# MANUAL

## I. CPU Benchmark:

→ We have implemented the CPU Benchmarking in C. Following are the steps to compile and run the CPU Benchmark:

**For compilation of CPU Benchmark**

i. Use gcc compiler to compile the program. For thread synchronization, we have used pthread library while compilation. Following command will compile the program:

→ **gcc cpu.c -o cpu -lm -pthread**

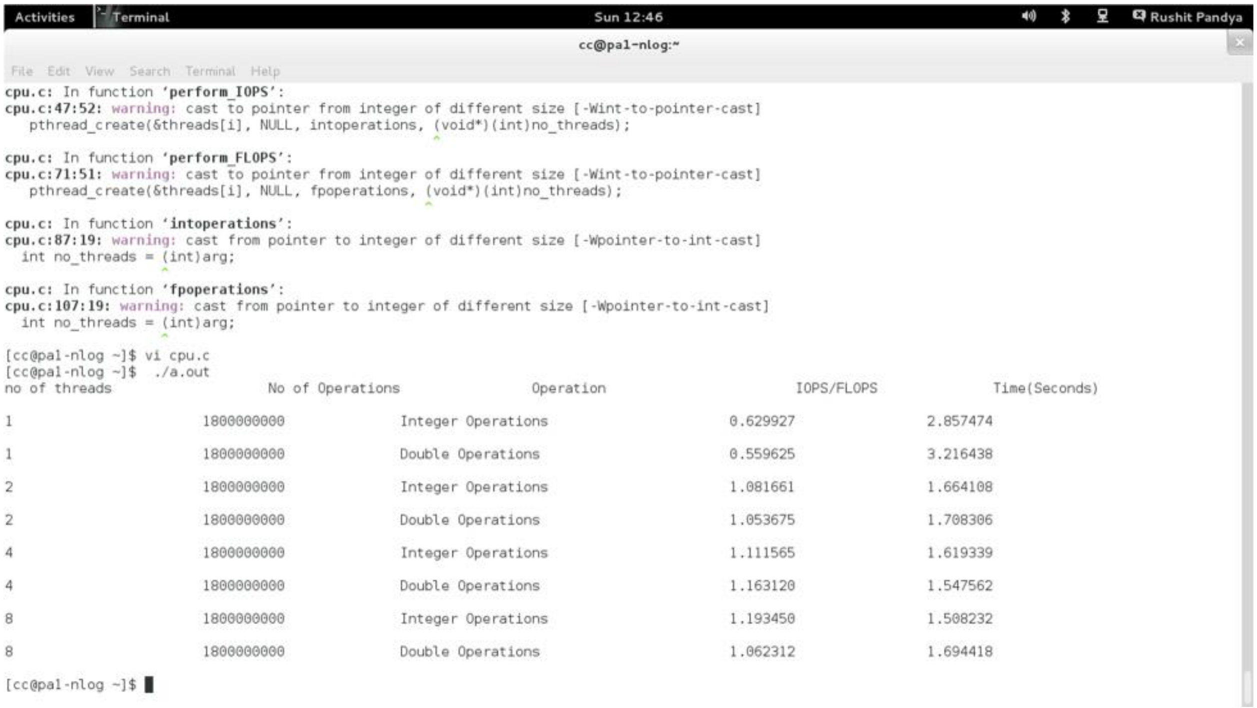
ii. As soon as we compile the C file, an object file named 'cpu' is generated.

**For running CPU Benchmark**

i. The object file generated after compilation will lead to the output of the program. Following command will run the object file:

→ **./cpu**

ii. The following command will display an output, wherein we can see the processor speed in Gflops for integer operation and double precision floating operation per second. Screenshot is displayed below:



```
cpu.c: In function 'perform_IOPS':
cpu.c:47:52: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   pthread_create(&threads[i], NULL, intoperations, (void*)(int)no_threads);
                                                    ^
cpu.c: In function 'perform_FLOPS':
cpu.c:71:51: warning: cast to pointer from integer of different size [-Wint-to-pointer-cast]
   pthread_create(&threads[i], NULL, fpoperations, (void*)(int)no_threads);
                                                   ^
cpu.c: In function 'intoperations':
cpu.c:87:19: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
  int no_threads = (int)arg;
                   ^
cpu.c: In function 'fpoperations':
cpu.c:107:19: warning: cast from pointer to integer of different size [-Wpointer-to-int-cast]
  int no_threads = (int)arg;
                   ^
[cc@pal-nlog ~]$ vi cpu.c
[cc@pal-nlog ~]$  ./a.out
no of threads                No of Operations              Operation                    IOPS/FLOPS             Time(Seconds)

1                       1800000000             Integer Operations                0.629927             2.857474

1                       1800000000             Double Operations                 0.559625             3.216438

2                       1800000000             Integer Operations                1.081661             1.664108

2                       1800000000             Double Operations                 1.053675             1.708306

4                       1800000000             Integer Operations                1.111565             1.619339

4                       1800000000             Double Operations                 1.163120             1.547562

8                       1800000000             Integer Operations                1.193450             1.508232

8                       1800000000             Double Operations                 1.062312             1.694418

[cc@pal-nlog ~]$ █
```

**CPU AVX Benchmarking:**

→We have implemented the CPU AVX Benchmarking in C. Following are the steps to compile and run the CPU AVX Benchmark:

**For compilation of CPU Benchmark**

i. Use gcc compiler to compile the program. For thread synchronization, we have used pthread library while compilation. Following command will compile the program:

```
→ gcc -mavx -o cpuavx CPU_AVX.c
```

ii. As soon as we compile the C file, an object file named 'cpuavx' is generated.

**For running CPU Benchmark**

i. The object file generated after compilation will lead to the output of the program. Following command will run the object file:

```
→ ./cpuavx
```

ii. The following command will display an output, wherein we can see the processor speed in Gflops for integer operation and double precision floating operation per second.

**Linpack Benchmarking:**

→We have implemented the Linpack Benchmarking implemented in C. Following are the steps to run the Linpack Benchmark:

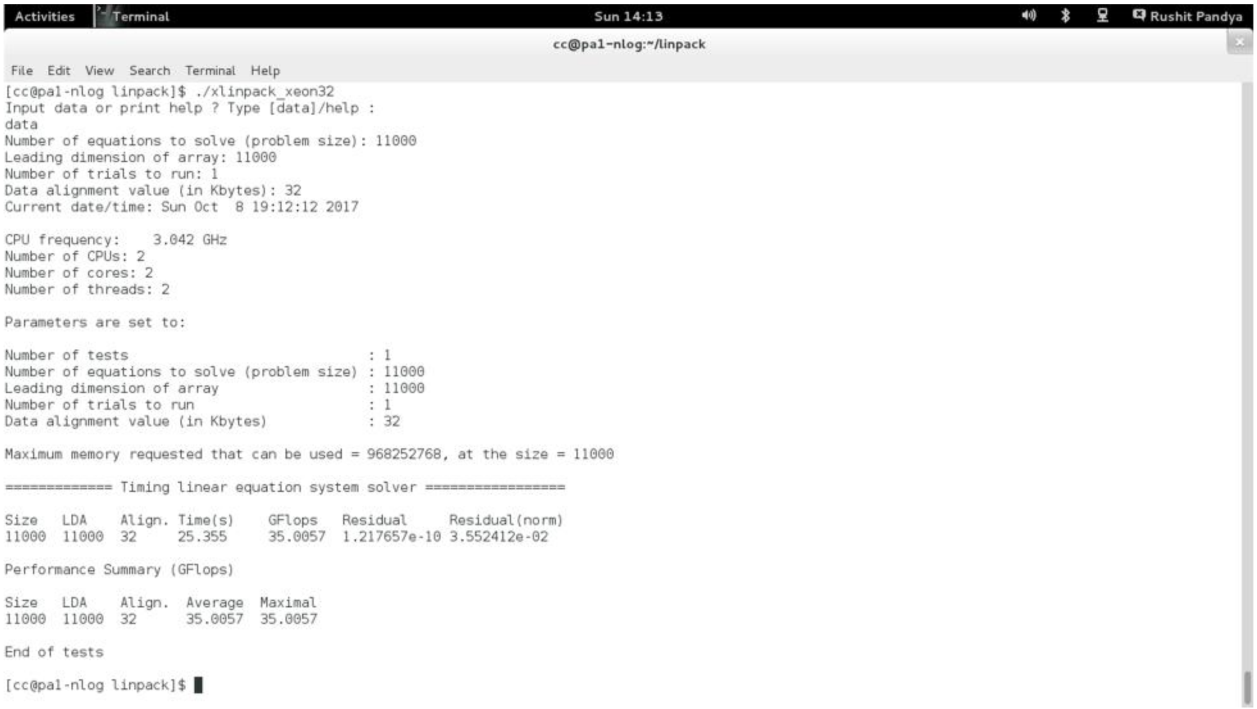
**For running Linpack Benchmark**

i. The object file generated will lead to the output of the program. Following command will run the object file:

```
→ ./xlinpack_xeon32
```

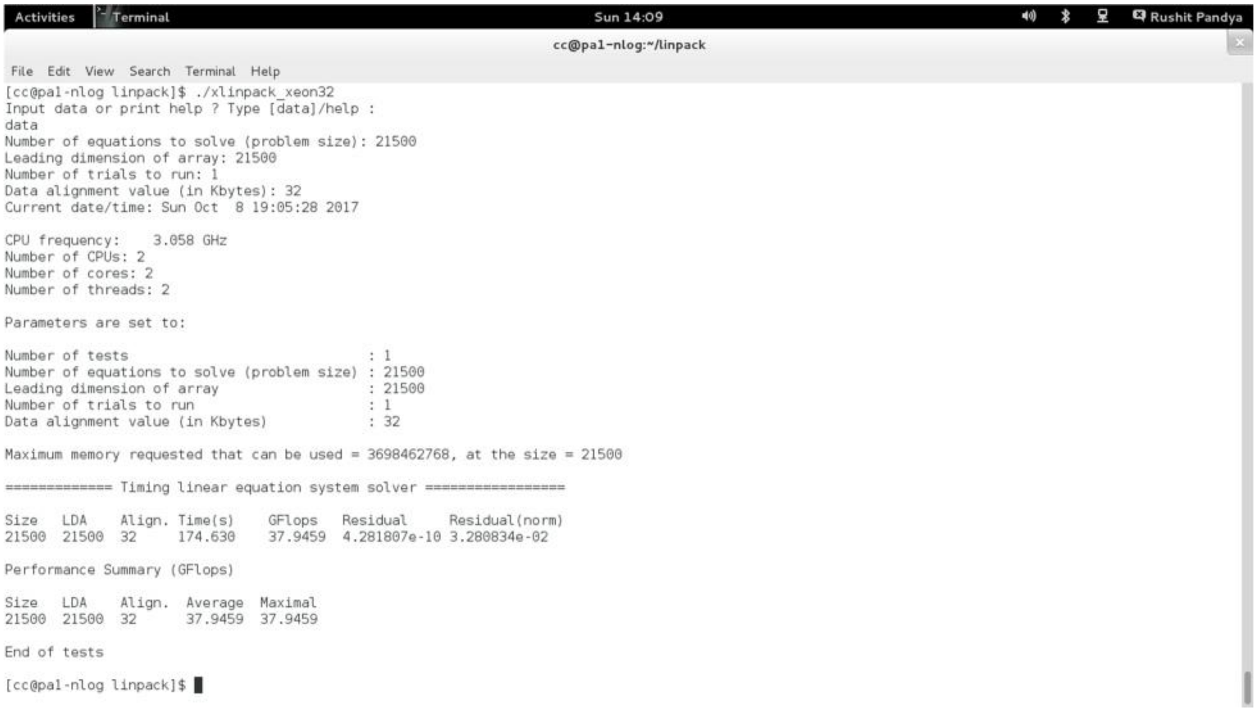
ii. Following are the parameters that need to be set: Number of tests, Number of equations to solve, Leading dimension of array, Number of trials to run and Data alignment value.

-For values, [1, 11000,11000,1,32] for the above parameters, we get the following result,



```
[cc@pa1-nlog linpack]$ ./xlinpack_xeon32
Input data or print help ? Type [data]/help :
data
Number of equations to solve (problem size): 11000
Leading dimension of array: 11000
Number of trials to run: 1
Data alignment value (in Kbytes): 32
Current date/time: Sun Oct  8 19:12:12 2017

CPU frequency:    3.042 GHz
Number of CPUs: 2
Number of cores: 2
Number of threads: 2

Parameters are set to:

Number of tests                             : 1
Number of equations to solve (problem size) : 11000
Leading dimension of array                  : 11000
Number of trials to run                     : 1
Data alignment value (in Kbytes)            : 32

Maximum memory requested that can be used = 968252768, at the size = 11000

============= Timing linear equation system solver =================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm)
11000  11000  32     25.355     35.0057  1.217657e-10 3.552412e-02

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
11000  11000  32      35.0057  35.0057

End of tests

[cc@pa1-nlog linpack]$ █
```

-For values, [1, 21500,21500,1,32] for the above parameters, we get the following result,



```
[cc@pa1-nlog linpack]$ ./xlinpack_xeon32
Input data or print help ? Type [data]/help :
data
Number of equations to solve (problem size): 21500
Leading dimension of array: 21500
Number of trials to run: 1
Data alignment value (in Kbytes): 32
Current date/time: Sun Oct  8 19:05:28 2017

CPU frequency:    3.058 GHz
Number of CPUs: 2
Number of cores: 2
Number of threads: 2

Parameters are set to:

Number of tests                             : 1
Number of equations to solve (problem size) : 21500
Leading dimension of array                  : 21500
Number of trials to run                     : 1
Data alignment value (in Kbytes)            : 32

Maximum memory requested that can be used = 3698462768, at the size = 21500

============= Timing linear equation system solver =================

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm)
21500  21500  32     174.630    37.9459  4.281807e-10 3.280834e-02

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
21500  21500  32      37.9459  37.9459

End of tests

[cc@pa1-nlog linpack]$ █
```

## II. Memory Benchmark:

→ We have implemented the Memory Benchmarking in C. Following are the steps to compile and run the Memory Benchmark:

**For compilation of Memory Benchmark**

i. Use gcc compiler to compile the program. For thread synchronization, we have used pthread library while compilation. Following command will compile the program:

→ **gcc memory.c -o memory -lm -pthread**

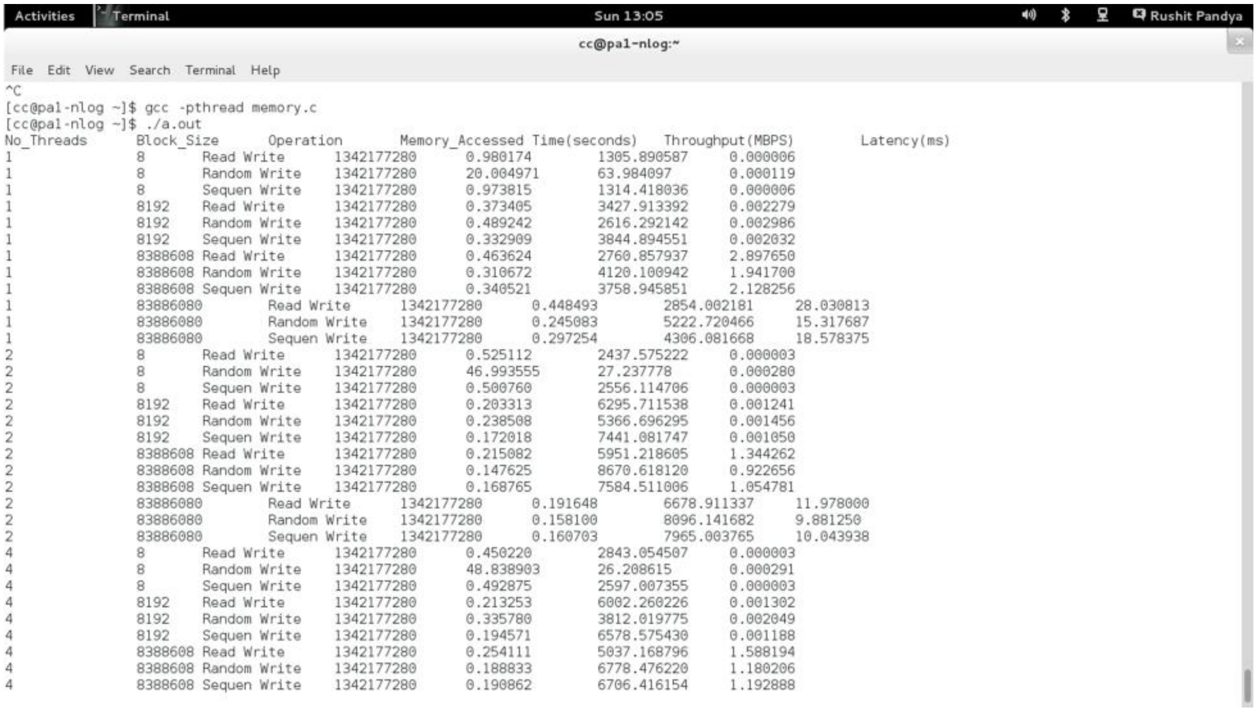
ii. As soon as we compile the C file, an object file named 'memory' is generated.

**For running Memory Benchmark**

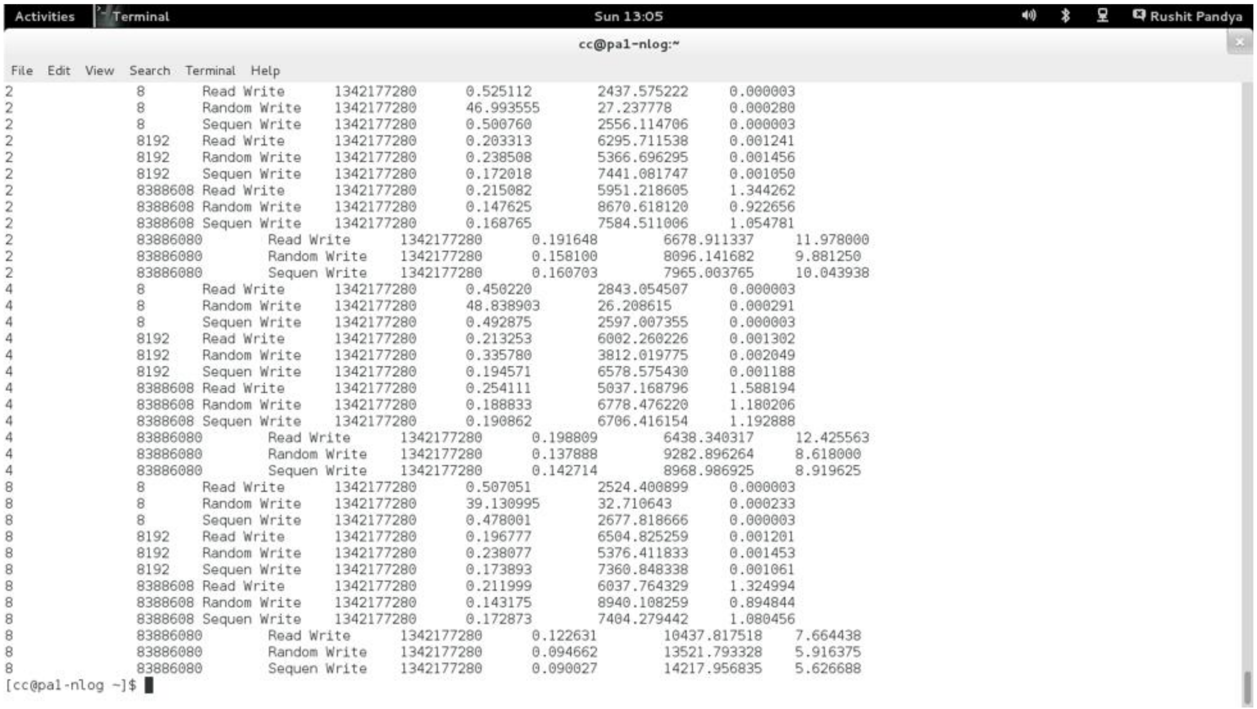
i. The object file generated after compilation will lead to the output of the program. Following command will run the object file:

→ **./memory**

ii. The following command will display an output, wherein we can see the throughput and latency of the processor. Screenshot is displayed below:



```
^C
[cc@pa1-nlog ~]$ gcc -pthread memory.c
[cc@pa1-nlog ~]$ ./a.out
No_Threads      Block_Size      Operation       Memory_Accessed Time(seconds)   Throughput(MBPS)        Latency(ms)
1               8       Read Write      1342177280      0.980174        1305.890587     0.000006
1               8       Random Write    1342177280      20.004971       63.984097       0.000119
1               8       Sequen Write    1342177280      0.973815        1314.418036     0.000006
1               8192    Read Write      1342177280      0.373405        3427.913392     0.002279
1               8192    Random Write    1342177280      0.489242        2616.292142     0.002986
1               8192    Sequen Write    1342177280      0.332909        3844.894551     0.002032
1               8388608 Read Write      1342177280      0.463624        2760.857937     2.897650
1               8388608 Random Write    1342177280      0.310672        4120.100942     1.941700
1               8388608 Sequen Write    1342177280      0.340521        3758.945851     2.128256
1               83886080        Read Write      1342177280      0.448493        2854.002181     28.030813
1               83886080        Random Write    1342177280      0.245083        5222.720466     15.317687
1               83886080        Sequen Write    1342177280      0.297254        4306.081668     18.578375
2               8       Read Write      1342177280      0.525112        2437.575222     0.000003
2               8       Random Write    1342177280      46.993555       27.237778       0.000280
2               8       Sequen Write    1342177280      0.500760        2556.114706     0.000003
2               8192    Read Write      1342177280      0.203313        6295.711538     0.001241
2               8192    Random Write    1342177280      0.238508        5366.696295     0.001456
2               8192    Sequen Write    1342177280      0.172018        7441.081747     0.001050
2               8388608 Read Write      1342177280      0.215082        5951.218605     1.344262
2               8388608 Random Write    1342177280      0.147625        8670.618120     0.922656
2               8388608 Sequen Write    1342177280      0.168765        7584.511006     1.054781
2               83886080        Read Write      1342177280      0.191648        6678.911337     11.978000
2               83886080        Random Write    1342177280      0.158100        8096.141682     9.881250
2               83886080        Sequen Write    1342177280      0.160703        7965.003765     10.043938
4               8       Read Write      1342177280      0.450220        2843.054507     0.000003
4               8       Random Write    1342177280      48.838903       26.208615       0.000291
4               8       Sequen Write    1342177280      0.492875        2597.007355     0.000003
4               8192    Read Write      1342177280      0.213253        6002.260226     0.001302
4               8192    Random Write    1342177280      0.335780        3812.019775     0.002049
4               8192    Sequen Write    1342177280      0.194571        6578.575430     0.001188
4               8388608 Read Write      1342177280      0.254111        5037.168796     1.588194
4               8388608 Random Write    1342177280      0.188833        6778.476220     1.180206
4               8388608 Sequen Write    1342177280      0.190862        6706.416154     1.192888
```



```
2               8       Read Write      1342177280      0.525112        2437.575222     0.000003
2               8       Random Write    1342177280      46.993555       27.237778       0.000280
2               8       Sequen Write    1342177280      0.500760        2556.114706     0.000003
2               8192    Read Write      1342177280      0.203313        6295.711538     0.001241
2               8192    Random Write    1342177280      0.238508        5366.696295     0.001456
2               8192    Sequen Write    1342177280      0.172018        7441.081747     0.001050
2               8388608 Read Write      1342177280      0.215082        5951.218605     1.344262
2               8388608 Random Write    1342177280      0.147625        8670.618120     0.922656
2               8388608 Sequen Write    1342177280      0.168765        7584.511006     1.054781
2               83886080        Read Write      1342177280      0.191648        6678.911337     11.978000
2               83886080        Random Write    1342177280      0.158100        8096.141682     9.881250
2               83886080        Sequen Write    1342177280      0.160703        7965.003765     10.043938
4               8       Read Write      1342177280      0.450220        2843.054507     0.000003
4               8       Random Write    1342177280      48.838903       26.208615       0.000291
4               8       Sequen Write    1342177280      0.492875        2597.007355     0.000003
4               8192    Read Write      1342177280      0.213253        6002.260226     0.001302
4               8192    Random Write    1342177280      0.335780        3812.019775     0.002049
4               8192    Sequen Write    1342177280      0.194571        6578.575430     0.001188
4               8388608 Read Write      1342177280      0.254111        5037.168796     1.588194
4               8388608 Random Write    1342177280      0.188833        6778.476220     1.180206
4               8388608 Sequen Write    1342177280      0.190862        6706.416154     1.192888
4               83886080        Read Write      1342177280      0.198809        6438.340317     12.425563
4               83886080        Random Write    1342177280      0.137888        9282.896264     8.618000
4               83886080        Sequen Write    1342177280      0.142714        8968.986925     8.919625
8               8       Read Write      1342177280      0.507051        2524.400899     0.000003
8               8       Random Write    1342177280      39.130995       32.710643       0.000233
8               8       Sequen Write    1342177280      0.478001        2677.818666     0.000003
8               8192    Read Write      1342177280      0.196777        6504.825259     0.001201
8               8192    Random Write    1342177280      0.238077        5376.411833     0.001453
8               8192    Sequen Write    1342177280      0.173893        7360.848338     0.001061
8               8388608 Read Write      1342177280      0.211999        6037.764329     1.324994
8               8388608 Random Write    1342177280      0.143175        8940.108259     0.894844
8               8388608 Sequen Write    1342177280      0.172873        7404.279442     1.080456
8               83886080        Read Write      1342177280      0.122631        10437.817518    7.664438
8               83886080        Random Write    1342177280      0.094662        13521.793328    5.916375
8               83886080        Sequen Write    1342177280      0.090027        14217.956835    5.626688
[cc@pa1-nlog ~]$ █
```

**Stream Benchmark:**

→We have implemented the Stream Benchmarking implemented in C. Following are the steps to compile and run the Stream Benchmark:

*For Stream Benchmarking, we have downloaded the package from net and it is in the Memory folder and the below mentioned stream.c file is in the Stream folder

**<u>For compilation of Stream Benchmark</u>**

i. Use gcc compiler to compile the program. Following command will compile the program:

→ **gcc stream.c -o stream**

ii. We need to use 'openmp' as the command, for multithreading

→gcc -fopenmp -D OPENMP stream.c -o stream export OMP_NUM_THREADS=2

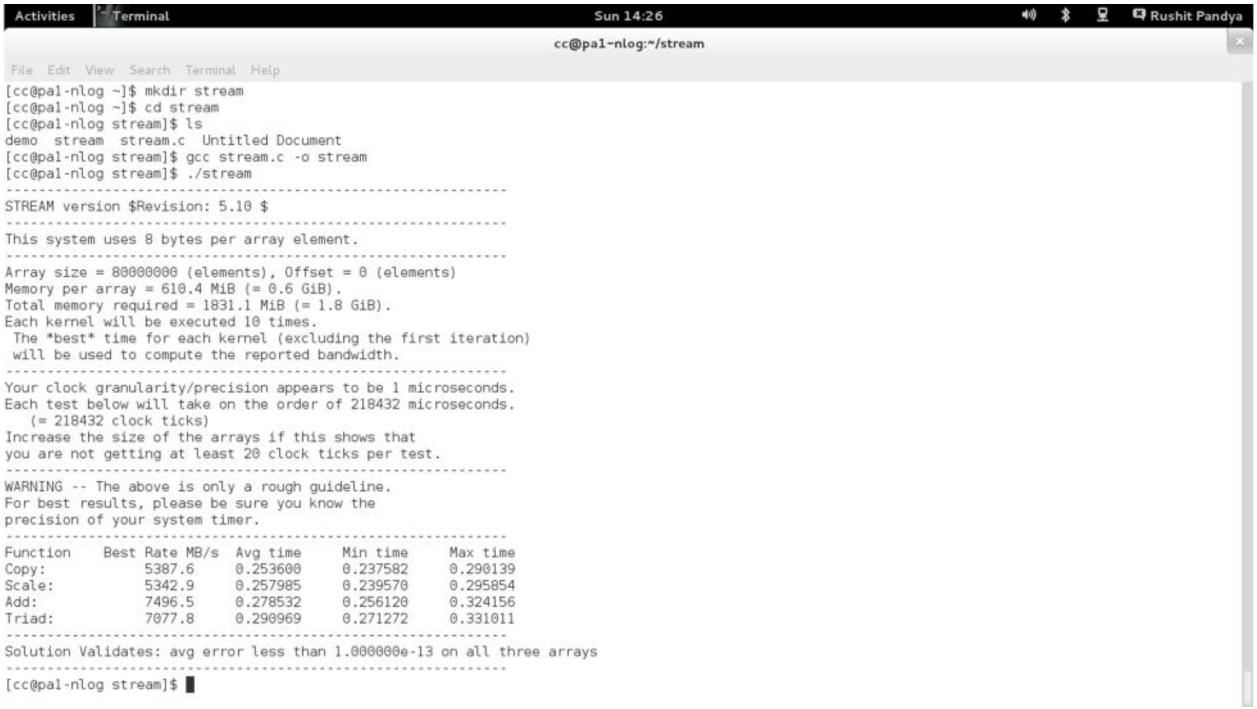
iii. As soon as we compile the C file, an object file named 'stream' is generated.

**<u>For running Stream Benchmark</u>**

i. The object file generated after compilation will lead to the output of the program. Following command will run the object file:

→ **./stream**

ii. The following command will display an output for which the Screenshot is displayed below:



```
[cc@pal-nlog ~]$ mkdir stream
[cc@pal-nlog ~]$ cd stream
[cc@pal-nlog stream]$ ls
demo  stream  stream.c  Untitled Document
[cc@pal-nlog stream]$ gcc stream.c -o stream
[cc@pal-nlog stream]$ ./stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 80000000 (elements), Offset = 0 (elements)
Memory per array = 610.4 MiB (= 0.6 GiB).
Total memory required = 1831.1 MiB (= 1.8 GiB).
Each kernel will be executed 10 times.
 The *best* time for each kernel (excluding the first iteration)
 will be used to compute the reported bandwidth.
-------------------------------------------------------------
Your clock granularity/precision appears to be 1 microseconds.
Each test below will take on the order of 218432 microseconds.
   (= 218432 clock ticks)
Increase the size of the arrays if this shows that
you are not getting at least 20 clock ticks per test.
-------------------------------------------------------------
WARNING -- The above is only a rough guideline.
For best results, please be sure you know the
precision of your system timer.
-------------------------------------------------------------
Function    Best Rate MB/s  Avg time     Min time     Max time
Copy:           5387.6     0.253600     0.237582     0.290139
Scale:          5342.9     0.257985     0.239570     0.295854
Add:            7496.5     0.278532     0.256120     0.324156
Triad:          7077.8     0.290969     0.271272     0.331011
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
[cc@pal-nlog stream]$ █
```

## III. Disk Benchmark:

→ We have implemented the Disk Benchmarking in C. Following are the steps to compile and run the Disk Benchmark:

**<u>For compilation of Disk Benchmark</u>**

i. Use gcc compiler to compile the program. For thread synchronization, we have used pthread library while compilation. Following command will compile the program:

→ **gcc disk.c -o disk -lm -pthread**

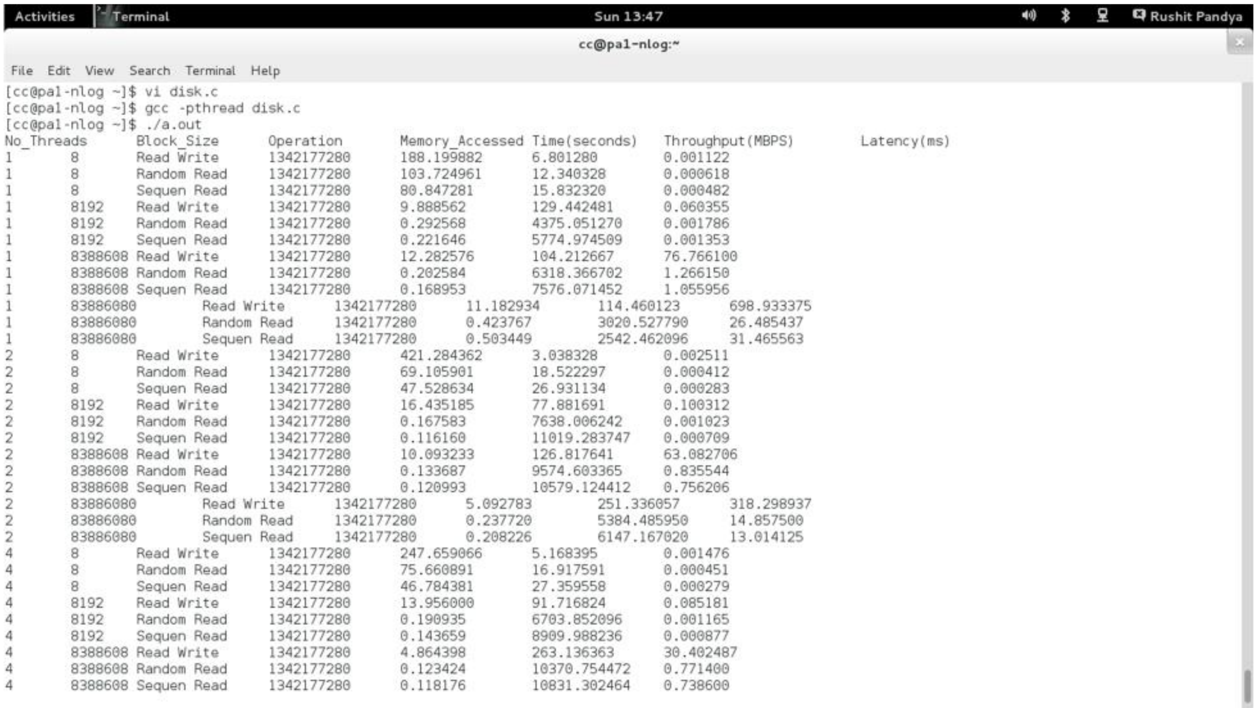
ii. As soon as we compile the C file, an object file named ‘disk’ is generated.

**<u>For running Disk Benchmark</u>**

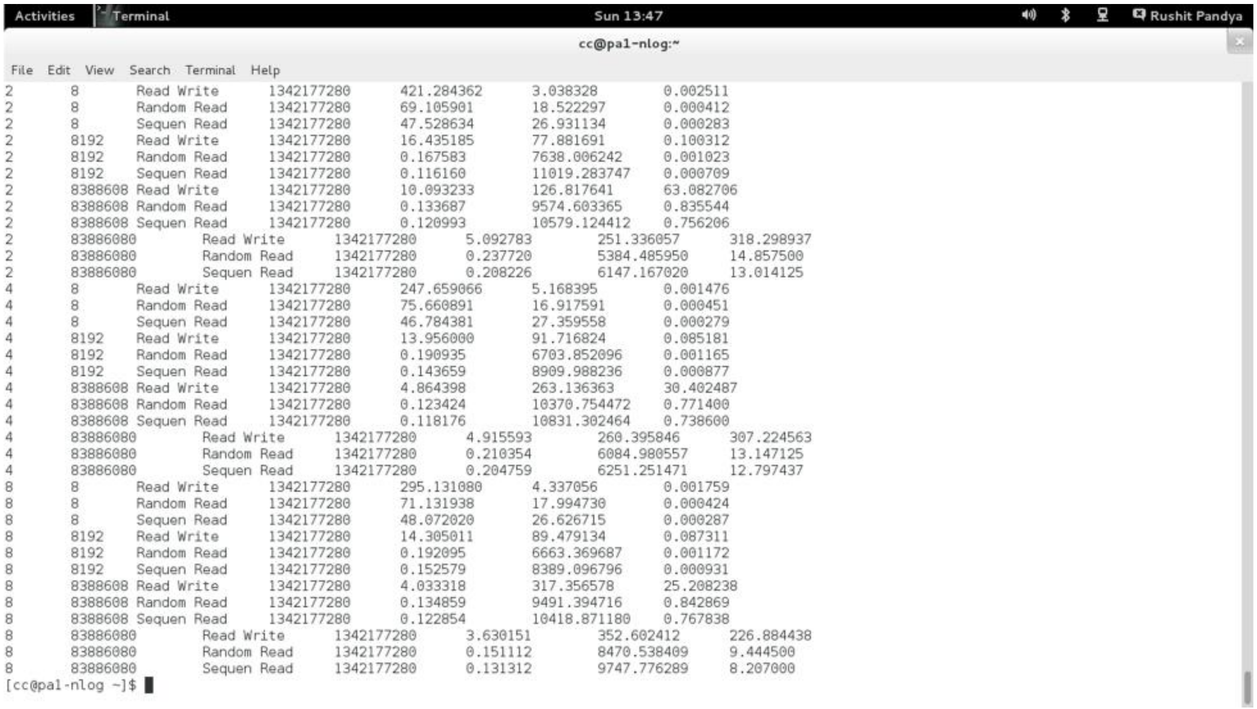
i. The object file generated after compilation will lead to the output of the program. Following command will run the object file:

→ **./disk**

ii. The following command will display an output, wherein we can see the throughput and latency of the processor. Screenshot is displayed below:



```
[cc@pa1-nlog ~]$ vi disk.c
[cc@pa1-nlog ~]$ gcc -pthread disk.c
[cc@pa1-nlog ~]$ ./a.out
No_Threads      Block_Size      Operation       Memory_Accessed Time(seconds)   Throughput(MBPS)        Latency(ms)
1       8       Read Write      1342177280      188.199882      6.801280        0.001122
1       8       Random Read     1342177280      103.724961      12.340328       0.000618
1       8       Sequen Read     1342177280      80.847281       15.832320       0.000482
1       8192    Read Write      1342177280      9.888562        129.442481      0.060355
1       8192    Random Read     1342177280      0.292568        4375.051270     0.001786
1       8192    Sequen Read     1342177280      0.221646        5774.974509     0.001353
1       8388608 Read Write      1342177280      12.282576       104.212667      76.766100
1       8388608 Random Read     1342177280      0.202584        6318.366702     1.266150
1       8388608 Sequen Read     1342177280      0.168953        7576.071452     1.055956
1       83886080        Read Write      1342177280      11.182934       114.460123      698.933375
1       83886080        Random Read     1342177280      0.423767        3020.527790     26.485437
1       83886080        Sequen Read     1342177280      0.503449        2542.462096     31.465563
2       8       Read Write      1342177280      421.284362      3.038328        0.002511
2       8       Random Read     1342177280      69.105901       18.522297       0.000412
2       8       Sequen Read     1342177280      47.528634       26.931134       0.000283
2       8192    Read Write      1342177280      16.435185       77.881691       0.100312
2       8192    Random Read     1342177280      0.167583        7638.006242     0.001023
2       8192    Sequen Read     1342177280      0.116160        11019.283747    0.000709
2       8388608 Read Write      1342177280      10.093233       126.817641      63.082706
2       8388608 Random Read     1342177280      0.133687        9574.603365     0.835544
2       8388608 Sequen Read     1342177280      0.120993        10579.124412    0.756206
2       83886080        Read Write      1342177280      5.092783        251.336057      318.298937
2       83886080        Random Read     1342177280      0.237720        5384.485950     14.857500
2       83886080        Sequen Read     1342177280      0.208226        6147.167020     13.014125
4       8       Read Write      1342177280      247.659066      5.168395        0.001476
4       8       Random Read     1342177280      75.660891       16.917591       0.000451
4       8       Sequen Read     1342177280      46.784381       27.359558       0.000279
4       8192    Read Write      1342177280      13.956000       91.716824       0.085181
4       8192    Random Read     1342177280      0.190935        6703.852096     0.001165
4       8192    Sequen Read     1342177280      0.143659        8909.988236     0.000877
4       8388608 Read Write      1342177280      4.864398        263.136363      30.402487
4       8388608 Random Read     1342177280      0.123424        10370.754472    0.771400
4       8388608 Sequen Read     1342177280      0.118176        10831.302464    0.738600
```



```
2       8       Read Write      1342177280      421.284362      3.038328        0.002511
2       8       Random Read     1342177280      69.105901       18.522297       0.000412
2       8       Sequen Read     1342177280      47.528634       26.931134       0.000283
2       8192    Read Write      1342177280      16.435185       77.881691       0.100312
2       8192    Random Read     1342177280      0.167583        7638.006242     0.001023
2       8192    Sequen Read     1342177280      0.116160        11019.283747    0.000709
2       8388608 Read Write      1342177280      10.093233       126.817641      63.082706
2       8388608 Random Read     1342177280      0.133687        9574.603365     0.835544
2       8388608 Sequen Read     1342177280      0.120993        10579.124412    0.756206
2       83886080        Read Write      1342177280      5.092783        251.336057      318.298937
2       83886080        Random Read     1342177280      0.237720        5384.485950     14.857500
2       83886080        Sequen Read     1342177280      0.208226        6147.167020     13.014125
4       8       Read Write      1342177280      247.659066      5.168395        0.001476
4       8       Random Read     1342177280      75.660891       16.917591       0.000451
4       8       Sequen Read     1342177280      46.784381       27.359558       0.000279
4       8192    Read Write      1342177280      13.956000       91.716824       0.085181
4       8192    Random Read     1342177280      0.190935        6703.852096     0.001165
4       8192    Sequen Read     1342177280      0.143659        8909.988236     0.000877
4       8388608 Read Write      1342177280      4.864398        263.136363      30.402487
4       8388608 Random Read     1342177280      0.123424        10370.754472    0.771400
4       8388608 Sequen Read     1342177280      0.118176        10831.302464    0.738600
```

```
4       83886080        Read Write      1342177280      4.915593        260.395846      307.224563
4       83886080        Random Read     1342177280      0.210354        6084.980557     13.147125
4       83886080        Sequen Read     1342177280      0.204759        6251.251471     12.797437
8       8       Read Write      1342177280      295.131080      4.337056        0.001759
8       8       Random Read     1342177280      71.131938       17.994730       0.000424
8       8       Sequen Read     1342177280      48.072020       26.626715       0.000287
8       8192    Read Write      1342177280      14.305011       89.479134       0.087311
8       8192    Random Read     1342177280      0.192095        6663.369687     0.001172
8       8192    Sequen Read     1342177280      0.152579        8389.096796     0.000931
8       8388608 Read Write      1342177280      4.033318        317.356578      25.208238
8       8388608 Random Read     1342177280      0.134859        9491.394716     0.842869
8       8388608 Sequen Read     1342177280      0.122854        10418.871180    0.767838
8       83886080        Read Write      1342177280      3.630151        352.602412      226.884438
8       83886080        Random Read     1342177280      0.151112        8470.538409     9.444500
8       83886080        Sequen Read     1342177280      0.131312        9747.776289     8.207000
[cc@pa1-nlog ~]$ █
```

## IOZone Benchmark:

→We have implemented the IOZone Benchmarking implemented in C. Following are the steps to compile and run the IOZone Benchmark:

*For IOZone Benchmarking, we have downloaded the package from net and it is in the Disk folder and the below mentioned makefile is in the iozone_394 folder within the src folder

**<u>For compilation of IOZone Benchmark</u>**

i. Use gcc compiler to compile the program. Following command will compile the program:

→ **make linux**

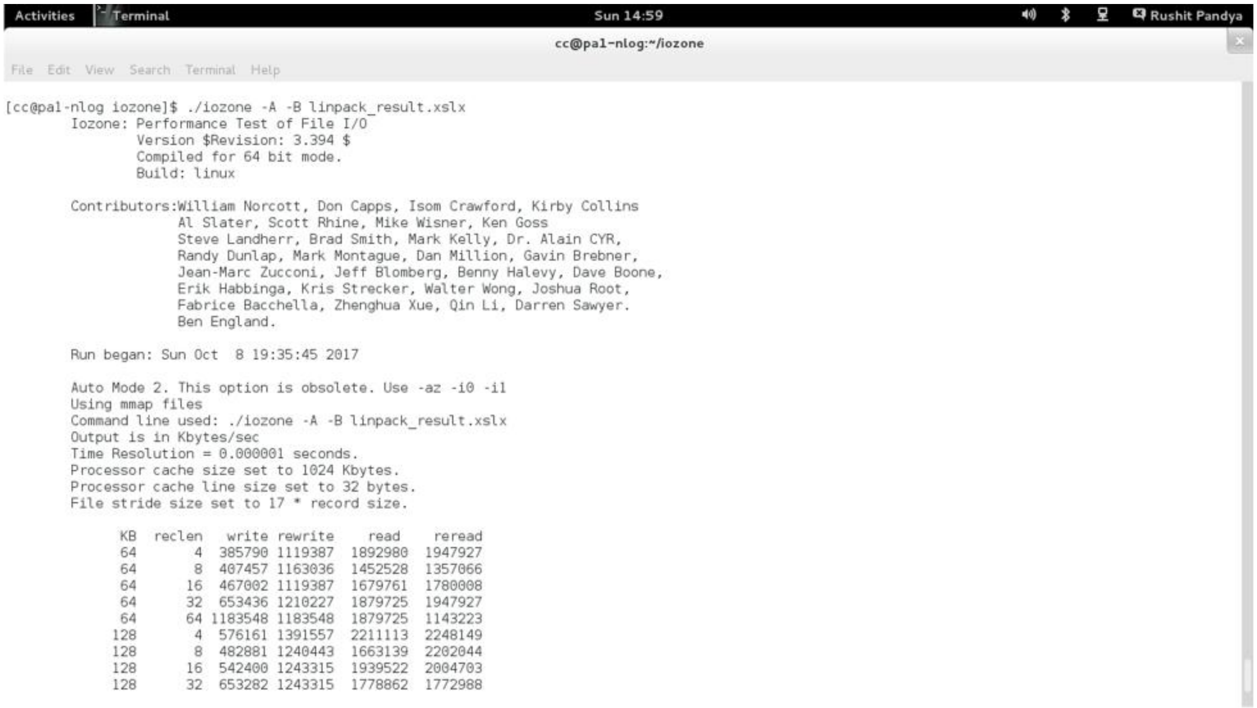
ii. As soon as we compile the C file, an object file named 'iozone' is generated.

**<u>For running IOZone Benchmark</u>**

i. The object file generated after compilation will lead to the output of the program. Following command will run the object file:

→ **./iozone -A -B linpack_result.xls**

ii. The following command will display an output for which the Screenshot is displayed below:



```
[cc@pal-nlog iozone]$ ./iozone -A -B linpack_result.xslx
        Iozone: Performance Test of File I/O
                Version $Revision: 3.394 $
                Compiled for 64 bit mode.
                Build: linux

        Contributors:William Norcott, Don Capps, Isom Crawford, Kirby Collins
                     Al Slater, Scott Rhine, Mike Wisner, Ken Goss
                     Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CYR,
                     Randy Dunlap, Mark Montague, Dan Million, Gavin Brebner,
                     Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
                     Erik Habbinga, Kris Strecker, Walter Wong, Joshua Root,
                     Fabrice Bacchella, Zhenghua Xue, Qin Li, Darren Sawyer.
                     Ben England.

        Run began: Sun Oct  8 19:35:45 2017

        Auto Mode 2. This option is obsolete. Use -az -i0 -i1
        Using mmap files
        Command line used: ./iozone -A -B linpack_result.xslx
        Output is in Kbytes/sec
        Time Resolution = 0.000001 seconds.
        Processor cache size set to 1024 Kbytes.
        Processor cache line size set to 32 bytes.
        File stride size set to 17 * record size.

              KB  reclen   write rewrite    read    reread
              64       4  385790 1119387  1892980  1947927
              64       8  407457 1163036  1452528  1357066
              64      16  467002 1119387  1679761  1780008
              64      32  653436 1210227  1879725  1947927
              64      64 1183548 1183548  1879725  1143223
             128       4  576161 1391557  2211113  2248149
             128       8  482881 1240443  1663139  2202044
             128      16  542400 1243315  1939522  2004703
             128      32  653282 1243315  1778862  1772988
```

## IV. Network Benchmark:

→ We have implemented the Network Benchmarking in Java. Following are the steps to compile and run the Network Benchmark:

**<u>For compilation of TCP Server</u>**

i. Use javac compiler to compile the program. Now, open 1$^{st}$ terminal and follow the commands to compile the program:

→ **javac TcpServer.java**

**For running TCP Server**

→ **java TcpServer**

**For compilation of TCP Client**

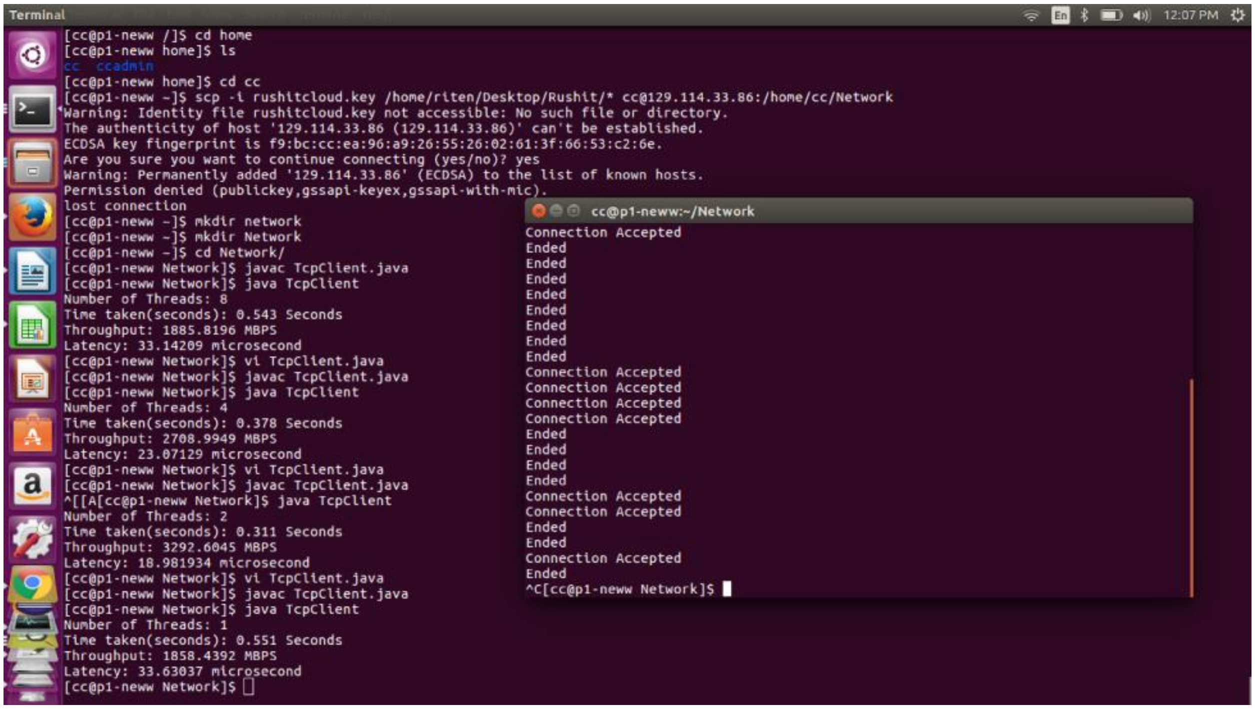
ii. Similarly for the TCP Client side, use javac compiler to compile the program. Now, open 2nd terminal and follow the commands to compile the program:

→ **javac TcpClient.java**

**For running TCP Client**

→ **java TcpClient**

Screenshot is displayed below:



```
[cc@p1-neww /]$ cd home
[cc@p1-neww home]$ ls
cc  ccadmin
[cc@p1-neww home]$ cd cc
[cc@p1-neww ~]$ scp -i rushitcloud.key /home/riten/Desktop/Rushit/* cc@129.114.33.86:/home/cc/Network
Warning: Identity file rushitcloud.key not accessible: No such file or directory.
The authenticity of host '129.114.33.86 (129.114.33.86)' can't be established.
ECDSA key fingerprint is f9:bc:cc:ea:96:a9:26:55:26:02:61:3f:66:53:c2:6e.
Are you sure you want to continue connecting (yes/no)? yes
Warning: Permanently added '129.114.33.86' (ECDSA) to the list of known hosts.
Permission denied (publickey,gssapi-keyex,gssapi-with-mic).
lost connection
[cc@p1-neww ~]$ mkdir network
[cc@p1-neww ~]$ mkdir Network
[cc@p1-neww ~]$ cd Network/
[cc@p1-neww Network]$ javac TcpClient.java
[cc@p1-neww Network]$ java TcpClient
Number of Threads: 8
Time taken(seconds): 0.543 Seconds
Throughput: 1885.8196 MBPS
Latency: 33.14209 microsecond
[cc@p1-neww Network]$ vi TcpClient.java
[cc@p1-neww Network]$ javac TcpClient.java
[cc@p1-neww Network]$ java TcpClient
Number of Threads: 4
Time taken(seconds): 0.378 Seconds
Throughput: 2708.9949 MBPS
Latency: 23.07129 microsecond
[cc@p1-neww Network]$ vi TcpClient.java
[cc@p1-neww Network]$ javac TcpClient.java
^[[A[cc@p1-neww Network]$ java TcpClient
Number of Threads: 2
Time taken(seconds): 0.311 Seconds
Throughput: 3292.6045 MBPS
Latency: 18.981934 microsecond
[cc@p1-neww Network]$ vi TcpClient.java
[cc@p1-neww Network]$ javac TcpClient.java
[cc@p1-neww Network]$ java TcpClient
Number of Threads: 1
Time taken(seconds): 0.551 Seconds
Throughput: 1858.4392 MBPS
Latency: 33.63037 microsecond
[cc@p1-neww Network]$
```

```
Connection Accepted
Ended
Ended
Ended
Ended
Ended
Ended
Ended
Ended
Connection Accepted
Connection Accepted
Connection Accepted
Connection Accepted
Ended
Ended
Ended
Ended
Connection Accepted
Connection Accepted
Ended
Ended
Connection Accepted
Ended
^C[cc@p1-neww Network]$
```

**For compilation of UDP Server**

i. Use javac compiler to compile the program. Now, open 1st terminal and follow the commands to compile the program:

→ **javac UdpServer.java**

**For running UDP Server**

→ **java UdpServer**

**For compilation of UDP Client**

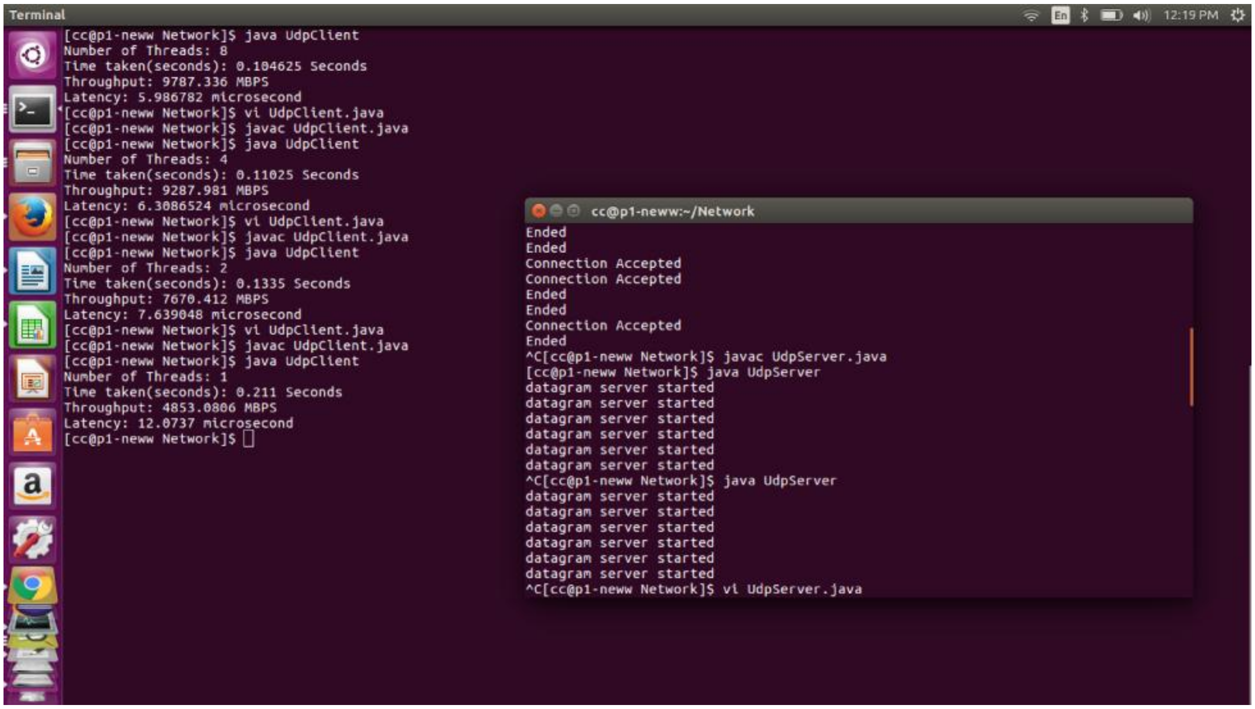
ii. Similarly for the UDP Client side, use javac compiler to compile the program. Now, open 2nd terminal and follow the commands to compile the program:

→ **javac UdpClient.java**

**For running UDP Client**

→ **java UdpClient**

Screenshot is displayed below:



```
[cc@p1-neww Network]$ java UdpClient
Number of Threads: 8
Time taken(seconds): 0.104625 Seconds
Throughput: 9787.336 MBPS
Latency: 5.986782 microsecond
[cc@p1-neww Network]$ vi UdpClient.java
[cc@p1-neww Network]$ javac UdpClient.java
[cc@p1-neww Network]$ java UdpClient
Number of Threads: 4
Time taken(seconds): 0.11025 Seconds
Throughput: 9287.981 MBPS
Latency: 6.3086524 microsecond
[cc@p1-neww Network]$ vi UdpClient.java
[cc@p1-neww Network]$ javac UdpClient.java
[cc@p1-neww Network]$ java UdpClient
Number of Threads: 2
Time taken(seconds): 0.1335 Seconds
Throughput: 7670.412 MBPS
Latency: 7.639048 microsecond
[cc@p1-neww Network]$ vi UdpClient.java
[cc@p1-neww Network]$ javac UdpClient.java
[cc@p1-neww Network]$ java UdpClient
Number of Threads: 1
Time taken(seconds): 0.211 Seconds
Throughput: 4853.0806 MBPS
Latency: 12.0737 microsecond
[cc@p1-neww Network]$ 
```

```
Ended
Ended
Connection Accepted
Connection Accepted
Ended
Ended
Connection Accepted
Ended
^C[cc@p1-neww Network]$ javac UdpServer.java
[cc@p1-neww Network]$ java UdpServer
datagram server started
datagram server started
datagram server started
datagram server started
datagram server started
datagram server started
^C[cc@p1-neww Network]$ java UdpServer
datagram server started
datagram server started
datagram server started
datagram server started
datagram server started
datagram server started
^C[cc@p1-neww Network]$ vi UdpServer.java
```

**IPerf Benchmark:**

→We have run the IPerf Benchmarking and following are the steps to run the IPerf Benchmark:

*For IPerf Benchmarking, we have downloaded the package from net and it was installed using the command: **sudo yum install iperf3**

**For running IPerf TCP Server Benchmark**

i. Open 1st terminal and follow the commands to run the program:

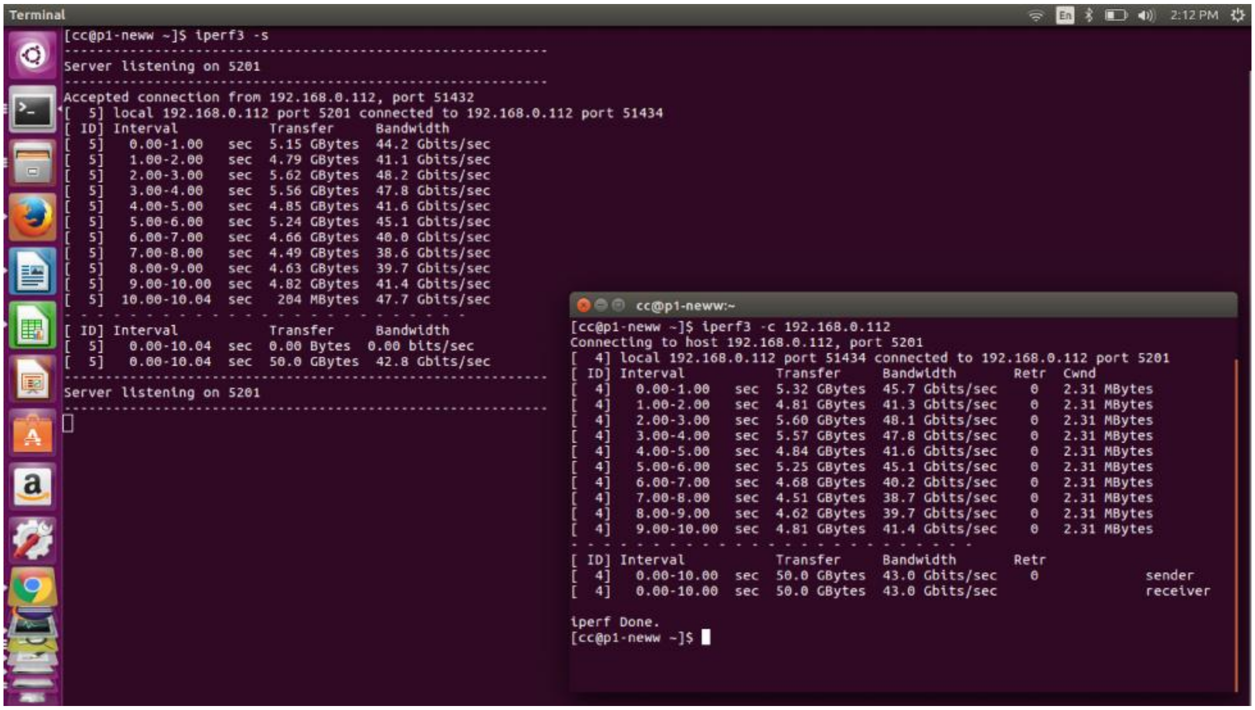
→ **iperf3 -s**

**For running IPerf TCP Client Benchmark**

i. Now, open 2nd terminal and follow the commands to run the program:

→ **iperf3 -c 192.168.0.112**

Screenshot is displayed below:



```
[cc@p1-neww ~]$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.0.112, port 51432
[  5] local 192.168.0.112 port 5201 connected to 192.168.0.112 port 51434
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-1.00   sec  5.15 GBytes  44.2 Gbits/sec
[  5]   1.00-2.00   sec  4.79 GBytes  41.1 Gbits/sec
[  5]   2.00-3.00   sec  5.62 GBytes  48.2 Gbits/sec
[  5]   3.00-4.00   sec  5.56 GBytes  47.8 Gbits/sec
[  5]   4.00-5.00   sec  4.85 GBytes  41.6 Gbits/sec
[  5]   5.00-6.00   sec  5.24 GBytes  45.1 Gbits/sec
[  5]   6.00-7.00   sec  4.66 GBytes  40.0 Gbits/sec
[  5]   7.00-8.00   sec  4.49 GBytes  38.6 Gbits/sec
[  5]   8.00-9.00   sec  4.63 GBytes  39.7 Gbits/sec
[  5]   9.00-10.00  sec  4.82 GBytes  41.4 Gbits/sec
[  5]  10.00-10.04  sec   204 MBytes  47.7 Gbits/sec
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth
[  5]   0.00-10.04  sec  0.00 Bytes  0.00 bits/sec
[  5]   0.00-10.04  sec  50.0 GBytes  42.8 Gbits/sec
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

cc@p1-neww:~

```
[cc@p1-neww ~]$ iperf3 -c 192.168.0.112
Connecting to host 192.168.0.112, port 5201
[  4] local 192.168.0.112 port 51434 connected to 192.168.0.112 port 5201
[ ID] Interval           Transfer     Bandwidth       Retr  Cwnd
[  4]   0.00-1.00   sec  5.32 GBytes  45.7 Gbits/sec    0   2.31 MBytes
[  4]   1.00-2.00   sec  4.81 GBytes  41.3 Gbits/sec    0   2.31 MBytes
[  4]   2.00-3.00   sec  5.60 GBytes  48.1 Gbits/sec    0   2.31 MBytes
[  4]   3.00-4.00   sec  5.57 GBytes  47.8 Gbits/sec    0   2.31 MBytes
[  4]   4.00-5.00   sec  4.84 GBytes  41.6 Gbits/sec    0   2.31 MBytes
[  4]   5.00-6.00   sec  5.25 GBytes  45.1 Gbits/sec    0   2.31 MBytes
[  4]   6.00-7.00   sec  4.68 GBytes  40.2 Gbits/sec    0   2.31 MBytes
[  4]   7.00-8.00   sec  4.51 GBytes  38.7 Gbits/sec    0   2.31 MBytes
[  4]   8.00-9.00   sec  4.62 GBytes  39.7 Gbits/sec    0   2.31 MBytes
[  4]   9.00-10.00  sec  4.81 GBytes  41.4 Gbits/sec    0   2.31 MBytes
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Retr
[  4]   0.00-10.00  sec  50.0 GBytes  43.0 Gbits/sec    0             sender
[  4]   0.00-10.00  sec  50.0 GBytes  43.0 Gbits/sec                  receiver

iperf Done.
[cc@p1-neww ~]$
```

**<u>For running IPerf UDP Server Benchmark</u>**

i. Open 1st terminal and follow the commands to run the program:

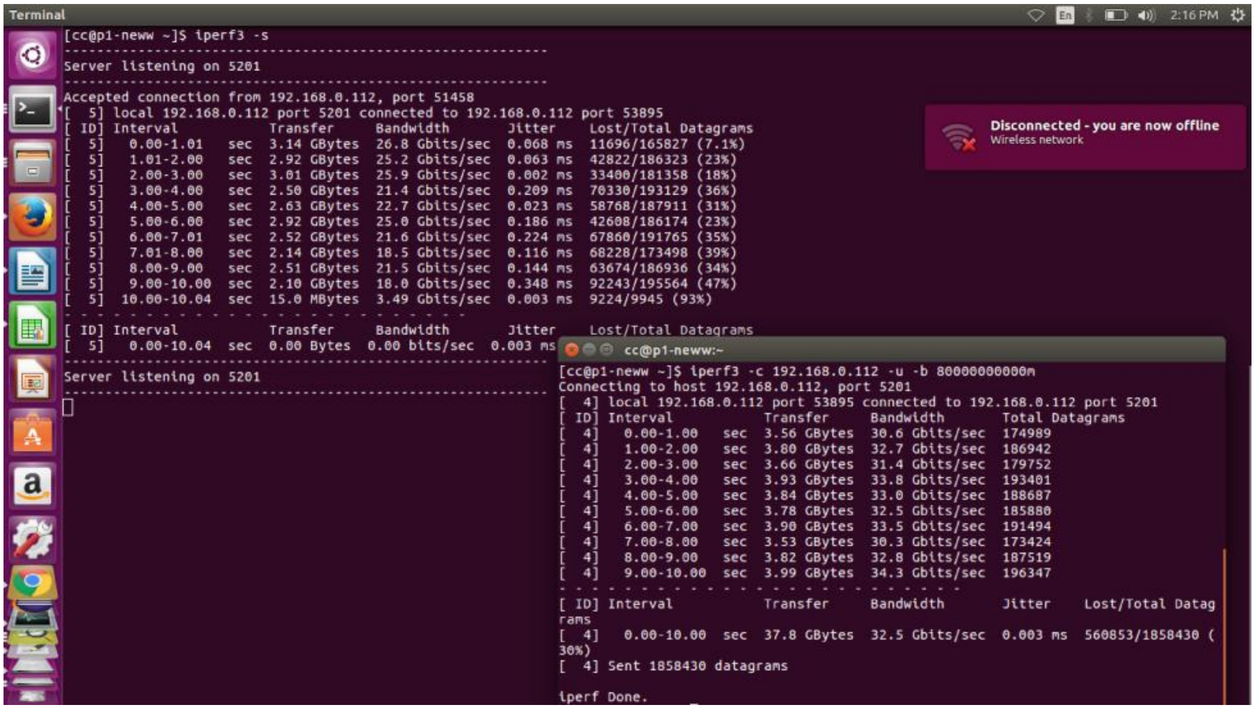
**→ iperf3 -s**

**<u>For running IPerf UDP Client Benchmark</u>**

i. Now, open 2nd terminal and follow the commands to run the program:

**→ iperf3 -c 192.168.0.112 -u -b 80000000000m**

Screenshot is displayed below:



```
[cc@p1-neww ~]$ iperf3 -s
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
Accepted connection from 192.168.0.112, port 51458
[  5] local 192.168.0.112 port 5201 connected to 192.168.0.112 port 53895
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-1.01   sec  3.14 GBytes  26.8 Gbits/sec  0.068 ms  11696/165827 (7.1%)
[  5]   1.01-2.00   sec  2.92 GBytes  25.2 Gbits/sec  0.063 ms  42822/186323 (23%)
[  5]   2.00-3.00   sec  3.01 GBytes  25.9 Gbits/sec  0.002 ms  33400/181358 (18%)
[  5]   3.00-4.00   sec  2.50 GBytes  21.4 Gbits/sec  0.209 ms  70330/193129 (36%)
[  5]   4.00-5.00   sec  2.63 GBytes  22.7 Gbits/sec  0.023 ms  58768/187911 (31%)
[  5]   5.00-6.00   sec  2.92 GBytes  25.0 Gbits/sec  0.186 ms  42608/186174 (23%)
[  5]   6.00-7.01   sec  2.52 GBytes  21.6 Gbits/sec  0.224 ms  67860/191765 (35%)
[  5]   7.01-8.00   sec  2.14 GBytes  18.5 Gbits/sec  0.116 ms  68228/173498 (39%)
[  5]   8.00-9.00   sec  2.51 GBytes  21.5 Gbits/sec  0.144 ms  63674/186936 (34%)
[  5]   9.00-10.00  sec  2.10 GBytes  18.0 Gbits/sec  0.348 ms  92243/195564 (47%)
[  5]  10.00-10.04  sec  15.0 MBytes  3.49 Gbits/sec  0.003 ms  9224/9945 (93%)
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datagrams
[  5]   0.00-10.04  sec  0.00 Bytes  0.00 bits/sec  0.003 ms
-----------------------------------------------------------
Server listening on 5201
-----------------------------------------------------------
```

```
[cc@p1-neww ~]$ iperf3 -c 192.168.0.112 -u -b 80000000000m
Connecting to host 192.168.0.112, port 5201
[  4] local 192.168.0.112 port 53895 connected to 192.168.0.112 port 5201
[ ID] Interval           Transfer     Bandwidth       Total Datagrams
[  4]   0.00-1.00   sec  3.56 GBytes  30.6 Gbits/sec  174989
[  4]   1.00-2.00   sec  3.80 GBytes  32.7 Gbits/sec  186942
[  4]   2.00-3.00   sec  3.66 GBytes  31.4 Gbits/sec  179752
[  4]   3.00-4.00   sec  3.93 GBytes  33.8 Gbits/sec  193401
[  4]   4.00-5.00   sec  3.84 GBytes  33.0 Gbits/sec  188687
[  4]   5.00-6.00   sec  3.78 GBytes  32.5 Gbits/sec  185880
[  4]   6.00-7.00   sec  3.90 GBytes  33.5 Gbits/sec  191494
[  4]   7.00-8.00   sec  3.53 GBytes  30.3 Gbits/sec  173424
[  4]   8.00-9.00   sec  3.82 GBytes  32.8 Gbits/sec  187519
[  4]   9.00-10.00  sec  3.99 GBytes  34.3 Gbits/sec  196347
- - - - - - - - - - - - - - - - - - - - - - - - -
[ ID] Interval           Transfer     Bandwidth       Jitter    Lost/Total Datag
rams
[  4]   0.00-10.00  sec  37.8 GBytes  32.5 Gbits/sec  0.003 ms  560853/1858430 (
30%)
[  4] Sent 1858430 datagrams

iperf Done.
```