**CS553 Programming Assignment #1**

**Benchmarking**

**Performance Evaluation**

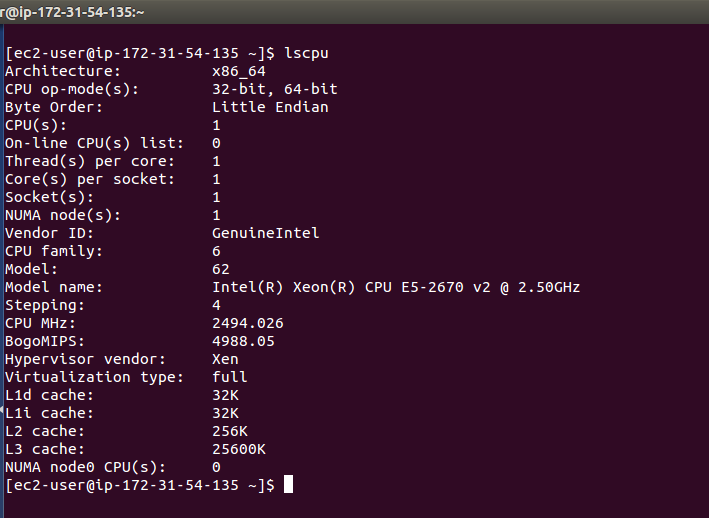
**Harsh C. Parikh**

**A20338453**

This document contains all experimental results of all the benchmarks. Each experiment is performed 3 times and average and standard deviation are calculated based on that.

**1. Introduction**

For all the benchmarks, I experimented on Amazon EC2, with 1 core CPU, Intel(R) Xeon(R) CPU E5-2670 v2 @ 2.50GHz. All Programs are done in Python language.



For CPU benchmark, I measured processor's speed in terms of GFLOPS and GIOPS at varying concurrency level for 1,2 and 4 threads. Moreover, I took the 600 samples for IOPS and FLOPS at the interval of 1 sec till 10 minutes.

For Disk benchmark, I measured the throughputs of Sequential Read, Sequential Write, Random Read and Random Write and latency for all of them and varied the block sizes (1B/1KB/1MB) for the number of threads (1 and 2).

For Network benchmark, I measured throughput and latency for both TCP and UDP protocols and varied the packet size (1B/1KB/64KB) and the number of threads (1 and 2)

**2. Experiment Results and Analysis**

In this section, all experiment results for each benchmark and explanations for the trede offs in the results.

**2.1 CPU Benchmarking**

* For CPU benchmark, I am finding GIOPS and GFLOPS value for different number of threads.
* In graph, I had done multiple experiments for 1,2 and 4 threads and according to that I am getting results for GFLOPS and GIOPS as follows.
* Theoretical peak performance of the processor is given by:

= (CPU speed in GHz) x (number of CPU cores) x (CPU instruction per cycle) x (number of CPUs per node)

=2.50 \* 1 \*8 \* 2

**= 40 Gflops**

and efficiency would be = (20.88/40)\*100 = **52.20 %.**

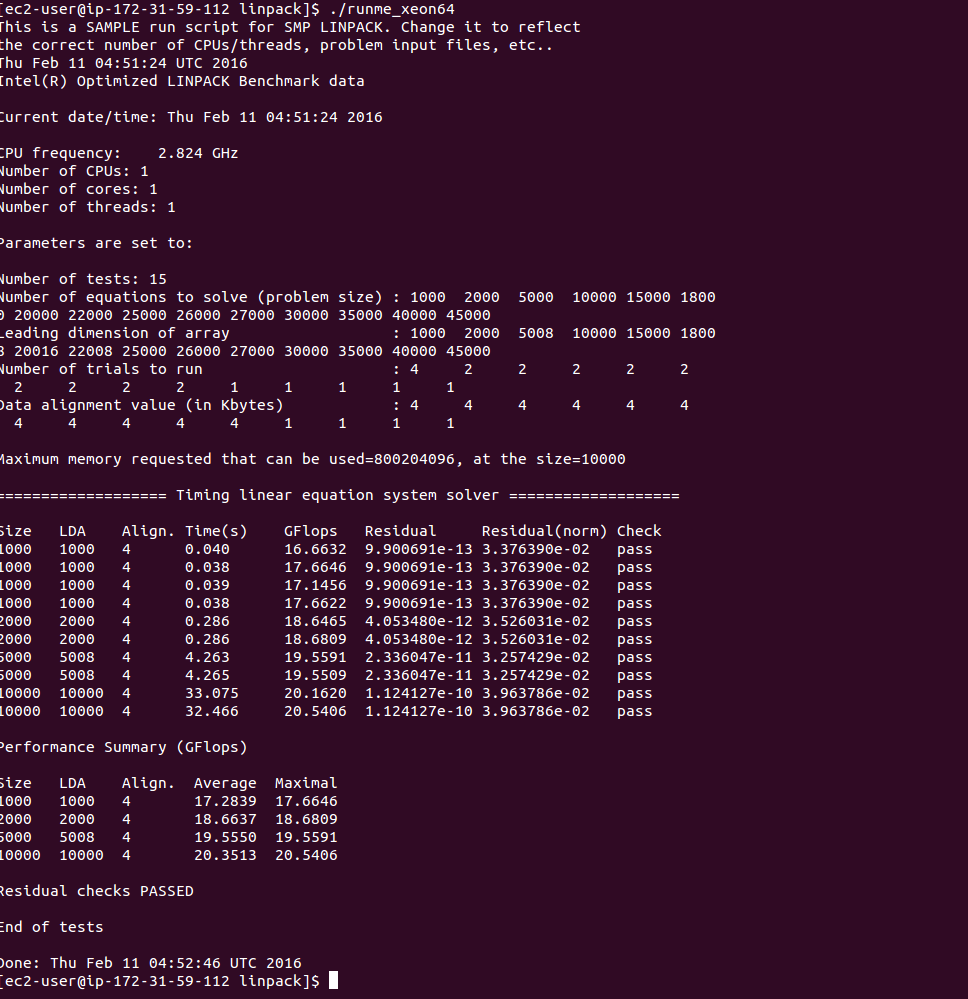
**Average and Standard Deviation for GFLOPS and IOPS are in following tables.**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of thread** | **Average (GIOPS)** | **Standard Deviation(GIOPS)** | **Average(GFLOPS)** | **Standard Deviation(GFLOPS)** |
| 1 | 24.2798 | 0.3541 | 20.12567 | 0.3011 |
| 2 | 24.4782 | 0.3631 | 20.8832 | 0.3166 |
| 4 | 24.9537 | 0.3777 | 21.43637 | 0.3199 |

Now, I have taken **600 samples of IOPS and FLOPS**. Here are the snap-shots of results. These results performed on **4 threads**.

Here, in the graph, I observe there are some glitches. That's because of operations were performed less in that time interval due to other process might use the core. So sometimes due to that we got these types of glitches as if less numbers of operations performed in that period.

Now, I had run the **Lin-pack benchmark and result is being shown in below screen..**



From practical performance, linpack gives me around **20.35 Gflops.**

**Efficiency = (20.53/40)\*100 = 51.32 %**

**2.2 Disk Benchmark**



From practical Throughput for reading 1MB or 1024 KB, I'm getting around = 4.977 GB/sec = 4977615 B/sec and as per theoretical Throughput, it would be around 6 GB/sec (as per wikipedia, our EC2 instance version is 3.0) . So efficiency would be around **82.83%.** Moreover on IOzone, throughput is around 6.40 GB/sec.

* And latency is = 0.747 ms .
* Here I have calculated the throughput of single thread and multi threaded and block size for sequential and random read/write. And their results are shown.
* Here, I'm getting less throughput for 2 threads compare to 1 thread because when in 2 threads, after completion of one thread, other can access the file. So 2 threads can't access a file at the same time. (Read it somewhere from Amazon Notes).
* The graph shows the trade offs between sequential and random read/write throughput according to block size. It's going upwards as we increase the block sizes And if we increase number of threads then throughput is sometimes decreasing because if one can't write in a file simultaneously. So that
* Sequential throughput is faster than random throughput for both read and write operations.
* Read throughput performs better in both sequential and random cases.
* Another Graph shows the trade offs between number of threads and throughput in sequential read, sequential write, random read and random write with respect to latency in milliseconds.
* Sequential operation works faster than random throughput for both read and write.
* In other graphs, I have used 1 KB block to measure trade-off between latency and thread. So, one can understand from graph that if multi-threaded operation takes lesser time for both of them.

**Throughput result of Sequential Read/Write Operations (results in MB/sec):**



**Random Read/Write Operations (results in MB/sec):**



**Average and Standard Deviation for 1 Byte are in following tables(results in MB/sec & latency in ms).**

**For Sequential Read (1 B):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Thread** | **Average(Throughput-Sequential Read)** | **Standard Deviation(Throughput-Sequential Read)** | **Average(Latency-Sequential Read)** | **Standard Deviation((Latency-Sequential Read)** |
| 1 | 24.138 | 4.22 | 0.0043 | 0.0036 |
| 2 | 22.339 | 3.88 | 0.0042 | 0.0027 |

**For Random read (1 B):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Thread** | **Average(Throughput-Random Read)** | **Standard Deviation(Throughput-Random Read)** | **Average(Latency-Random Read)** | **Standard Deviation((Latency-Random Read)** |
| 1 | 18.45 | 6.99 | 0.0163 | 0.006 |
| 2 | 20.562 | 5.29 | 0.0071 | 0.004 |

**For Sequential write (1 B):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Thread** | **Average(Throughput-Sequential Write)** | **Standard Deviation(Throughput-Sequential Write)** | **Average(Latency-Sequential write)** | **Standard Deviation((Latency-Sequential Write)** |
| 1 | 15.442 | 2.88 | 0.081 | 0.007 |
| 2 | 18.996 | 4.22 | 0.096 | 0.006 |

**For Random write (1 B):**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of Thread** | **Average(Throughput-Random write)** | **Standard Deviation(Throughput-Random write)** | **Average(Latency-Random write)** | **Standard Deviation((Latency-Random write)** |
| 1 | 14.998 | 7.12 | 0.0101 | 0.006 |
| 2 | 15.545 | 3.65 | 0.0123 | 0.008 |

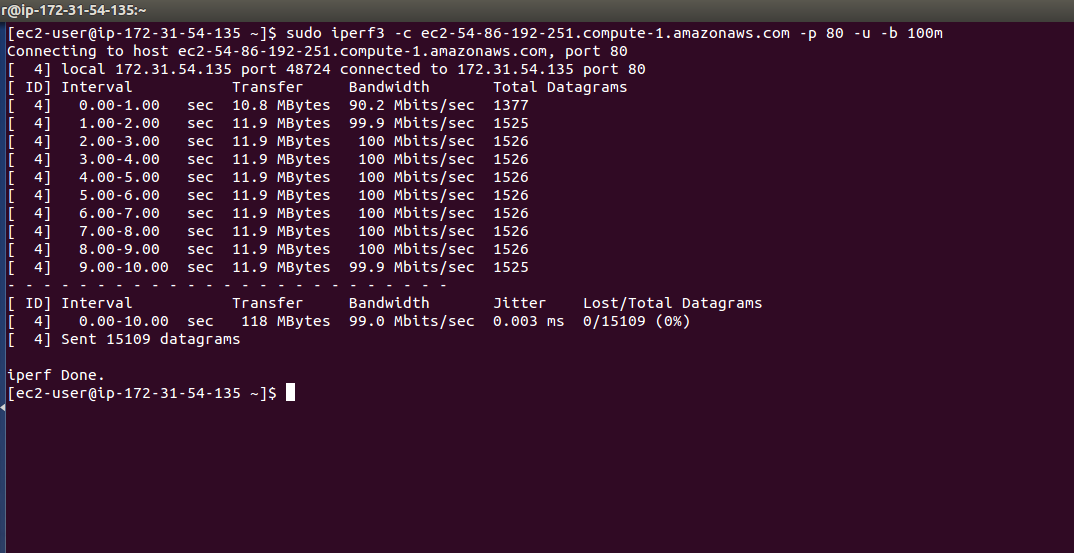
****

**2.3 Network Benchmark**

* The following graph shows the trade offs between Latency and throughput according to block size. It's going upwards as we increase the block sizes in both TCP and UDP .
* In Network, I observer that TCP throughput is lesser than UDP because UDP is a connectionless protocol and so in UDP, there is no guarantee of receiving bytes.
* On the other side, TCP sends an acknowledgement from server. So for that reason, in TCP, we are getting higher latency compare to UDP.
* The graph shows the trade offs between Latency and throughput according to block size. It's going upwards as we increase the block sizes in both TCP and UDP .
* In Network, I observer that TCP throughput is lesser than UDP because UDP is a connectionless protocol and so in UDP, there is no guarantee of receiving packages.
* On the other side, TCP sends an acknowledgement from server. So for that reason, in TCP, we are getting higher latency compare to UDP.

**These are two snap-shots of UDP and TCP Iperf performance.**

**UDP Graph: -**



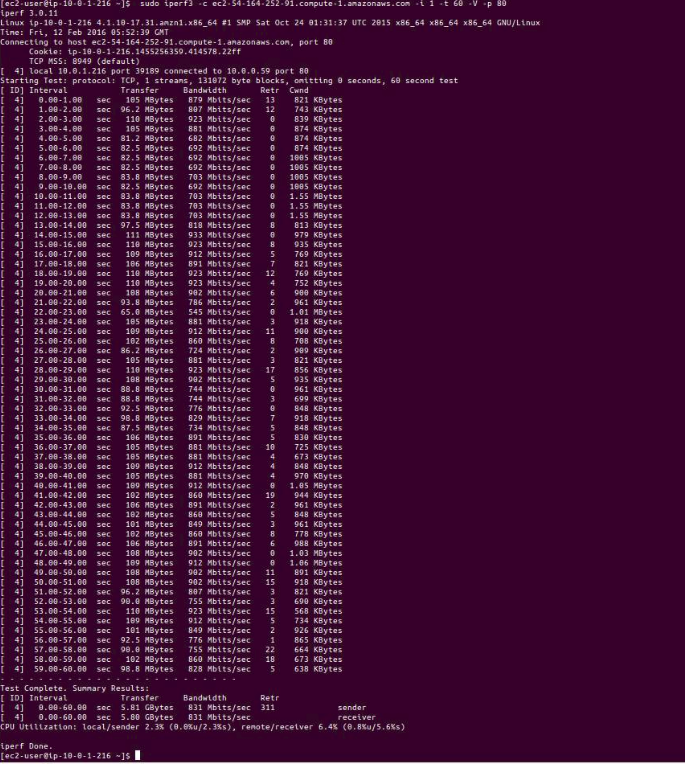
Here, in this Iperf, they have given Bandwidth and packets those are being transferred from transmitter to receiver side. In UDP, we can't send more than 65536 bytes package at the same time.

Moreover, in UDP, we need to send packets one by one and in TCP, we can send number of bytes together. So in screen-shot, we get the idea about that.

Efficiency comparison between Iperf and My result.

**Efficiency = (759/979)\*100 = 77.52%**

**TCP Graph:**



**Throughput (1 Thread) in MB/sec:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of blocks** | **Average(Throughput-TCP)** | **Standard Deviation(Throughput-TCP)** | **Average(Throughput-UDP)** | **Standard Deviation(Throughput-UDP)** |
| 1 B | 1.023 | 0.324 | 2.085 | 1.1445 |
| 1 KB | 328.614 | 27.76 | 452.225 | 29.897 |
| 64 KB | 759.314 | 85.14 | 1310.124 | 110.255 |

**Latency (1 Thread) in ms:**

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **Number of blocks** | **Average(Latency-TCP)** | **Standard Deviation(Latency-TCP)** | **Average(Latency-UDP)** | **Standard Deviation(Latency-UDP)** |
| 1 B | 0.147 | 0.005 | 0.077 | 0.0025 |
| 1 KB | 0.506 | 0.002 | 0.498 | 0.05 |
| 64 KB | 12.372 | 1.992 | 8.0079 | 1.56 |

**Throughput result of TCP and UDP (results in MB):**

****

****