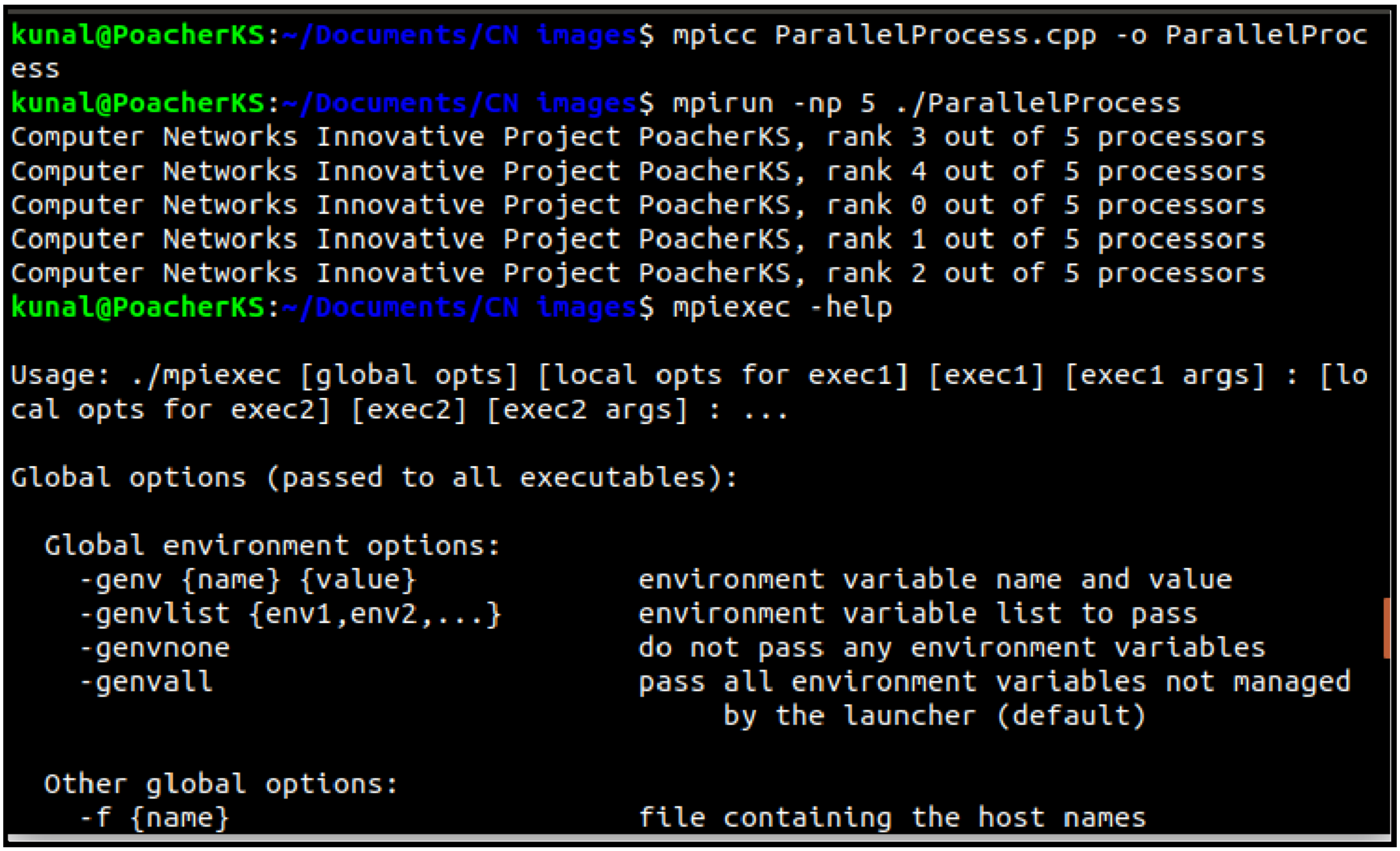
1. In this project we aimed to establish a local Beowulf Cluster and using MPI and SSH aimed at creating a low-cost supercomputer that can run the processes faster then what single machines can do, after the implementation phase we have completed the procedure and achieved faster computation speed.





2. The above shown picture is of the final Beowulf Cluster we created having Laptop as Master and PC as Slave Node in the Local Cluster.

3. We were able to setup the MPICH2 on Laptop and Computer and Secure Shell on both nodes too, then we passed the information using Host names since we have already allocated Static IP addresses to both of them previously, the connection we established was confirmed by using Scapy Library and Wireshark Network Protocol Analyzer tool which confirmed the ports that were open which was the same port we used for connection.



```
kunal@PoacherKS:~/Documents/CN images$ mpicc ParallelProcess.cpp -o ParallelProc
ess
kunal@PoacherKS:~/Documents/CN images$ mpirun -np 5 ./ParallelProcess
Computer Networks Innovative Project PoacherKS, rank 3 out of 5 processors
Computer Networks Innovative Project PoacherKS, rank 4 out of 5 processors
Computer Networks Innovative Project PoacherKS, rank 0 out of 5 processors
Computer Networks Innovative Project PoacherKS, rank 1 out of 5 processors
Computer Networks Innovative Project PoacherKS, rank 2 out of 5 processors
kunal@PoacherKS:~/Documents/CN images$ mpiexec -help

Usage: ./mpiexec [global opts] [local opts for exec1] [exec1] [exec1 args] : [lo
cal opts for exec2] [exec2] [exec2 args] : ...

Global options (passed to all executables):

  Global environment options:
    -genv {name} {value}             environment variable name and value
    -genvlist {env1,env2,...}        environment variable list to pass
    -genvnone                        do not pass any environment variables
    -genvall                         pass all environment variables not managed
                                          by the launcher (default)

  Other global options:
    -f {name}                        file containing the host names
```

4. Finally We ran a parallel process that prints a simple String literal in C++, in it we found that by using Cluster's computational power, the execution was way faster as compared to what happens when we ran the program individually on nodes. Hence the creation of our low-cost Supercomputer like Beowulf Cluster was a success.

Conclusion

1. To conclude the project, we can say that the execution of programmes that require parallel computations is much faster on a clustered system, like in the case of the above constructed Beowulf Cluster.

2. The programmes run much faster on them as compared to their time requirements on single node computers. Beowulf clusters can communicate among themselves using Open-SSH and Message Passing Interface which is a special tool for communication within nodes in a Private Network. Hence to increase productivity we can use 2 simple computers, merge them in a cluster and create a workable Supercomputer.

References

1. What’s a Beowulf ? Blog post.
2. Beowulf’ Good for ? Blog post.
3. Beowulf Cluster Computing with Linux. Book
4. Beowulf Project Overview. Blog.
5. Building Your own Beowulf Cluster. Wired Article.
6. Implementation of a Beowulf Cluster. Research Paper.