

# User Manual

for

## Programming Assignment 1

Manthan Patel  
(A-20413535)  
mpatel120@hawk.iit.edu

# Index

- 1) CPU Benchmark
  - a) Commands for compilation and running project
  - b) use of Makefile
  - c) SLURM Job command
- 2) Disk Benchmark
  - a) Commands for compilation and running project
  - b) use of Makefile
  - c) SLURM Job command
- 3) Memory Benchmark
  - a) Commands for compilation and running project
  - b) use of Makefile
  - c) SLURM Job command
- 4) Network Benchmark
  - 4.1) TCP Network Benchmarking
    - a) Commands for compilation and running project
    - b) use of Makefile
    - c) SLURM Job command
  - 4.2) UDP Network Benchmarking
    - a) Commands for compilation and running project
    - b) use of Makefile
    - c) SLURM Job command

# 1. CPU Benchmark

This benchmark has been developed in C language and used POSIX library for threading. The purpose of this benchmark is to test the cpu speed for very large number of arithmetic operations.

## Commands for compilation and running project

To compile the .c file on terminal the command is -

**gcc -Wall MyCPUBench.c -o MyCPUBench -lpthread**

and for running after successful compilation, the code for execution is -

**./MyCPUBench <\*.dat file name >**  
**or**

**./MyCPUBench**

(I have kept an else part for testing constant values in main method)

## Using Makefile

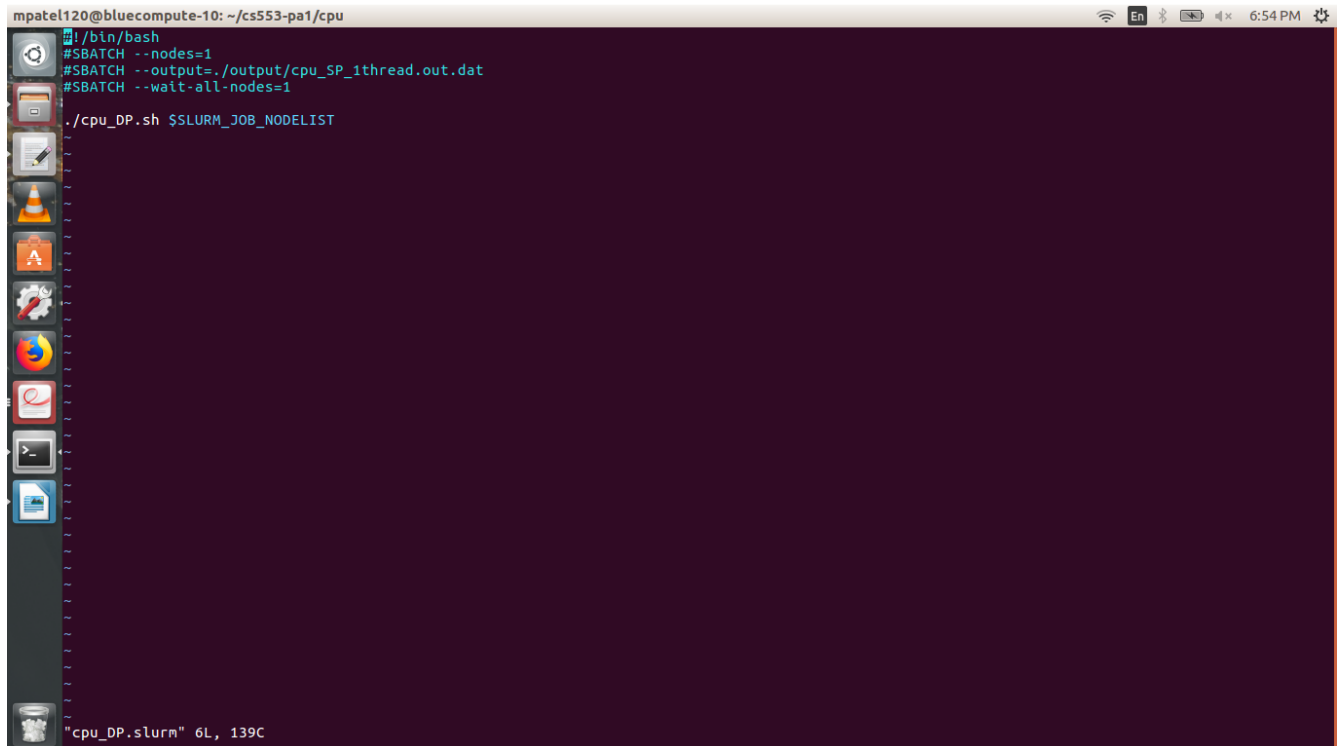
Use “**make all**” command in terminal and it will remove all .o files and executable file **MyCPUBench** . And then it will compile the code and create new **MyCPUBench** executable file.

**Note: The makefile was not deleting or not creating a correct executable file. Therefore, I have made changes in makefile. Still if the output dose not come in appropriate form use the command line compile command from above.**

# SLURM Job command

The command for slurm job is -

**sbatch <.slurm file name>**

A screenshot of a terminal window on a Linux system. The window title is 'mpatel120@bluecompute-10: ~/cs553-pa1/cpu'. The terminal shows the following commands and output:

```
# ./bin/bash
#SBATCH --nodes=1
#SBATCH --output=./output/cpu_SP_1thread.out.dat
#SBATCH --wait-all-nodes=1
./cpu_DP.sh $SLURM_JOB_NODELIST
```

The terminal background is dark purple. On the left side, there is a vertical dock with various application icons. At the bottom of the terminal window, a status bar shows '"cpu\_DP.slurm" 6L, 139C'. The system tray at the top right of the window shows icons for network, battery, and time (6:54 PM).

( SLURM file for CPU DP code )

I have written one “**run\_all\_cpus.sh**” file having command for all the batch files to run.

## 2. Disk Benchmark

This benchmark has been developed in C language with the help of POSIX threads for multi threading. This benchmark is for testing the speed for disk operations(read-write sequential/random) and the disk latency.

### Commands for compilation and running project

To compile the .c file on terminal the command is -

```
gcc -Wall -o MyDiskBench MyDiskBench.c -lpthread
```

and for running after successful compilation, the code for execution is -

```
./ MyDiskBench <*.dat file name >  
or
```

```
./ MyDiskBench
```

(I have kept an else part for testing constant values in main method)

### Using Makefile

Use “**make all**” command in terminal and it will remove all .o files and executable file **MyDiskBench** . And then it will compile the code and create new **MyDiskBench** executable file.

**Note: If the makefile dose not help in getting proper output. Please try to remove **MyDiskBench** executable and run the above compilation command.**

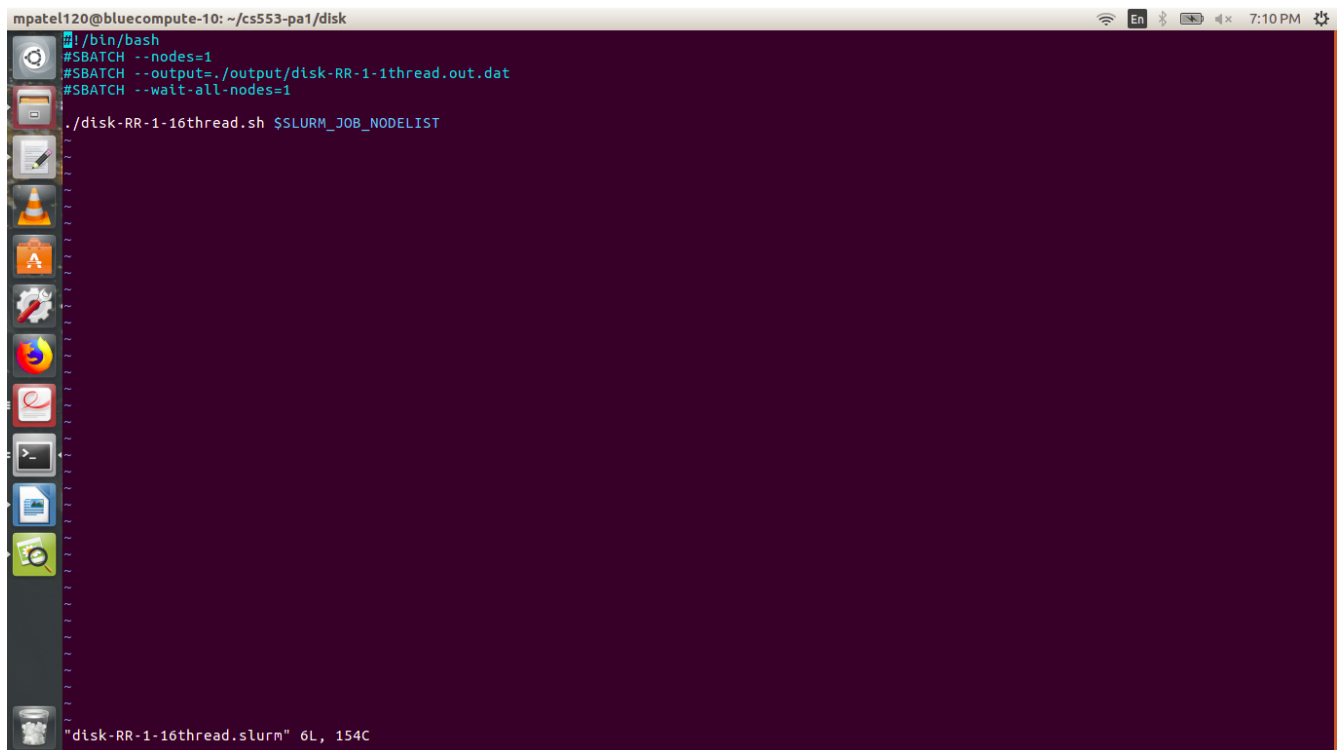
# SLURM Job command

The command for slurm job is -

**sbatch <.slurm file name>**

for example:

**sbatch disk-RR-1-1thread.slurm**

A terminal window screenshot showing a user named mpatel120 on a machine named bluecompute-10. The user has submitted a SLURM batch job using the command 'sbatch disk-RR-1-1thread.slurm'. The terminal output shows the job configuration: '#SBATCH --nodes=1', '#SBATCH --output=./output/disk-RR-1-1thread.out.dat', and '#SBATCH --wait-all-nodes=1'. The user then runs './disk-RR-1-16thread.sh \$SLURM\_JOB\_NODELIST'. The terminal status bar at the bottom indicates the job 'disk-RR-1-16thread.slurm' is running on 6L, 154C. The terminal window has a dark purple background and a sidebar with various application icons on the left. The top of the window shows the user's name, machine name, and the current directory path '~/.cs553-pa1/disk'. The system clock in the top right corner shows 7:10 PM.

( **SLURM file** - disk-RR-1-16thread.slurm )

I have written one “**run\_all\_jobs.sh**” file having command for all the batch files to run.

## 3. Memory Benchmark

I have implemented this benchmark to test the RAM read/write operation speed using different block size (1KB, 32KB,etc.). This benchmark is written in C language with POSIX library for threading.

### Commands for compilation and running project

To compile the .c file on terminal the command is -

```
gcc -Wall MyRAMBench.c -pthread -o MyRAMBench
```

and for running after successful compilation, the code for execution is -

```
./ MyRAMBench <*.dat file name >  
or  
./ MyRAMBench
```

(I have kept an else part for testing constant values in main method to reduce command line arguments)

### Using Makefile

Use “**make all**” command in terminal and it will remove all .o files and executable file **MyRAMBench** . And then it will compile the code and create new **MyRAMBench** executable file.

**Note:** If the makefile dose not help in getting proper output. Please try to remove **MyRAMBench** executable and run the above compilation command.

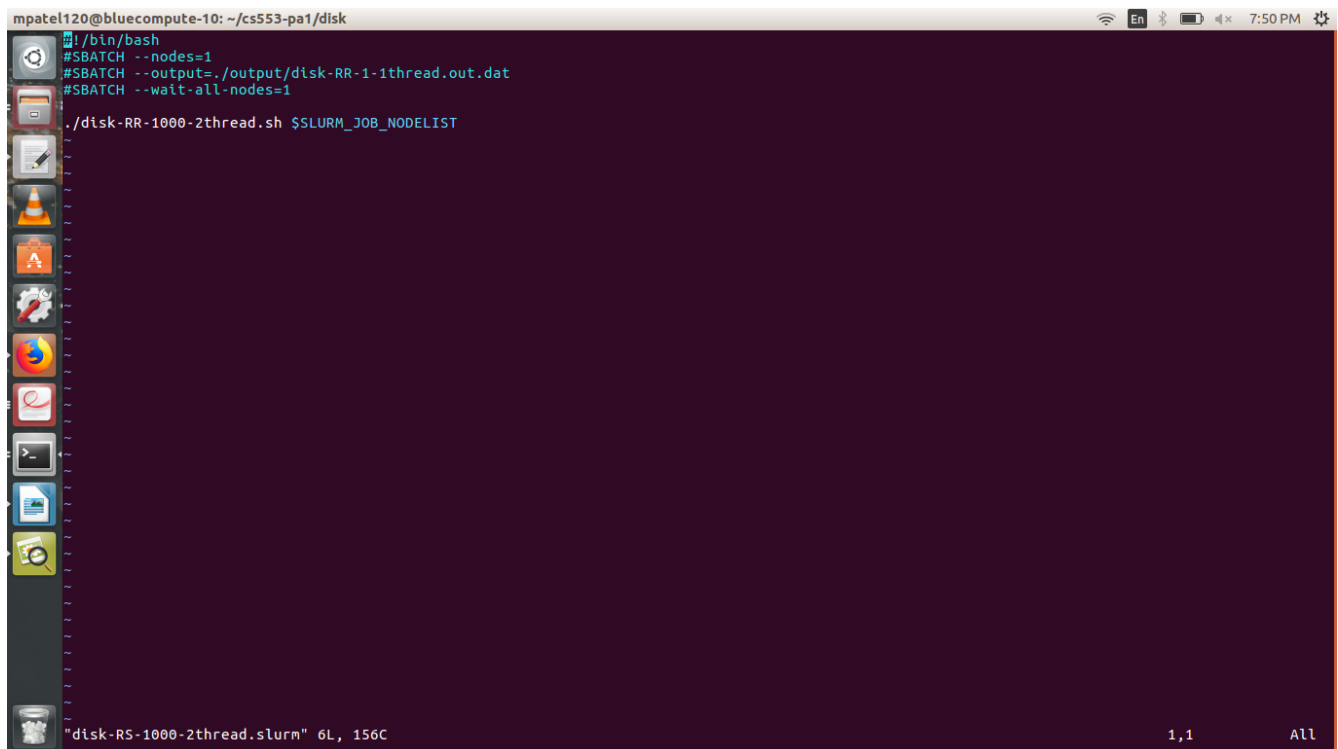
# SLURM Job command

The command for slurm job is -

**sbatch <.slurm file name>**

for example:

**sbatch memory-RWR-1.slurm**



```
mpatel120@bluecompute-10: ~/cs553-pa1/disk
# /bin/bash
#SBATCH --nodes=1
#SBATCH --output=./output/disk-RR-1-1thread.out.dat
#SBATCH --wait-all-nodes=1
./disk-RR-1000-2thread.sh $SLURM_JOB_NODELIST
```

**(SLURM FILE - disk-RS-1000-2thread.slurm)**

I have included one “**run\_all\_batch.sh**” file which has all the commands for job creation. It is executable, too.



## 4. Network Benchmark

This benchmark is used for testing of network speed and latency of round trip time (RTT) for a ping to given host. This benchmark has been written in C language and has included socket programming (using both UDP and TCP protocols) and POSIX library for multi threading.

Network Benchmark has main two part: client and server for which there will be a requirement of 2 concurrent nodes for each(client and server). Therefore, this benchmark runs a little differently. Also, the job creation will use 2 tasks and 2 nodes for running the benchmark.

### A) TCP Network Benchmark

#### Commands for compilation and running project

To compile the .c file on terminal the command is -

```
gcc -Wall -o MyNETBench-TCP MyNETBench-TCP.c -lpthread
```

After successful compilation, the code for execution on one terminal or node is -

```
./ MyNETBench-TCP server <*.dat file name >
```

and on other node or terminal

```
./ MyNETBench-TCP client <*.dat file name > localhost
```

(There will be total 4 command line arguments(including ./MyNETBench-TCP) for client and 3 for server.)

## Using Makefile

Use “**make all**” command in terminal and it will remove all .o files and executable file **MyNETBench-TCP**. And then it will compile the code and create new MyNETBench-TCP executable file.

**Note: If the makefile dose not help in getting proper output. Please try to remove **MyNETBench-TCP** executable and run the above compilation command.**

## SLURM Job command

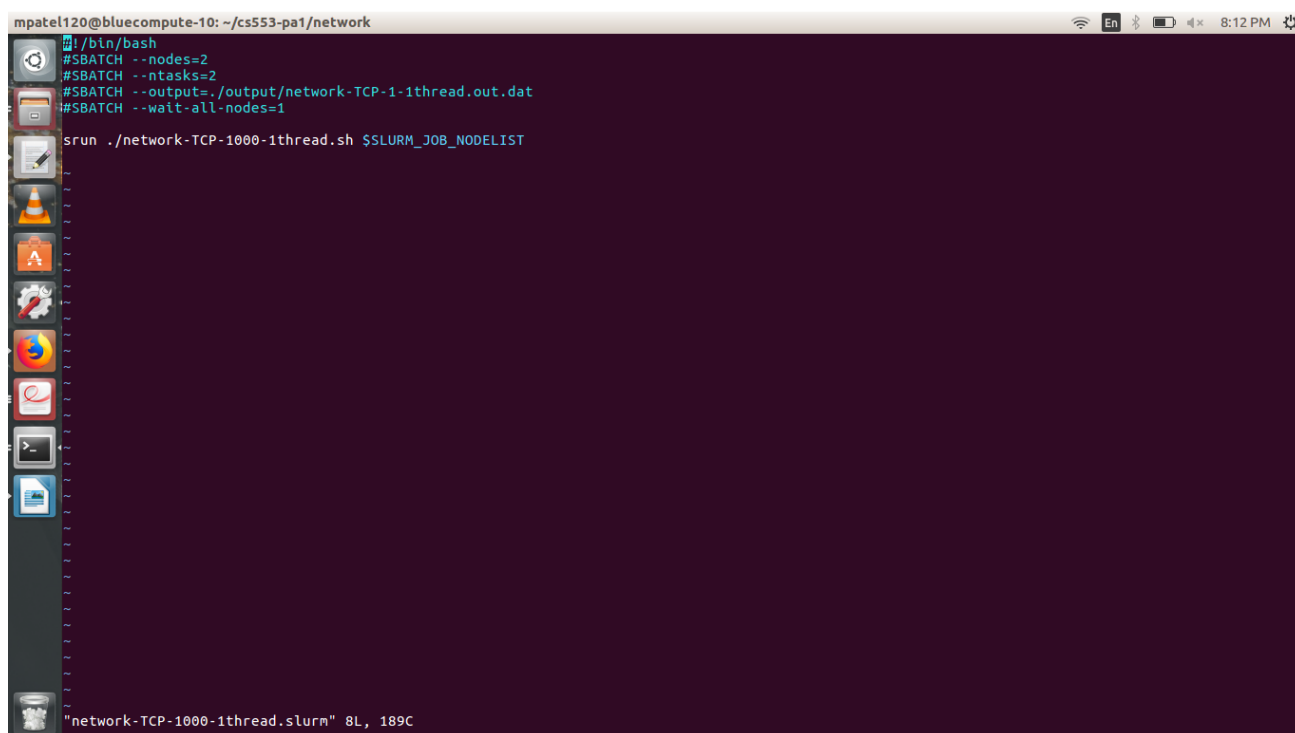
The command for slurm job is -

**sbatch <.slurm file name>**

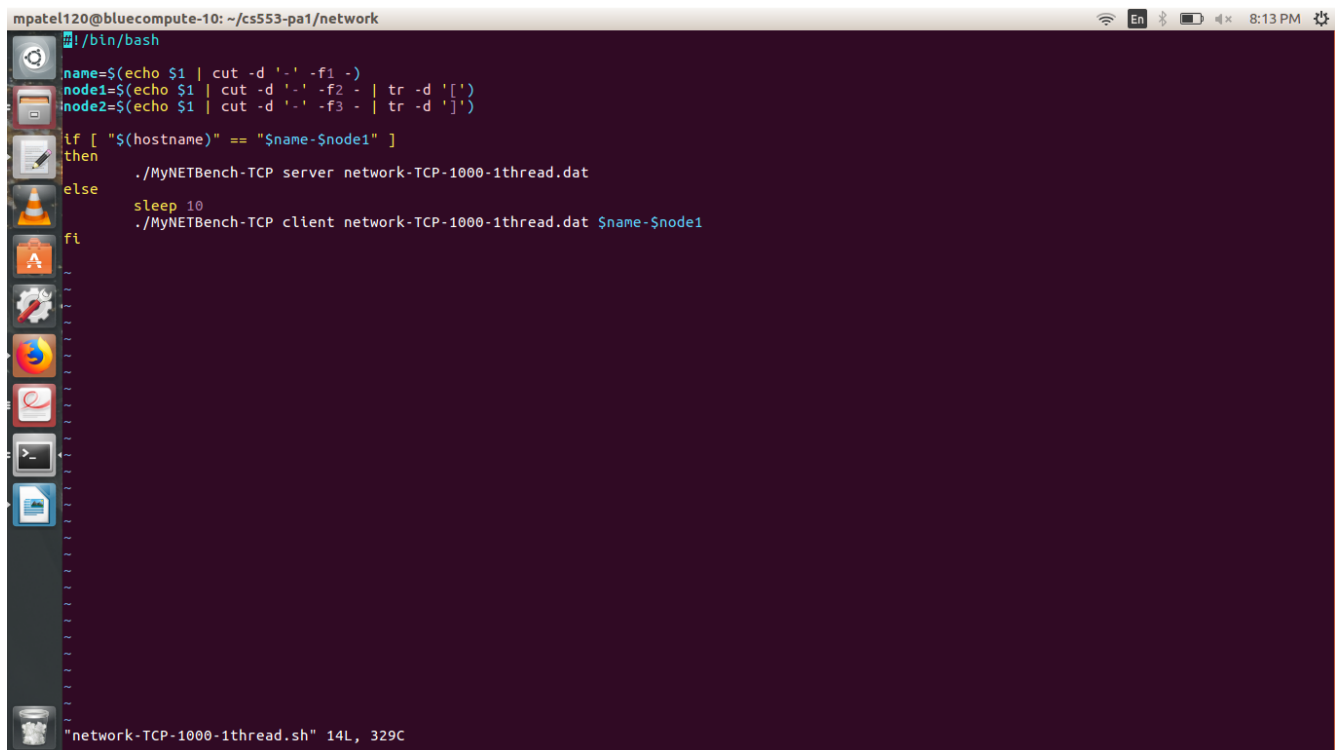
for example:

**sbatch network-TCP-32000-1thread.slurm**

The SLURM file and .sh files are different as these files will request 2 nodes for running server and client on each node. The code for SLURM file is given below -



The code for .sh file can be written as below -



The screenshot shows a terminal window titled "mpatel120@bluecompute-10: ~/cs553-pa1/network". The terminal is running a shell script named "network-TCP-1000-1thread.sh". The script defines variables for node names and uses an if statement to run either a server or a client command based on the hostname. The script is being executed in a terminal window with a dark background and a light-colored text. The terminal window has a standard Ubuntu-style dock on the left side with various application icons. The status bar at the top right shows the time as 8:13 PM and some system icons.

```
mpatel120@bluecompute-10: ~/cs553-pa1/network
#!/bin/bash
name=$(echo $1 | cut -d '-' -f1 -)
node1=$(echo $1 | cut -d '-' -f2 - | tr -d '[')
node2=$(echo $1 | cut -d '-' -f3 - | tr -d ']')

if [ "$(hostname)" == "$name-$node1" ]
then
    ./MyNETBench-TCP server network-TCP-1000-1thread.dat
else
    sleep 10
    ./MyNETBench-TCP client network-TCP-1000-1thread.dat $name-$node1
fi

"network-TCP-1000-1thread.sh" 14L, 329C
```

Also, for simplification I have made one command runAllBatch.sh file which has included all TCP and UDP batch assignment command.

## **B) UDP Network Benchmark**

### **Commands for compilation and running project**

To compile the .c file on terminal the command is -

**`gcc -Wall -o MyNETBench-UDP MyNETBench-UDP.c -lpthread`**

After successful compilation, the code for execution on one terminal or node is -

**`./ MyNETBench- UDP server <*.dat file name >`**

and on other node or terminal

**`./ MyNETBench- UDP client <*.dat file name > localhost`**

(There will be total 4 command line arguments(including `./MyNETBench-UDP`) for client and 3 for server.)

## Using Makefile

Use “**`make all`**” command in terminal and it will remove all .o files and executable file **`MyNETBench-UDP`** . And then it will compile the code and create new `MyNETBench-TCP` executable file.

**Note: If the makefile dose not help in getting proper output. Please try to remove `MyNETBench- UDP` executable and run the above compilation command.**

## SLURM Job command

The command for slurm job is -

**`sbatch <.slurm file name>`**

for example:

**`sbatch network-UDP-32000-1thread.slurm`**

The SLURM file and .sh files are different as these files will request 2 nodes for running server and client on each node. The code for SLURM and .sh files is same as the code given for TCP SLURM and .sh files.