

## Report

### Cloud Computing Assignment-1

#### Experiment Environment

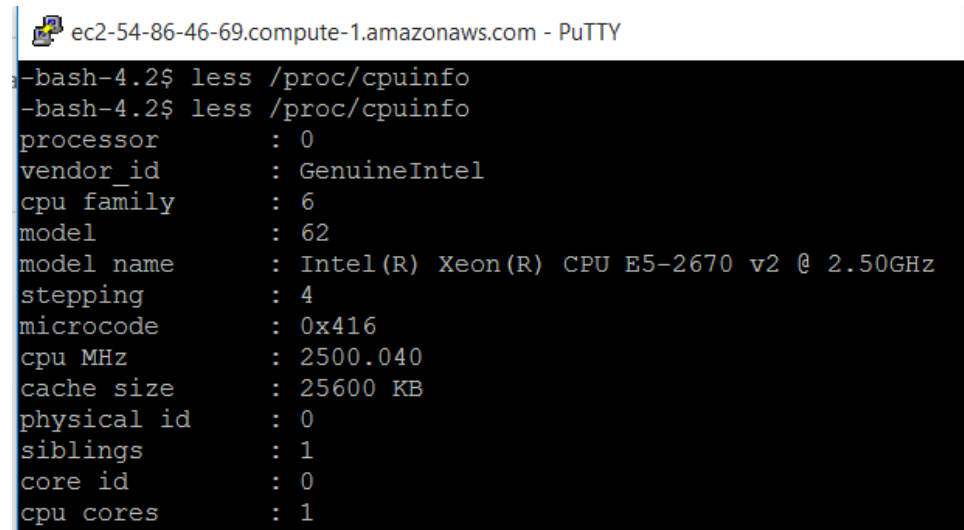
Amazon EC2 t2 micro

Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz

cpu cores : 1

cpu MHz : 2500.040

cache size : 25600 KB



```
ec2-54-86-46-69.compute-1.amazonaws.com - PuTTY
-bash-4.2$ cat /proc/cpuinfo
-bash-4.2$ cat /proc/cpuinfo
processor           : 0
vendor_id          : GenuineIntel
cpu family         : 6
model              : 62
model name         : Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz
stepping           : 4
microcode          : 0x416
cpu MHz            : 2500.040
cache size         : 25600 KB
physical id        : 0
siblings           : 1
core id            : 0
cpu cores          : 1
```

## 1) CPU Experiments

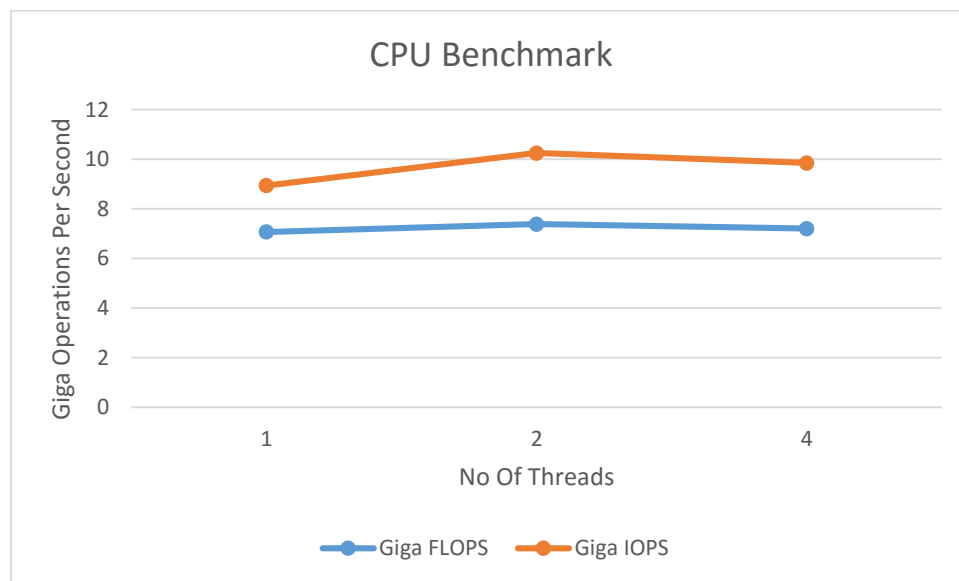
- Measuring Giga Flops and Giga Iops

In this experiment the intent was to measure Giga Flops and Giga Iops i.e. the number of floating point and Integer operations CPU can handle in a single second.

The processor speed was measured with with varying level of concurrency with 1,2 and 4 threads.

Each experiment was carried out three times and the average results are documented in the following table.

|   | Average Giga Flops | Standard Deviation Flops | Average Giga Iops | Standard Deviation Iops |
|---|--------------------|--------------------------|-------------------|-------------------------|
| 1 | 7.07               | 0.01163                  | 8.94              | 0.0652                  |
| 2 | 7.39               | 0.06121                  | 10.25             | 0.03621                 |
| 4 | 7.21               | 0.21645                  | 9.85              | 0.01939                 |



### Evaluation :

- As we can see from the above graph , we obtain more Integer operations per second than float operations per seconds as integer operations can be calculated faster by CPU.
- The pattern for operations per second almost remains constant as there is only one underlying hardware thread , which means increasing the number of threads will not make any major changes to no of operations performed in one second.

- Theoretical peak performance is measured by using following formula

$$\begin{aligned}\text{Theoretical Performance} &= \text{Ghz} * \text{IPC} * \text{Cores} \\ &= 2.5 * 16 * 1 \\ &= 40 \text{ Giga Flops}\end{aligned}$$

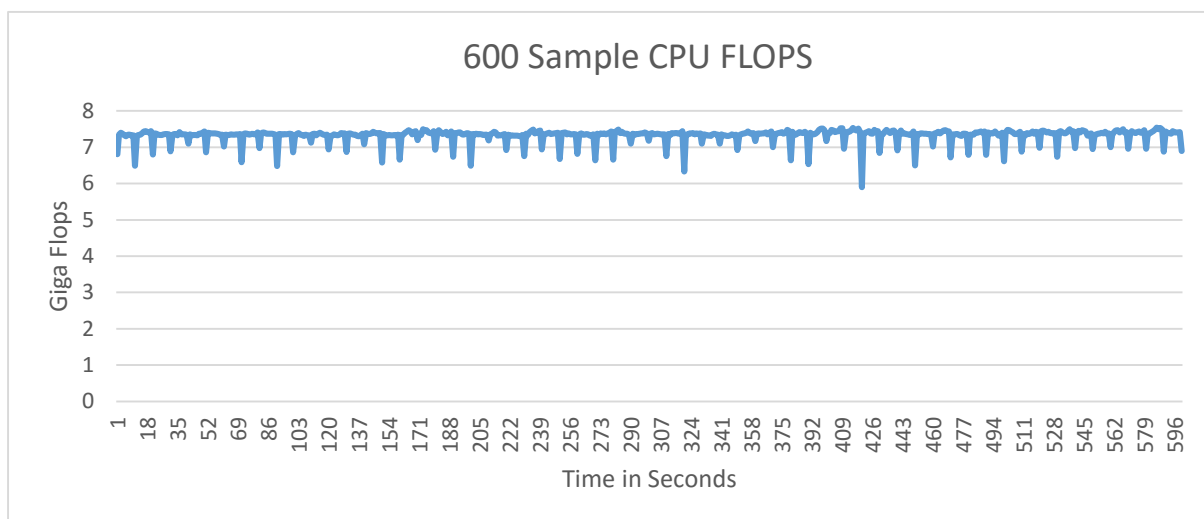
$$\begin{aligned}\text{Efficiency achieved as compared to theoretical performance} &= (7.22/40) * 100 \\ &= 18.05\%\end{aligned}$$

**Efficiency achieved = 18.05 % as compared to theoretical performance**

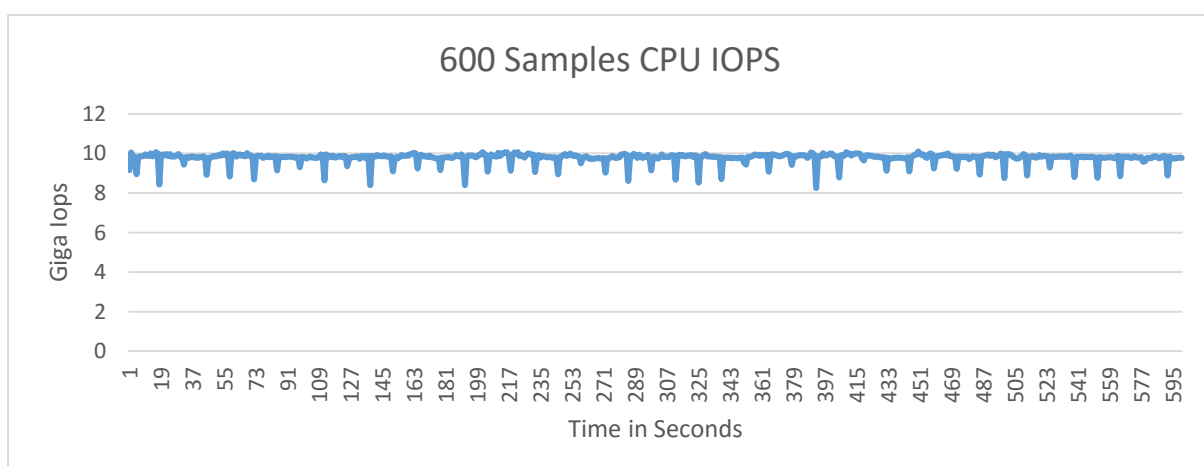
- Running the Benchmark and taking 600 Samples for FLOPS and IOPS

The CPU Benchmark was performed with 4 threads to obtain 600 samples for flops and iops and the results obtained are shown in the below graph. The 600 Samples are place in Samples\_FLOPS\_IOPS.xlsx file.

### FLOPS



### IOPS



**Evaluation :** As we see from the above graphs we obtain a constant pattern of number of floating point operations per second & Integer operations per second when carried out with 4 threads . The few negative spikes which we see in the graphs gives us the indication that some part of CPU might be running many background processes, because of which there is a slight dip in IOPS and Flops at some intervals.

- Linpack Benchmark

```
CPU frequency: 2.813 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 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=800204096, at the size=10000

----- Timing linear equation system solver -----

Size LDA Align. Time(s) GFlops Residual Residual(norm) Check
1000 1000 4 0.041 16.2250 9.900691e-13 3.376390e-02 pass
1000 1000 4 0.040 16.7715 9.900691e-13 3.376390e-02 pass
1000 1000 4 0.039 17.1183 9.900691e-13 3.376390e-02 pass
1000 1000 4 0.039 17.0314 9.900691e-13 3.376390e-02 pass
2000 2000 4 0.296 18.0411 4.053480e-12 3.526031e-02 pass
2000 2000 4 0.294 18.1604 4.053480e-12 3.526031e-02 pass
5000 5008 4 4.309 19.3503 2.336047e-11 3.257429e-02 pass
5000 5008 4 4.297 19.4045 2.336047e-11 3.257429e-02 pass
```

- The best Performance achieved using linpack Benchmark was around 19.5 Giga Flops.
- Linpack Benchmark efficiency achieved as compared to theoritical performance is

$$= (\text{Linpack Performance} / \text{Theoritical Performance}) * 100$$

$$= (19.5/40)*100$$

$$= 48.75 \%$$

## 2) Disk Experiments

- For disk experiments , we need to perform sequential and Random read write operations using varying block sizes of (1B,1KB,1MB) along with varying concurrency of 1 and 2 threads.

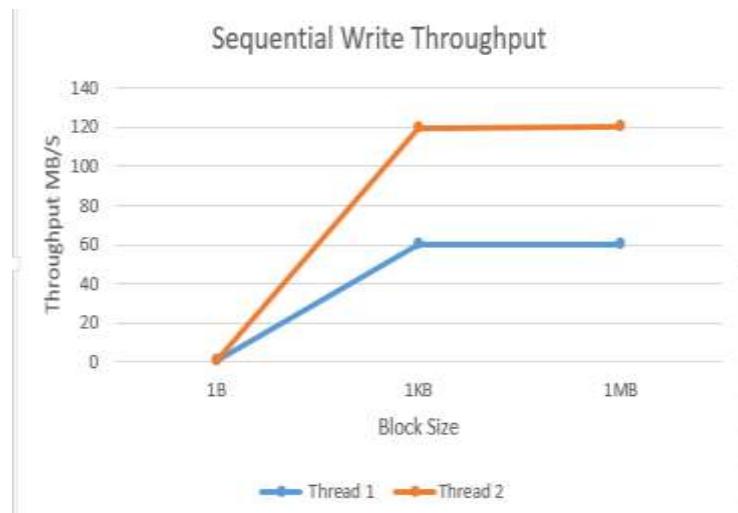
### Sequential Write

The following results are obtained when sequential write operations are performed for varying block sizes of (1B,1KB,1MB) and varying concurrency of 1 and 2 threads.

- For 1Byte Block total data written is **50 MB**.
- For 1 KB and 1MB Block data written in **5 GB** to carry out the experiments precisely.

| No of Threads | Throughput(MB/Second) |                |                |
|---------------|-----------------------|----------------|----------------|
|               | Block Size 1B         | Block Size 1KB | Block Size 1MB |
| 1             | 0.76 MB/s             | 60.19 MB/s     | 60.34 MB/s     |
| 2             | 1.00 MB/s             | 119.7 MB/s     | 120.33 MB/s    |

| No of Threads | Latency (ms)  |                |                |
|---------------|---------------|----------------|----------------|
|               | Block Size 1B | Block Size 1KB | Block Size 1MB |
| 1             | 0.00124 ms    | 0.0162 ms      | 16.5 ms        |
| 2             | 0.00094 ms    | 0.0081 ms      | 8.3 ms         |



### Evaluation :

- From the above graph for sequential write we observe that better performance for throughput was achieved using 2 threads.
- While using 1Byte buffer block the throughput achieved was very less as compared to the throughput achieved by performing the experiments using 1KB and 1MB block sizes.
- As the throughput increases the latency i.e. the time required to transfer each block size decreases.

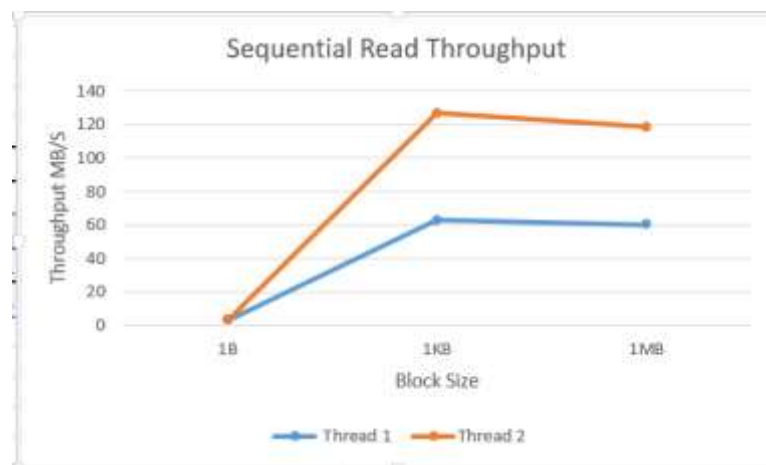
## Sequential Read

The following results are obtained when sequential read operations are performed for varying block sizes of (1B,1KB,1MB) and varying concurrency of 1 and 2 threads.

- For 1B, 1KB and 1MB block sizes total data read is **1GB text file** to carry out the experiments precisely.

| No of Threads | Throughput(MB/Second) |                   |                   |
|---------------|-----------------------|-------------------|-------------------|
|               | Block Size<br>1B      | Block Size<br>1KB | Block Size<br>1MB |
| 1             | 3.11 MB/s             | 63.00 MB/s        | 60.13/ MB/s       |
| 2             | 3.08 MB/s             | 127.05 MB/s       | 118.6 MB/s        |

| No of Threads | Latency (ms)     |                   |                   |
|---------------|------------------|-------------------|-------------------|
|               | Block Size<br>1B | Block Size<br>1KB | Block Size<br>1MB |
| 1             | 0.0003 ms        | 0.0154 ms         | 16.6 ms           |
| 2             | 0.0003 ms        | 0.0076 ms         | 8.4 ms            |



## Evaluation :

- From the above graph for sequential read we observe that better performance for throughput was achieved using 2 threads.
- While using 1Byte buffer block the throughput achieved was very less as compared to the throughput achieved by performing the experiments using 1KB and 1MB block sizes.

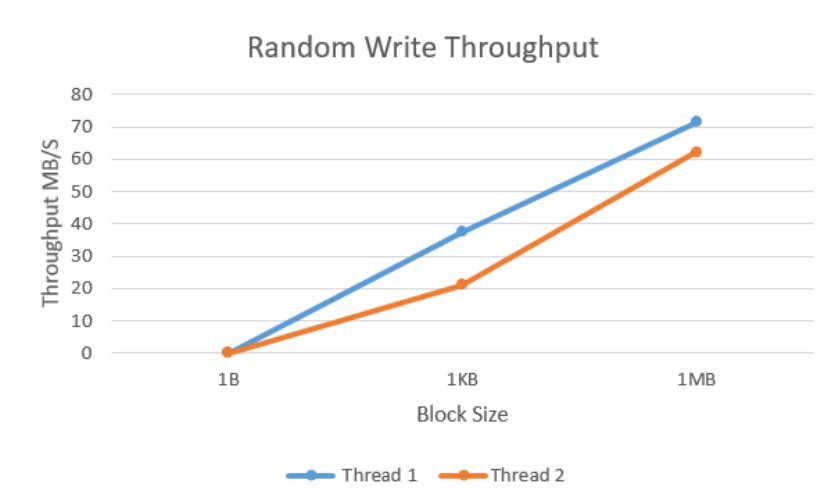
## Random Write

The following results are obtained when Random write operations are performed for varying block sizes of (1B,1KB,1MB) and varying concurrency of 1 and 2 threads.

- For 1Byte Block total data written is **10 MB**.
- For 1 KB and 1MB Block data written in **1 GB** to carry out the experiments precisely.

|               | Throughput(MB/Second) |                   |                   |
|---------------|-----------------------|-------------------|-------------------|
| No of Threads | Block Size<br>1B      | Block Size<br>1KB | Block Size<br>1MB |
| 1             | 0.0257 MB/s           | 37.56 MB/s        | 71.40 MB/s        |
| 2             | 0.0366 MB/s           | 21.22 MB/s        | 62.17 MB/s        |

|               | Latency (ms)     |                   |                   |
|---------------|------------------|-------------------|-------------------|
| No of Threads | Block Size<br>1B | Block Size<br>1KB | Block Size<br>1MB |
| 1             | 0.037 ms         | 0.026 ms          | 14 ms             |
| 2             | 0.030 ms         | 0.046ms           | 16 ms             |



## Evaluation :

- From the above graph for Random Write as the size of the buffer block increases there is constant increase in the throughput achieved.
- Moreover better performance was achieved using 2 threads as compared to 1 thread.

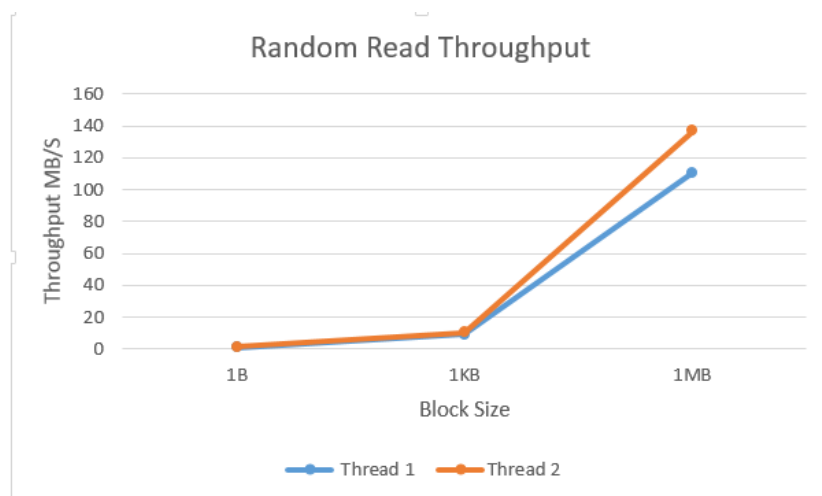
## Random Read

The following results are obtained when Random read operations are performed for varying block sizes of (1B,1KB,1MB) and varying concurrency of 1 and 2 threads.

- For 1B block sizes total data read is **50MB**.
- For 1KB and 1MB block sizes total data read is **1GB text file** to carry out the experiments precisely.

| No of Threads | Throughput(MB/Second) |                |                |
|---------------|-----------------------|----------------|----------------|
|               | Block Size 1B         | Block Size 1KB | Block Size 1MB |
| 1             | 1 MB/s                | 9.07 MB/s      | 110.75 MB/s    |
| 2             | 1.34 MB/s             | 10.32 MB/s     | 136.89 MB/s    |

| No of Threads | Latency (ms)  |                |                |
|---------------|---------------|----------------|----------------|
|               | Block Size 1B | Block Size 1KB | Block Size 1MB |
| 1             | 0.0009 ms     | 0.026 ms       | 9.0 ms         |
| 2             | 0.0008 ms     | 0.046ms        | 7.3 ms         |



## Evaluation :

- While performing random read experiments with 1B and 1KB blocks we were not able to achieve high throughputs.
- While running the same experiment using 1MB block sizes higher values of throughput was achieved.
- The performance achieved using 1 thread and 2 threads was almost same.



- IOZONE Benchmark

```
ec2-54-86-46-69.compute-1.amazonaws.com - PuTTY
-bash-4.2$ ./iozone
Usage: For usage information type iozone -h

-bash-4.2$ ./iozone -a
iozone: Performance Test of File I/O
Version $Revision: 3.434 $
Compiled for 64 bit mode.
Build: linux

Contributors:William Morcott, Don Capps, Isaac Crawford, Kirby Collins
Al Slater, Scott Rhine, Mike Warner, Ken Ross,
Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CHU,
Bandy Dunlap, Mark Montague, Dan Million, Gavin Brehmer,
Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
Erik Habbitz, Kris Strecker, Walter Wong, Joshua Root,
Fabrice Bacchella, Zhonghua Xue, Qin Li, Darren Sawyer,
Vangel Bojakhli, Ben England, Vikentsi Lapa,
Alexey Skidanov.

Run began: Thu Feb 11 23:46:03 2016

Auto Mode
Command line used: ./iozone -a
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 random random bkwrd record stride
read rewrite read write read write read write read write
64 4 1103556 2067979 5735102 15372805 10402178 2561267 4264768 4511132 8026779 2298136 2278618 48
64 8 1133775 1539680 7190197 20462193 15872895 2681589 3389453 4897948 19462178 2361287 2487186 64
64 16 1305975 2279428 3363812 15972805 32802017 2378636 3022727 4818132 3363611 3057133 2892445 84
64 32 1302258 1484662 7781156 20662193 15972805 1937893 6421025 3702156 8318831 1847921 3732356 32
64 64 1598880 2581267 15972805 20662193 15872805 1528986 6421025 1679781 2823882 2867186 2581267 79
128 4 1103557 2279427 6486138 14201794 20779207 1501110 1189160 4396573 8026779 1967285 1105484 84
```

- I have run the iozone benchmark to test the Disk Speed.

```
ec2-54-86-46-69.compute-1.amazonaws.com - PuTTY
-bash-4.2$ ./iozone -a -i 0 -i 1 -i 2 -s 2097152 -r 1024
iozone: Performance Test of File I/O
Version $Revision: 3.434 $
Compiled for 64 bit mode.
Build: linux

Contributors:William Morcott, Don Capps, Isaac Crawford, Kirby Collins
Al Slater, Scott Rhine, Mike Warner, Ken Ross,
Steve Landherr, Brad Smith, Mark Kelly, Dr. Alain CHU,
Bandy Dunlap, Mark Montague, Dan Million, Gavin Brehmer,
Jean-Marc Zucconi, Jeff Blomberg, Benny Halevy, Dave Boone,
Erik Habbitz, Kris Strecker, Walter Wong, Joshua Root,
Fabrice Bacchella, Zhonghua Xue, Qin Li, Darren Sawyer,
Vangel Bojakhli, Ben England, Vikentsi Lapa,
Alexey Skidanov.

Run began: Fri Feb 12 00:38:08 2016

Auto Mode
File size set to 2097152 kB
Record Size 1024 kB
Command line used: ./iozone -a -i 0 -i 1 -i 2 -s 2097152 -r 1024
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 random random bkwrd record stride
read rewrite read write read write read write read write
2097152 1024 62618 63367 43194 57759 35930 66243

iozone test complete.
-bash-4.2$
-bash-4.2$
-bash-4.2$
-bash-4.2$
```

- Results obtained on iozone to test a file of 2GB using block size of 1MB.

| Sequential Write<br>MB/s | Sequential Read<br>MB/s | Random write<br>MB/s | Randon Read<br>MB/s |
|--------------------------|-------------------------|----------------------|---------------------|
| 63 MB/s                  | 42.18 MB/s              | 64.69 MB/s           | 54.6 MB/s           |

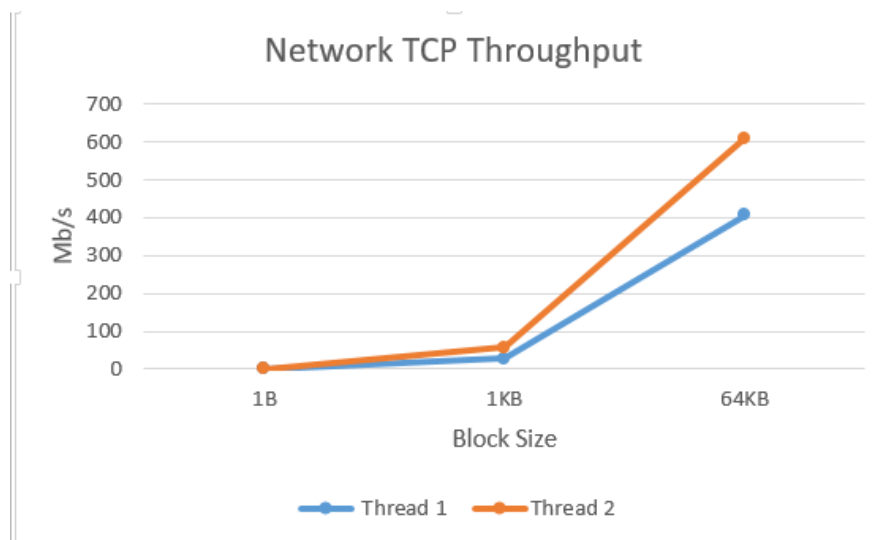
### 3) Network Experiments

For Network experiments , we need to measure the network speed between two instances using varying buffer size of 1B, 1KB and 64 KB with varying level of concurrency using 1 and 2 threads.

#### TCP

| No of Threads | Throughput (Mb/Second) |                   |                    |
|---------------|------------------------|-------------------|--------------------|
|               | Block Size<br>1B       | Block Size<br>1KB | Block Size<br>64KB |
| 1             | 0.01 Mb/s              | 28.67 Mb/s        | 407.77 Mb/s        |
| 2             | 0.017 Mb/s             | 57 Mb/s           | 611 Mb/s           |

| No of Threads | Latency (ms)     |                   |                    |
|---------------|------------------|-------------------|--------------------|
|               | Block Size<br>1B | Block Size<br>1KB | Block Size<br>64KB |
| 1             | 0.008 ms         | 0.04 ms           | 2.4 ms             |
| 2             | 0.045 ms         | 0.031 ms          | 1.9 ms             |



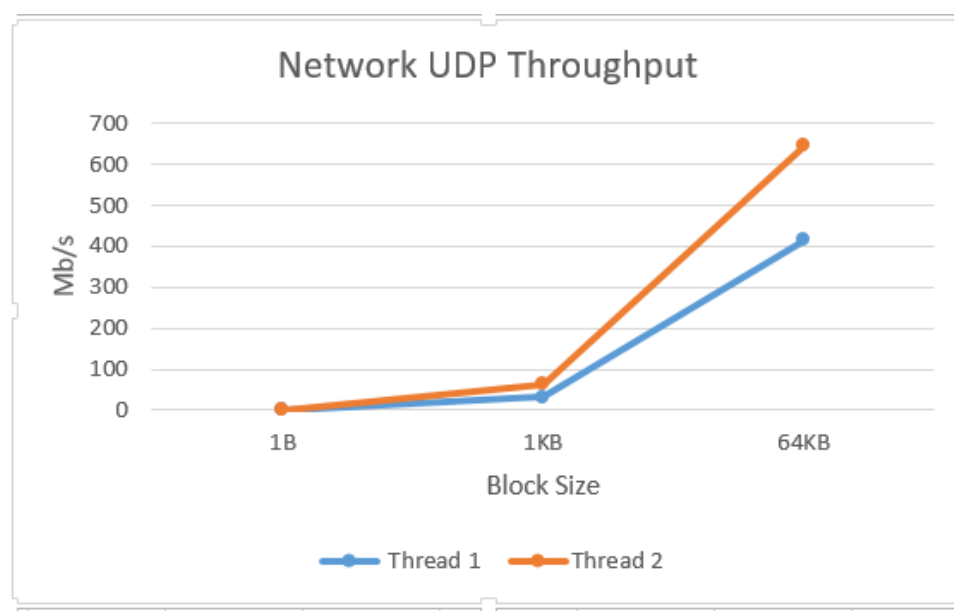
#### Evaluation :

- From the above graph we observe that as the Block Size increases there is a steep growth in throughput achieved.
- For the block sizes of 1B and 1KB the throughput achieved was low as compared to the throughput achieved with block sizes of 64KB.
- The performance achieved with 2 threads was better than the performance achieved with single thread.

## UDP

|               | Throughput (Mb/Second) |                   |                    |
|---------------|------------------------|-------------------|--------------------|
| No of Threads | Block Size<br>1B       | Block Size<br>1KB | Block Size<br>64KB |
| 1             | 0.02 Mb/s              | 31.84 Mb/s        | 414.91 Mb/s        |
| 2             | 0.03 Mb/s              | 63.67 Mb/s        | 645.37 Mb/s        |

|               | Latency (ms)     |                   |                    |
|---------------|------------------|-------------------|--------------------|
| No of Threads | Block Size<br>1B | Block Size<br>1KB | Block Size<br>64KB |
| 1             | 0.006 ms         | 0.03 ms           | 2.1 ms             |
| 2             | 0.003 ms         | 0.025 ms          | 1.8 ms             |



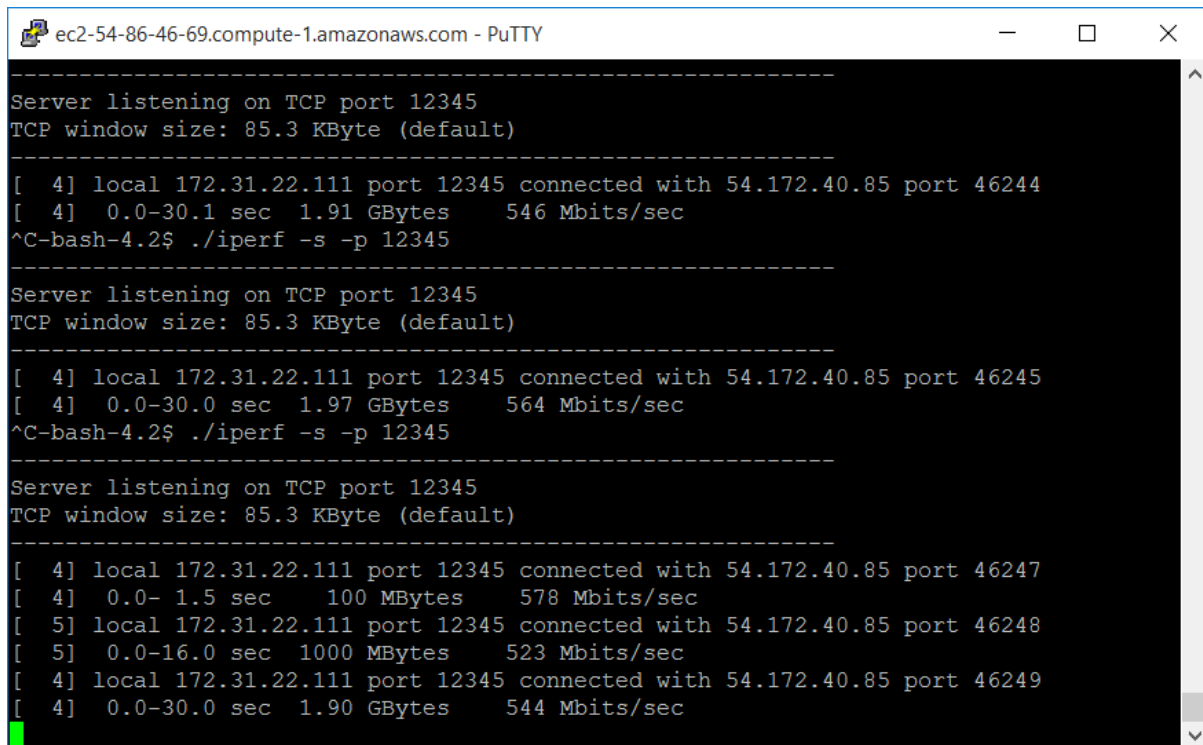
### Evaluation :

- The result obtained after running the experiments using UDP protocol was slightly faster than TCP Protocol.
- As in TCP , in UDP too there was a steep growth in throughput achieved as the buffer block size increases.

- IPERF Benchmark

- Execute command  
**./iperf -s -p 12345**

This command starts the server instance with IP 12345



The screenshot shows a PuTTY terminal window titled "ec2-54-86-46-69.compute-1.amazonaws.com - PuTTY". The terminal displays the output of the command `./iperf -s -p 12345` executed three times. Each execution shows the server listening on TCP port 12345 with a window size of 85.3 KByte. The first two runs show a single connection from 54.172.40.85 port 46244, with results of 1.91 GBytes at 546 Mbits/sec and 1.97 GBytes at 564 Mbits/sec respectively. The third run shows multiple connections: a 100 MByte test at 578 Mbits/sec, a 1000 MByte test at 523 Mbits/sec, and a final 1.90 GByte test at 544 Mbits/sec. The terminal has a green cursor at the bottom left.

```
-----  
Server listening on TCP port 12345  
TCP window size: 85.3 KByte (default)  
-----  
[ 4] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46244  
[ 4] 0.0-30.1 sec 1.91 GBytes 546 Mbits/sec  
^C-bash-4.2$ ./iperf -s -p 12345  
-----  
Server listening on TCP port 12345  
TCP window size: 85.3 KByte (default)  
-----  
[ 4] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46245  
[ 4] 0.0-30.0 sec 1.97 GBytes 564 Mbits/sec  
^C-bash-4.2$ ./iperf -s -p 12345  
-----  
Server listening on TCP port 12345  
TCP window size: 85.3 KByte (default)  
-----  
[ 4] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46247  
[ 4] 0.0- 1.5 sec 100 MBytes 578 Mbits/sec  
[ 5] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46248  
[ 5] 0.0-16.0 sec 1000 MBytes 523 Mbits/sec  
[ 4] local 172.31.22.111 port 12345 connected with 54.172.40.85 port 46249  
[ 4] 0.0-30.0 sec 1.90 GBytes 544 Mbits/sec
```

## References

- 1) <http://www.binarytides.com/udp-socket-programming-in-java/>
- 2) <http://tutorials.jenkov.com/java-nio/index.html>
- 3) <https://www.cs.uic.edu/~troy/spring05/cs450/sockets/socket.html>
- 4) [http://www.iozone.org/docs/IOzone\\_msword\\_98.pdf](http://www.iozone.org/docs/IOzone_msword_98.pdf)
- 5) [https://www.veritas.com/support/en\\_US/article.HOWTO64302](https://www.veritas.com/support/en_US/article.HOWTO64302)