CS553-Assignment 1

Performance

Introduction

This document talks about the program results and their analysis, represents result in graphical or tabular format, calculates theoretical performance of different parts of the computer systems and compares it with the standard benchmarks.

CPU, Memory, Disk and Network benchmarks were run on KVM baremetal instance having following configuration:

```
[cc@pa1-jagruti-abhishek ~]$ lscpu
Architecture:          x86_64
CPU op-mode(s):        32-bit, 64-bit
Byte Order:            Little Endian
CPU(s):                48
On-line CPU(s) list:   0-47
Thread(s) per core:    2
Core(s) per socket:    12
Socket(s):             2
NUMA node(s):          2
Vendor ID:             GenuineIntel
CPU family:            6
Model:                 63
Model name:            Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
Stepping:              2
CPU MHz:               1987.253
BogoMIPS:              4603.58
Virtualization:        VT-x
L1d cache:             32K
L1i cache:             32K
L2 cache:              256K
L3 cache:              30720K
NUMA node0 CPU(s):     0,2,4,6,8,10,12,14,16,18,20,22,24,26,28,30,32,34,36,38,40,42,44,46
NUMA node1 CPU(s):     1,3,5,7,9,11,13,15,17,19,21,23,25,27,29,31,33,35,37,39,41,43,45,47
[cc@pa1-jagruti-abhishek ~]$
```

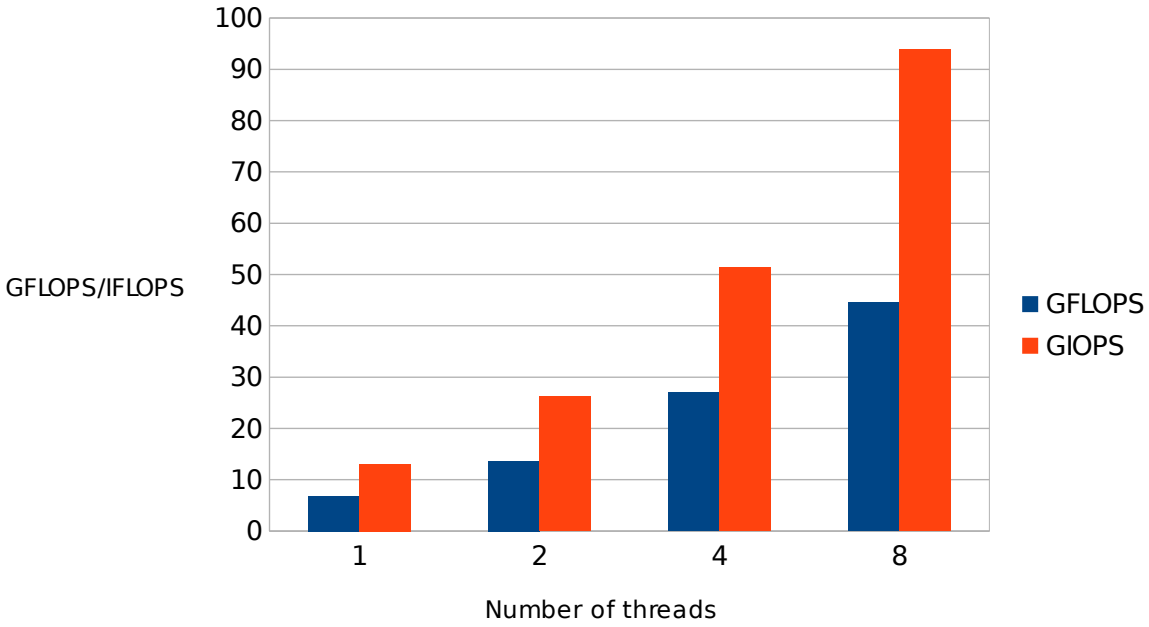
CPU

Questions a), b) and c)

Source code is available in the same directory as this file with file names ‘cpu_benchmark.c’ and ‘cpu_benchmark_2.c’.

‘cpu_benchmark.c’ measures processor speed in terms of double precision floating point operations per second (GFLOPS, 10 9 FLOPS) and integer operations per second (GIOPS, 10 9 IOPS) at varying level of concurrency for 1, 2, 4 and 8 threads. AVX instructions are also used to make sure that the performance is better.

The output is shown in graphical format below:



Above graph shows throughput values (GFLOPS/GIOPS) for 1, 2, 4 and 8 threads. It can be seen from the above graph that throughput increases with increase in number of threads.

Question d)

Theoretical performance of CPU = Speed (GHz) * No of CPU cores * CPU instruction per cycle * No of CPU per node

Taking into consideration the configuration of virtual machine (shown above), theoretical performance of CPU can be given as,

Theoretical performance of CPU = 2.30*24*8*2 = 883.2 GFLOPS

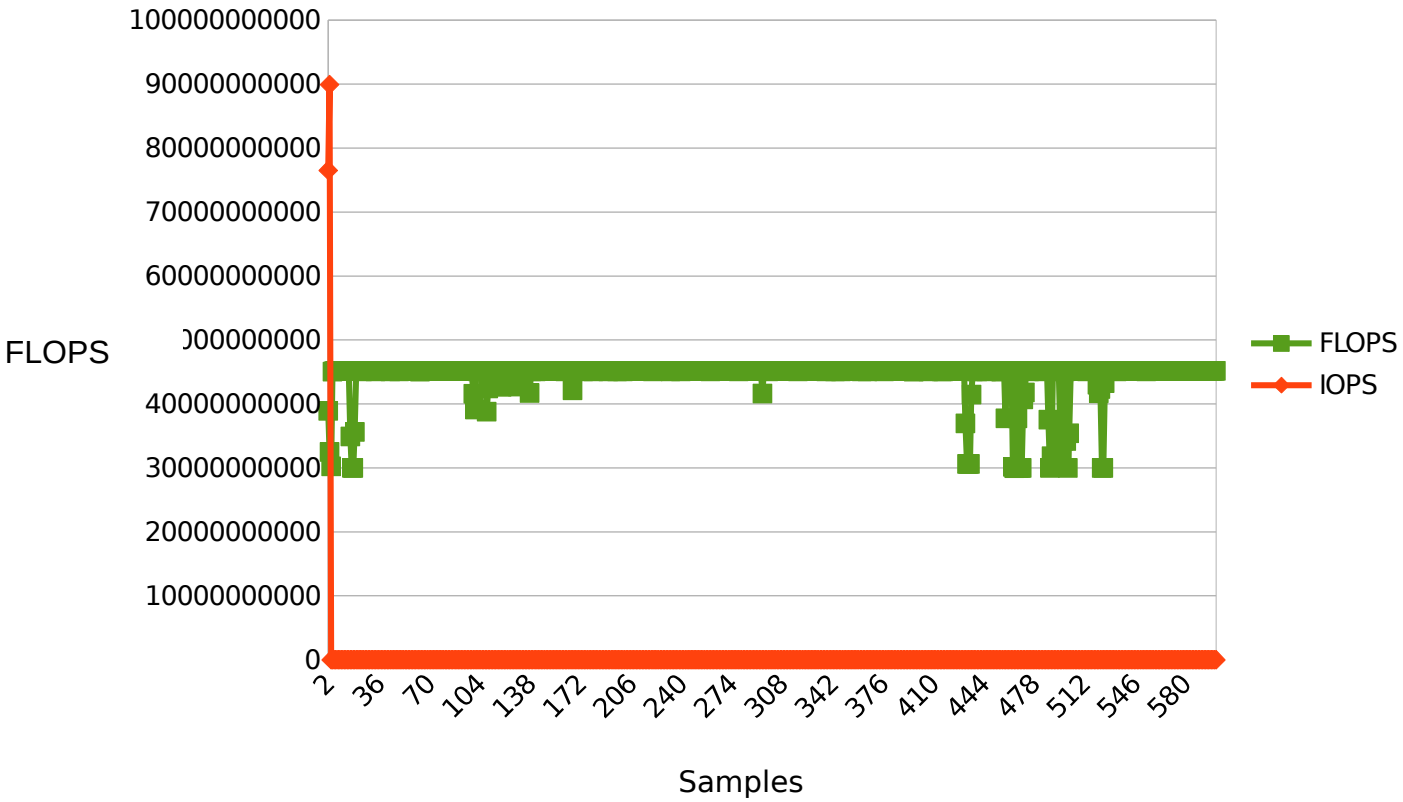
Question e)

Efficiency achieved is = (44.51/883.2)*100 = 5.04%

Efficiency obtained is too low because the maximum number of threads used in our programs is 8.

Question f)

'cpu_benchmark_2.c' calculates GFLOPS and IFLOPS at every second for about 10 mins and takes about 600 samples/second. The result is shown in graphical format below for FLOPS:



Question g)

Linpack benchmark was run on the same instance using double precision floating point and it gives following output

```
2017-10-10 01:47:35 (1.24 MB/s) - '/tmp/l_lpk.tgz' saved [27931818/27931818]

[root@pa1-abhi-jag tmp]# tar -xzf /tmp/l_lpk.tgz -C /tmp/
[root@pa1-abhi-jag tmp]# cp -a /tmp/linpack_11.1.2/benchmarks/linpack/ /usr/share/
[root@pa1-abhi-jag tmp]# ln -sf /usr/share/linpack/runme_xeon64 /usr/sbin/
[root@pa1-abhi-jag tmp]# ln -sf /usr/share/linpack/xlinpack_xeon64 /usr/sbin/
[root@pa1-abhi-jag tmp]# sed -i s'|./xlinpack_$arch lininput_$arch|/usr/sbin/xlinpack_$arch /usr/share/linpack/lininput_$arch|g' /usr/sb
[root@pa1-abhi-jag tmp]# CPU=$(cat /proc/cpuinfo | grep "model name" | tail -1)
[root@pa1-abhi-jag tmp]# COUNT=$(cat /proc/cpuinfo | grep processor | wc -l)
[root@pa1-abhi-jag tmp]# echo "CPU : $CPU"
CPU : model name        : Intel(R) Xeon(R) CPU E5-2670 v3 @ 2.30GHz
[root@pa1-abhi-jag tmp]# echo "COUNT : $COUNT"
COUNT : 48
[root@pa1-abhi-jag tmp]# runme_xeon64
This is a SAMPLE run script for SMP LINPACK. Change it to reflect
the correct number of CPUs/threads, problem input files, etc..
Tue Oct 10 01:50:08 UTC 2017
Intel(R) Optimized LINPACK Benchmark data

Current date/time: Tue Oct 10 01:50:08 2017

CPU frequency:    3.099 GHz
Number of CPUs: 2
Number of cores: 24
Number of threads: 48

Parameters are set to:

Number of tests: 15
Number of equations to solve (problem size) : 1000  2000  5000  10000 15000 18000 20000 22000 25000 26000 27000 30000 35000 40000 45000
Leading dimension of array                  : 1000  2000  5008  10000 15000 18008 20016 22008 25000 26000 27000 30000 35000 40000 45000
Number of trials to run                     : 4     2     2     2     2     2     2     2     2     2     1     1     1     1     1
Data alignment value (in Kbytes)            : 4     4     4     4     4     4     4     4     4     4     4     1     1     1     1
```

```
Number of tests: 15
Number of equations to solve (problem size) : 1000  2000  5000  10000 15000 18000 20000 22000 25000 26000 27000 30000 35000 40000 45000
Leading dimension of array                  : 1000  2000  5008  10000 15000 18008 20016 22008 25000 26000 27000 30000 35000 40000 45000
Number of trials to run                     : 4     2     2     2     2     2     2     2     2     2     1     1     1     1     1
Data alignment value (in Kbytes)            : 4     4     4     4     4     4     4     4     4     4     4     1     1     1     1

Maximum memory requested that can be used=16200901024, at the size=45000

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

Size   LDA    Align. Time(s)    GFlops   Residual     Residual(norm) Check
1000   1000   4      0.013      53.3441  9.216516e-13 3.143069e-02   pass
1000   1000   4      0.009      77.5424  9.216516e-13 3.143069e-02   pass
1000   1000   4      0.009      77.2139  9.216516e-13 3.143069e-02   pass
1000   1000   4      0.009      77.5044  9.216516e-13 3.143069e-02   pass
2000   2000   4      0.055      97.3176  3.356482e-12 2.919728e-02   pass
2000   2000   4      0.050      106.6565 3.356482e-12 2.919728e-02   pass
5000   5008   4      0.446      186.8351 2.266876e-11 3.160975e-02   pass
5000   5008   4      0.465      179.2635 2.266876e-11 3.160975e-02   pass
10000  10000  4      1.516      439.7558 7.851564e-11 2.768541e-02   pass
10000  10000  4      1.509      442.0355 7.851564e-11 2.768541e-02   pass
15000  15000  4      3.823      588.6502 1.434079e-10 2.258698e-02   pass
15000  15000  4      3.806      591.3462 1.434079e-10 2.258698e-02   pass
18000  18008  4      7.161      543.0658 2.422183e-10 2.652588e-02   pass
18000  18008  4      7.062      550.6485 2.422183e-10 2.652588e-02   pass
20000  20016  4      8.894      599.7239 3.319940e-10 2.938874e-02   pass
20000  20016  4      8.945      596.3100 3.319940e-10 2.938874e-02   pass
22000  22008  4      11.568     613.7510 3.294611e-10 2.413173e-02   pass
22000  22008  4      11.627     610.6357 3.294611e-10 2.413173e-02   pass
25000  25000  4      16.505     631.1813 3.900263e-10 2.217940e-02   pass
25000  25000  4      16.378     636.1093 3.900263e-10 2.217940e-02   pass
26000  26000  4      18.377     637.6684 4.193180e-10 2.204900e-02   pass
26000  26000  4      18.312     639.9627 4.193180e-10 2.204900e-02   pass
27000  27000  4      20.236     648.5138 4.375310e-10 2.133623e-02   pass
30000  30000  1      27.458     655.6167 5.178598e-10 2.041409e-02   pass
35000  35000  1      42.838     667.2964 8.075694e-10 2.344251e-02   pass
40000  40000  1      62.301     684.9003 1.163169e-09 2.586928e-02   pass
45000  45000  1      89.079     682.0248 1.275890e-09 2.244795e-02   pass

Performance Summary (GFlops)

Size   LDA    Align.  Average  Maximal
1000   1000   4       71.4012  77.5424
2000   2000   4       101.9871 106.6565
5000   5008   4       183.0493 186.8351
10000  10000  4       440.8956 442.0355
15000  15000  4       589.9982 591.3462
18000  18008  4       546.8571 550.6485
20000  20016  4       598.0170 599.7239
22000  22008  4       612.1933 613.7510
25000  25000  4       633.6453 636.1093
26000  26000  4       638.8156 639.9627
27000  27000  4       648.5138 648.5138
30000  30000  1       655.6167 655.6167
35000  35000  1       667.2964 667.2964
40000  40000  1       684.9003 684.9003
45000  45000  1       682.0248 682.0248

Residual checks PASSED

End of tests
```

Linpack benchmark maximum throughput value = 684 GFLOPS

Its efficiency as compared to theoretical performance is = (684/883.2) *100 = 77.44%

Efficiency achieved in our case is low as the number of threads used by linpack benchmark is too high i.e. 48 threads.

GPU

GPU source code is available in the same directory with file name ‘CUDA.cu’. We were not able to run the code on hardware as hardware was not free on chameleon but we ran the code locally.

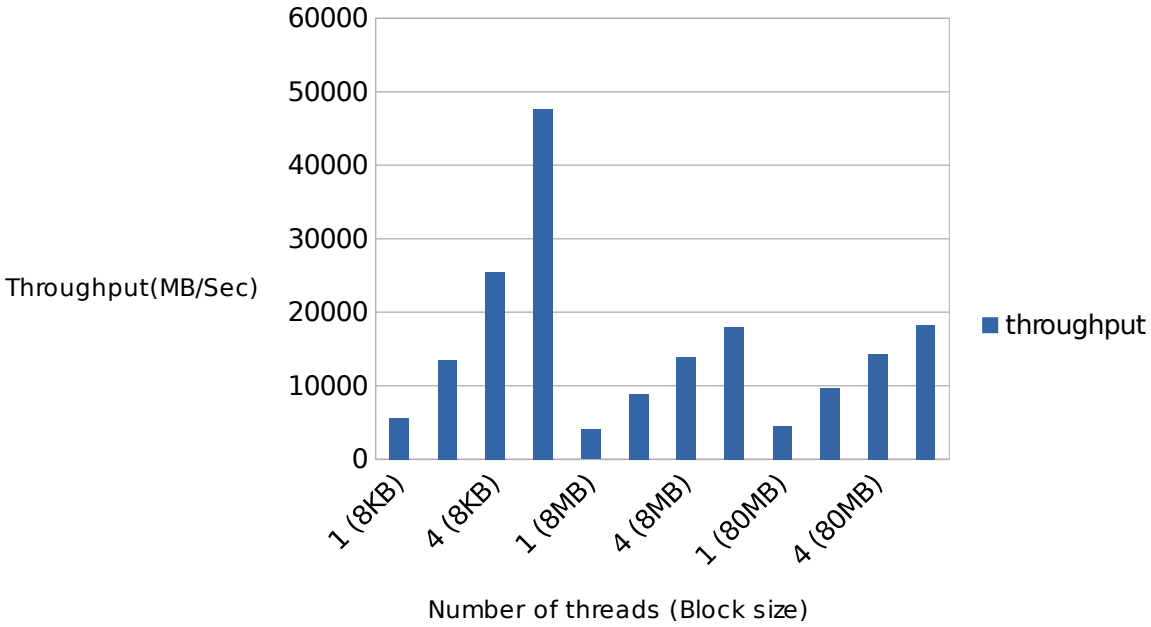
Memory

Questions a) b) c) d) e)

Memory benchmark is performed to measure the speed of memory in terms of throughput and latency. Speed is measured using different types of operations such as read/write, write sequential and write random with varying block sizes

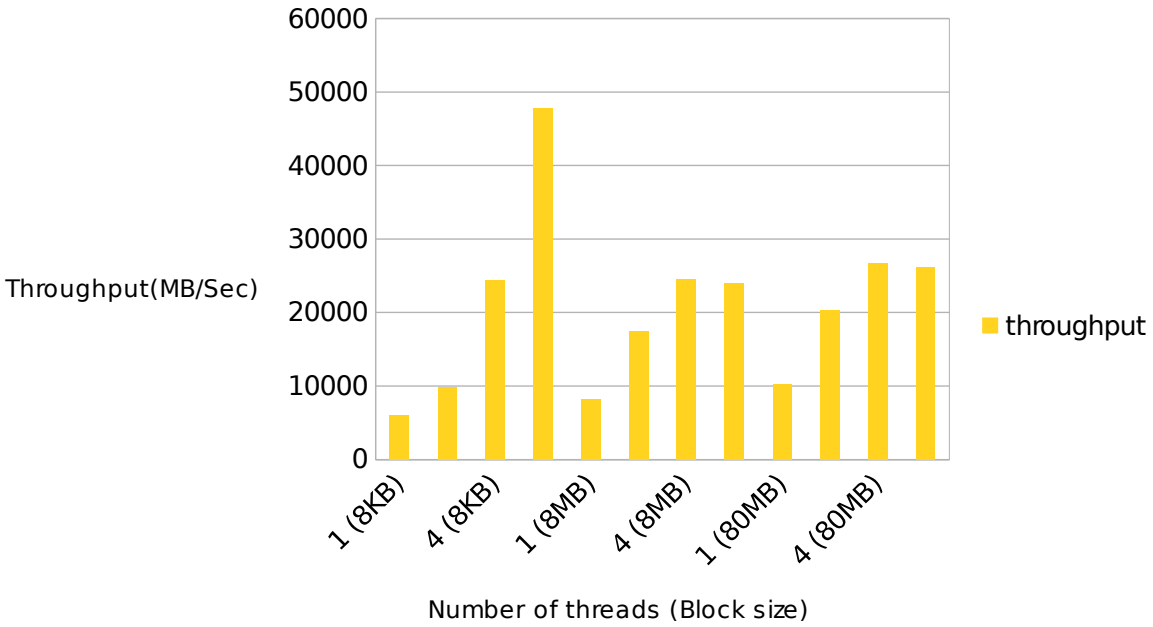
(8B, 8KB, 8MB, 80MB) and varying level of concurrency (1 thread, 2 threads, 4 threads, and 8 threads). Throughput is measured in MB/sec and latency is measured in microseconds. The output of the program is shown below in graphical format.

Memory throughput for read/write operations



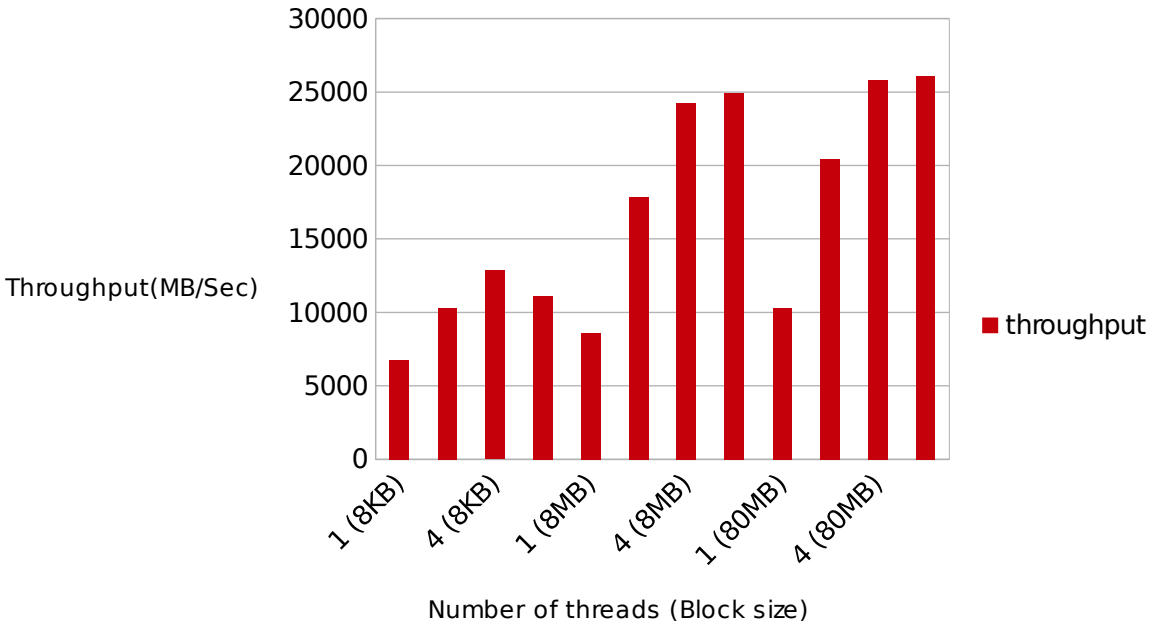
Above graph gives values of memory throughput in MB/sec for operation type read/write for 1, 2, 4 and 8 threads having block sizes 8KB, 8MB and 80 MB. Throughput value increases with increase in number of threads as we have used strong scaling.

Memory throughput for write sequential operations



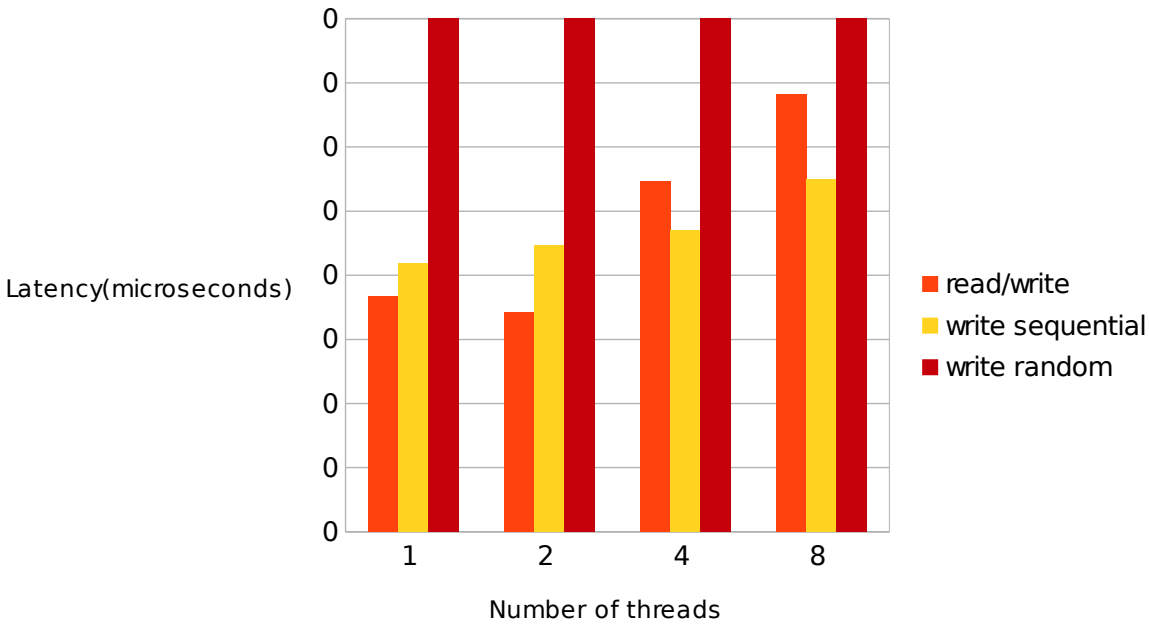
Above graph gives values of memory throughput in MB/sec for operation type write sequential for 1, 2, 4 and 8 threads having block sizes 8KB, 8MB and 80 MB. Throughput value increases with increase in number of threads as we have used strong scaling.

Memory throughput for write random operations



Above graph gives values of memory throughput in MB/sec for operation type write random for 1, 2, 4 and 8 threads having block sizes 8KB, 8MB and 80 MB. Throughput value increases with increase in number of threads as we have used strong scaling.

Memory latency (Block size – 8 bytes)



Above graph gives memory latency(microseconds) values for operation types read/write, write sequential and write random, having block size 8Bytes with varying level of concurrency (1,2,4,8 threads).

Question f)

Theoretical performance of memory can be calculated as follows:

Bus width is 64 bits that means can pass 8 bytes of data(64bits/8=8bytes). It is DD4.

Therefore, Memory speed = 8 * 1987 = 15896 MB/sec

Question g)

We ran the stream benchmark on the same instance if gives following output

```
[root@pa1-abhi-jag cc]# vi stream.c
[root@pa1-abhi-jag cc]# gcc -O stream.c -o stream
[root@pa1-abhi-jag cc]# ./stream
-------------------------------------------------------------
STREAM version $Revision: 5.10 $
-------------------------------------------------------------
This system uses 8 bytes per array element.
-------------------------------------------------------------
Array size = 10000000 (elements), Offset = 0 (elements)
Memory per array = 76.3 MiB (= 0.1 GiB).
Total memory required = 228.9 MiB (= 0.2 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 8837 microseconds.
   (= 8837 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:           12258.7     0.013063     0.013052     0.013087
Scale:          11974.3     0.013396     0.013362     0.013417
Add:            13698.7     0.017535     0.017520     0.017552
Triad:          13491.4     0.017802     0.017789     0.017814
-------------------------------------------------------------
Solution Validates: avg error less than 1.000000e-13 on all three arrays
-------------------------------------------------------------
[root@pa1-abhi-jag cc]#
```

Stream benchmark gives 13698 MB/sec

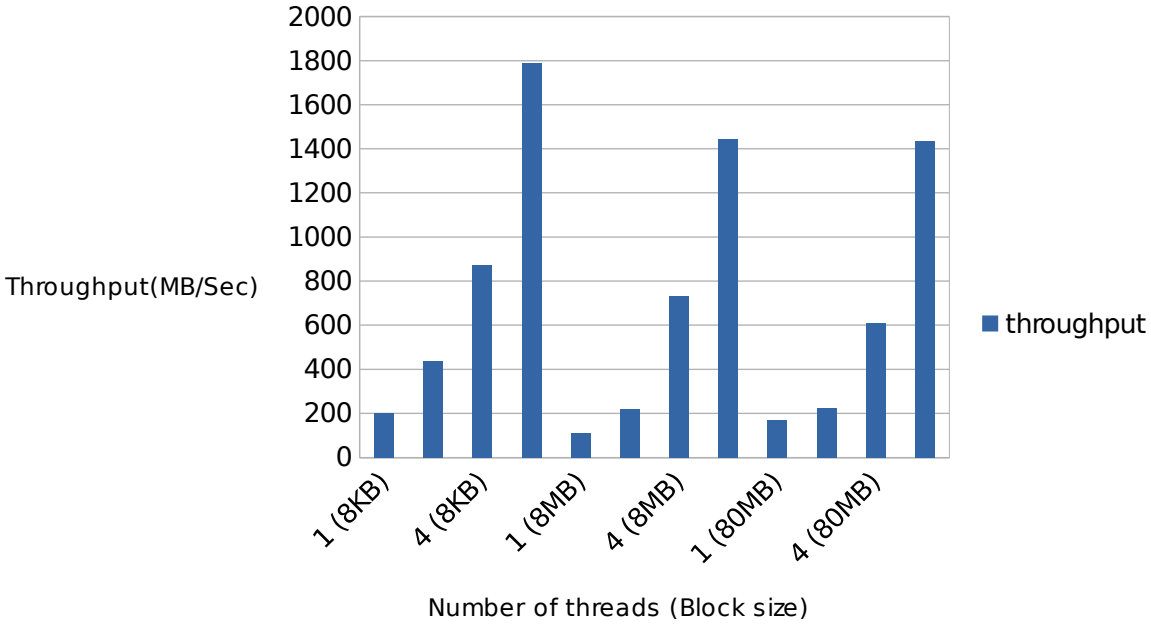
Its efficiency as compared to theoretical performance is = (13698/15896) *100 = 86.17%

Disk

Questions a) b) c) d)

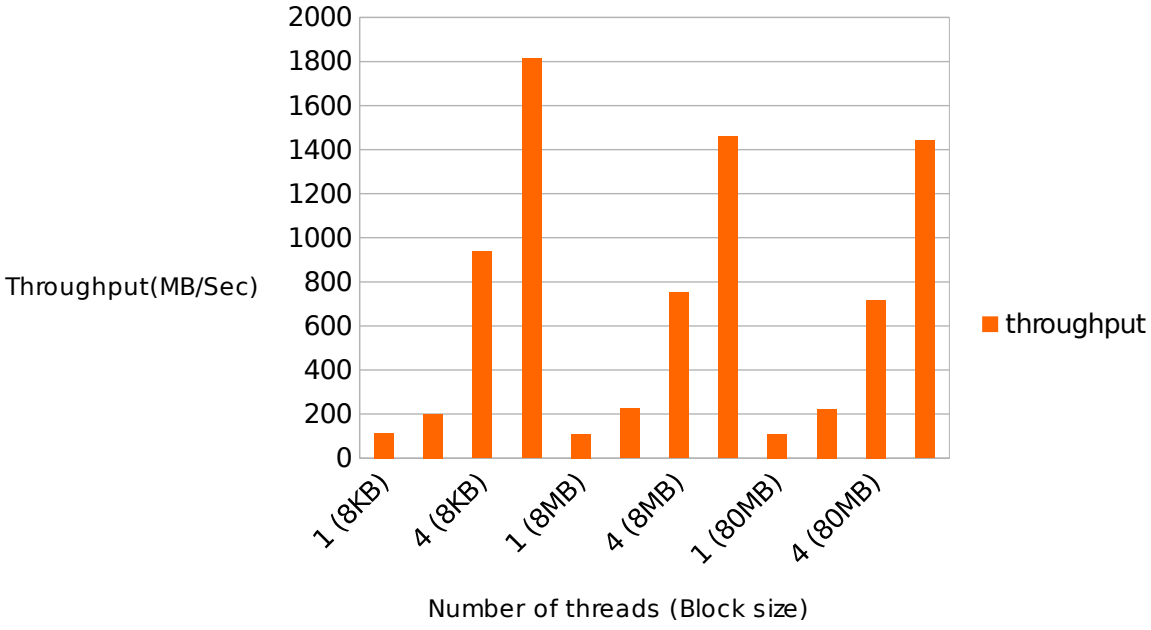
Disk benchmark is performed to measure the speed of disk in terms of throughput and latency. Speed is measured using different types of operations such as read/write, read sequential and read random with varying block sizes (8B, 8KB, 8MB, 80MB) and varying level of concurrency (1 thread, 2 threads, 4 threads, and 8 threads). Throughput is measured in MB/sec and latency is measured in milliseconds. The output of the program is shown below in graphical format.

Disk throughput for read/write operations



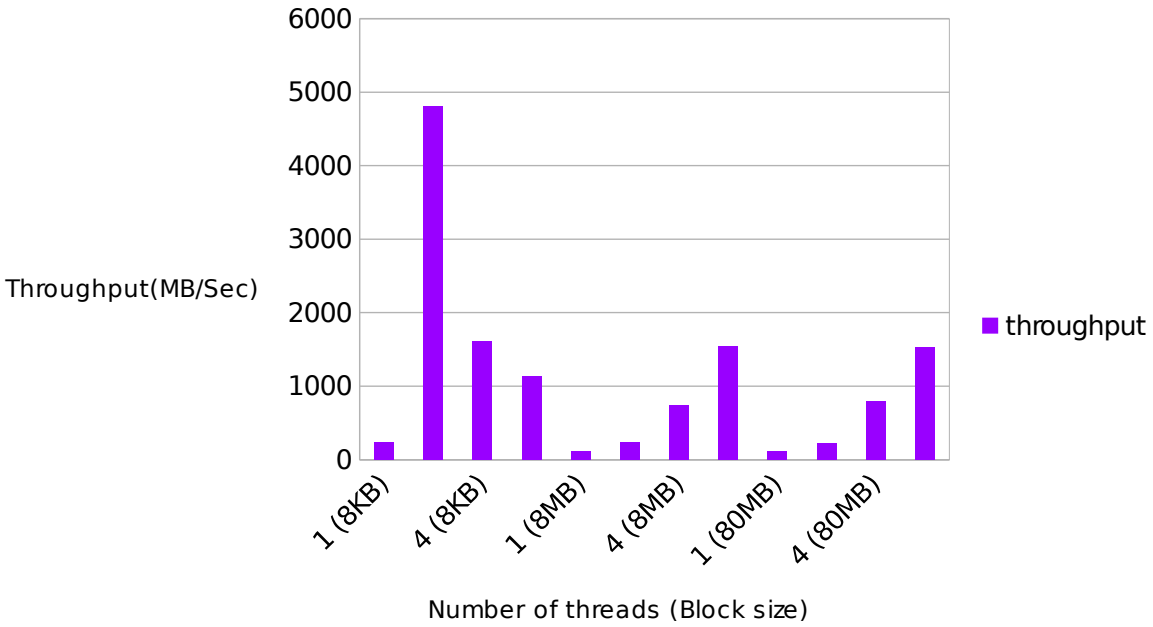
Above graph gives values of disk throughput in MB/sec for operation type read/write, with varying concurrency level (1, 2, 4 and 8 threads) having block sizes 8KB, 8MB and 80 MB. Throughput value increases with increase in number of threads as we have used strong scaling.

Disk throughput for read sequential operations



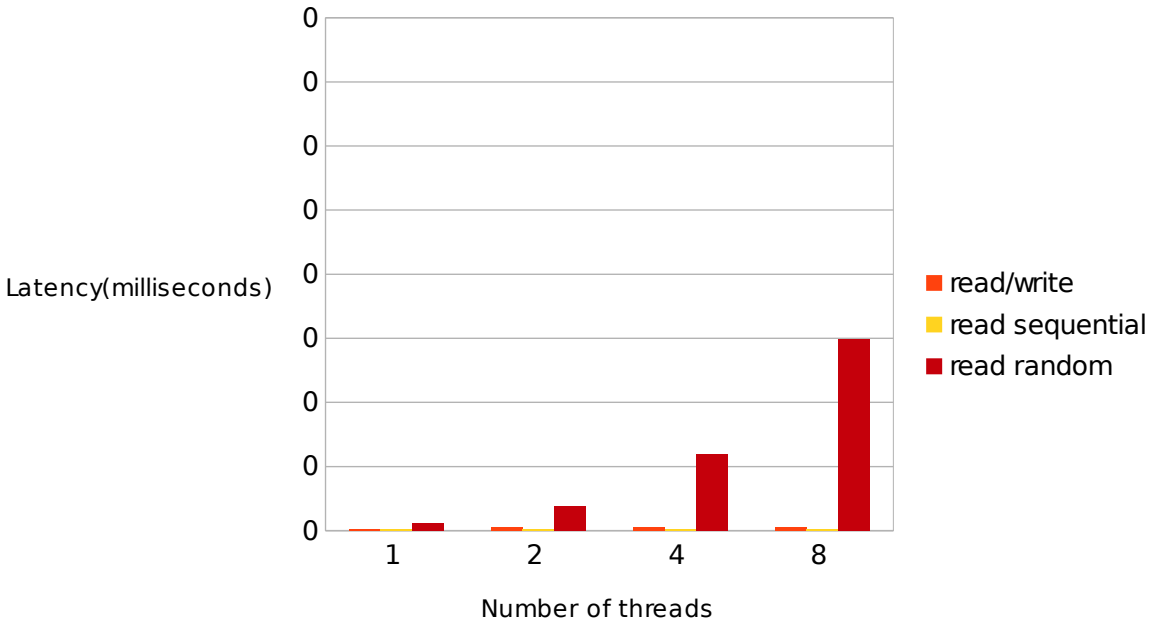
Above graph gives values of disk throughput in MB/sec for operation type read sequential, with varying concurrency level (1, 2, 4 and 8 threads) having block sizes 8KB, 8MB and 80 MB. Throughput value increases with increase in number of threads as we have used strong scaling.

Disk throughput for read random operations



Above graph gives values of disk throughput in MB/sec for operation type read random, with varying concurrency level (1, 2, 4 and 8 threads) having block sizes 8KB, 8MB and 80 MB.

Disk latency (Block size – 8 bytes)



Above graph gives memory latency(milliseconds) values for operation types read/write, read sequential and read random, having block size 8Bytes with varying level of concurrency (1,2,4,8 threads).

Question e)

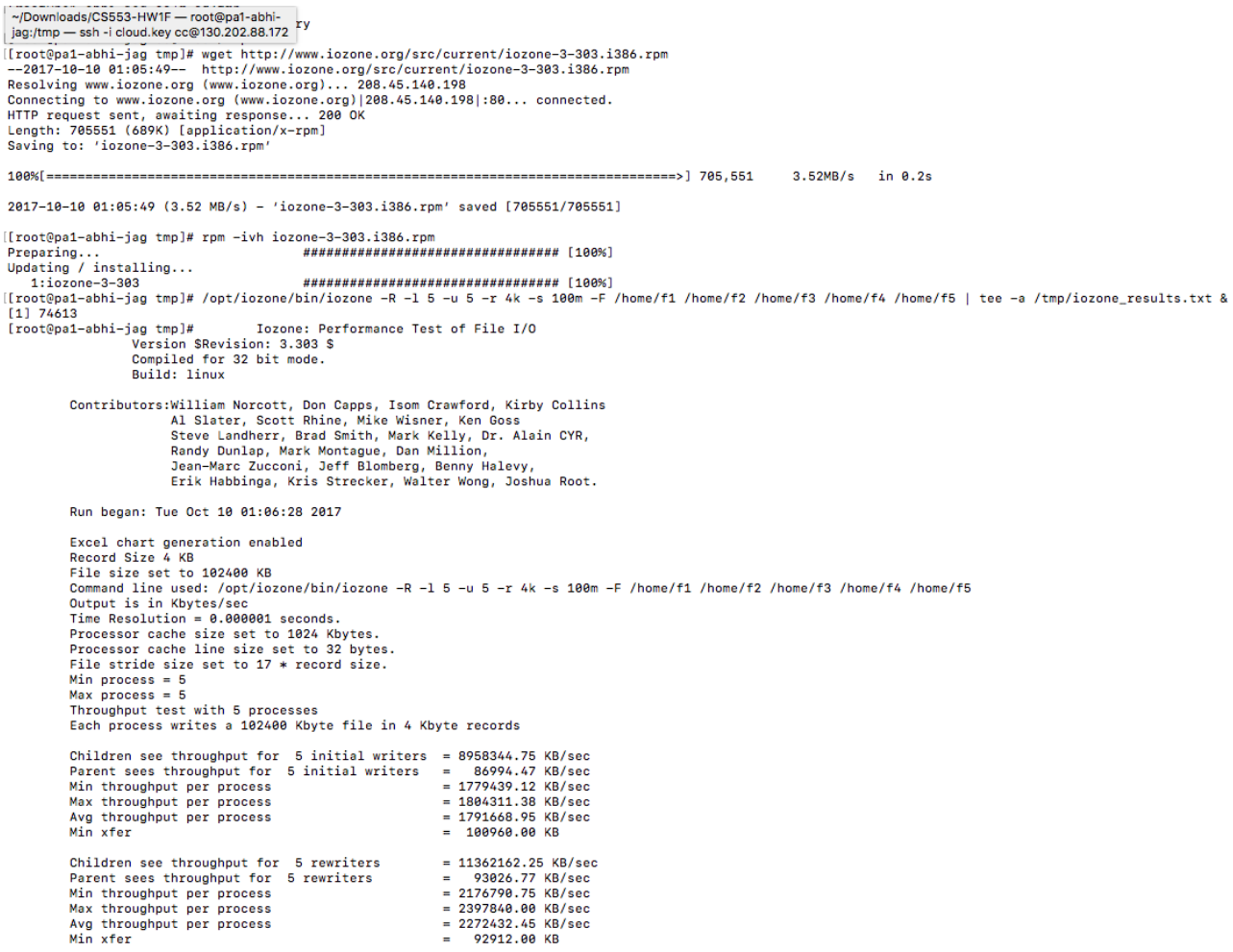
As seen in the graphs, sometimes the performance achieved decreases with increase in number of threads because the file can be accessed by only 1 thread at a time. Second thread has to wait sometimes until 1st thread finishes its operation. Because of dividing an amount of work between too many threads, each thread has less amount of work but the overhead of creating and terminating thread is more.

It is a solid state memory disk (SSD).

Question f)

Theoretical performance of disk is given as 6000 MB/s approximately in Wikipedia.

We ran the IOZone benchmark on the same instance if gives following output



~/Downloads/CS553-HW1F — root@pa1-abhi-jag:/tmp — ssh -i cloud.key cc@130.202.88.172

```
re-readers                                = 25401069.50 KB/sec
re-readers                                = 24865273.60 KB/sec
Min throughput per process                = 4296570.50 KB/sec
Max throughput per process                = 5717449.00 KB/sec
Avg throughput per process                = 5080213.90 KB/sec
Min xfer                                  =   76896.00 KB

Children see throughput for 5 reverse readers   = 19924483.75 KB/sec
Parent sees throughput for 5 reverse readers    = 19610841.61 KB/sec
Min throughput per process                = 3292567.50 KB/sec
Max throughput per process                = 4639339.50 KB/sec
Avg throughput per process                = 3984896.75 KB/sec
Min xfer                                  =   72756.00 KB

Children see throughput for 5 stride readers    = 20778434.25 KB/sec
Parent sees throughput for 5 stride readers     = 20458238.79 KB/sec
Min throughput per process                = 3605557.25 KB/sec
Max throughput per process                = 4626790.00 KB/sec
Avg throughput per process                = 4155686.85 KB/sec
Min xfer                                  =   79888.00 KB

Children see throughput for 5 random readers    = 19885583.25 KB/sec
Parent sees throughput for 5 random readers     = 19577725.65 KB/sec
Min throughput per process                = 3478291.25 KB/sec
Max throughput per process                = 4310470.00 KB/sec
Avg throughput per process                = 3977116.65 KB/sec
Min xfer                                  =   82616.00 KB

Children see throughput for 5 mixed workload    = 15264947.25 KB/sec
Parent sees throughput for 5 mixed workload     =  170634.73 KB/sec
Min throughput per process                = 2093939.00 KB/sec
Max throughput per process                = 4420253.00 KB/sec
Avg throughput per process                = 3052989.45 KB/sec
Min xfer                                  =   47652.00 KB

Children see throughput for 5 random writers    = 10787675.88 KB/sec
Parent sees throughput for 5 random writers     =   91794.29 KB/sec
Min throughput per process                = 2045460.75 KB/sec
Max throughput per process                = 2302053.25 KB/sec
Avg throughput per process                = 2157535.17 KB/sec
Min xfer                                  =   90984.00 KB

Children see throughput for 5 pwrite writers    = 8888532.62 KB/sec
Parent sees throughput for 5 pwrite writers     =   95151.98 KB/sec
Min throughput per process                = 1765779.00 KB/sec
Max throughput per process                = 1797848.88 KB/sec
Avg throughput per process                = 1777706.52 KB/sec
Min xfer                                  =  100600.00 KB

Children see throughput for 5 pread readers     = 16112060.75 KB/sec
Parent sees throughput for 5 pread readers      = 8554165.06 KB/sec
Min throughput per process                = 2653412.00 KB/sec
Max throughput per process                = 3812207.00 KB/sec
Avg throughput per process                = 3222412.15 KB/sec
Min xfer                                  =   71284.00 KB


"Throughput report Y-axis is type of test X-axis is number of processes"
"Record size = 4 Kbytes "
"Output is in Kbytes/sec"
```

```
        Avg throughput per process                      = 3977116.65 KB/sec
        Min xfer                                        =   82616.00 KB

        Children see throughput for 5 mixed workload    = 15264947.25 KB/sec
        Parent sees throughput for 5 mixed workload     =  170634.73 KB/sec
        Min throughput per process                      = 2093939.00 KB/sec
        Max throughput per process                      = 4420253.00 KB/sec
        Avg throughput per process                      = 3052989.45 KB/sec
        Min xfer                                        =   47652.00 KB

        Children see throughput for 5 random writers    = 10787675.88 KB/sec
        Parent sees throughput for 5 random writers     =   91794.29 KB/sec
        Min throughput per process                      = 2045460.75 KB/sec
        Max throughput per process                      = 2302053.25 KB/sec
        Avg throughput per process                      = 2157535.17 KB/sec
        Min xfer                                        =   90984.00 KB

        Children see throughput for 5 pwrite writers    = 8888532.62 KB/sec
        Parent sees throughput for 5 pwrite writers     =   95151.98 KB/sec
        Min throughput per process                      = 1765779.00 KB/sec
        Max throughput per process                      = 1797848.88 KB/sec
        Avg throughput per process                      = 1777706.52 KB/sec
        Min xfer                                        =  100600.00 KB

        Children see throughput for 5 pread readers     = 16112060.75 KB/sec
        Parent sees throughput for 5 pread readers      = 8554165.06 KB/sec
        Min throughput per process                      = 2653412.00 KB/sec
        Max throughput per process                      = 3812207.00 KB/sec
        Avg throughput per process                      = 3222412.15 KB/sec
        Min xfer                                        =   71284.00 KB


"Throughput report Y-axis is type of test X-axis is number of processes"
"Record size = 4 Kbytes "
"Output is in Kbytes/sec"

"  Initial write " 8958344.75

"        Rewrite " 11362162.25

"           Read " 25469894.50

"        Re-read " 25401069.50

"   Reverse Read " 19924483.75

"    Stride read " 20778434.25

"    Random read " 19885583.25

" Mixed workload " 15264947.25

"   Random write " 10787675.88

"         Pwrite " 8888532.62

"          Pread " 16112060.75


iozone test complete.
[root@pa1-abhi-jag tmp]#
```

~/Downloads/CS553-HW1F — root@pa1-abhi-jag:/tmp — ssh -i cloud.key cc@130.202.88.172

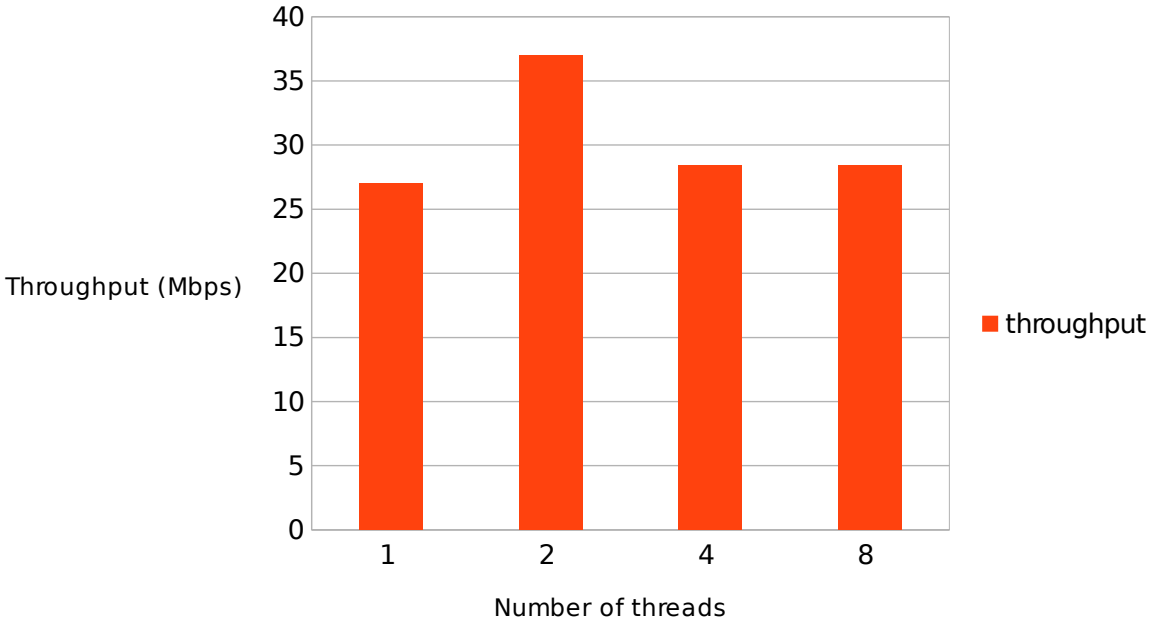
IOZone benchmark gives 3222 MB/sec average throughput.

Its efficiency as compared to theoretical performance is = (3222/6000) * 100 = 53.7%

Network

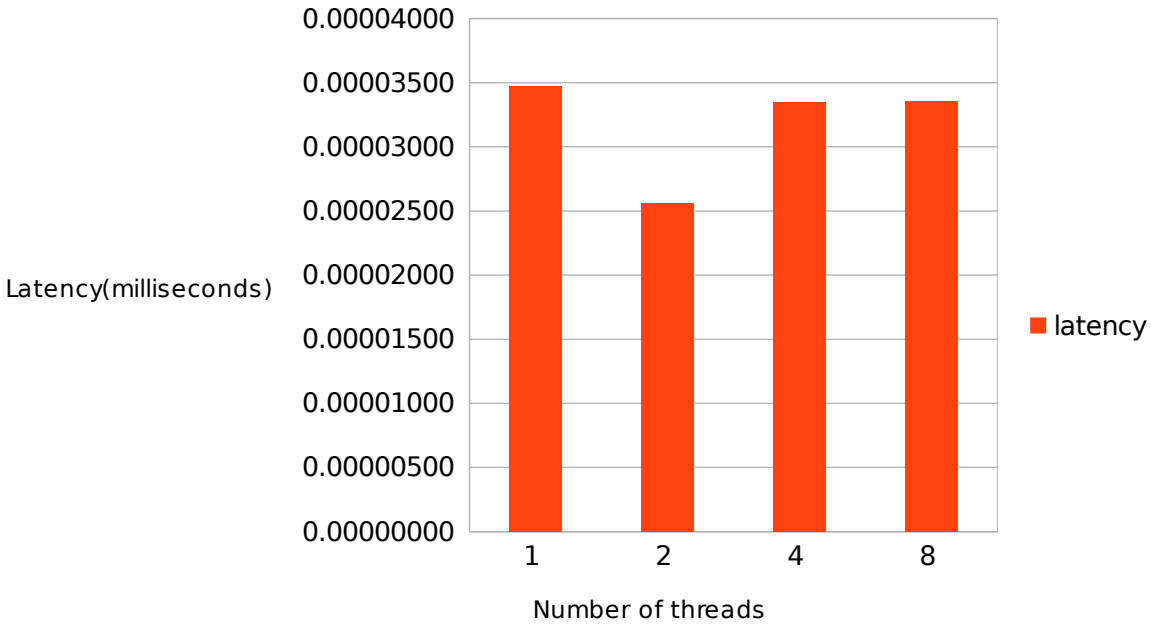
Network benchmark has done taking loopback into consideration. The experiment performed are basically done on single nodes but we have given results using different IPs as well.

Network throughput for UDP, Block size – 64KB



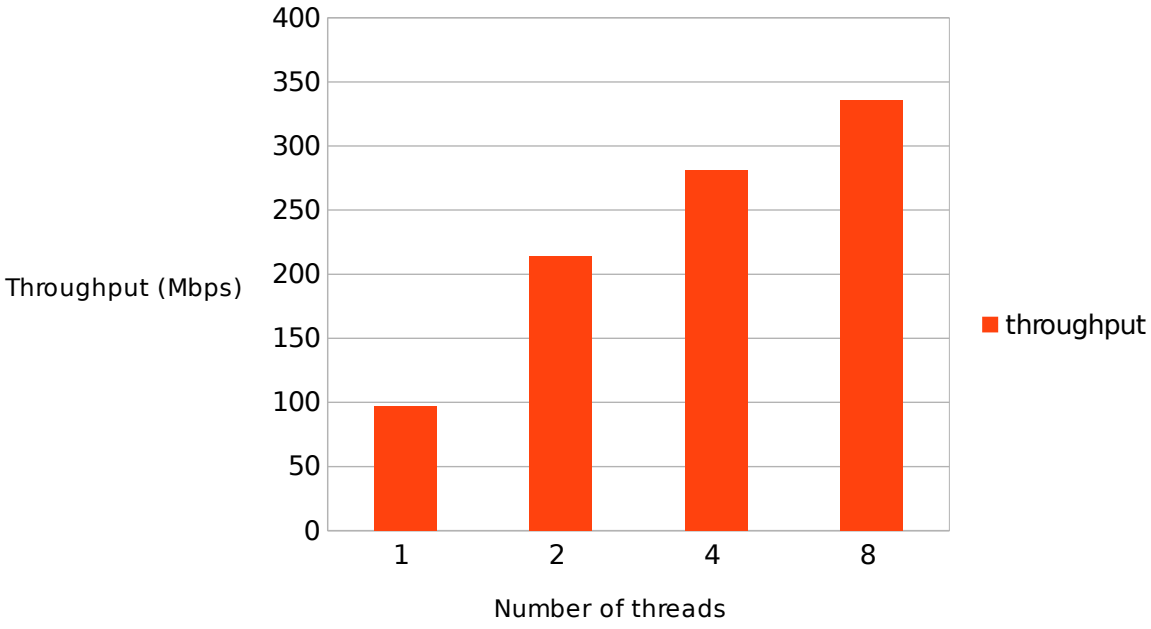
Since, UDP is unsecured we may see from the above graph that the throughput is not in accordance with the number of threads. Sometimes increase in number of threads doesn't affect the transmission done following UDP protocol.

Network latency for UDP, Block size – 64KB



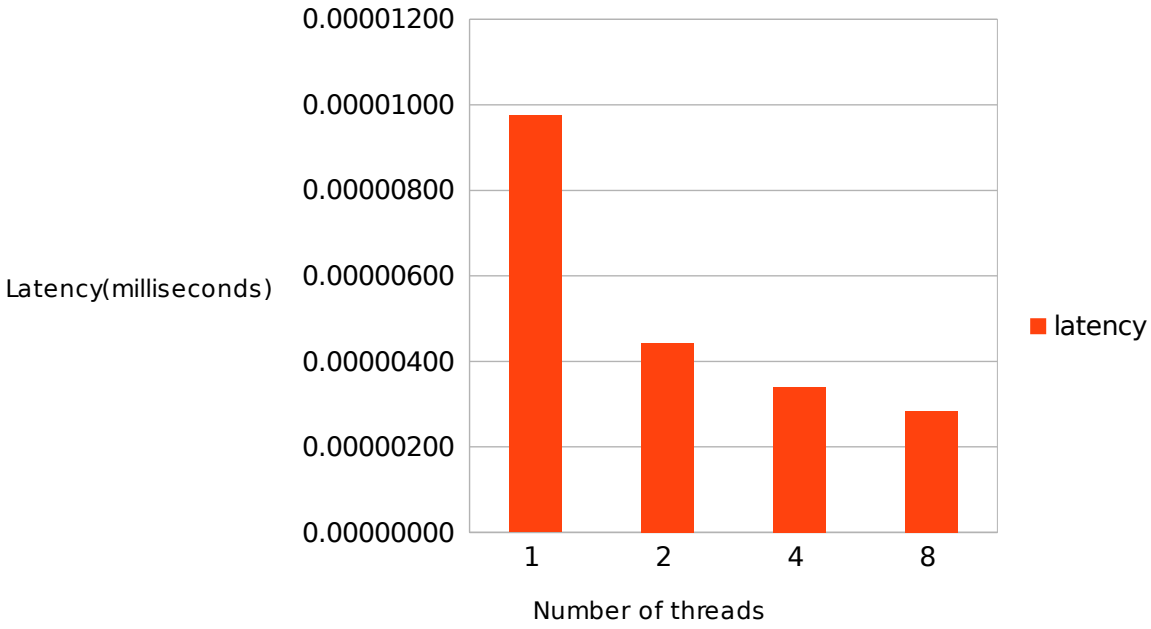
As seen from the above graph, UDP run time cannot be determined based on theory because of it being a wireless mode of transmission, packet loss keep on occurring affecting the speed of transfer.

Network throughput for TCP, Block size – 64KB



TCP is the perfect representation of theoretical calculation. The throughput increases with the increase in number of threads because we have followed strong scaling.

Network latency for TCP, Block size – 64KB



As seen from above graph, latency reduces as we increase the number of threads as the wait time of the packets to be transmitted reduces in proportion with the increase in number of threads.

We ran **IPERF** benchmark and it gave following result:

```
[iperf -c 10.140.81.19  -u
[^C[root@pa1-abhi-jag cc]# iperf -c 130.202.88.172 -u
------------------------------------------------------------
Client connecting to 130.202.88.172, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.140.81.19 port 57077 connected with 130.202.88.172 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 906 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 5 tries.
```

As we can see in the snapshot, there is a loss of acknowledgment from the datagram. It is happening because we are transmitting data through UDP while in the other case if we have a look in the snapshot below, we can see that the data transmitted through TCP has no loss and is more reliable.

```
------------------------------------------------------------
Client connecting to 10.140.81.19, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11.76 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.140.81.19 port 51943 connected with 10.140.81.19 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.16 GBytes  1000 Mbits/sec
[  3] Sent 1700679 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 2 tries.
[root@pa1-abhi-jag cc]# iperf -c 130.202.88.172 -u -b 1000m
------------------------------------------------------------
Client connecting to 130.202.88.172, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11.76 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.140.81.19 port 33768 connected with 130.202.88.172 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.16 GBytes  1000 Mbits/sec
[  3] Sent 850354 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 5 tries.
[root@pa1-abhi-jag cc]# iperf -u -s -p5000
------------------------------------------------------------
Server listening on UDP port 5000
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
^C[root@pa1-abhi-jag cc]# iperf -c 130.202.88.172 -u
------------------------------------------------------------
Client connecting to 130.202.88.172, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.140.81.19 port 47098 connected with 130.202.88.172 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 906 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 5 tries.
[root@pa1-abhi-jag cc]# iperf -c 130.202.88.172 -d
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 130.202.88.172, TCP port 5001
TCP window size: 1.15 MByte (default)
------------------------------------------------------------
[  5] local 10.140.81.19 port 49792 connected with 130.202.88.172 port 5001
[  4] local 10.140.81.19 port 5001 connected with 130.202.88.172 port 49792
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  10.9 GBytes  9.36 Gbits/sec
[  4]  0.0-10.0 sec  10.9 GBytes  9.35 Gbits/sec
[root@pa1-abhi-jag cc]#
```

```
[root@pa1-abhi-jag cc]# iperf -c 130.202.88.172 -u -b 1000m
------------------------------------------------------------
Client connecting to 130.202.88.172, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11.76 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.140.81.19 port 33768 connected with 130.202.88.172 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.16 GBytes  1000 Mbits/sec
[  3] Sent 850354 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 5 tries.
[root@pa1-abhi-jag cc]# iperf -u -s -p5000
------------------------------------------------------------
Server listening on UDP port 5000
Receiving 1470 byte datagrams
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
^C[root@pa1-abhi-jag cc]# iperf -c 130.202.88.172 -u
------------------------------------------------------------
Client connecting to 130.202.88.172, UDP port 5001
Sending 1470 byte datagrams, IPG target: 11215.21 us (kalman adjust)
UDP buffer size:  208 KByte (default)
------------------------------------------------------------
[  3] local 10.140.81.19 port 47098 connected with 130.202.88.172 port 5001
[ ID] Interval       Transfer     Bandwidth
[  3]  0.0-10.0 sec  1.25 MBytes  1.05 Mbits/sec
[  3] Sent 906 datagrams
read failed: Connection refused
[  3] WARNING: did not receive ack of last datagram after 5 tries.
[root@pa1-abhi-jag cc]# iperf -c 130.202.88.172 -d
------------------------------------------------------------
Server listening on TCP port 5001
TCP window size: 85.3 KByte (default)
------------------------------------------------------------
------------------------------------------------------------
Client connecting to 130.202.88.172, TCP port 5001
TCP window size: 1.15 MByte (default)
------------------------------------------------------------
[  5] local 10.140.81.19 port 49792 connected with 130.202.88.172 port 5001
[  4] local 10.140.81.19 port 5001 connected with 130.202.88.172 port 49792
[ ID] Interval       Transfer     Bandwidth
[  5]  0.0-10.0 sec  10.9 GBytes  9.36 Gbits/sec
[  4]  0.0-10.0 sec  10.9 GBytes  9.35 Gbits/sec
[root@pa1-abhi-jag cc]#
```