

**CS 553 Cloud Computing  
Programming Assignment 1  
Hariprasad Ravi Kumar  
(A20348609)**

**Performance Evaluation**

## CPU Benchmarking:

### System Configuration:

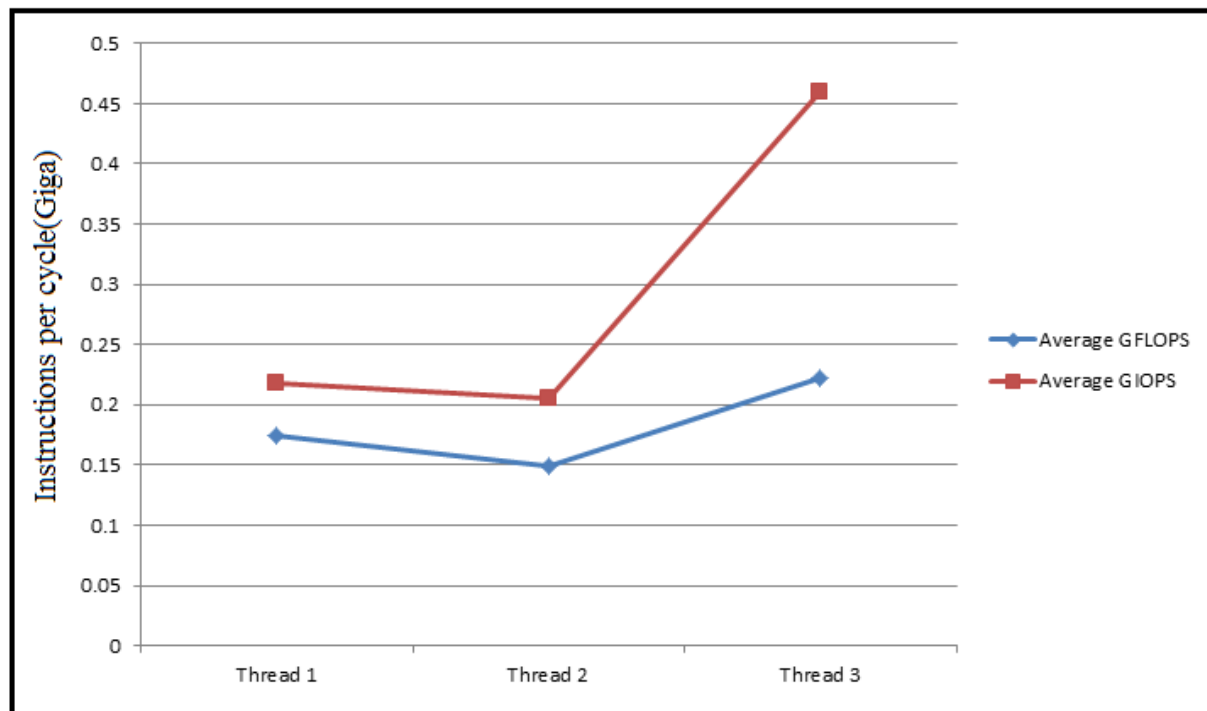
- Operating System: Ubuntu 14.4
- RAM: 8 GB
- No of Physical cores: 4
- No of logical cores: 8.

No. of Threads	Average GFLOPS	Average GIOPS
1	0.174785	0.217382
2	0.149712	0.205318
4	0.221784	0.459432

### Graph for CPU Benchmarking:

- X Axis: Number of Threads
- Y Axis: Instructions per cycle in giga [ GFLOPS and GIOPS]

The X axis is number of threads and Y axis is instructions per cycle having both GFLOPS and GIOPS. The minimum numbers of threads are 1 and maximum are 4. From the graph we can observe that the FLOPS and IOPS are highest when we run on a 4 thread.



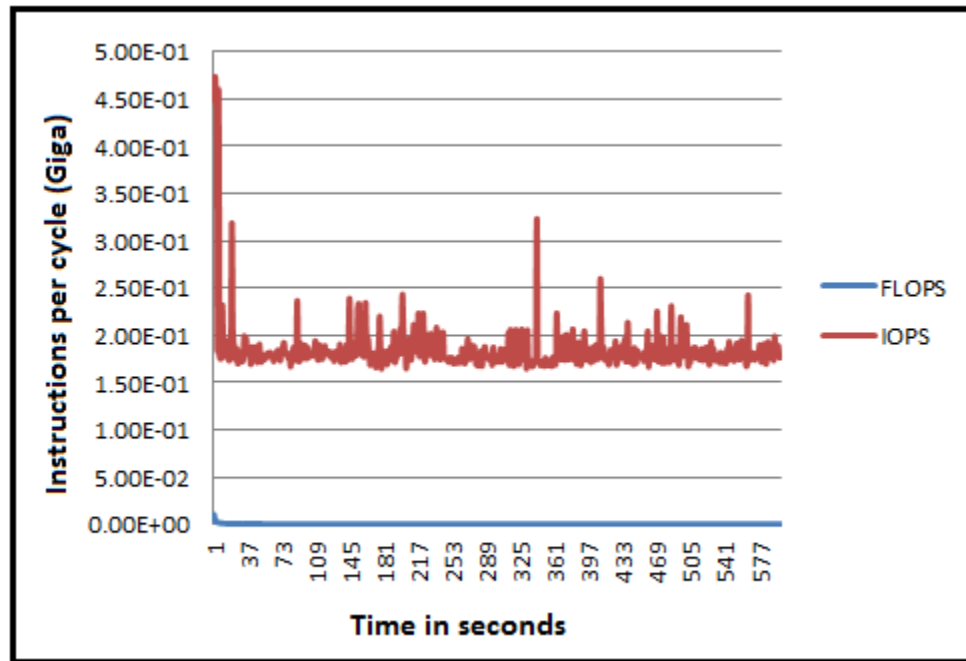
**Where:** X Axis: Number of Threads

Y Axis: Instructions per cycle in giga

Graph for CPU Benchmarking for 600 samples using thread 4:

- X Axis: Time (0 to 10min)
- Y Axis: Instructions per cycle in giga [ GFLOPS and GIOPS]

The X axis is Time in seconds for 600 seconds and Y axis is instructions per cycle having both GFLOPS and GIOPS. The thread used is 4. From the graph we can observe that the FLOPS has high readings and IOPS as a constant flow.



**Where:** X Axis: Time in 600 seconds

Y Axis: Instructions per cycle in giga

**Theoretical Performance:** Number of cores \* Number of Instructions per cycle \* Clock Speed = 76.8 GFLOPS

**Efficiency achieved compared to theoretical value:** 40.69%

## Linpack Results:

```
ec2-user@ip-172-31-50-176:~/linpack_11.1.2/benchmarks/linpack
CPU frequency: 3.190 GHz
Number of CPUs: 1
Number of cores: 1
Number of threads: 1
Parameters are set to:
Number of tests: 15
Number of equations to solve (problem size) : 1000 2000 5000 10000 15000 18000
0 20000 22000 25000 26000 27000 30000 35000 40000 45000
Leading dimension of array : 1000 2000 5008 10000 15000 18000
8 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 1 1 1 1
Maximum memory requested that can be used=800204096, at the size=10000
===== Timing linear equation system solver =====
Size LDA Align. Time(s) GFlops Residual Residual(norm) Check
1000 1000 4 0.039 16.9551 9.900691e-13 3.376390e-02 pass
1000 1000 4 0.037 17.9976 9.900691e-13 3.376390e-02 pass
1000 1000 4 0.039 16.9984 9.900691e-13 3.376390e-02 pass
1000 1000 4 0.037 17.8477 9.900691e-13 3.376390e-02 pass
2000 2000 4 0.281 18.9993 4.053480e-12 3.526031e-02 pass
2000 2000 4 0.276 19.3699 4.053480e-12 3.526031e-02 pass
5000 5000 4 4.142 20.1331 2.336047e-11 3.257420e-02 pass
5000 5000 4 4.115 20.2612 2.336047e-11 3.257420e-02 pass
10000 10000 4 32.804 20.3286 1.124127e-10 3.963786e-02 pass
10000 10000 4 32.654 20.4223 1.124127e-10 3.963786e-02 pass
Performance Summary (GFlops)
Size LDA Align. Average Maximal
1000 1000 4 17.4497 17.9976
2000 2000 4 19.1846 19.3699
5000 5000 4 20.1972 20.2612
10000 10000 4 20.3754 20.4223
Residual checks PASSED
End of tests
Done: Fri Feb 12 02:12:06 UTC 2016
[ec2-user@ip-172-31-50-176 linpack]$
```

The maximum GFLOPS achieved 20.4223

The efficiency of the results from Linpack when compared with theoretical performance is 49.18%

## **DISK Benchmarking:**

- All the experiments were performed on a VM of 20GB hard disk. The read and write operations to and from the memory are performed using this file system.
- The accuracy of the result could have been increased if we had a dual boot or the entire system running on Ubuntu. The following results are based on above configuration.
- The 1 MB access speeds are from cached space. Thus, we consider theoretical speed(max) of 3Gbps.

### **1 Byte**

Threads	Seq Read	Seq Write	Ran Read	Ran Write	Seq Read Latency	Seq Write Latency	Ran Read Latency	Ran Write Latency
1	0.040291	0.041210	0.040041	0.033062	0.236698	0.231252	0.238176	0.288446
2	0.035598	0.032814	0.036175	0.032814	0.267901	0.290627	0.2636265	0.334745

### **1 KB**

Threads	Seq Read	Seq Write	Ran Read	Ran Write	Seq Read Latency	Seq Write Latency	Ran Read Latency	Ran Write Latency
1	1261.708656	193.532006	2977.324695	161.361946	0.774000	5.046000	0.328000	6.052000
2	1163.960072	356.995979	1266.618029	206.614302	0.839000	2.735500	0.771000	4.726500

### **1 MB**

Threads	Seq Read	Seq Write	Ran Read	Ran Write	Seq Read Latency	Seq Write Latency	Ran Read Latency	Ran Write Latency
1	8568.980291	490.436488	3447.087211	450.673751	1.167000	20.39000	2.901000	22.18900
2	8960.573477	415.541242	3770.028275	564.461504	1.116000	24.065000	2.652500	17.716000

Seq Read: Sequential read in Mbps

Seq Write: Sequential write in Mbps

Ran Read: Random read in Mbps

Ran Write: Random write in Mbps

Seq Latency: Sequential Latency in ms

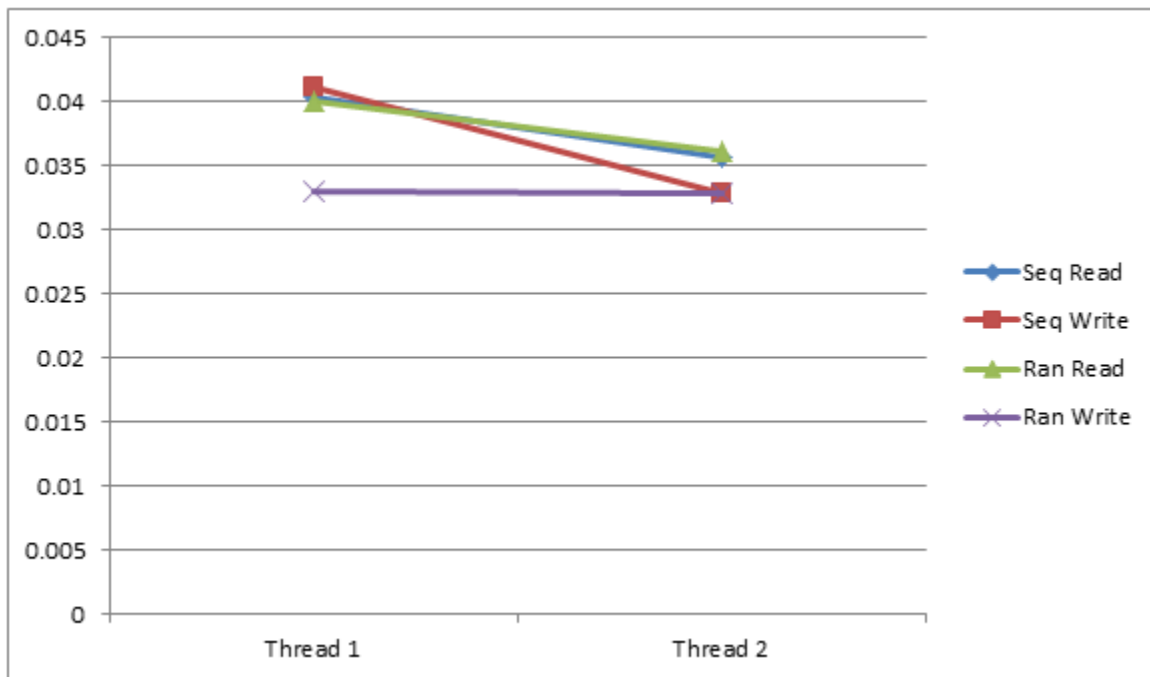
Ran Latency: Random Latency in ms

### Graph for DISK Benchmarking:

For each of 1BYTE, 1KB, 1MB we plot graphs for sequential read, sequential write, random read, random write and latency for 1 and 2 threads. The latency has both values of random and sequential read and write in it. From the graphs we notice that the performance is best for 1 thread.

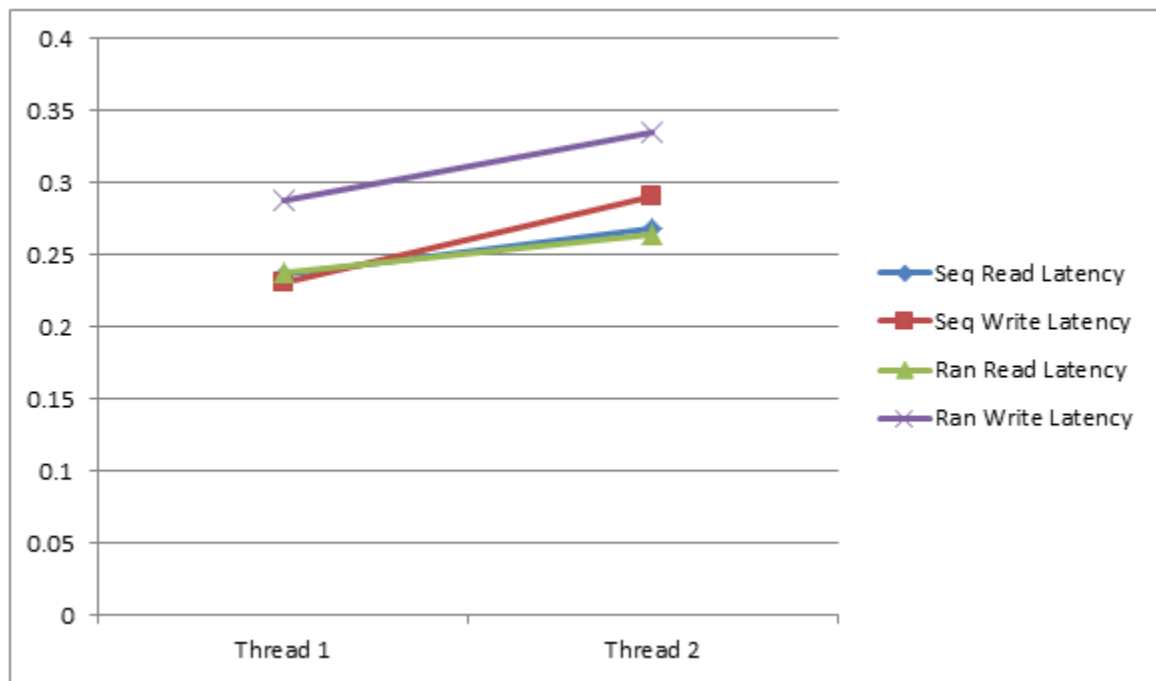
### 1 BYTE:

**1Byte Sequential and Random Speed: Read and Write Throughput**



- **X Axis:** Number of Threads
- **Y Axis:** Speed in MBPS

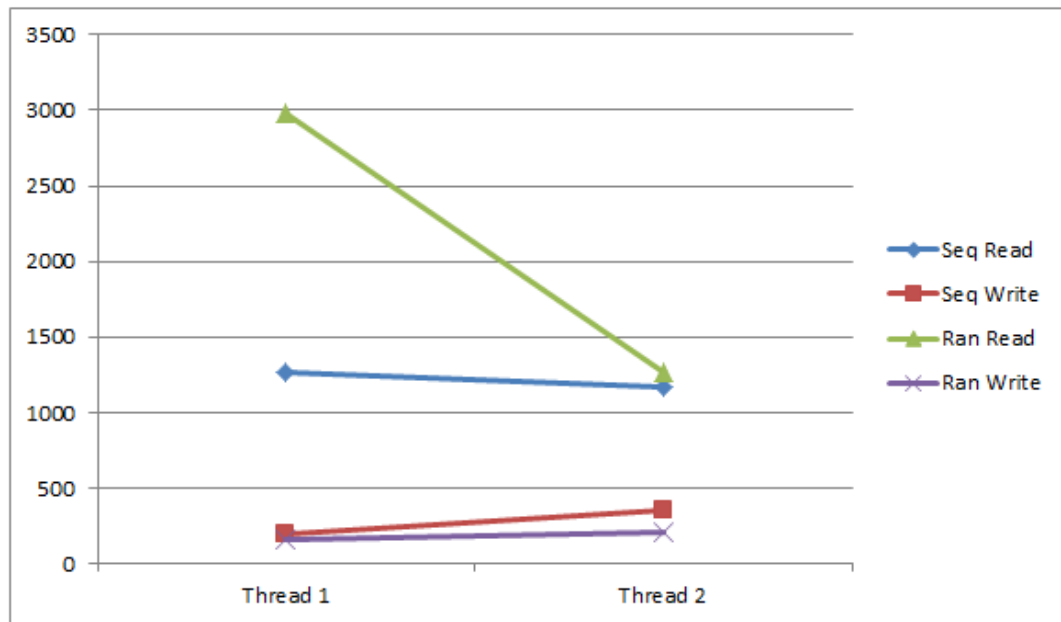
### 1Byte Sequential and Random Speed: Read and Write Latency



- **X Axis:** Number of Threads
- **Y Axis:** Latency in milliseconds

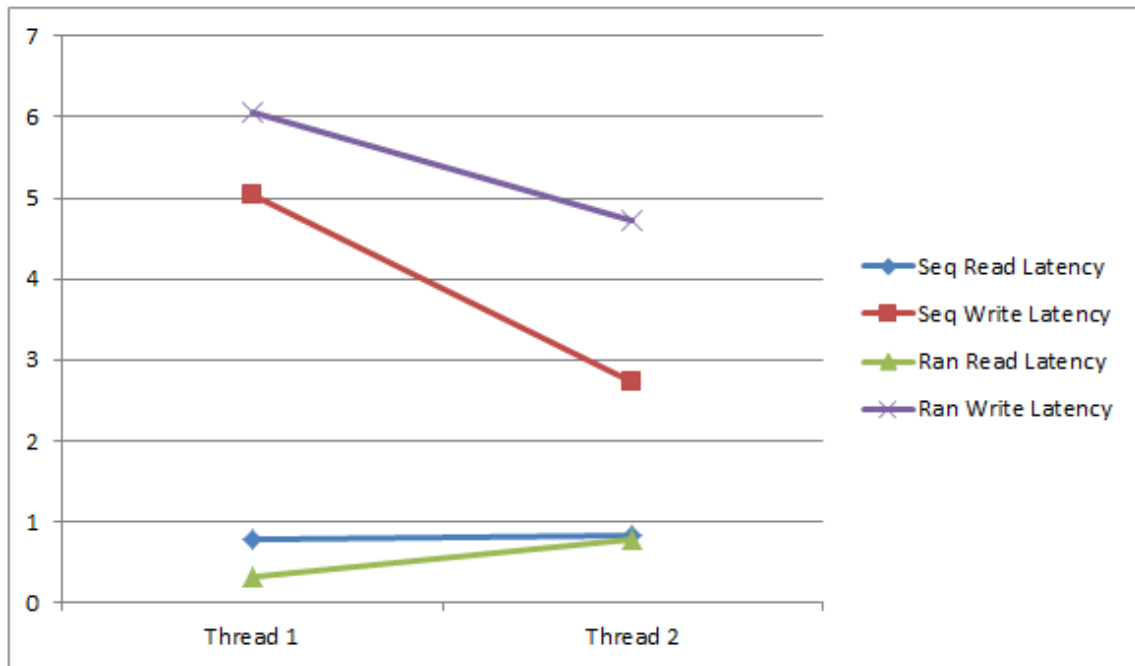
### 1 KILOBYTE:

#### 1KiloByte Sequential and Random Speed: Read and Write Throughput



- **X Axis:** Number of Threads
- **Y Axis:** Speed in MBPS

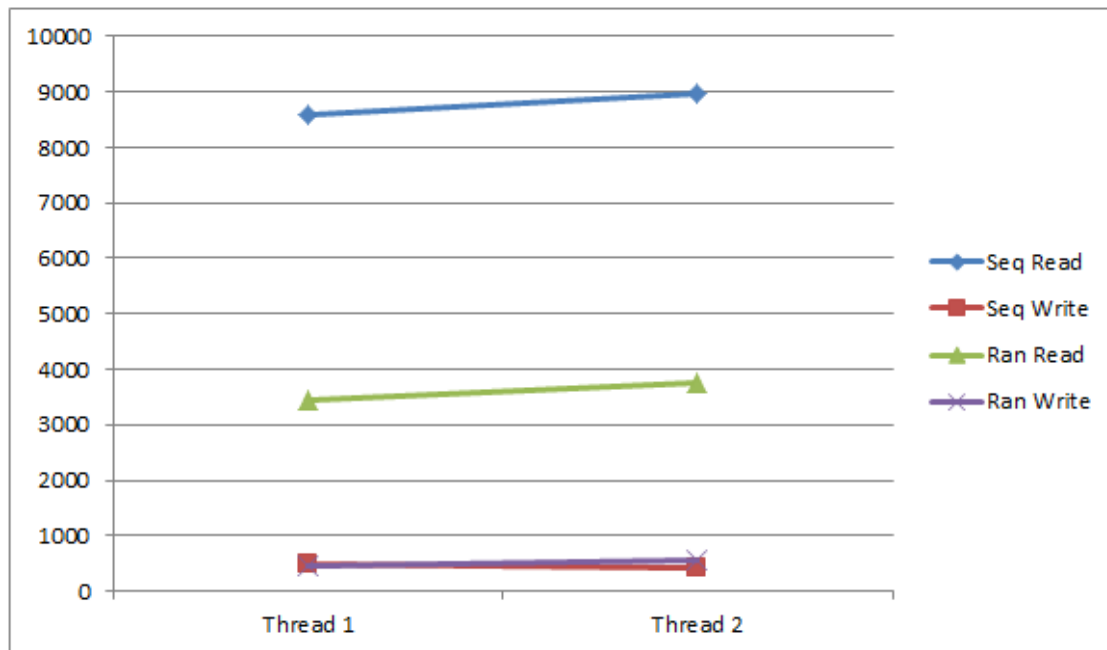
### 1KiloByte Sequential and Random Speed: Read and Write Latency



- **X Axis:** Number of Threads
- **Y Axis:** Latency in milliseconds

### 1 MEGABYTE:

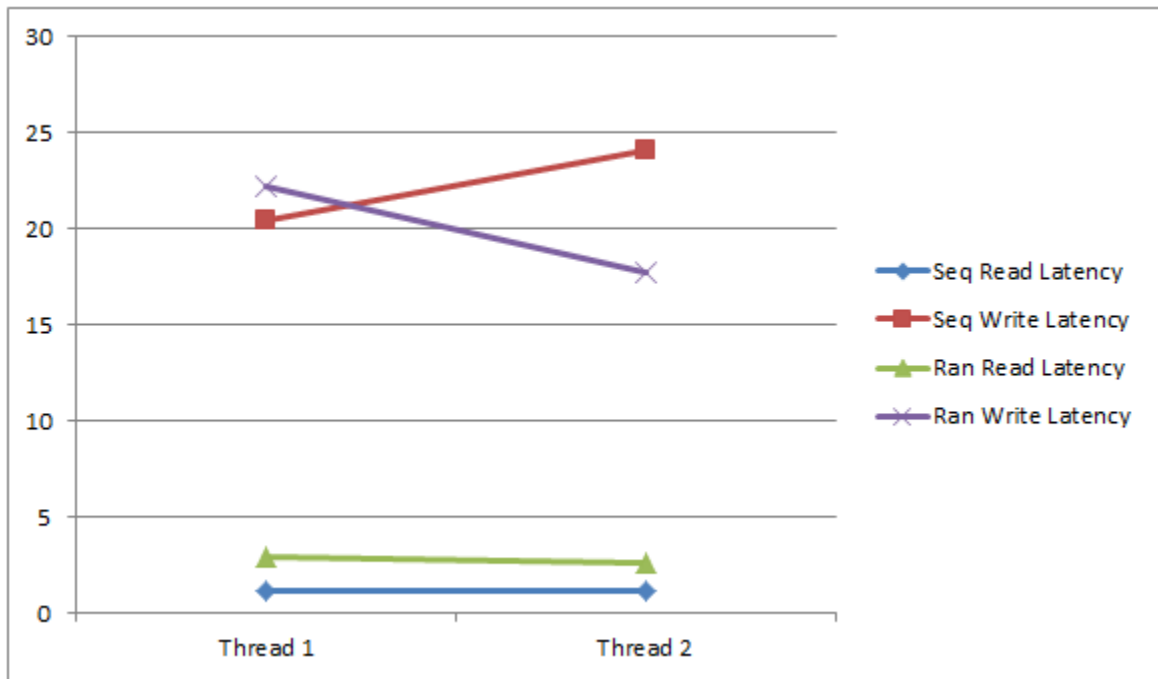
#### 1MegaByte Sequential and Random Speed: Read and Write Throughput



- **X Axis:** Number of Threads
- **Y Axis:** Speed in MBPS



### 1MegaByte Sequential and Random Speed: Read and Write Latency



- **X Axis:** Number of Threads
- **Y Axis:** Latency in milliseconds

## **Memory Benchmarking:**

### **System Configuration:**

- Operating System: Ubuntu 14.4
- RAM: 8 GB
- No of Physical cores: 4
- No of logical cores: 8

### **1 Thread:**

Random Access:

Block size	1 Byte	1 KiloByte	1 MegaByte
Speed	24.035342 MB/sec	10393.385483 MB/sec	2824.061000 MB/sec
Latency	0.0396780 ms	0.0093960 ms	0.0354100 ms

Sequential Access:

Block size	1 Byte	1 KiloByte	1 MegaByte
Speed	71.904872 MB/sec	13828.412631 MB/sec	5705.482969 MB/sec
Latency	0.01326300 ms	0.00706200 ms	0.01752700 ms

### **2 Threads:**

Random Access:

Block size	1 Byte	1 KiloByte	1 MegaByte
Speed	22.020234 MB/sec	8637.559703 MB/sec	5108.426349 MB/sec
Latency	0.0433090 ms	0.0113060 ms	0.01957550 ms

Sequential Access:

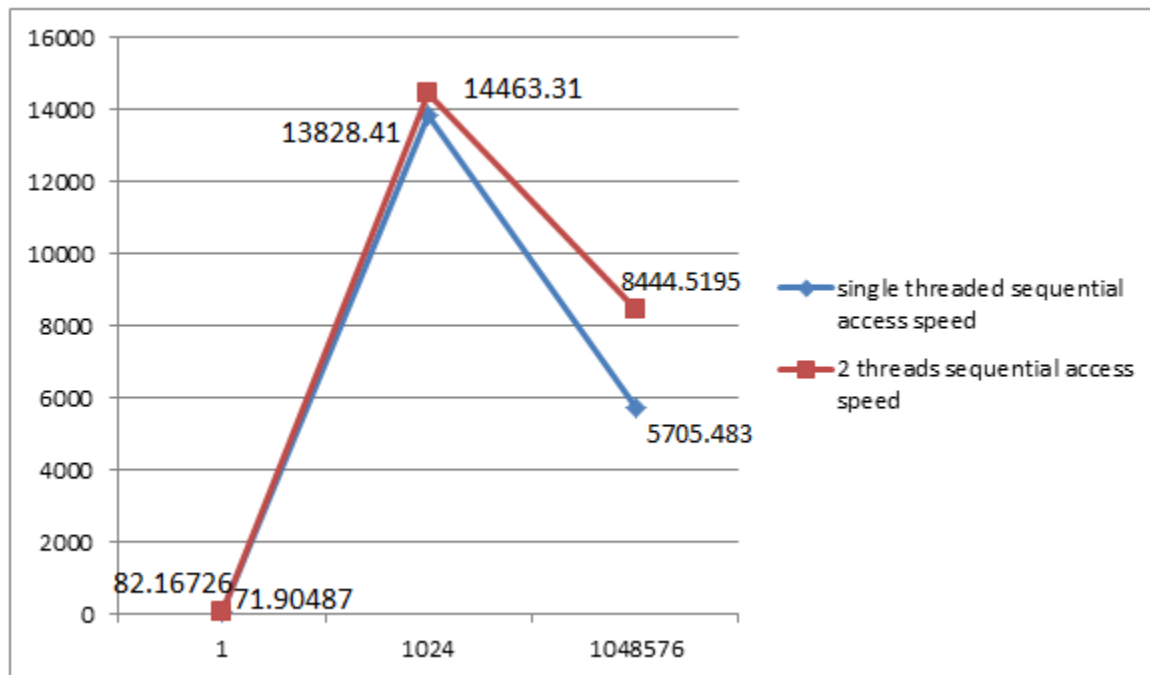
Block size	1 Byte	1 KiloByte	1 MegaByte
Speed	82.167261 MB/sec	14463.307168 MB/sec	8444.519507 MB/sec
Latency	0.011606500 ms	0.00675200 ms	0.01184200 ms

## Graph for Memory Benchmarking:

For each of 1BYTE, 1KB, 1MB we plot graphs for sequential read + write, random read + write and latency for 1 and 2 threads. The latency has both values of random and sequential in it. From the graphs we notice that the performance is best for 1 thread.

## Sequential Access Speed:

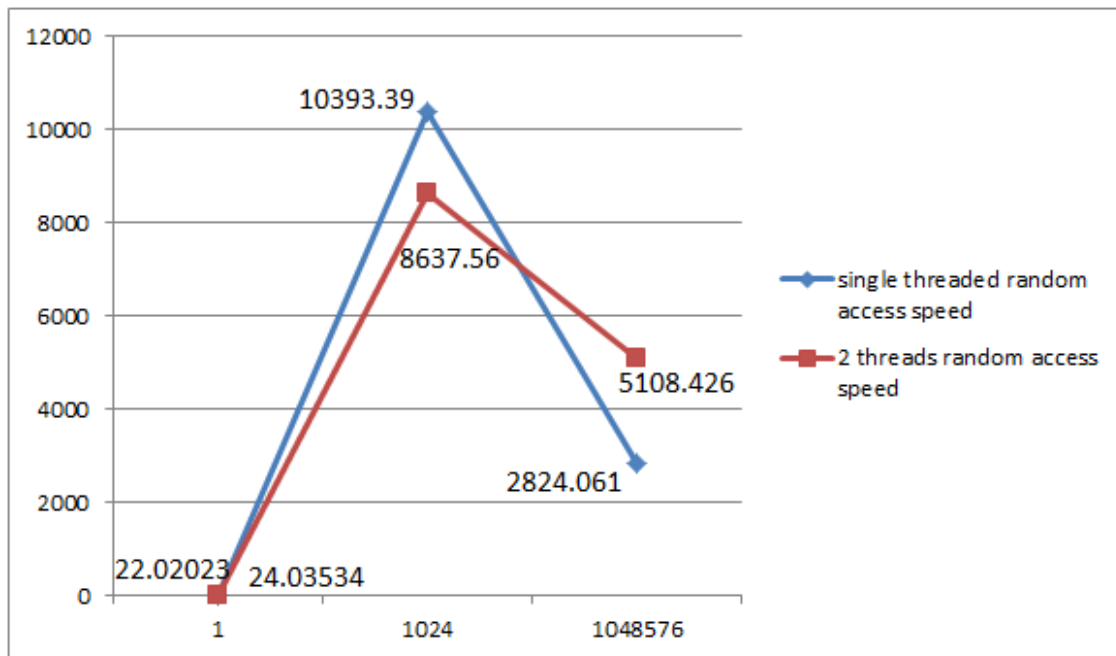
**Sequential Access Speed for Both Threads**



- **X Axis:** Size of block
- **Y Axis:** Speed in MBPS

## Random Access Speed:

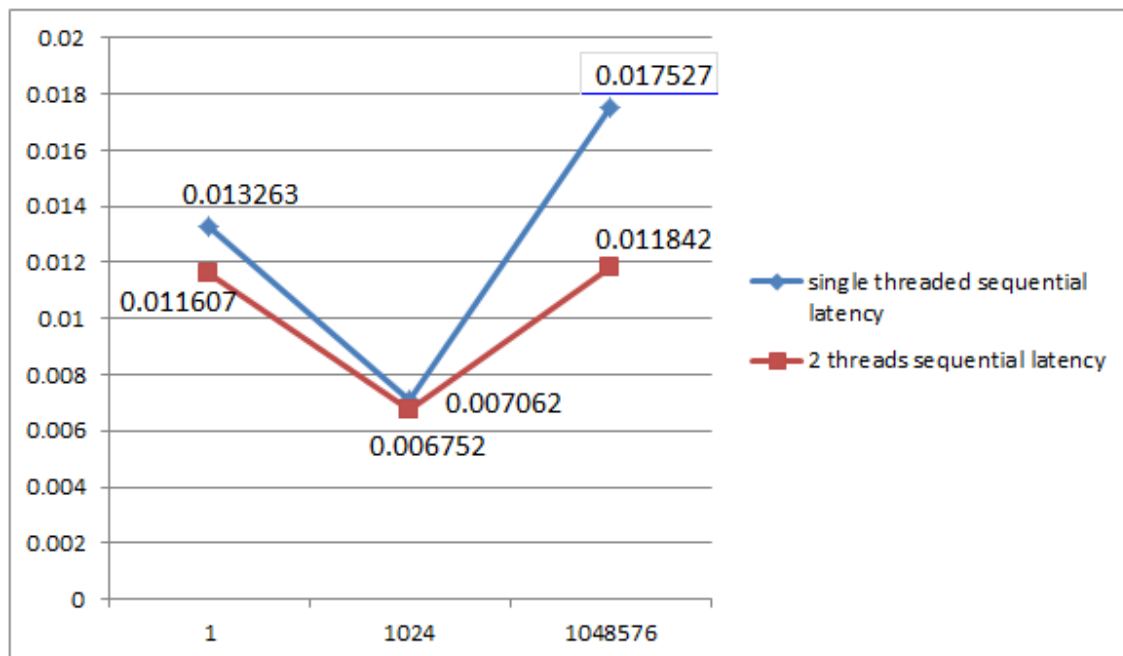
Random Access Speed for Both Threads



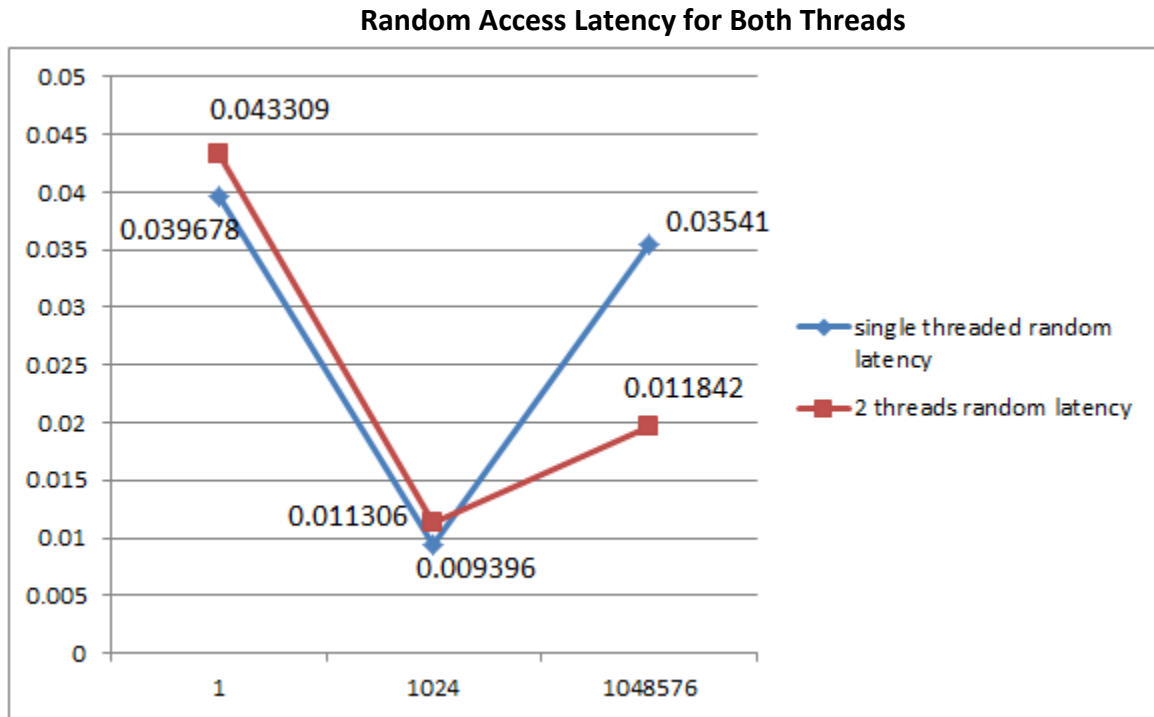
- **X Axis:** Size of block
- **Y Axis:** Speed in MBPS

## Latency graph:

Sequential Access Latency for Both Threads



- **X Axis:** Size of block
- **Y Axis:** Latency in milliseconds



- **X Axis:** Size of block
- **Y Axis:** Latency in milliseconds

Theoretical Performance: Base DRAM clock frequency \* No. of data transfer per clock \*  
Memory Bus width \* No. of Interfaces: 1600\*2\*64\*2  
Memory Bandwidth: 51.2 GB/s

### STREAM Results:

```

ec2-user@ip-172-31-50-176:~$
Permission denied (publickey).
hari@hari-550P5C-550P7C:~$ cd Documents
hari@hari-550P5C-550P7C:~/Documents$ ssh -i "Harikey.pem" ec2-user@ec2-54-164-35-14.compute-1.amazonaws.com
Last login: Fri Feb 12 02:22:51 2016 from 208-59-146-76.c3-0.mcn-ubr1.chi-mcn.il.cable.rcn.com

      _ _      _ _      _ _      _ _      _ _      _ _
     / /      / /      / /      / /      / /      / /
    /_/      /_/      /_/      /_/      /_/      /_/

Amazon Linux AMI

https://aws.amazon.com/amazon-linux-ami/2015.09-release-notes/
[ec2-user@ip-172-31-50-176 ~]$ dir
a.o               lnpack_11.1.2      stream.c
Cloud_computing \ assignment1 l_lpk_p_11.1.2.005.tgz
[ec2-user@ip-172-31-50-176 ~]$ gcc stream.c
[ec2-user@ip-172-31-50-176 ~]$ ./a.out
-----
STREAM version $Revision: 5.10 $
-----
This system uses 8 bytes per array element.
-----
Array size = 20000000 (elements), Offset = 0 (elements)
Memory per array = 152.6 MiB (= 0.1 GiB).
Total memory required = 457.8 MiB (= 0.4 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 52164 microseconds.
(= 52164 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:         5967.6      0.054407     0.053023     0.055108
Scale:        5815.8      0.055840     0.055023     0.056612
Add:          8478.4      0.057923     0.056614     0.059074
Triad:        7838.4      0.062088     0.061237     0.062976
-----
Solution Validates: avg error less than 1.000000e-13 on all three arrays
[ec2-user@ip-172-31-50-176 ~]$

```

The result of stream benchmark tool gives the memory bandwidth as 32.68 divided into 4 micro benchmarks: COPY, SCALE, SUM, and TRIAD. The efficiency achieved by STREAM benchmark tool when compared with our theoretical performance is 61.8%