**PERFORMANCE EVALUATION**

**DESIGNED BY:**

SAMI AHMAD KHAN

A20352677

COURSE: CS 553 CLOUD COMPUTING

|  |
| --- |
|  |
| BENCHMARKING TOOL |
|  |
| PA 1 |

# 1. INTRODUCTION

The purpose of this assignment was to benchmark different computer components like CPU, Memory, Disk and Network. Benchmarking is done on them by calculating their operations per second by causing load on them and hence evaluating the performance for the same.

This document has experimental values of all the benchmarks. First I have described the environment, then the specifications of the experiment for each benchmark is presented and at last all the results will be shown and analyzed.

# 2. EXPERIMENTAL ENVIRONMENT

For all the benchmarks, I have performed all the experiments on Amazon **AWS.**

**Amazon AWS Information:**

It is a collection of remote computing services, also called web services, that make up a cloud-computing platform offered by Amazon.com. These services operate from 11 geographical regions[2] across the world. The most central and well-known of these services arguably include Amazon Elastic Compute Cloud, also known as "EC2", and Amazon Simple Storage Service, also known as "S3".

Amazon Linux **AMI 2015.09.1 (HVM)**

SSD Volume Type **- ami-f0091d9**

Instance ID: **i-8851de4c**

Instance Type**: t2.micro**

# 3. OVERALL EXPERIMENTS

**1. CPU Benchmarking:**

Measuring the processor speed, in terms of floating point operations per second (Giga FLOPS, 109 FLOPS) and integer operations per second (Giga IOPS, 109 IOPS) at varying concurrency levels (1 and 2)

**2. Memory Benchmarking :**

* Measuring the memory speed of the host  by using read and write operations (e.g. memcpy), sequential access, random access, varying block sizes (1B, 1KB, 1MB), and varying the concurrency (1 thread & 2 threads)  .
* The metrics measured are throughput (Megabytes per second, MB/sec) and latency (milliseconds, ms)

**3. Network Benchmarking:**

Measuring the network speed between 2 instances . The parameter space includes the TCP protocol stack, UDP, varying packet/buffer size (1B, 1KB, 64KB), and varying the concurrency (1 thread & 2 threads).

# 4. EXPERIMENT RESULTS AND ANALYSIS

This section consists of experimental results for each benchmark and explanations for the same.

## 4.1 CPU Benchmark Results

For finding CPU benchmark results, I found out GFLOPS and GIOPS value for different number of threads, i.e. 1, 2 and 4 threads.

a). GFLOPS

|  |  |  |
| --- | --- | --- |
| Thread # | Average GFLOPS | Latency (in Millisecond) |
| 1 | 1.0876 | 2201.7254 |
| 2 | 3.8153 | 2987.3245 |
| 4 | 6.1541 | 3001.1233 |

b). GIOPS

|  |  |  |
| --- | --- | --- |
| Thread # | Average GIOPS | Latency (in Millisecond) |
| 1 | 2.1342 | 2109.1134 |
| 2 | 4.6477 | 2678.7501 |
| 4 | 7.0012 | 3091.90 |

From this graph above, it is very well evident that with the increase in the number of threads, the computation power of the cores increases. And hence the overall performance.

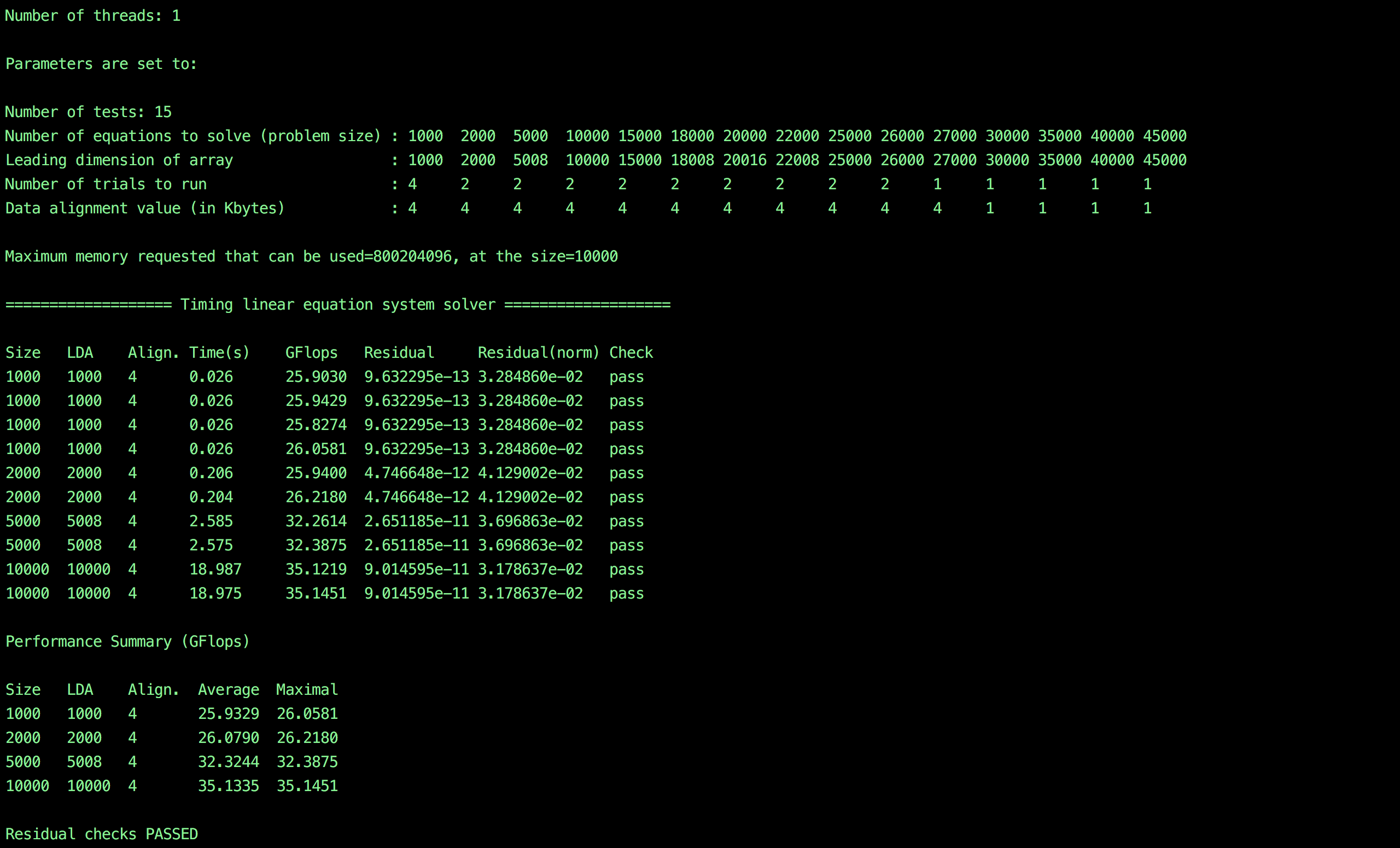
**THEORITICAL CALCULATIONS**

**Theoretical peak performance** of CPU is given by,

No. of Cores \* CPU Speed(GHz) \* Instructions per cycle = 2 \* 2.7 \* 420.8 GFLOPS

**LINPACK Comparison:**

A benchmark tool was provided to compare my CPU benchmark with the LINPACK’s benchmark



**600 Samples:** I ran the benchmark on floating point and integer

instructions and 4 threads for a 10-minute period for each one, and took samples every second on how many instructions per second were achieved during the experiment.

## 4.2 MEMORY BENCHMARK

For Memory benchmarking, “memcpy” operation is done on the memory using the code written in C. The memcpy operation copies bytes from one location to another using Random memory access operation and Sequential Memory Access operation. The benchmark executes for different memory block sizes of 1Byte, 1KBye and 1Mbyte.

Below are the results of Random and Sequential Access:

a) **RANDOM ACCESS - Throughput**

b) **RANDOM ACCESS – Latency**

**c) SEQUENTIAL ACCESS - Throughput**

**d). SEQUENTIAL ACCESS - Latency**

RANDOM ACCESS

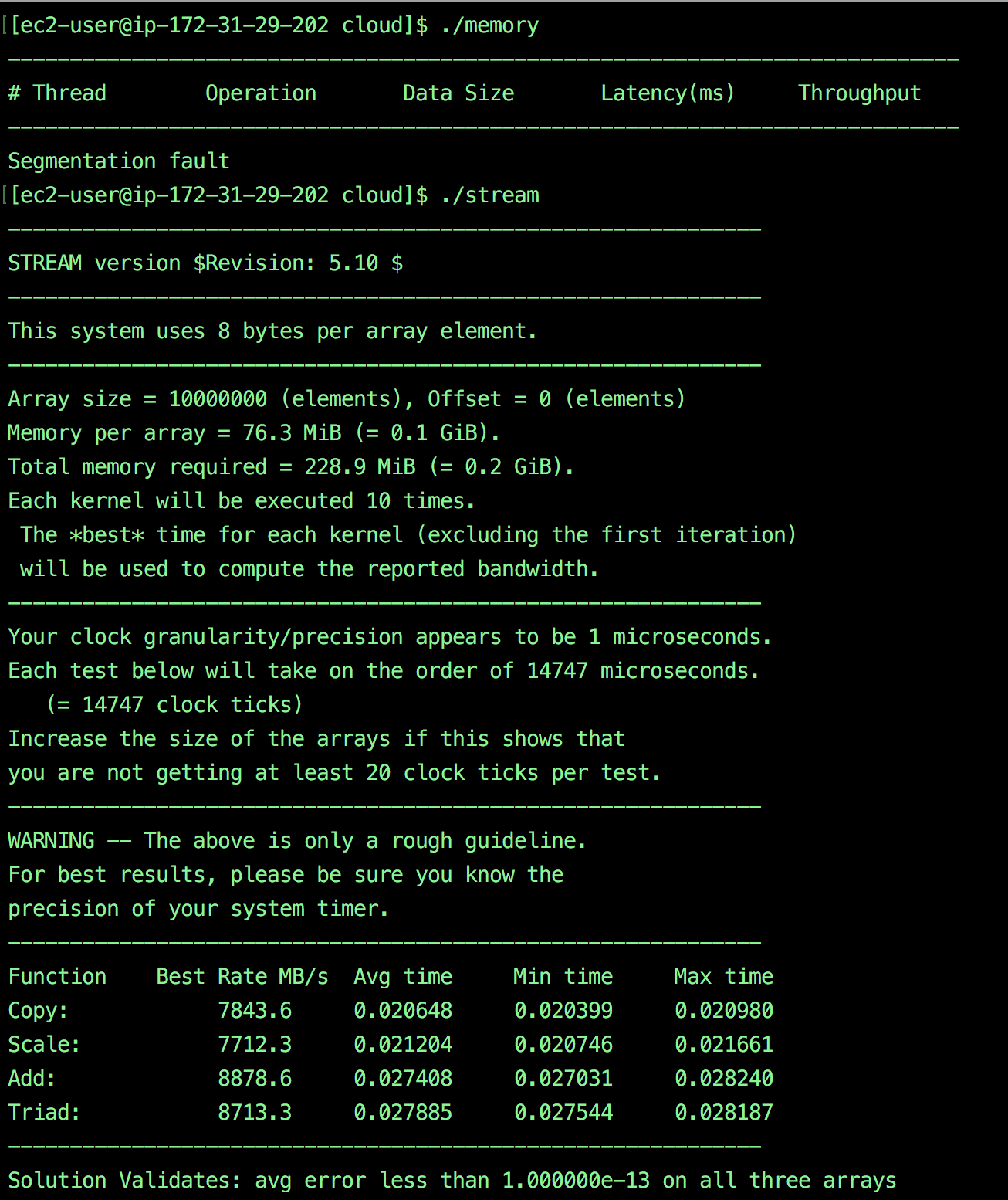
|  |  |  |  |
| --- | --- | --- | --- |
| # of Thread | Block Size | Average Throughput (Mbps) | Average Latency (in MilliSecond) |
| 1 | 1B | 18 | 0.000018 |
| 1 | 1KB | 2766 | 0.000201 |
| 1 | 1MB | 3400 | 0.246233 |
| 2 | 1B | 19 | 0.000011 |
| 2 | 1KB | 2592 | 0.000402 |
| 2 | 1MB | 2680 | 0.275689 |

SEQUENTIAL ACCESS

|  |  |  |  |
| --- | --- | --- | --- |
| # of Thread | Block Size | Average Throughput (Mbps) | Average Latency (in MilliSecond) |
| 1 | 1B | 193 | 0.000004 |
| 1 | 1KB | 286 | 0.000132 |
| 1 | 1MB | 3927 | 0.154232 |
| 2 | 1B | 122 | 0.000011 |
| 2 | 1KB | 294 | 0.000235 |
| 2 | 1MB | 4371 | 0.275689 |

**RUNNING STREAM BENCHMARK TOOL**

The stream benchmark tool was run in order to compare my own benchmark tool



## 4.3 NETWORK BENCHMARK

This benchmark tool will measure the network speed by running two instances of a Server and a Client and sending data from Client to Server 1000 times and then downloading the same from the Server. And then calculating the total time and hence the total network speed. The tool runs on both UDP and TCPThe results are as follows:

Throughput --> in MB / second

Latency --> in MilliSecond

**TCP Throughput**

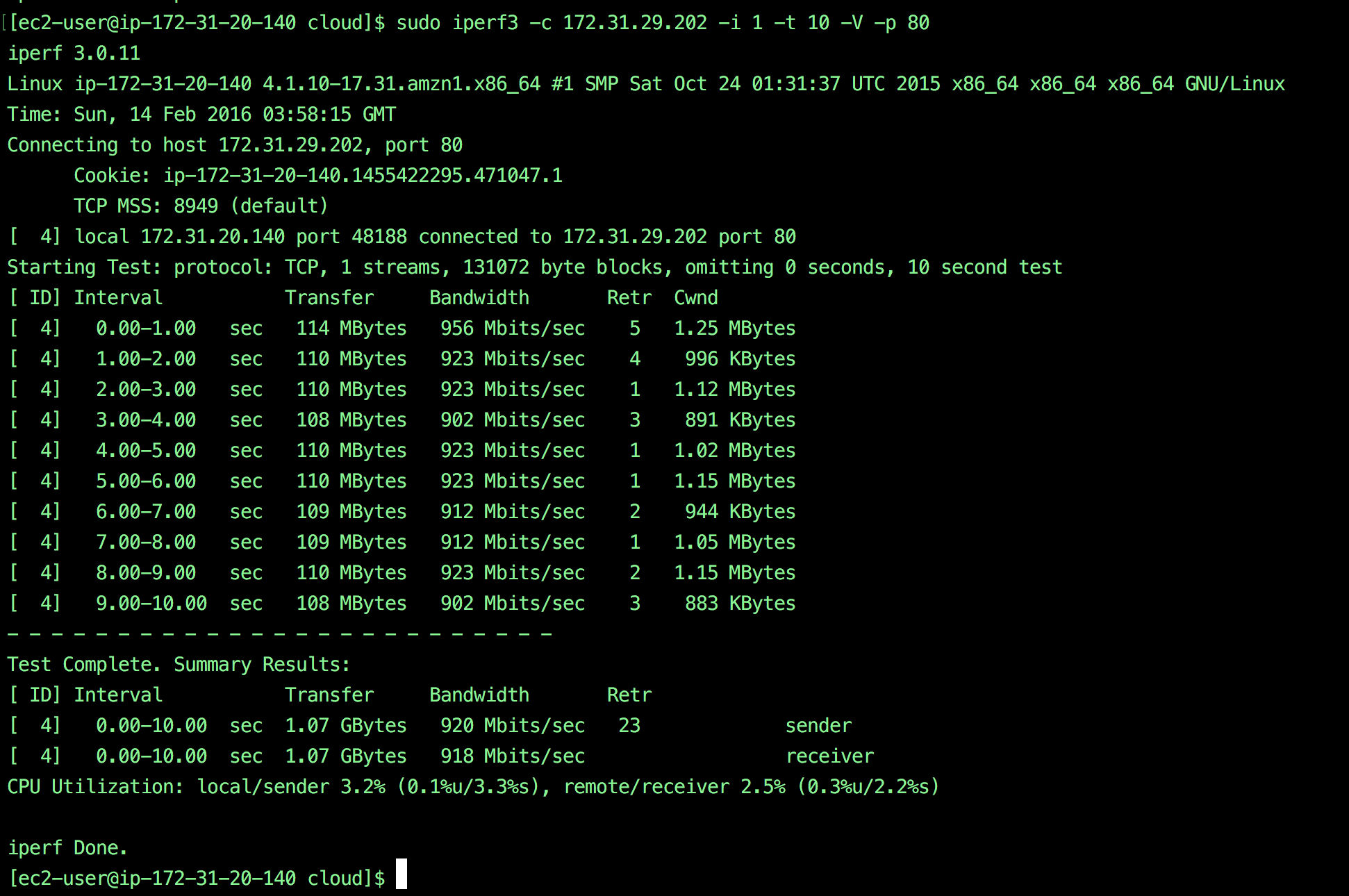
|  |  |  |  |
| --- | --- | --- | --- |
| # of Threads | Packet Size | Avg Throughput (MBps) | Latency (ms) |
| 1 | 1B | 0.0015 | 51 |
| 1 | 1KB | 8.547 | 117 |
| 1 | 1MB | 41.955 | 2908 |
| 2 | 1B | 0.0023 | 48 |
| 2 | 1KB | 30.213 | 133 |
| 2 | 1MB | 87.458 | 5418 |

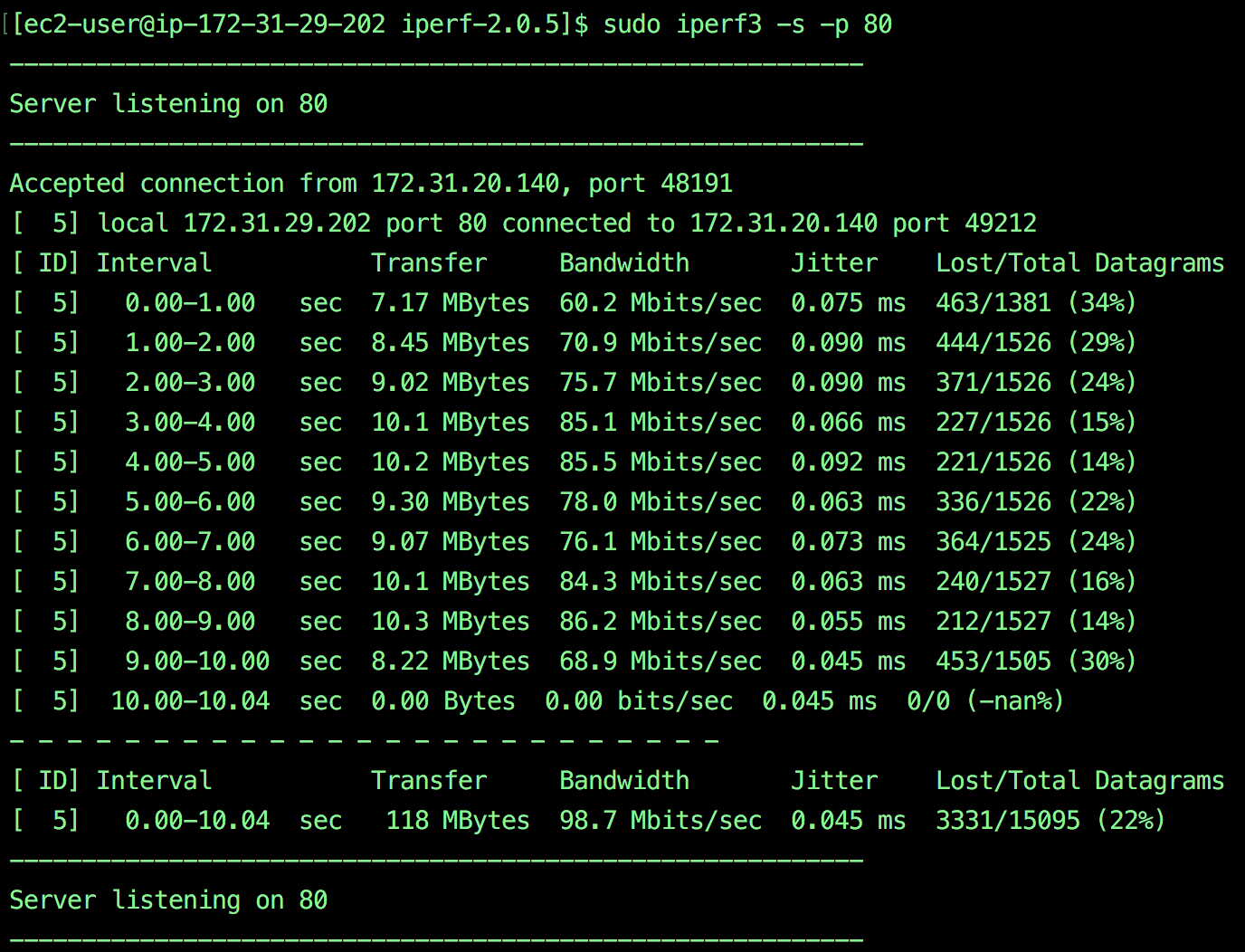
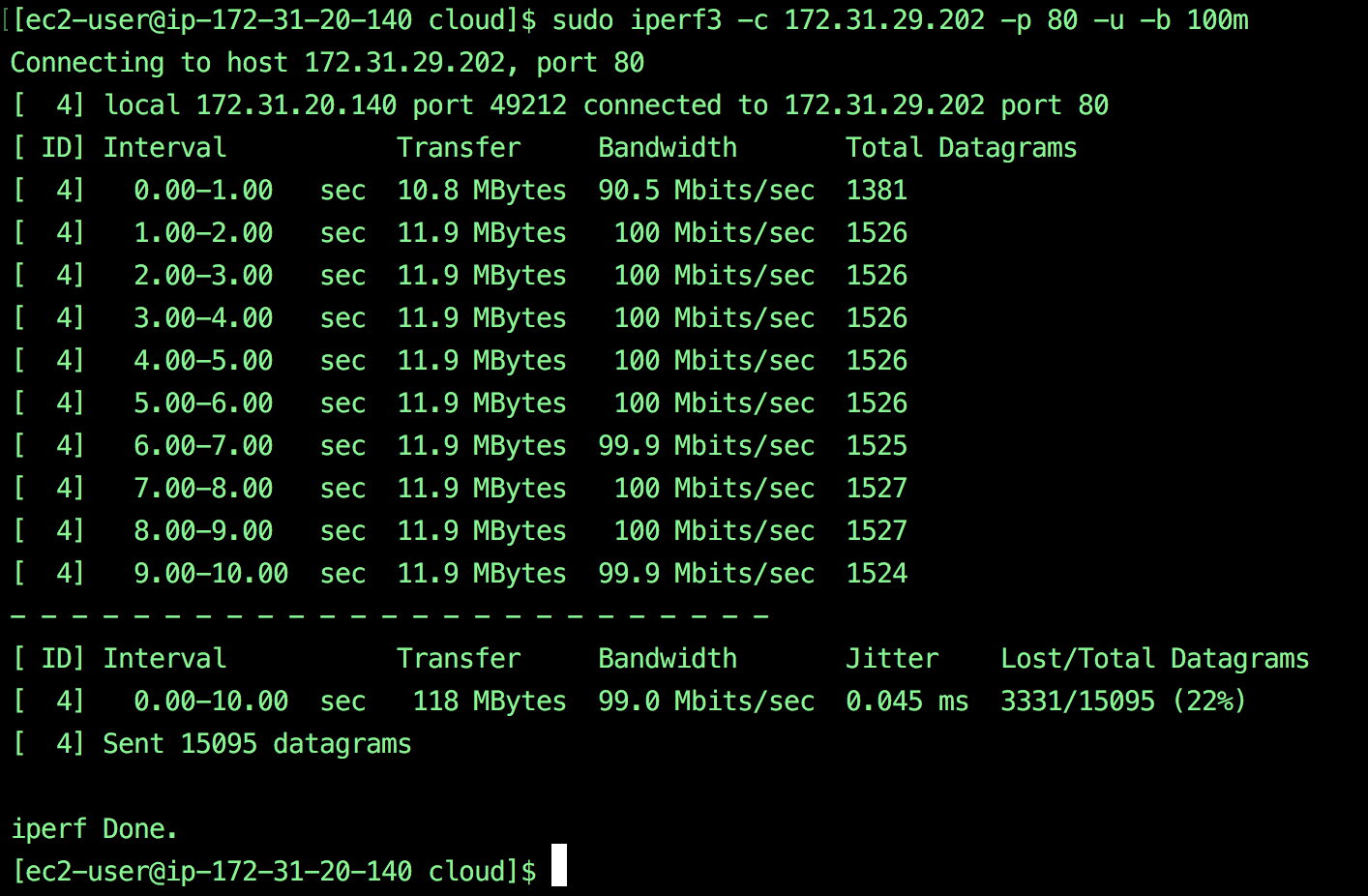
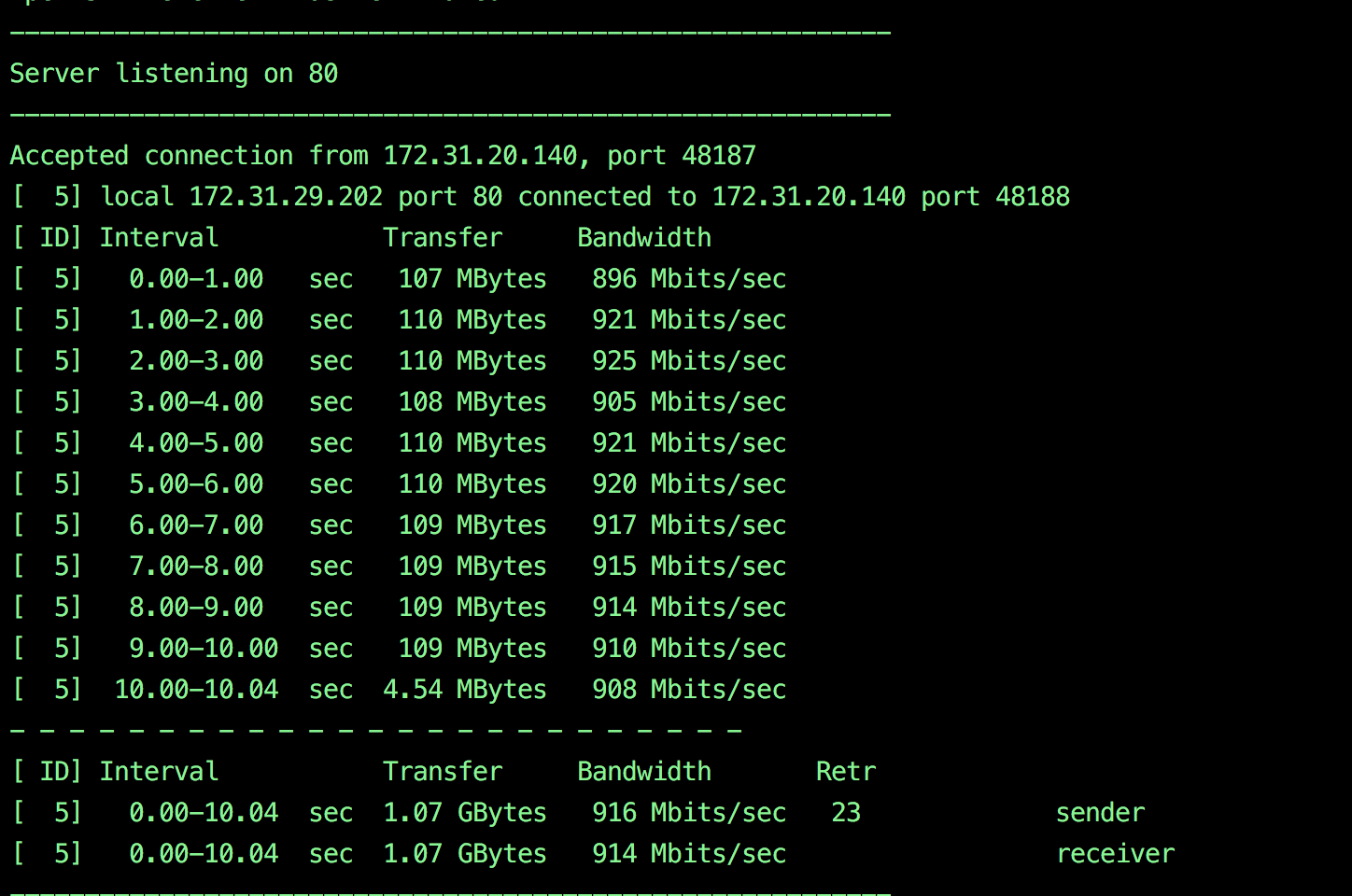
**UDP Throughput**

**UDP Latency**

## 

|  |  |  |  |
| --- | --- | --- | --- |
| # of Threads | Packet Size | Avg. Throughput( Mbps) | Latency (in Millisecond) |
| 1 | 1B | 0.0019 | 53 |
| 1 | 1KB | 12.313 | 69 |
| 1 | 1MB | 62.876 | 1189 |
| 2 | 1B | 0.0036 | 76 |
| 2 | 1KB | 37.65 | 152 |
| 2 | 1MB | 98.222 | 4789 |

**RUNNING IPERF **

****