CS 553 Cloud Computing

Programming Assignment 1

VICKYBEN PATEL (CWID A20370450)

BENCHMARKING:

The goal of benchmark is measuring and evaluating computational performance. Here we did benchmarking on three parts

CPU
MEMORY
DISK

CPU BENCHMARK

Program Design: CPU – Measure the processor speed, in terms of floating point operations per second (Giga FLOPS,109 FLOPS) and integer operations per second (Giga IOPS, 109 IOPS)

I have performed arithmetic operations for integer and floating values with loop. I have used threads to achieve parallelism. Each thread will do integer and floating point operations repeatedly and note down start time and end time then GFLOPS and GIOPS values are calculated.

Design Improvements:

- I have performed arithmetic operations with loop and it will be extended for more complex operations.

Methods explanation:

- main() – In this method threads are created and call to other methods for perform execution.
- cpuI() - In this method Integer arithmetic operations are performed.
- cpuF() – In this method Floating arithmetic operations are performed.

Manual

Steps to run the code

- Open Terminal
- Connect EC2 using **ssh -i "biya812.pem" ec2-user@ec2-52-91-94-113.compute-1.amazonaws.com**
- To transfer C file to EC2 use command **scp -i /home/biya/biya812.pem /home/biya/cpu.c ec2-user@54.85.180.114:/home/ec2-user/PA1**
- For compiling C file in Linux, compiler must be installed using **sudo yum group install "Development Tools"**
- Set folder where the file is located.
- Then run the command **gcc cpu.c -lpthread**
  gcc command used for compiling the file and –lpthread for thread execution.
- Once, code gets compiled successfully then for execution of the program use **./a.out** command.

Source Code

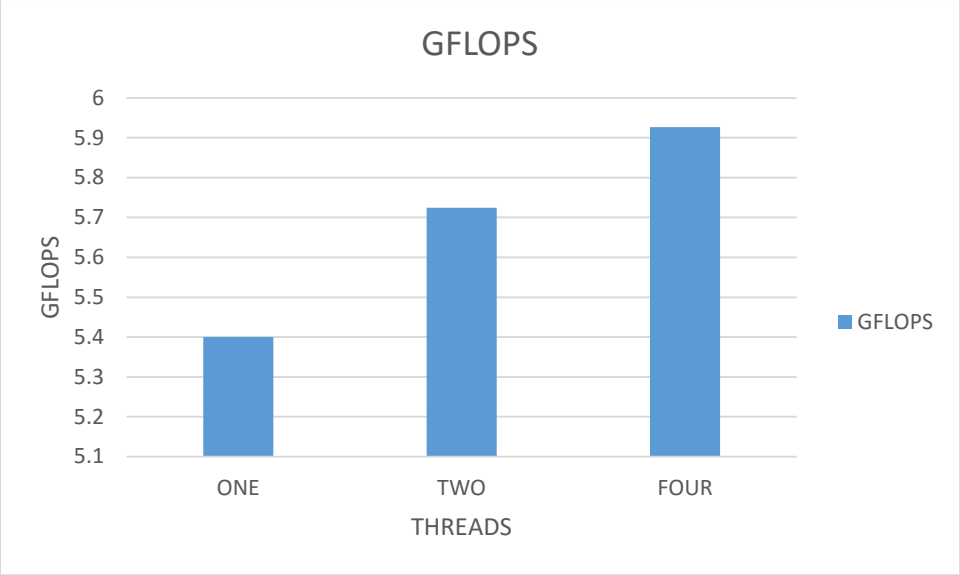
Source code can be found in cpu source code folder

Performance Table:

Number of Threads	GFLOPS	GIOPS
1	5.400000	6.075000
2	5.724532	6.527489
4	5.926829	6.854163

Graphs:

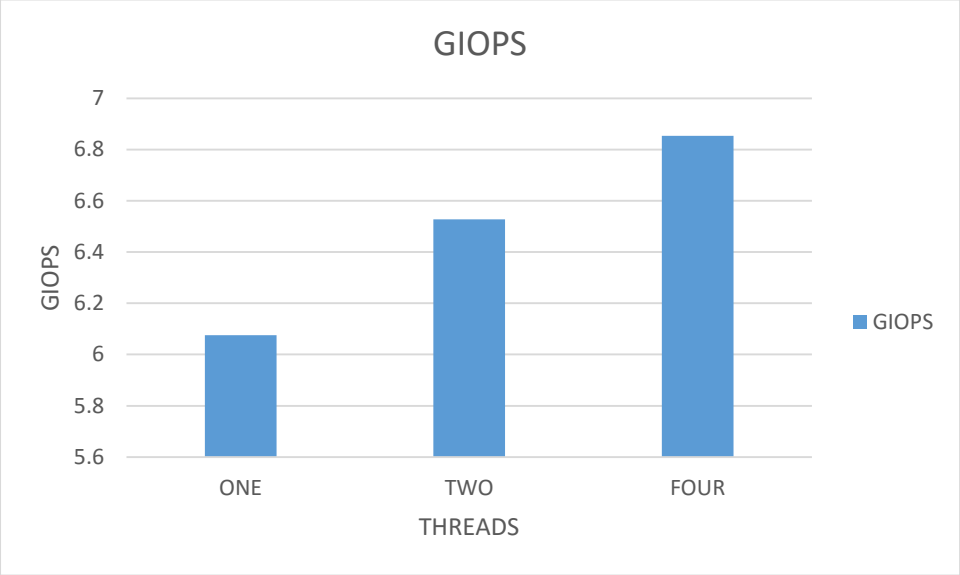
GRAPH -1 :



X-axis - threads
Y-axis - GFLOPS

- As no of threads increases GFLOPS increase

GRAPH -2



X-axis - threads
Y-axis - GIOPS

As number of threads increase GIOPS increase.
GFLOPS is lower than GIOPS which says floating point operations takes more time than integer operations

MEMORY BENCHMARK

Program Design: Memory – Measure the memory speed of host

To measure the memory speed I created two array of size 30 MB in memory. Depending upon type of access either sequential or random I copy different block size from one array to another array. For calculation of latency and throughput I calculate time difference between this operations for threads and average it.

Design Improvements

Instead of array I could have used list for dynamic memory allocation

Methods Explanation

- main() – In this method threads are created and call to other methods for perform execution.
- SeqRWB() - In this method sequential read write operation perform for 1 B block size for this I create two arrays of size 30 MB and fill one array then perform memcpy() function for 1 B block size.
- SeqRWKB() – In this method sequential read write operation perform for 1 KB block size for this I create two arrays of size 30 MB and fill one array then perform memcpy() function for 1 KB block size.
- SeqRWMB()- In this method sequential read write operation perform for 1 MB block size for this I create two arrays of size 30 MB and fill one array then perform memcpy() function for 1 MB block size.
- RndmRWB() - In this method sequential read write operation perform for 1 B block size for this I create two arrays of size 30 MB and fill one array then perform memcpy() function for 1 B block size.
- RndmRWKB() - In this method sequential read write operation perform for 1 KB block size for this I create two arrays of size 30 MB and fill one array then perform memcpy() function for 1 KB block size.
- RndmRWMB() - In this method sequential read write operation perform for 1 MB block size for this I create two arrays of size 30 MB and fill one array then perform memcpy() function for 1 MB block size.

Manual

Steps to run the code

- Open Terminal
- Connect EC2 using **ssh -i "biya812.pem" ec2-user@ec2-52-91-94-113.compute-1.amazonaws.com**

- To transfer C file to EC2 use command **scp -i /home/biya/biya812.pem /home/biya/memory.c ec2-user@54.85.180.114:/home/ec2-user/PA1**
- For compiling C file in Linux, compiler must be installed using **sudo yum group install "Development Tools"**
- Set folder where the file is located.
- Then run the command **gcc memory.c -lpthread**
  gcc command used for compiling the file and –lpthread for thread execution.
- Once, code gets compiled successfully then for execution of the program use **./a.out** command.

Source Code

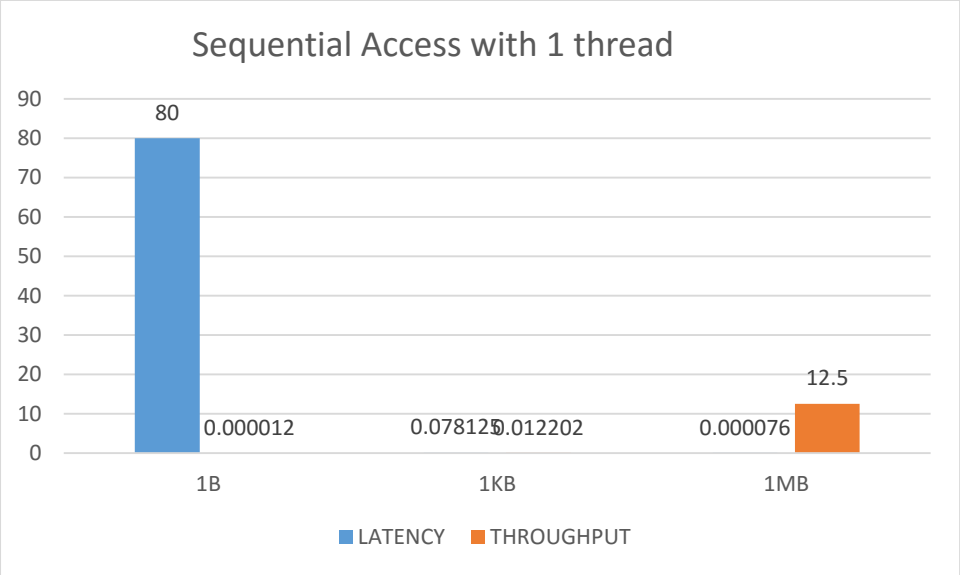
Source code can be found in memory source code folder

Performance

Below graphs shows the throughput and latency for different threads on different data size. Throughput is inversely proportional to latency which is being depicted below in the graph. Sequential access throughput is higher than the random access throughput. As, for sequential access, pointer has to move sequentially, whereas for the random access, pointer has to move to the random

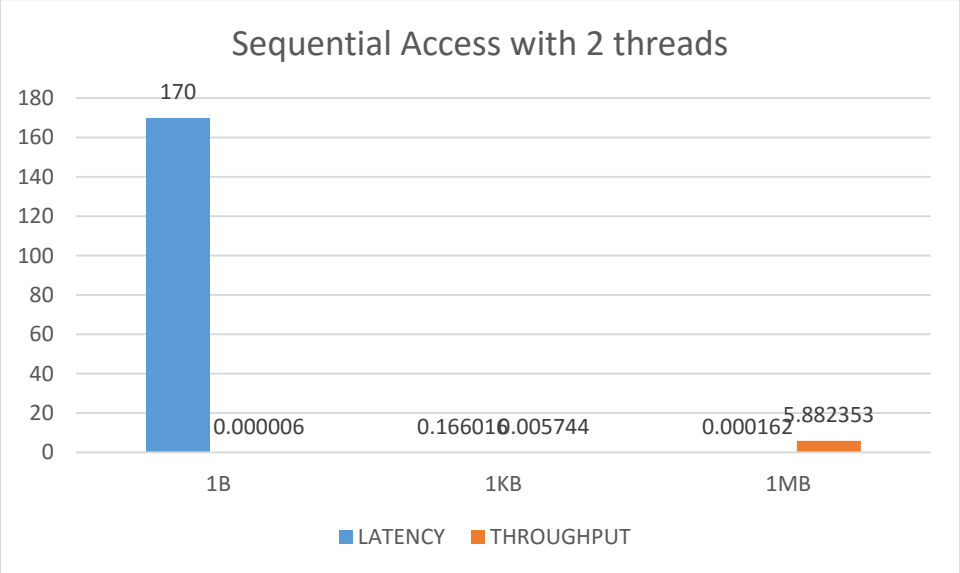
Sequential Access with 1 Thread

BLOCK SIZE	LATENCY	THROUGHPUT
1B	80	0.000012
1KB	0.078125	0.012202
1MB	0.000076	12.5



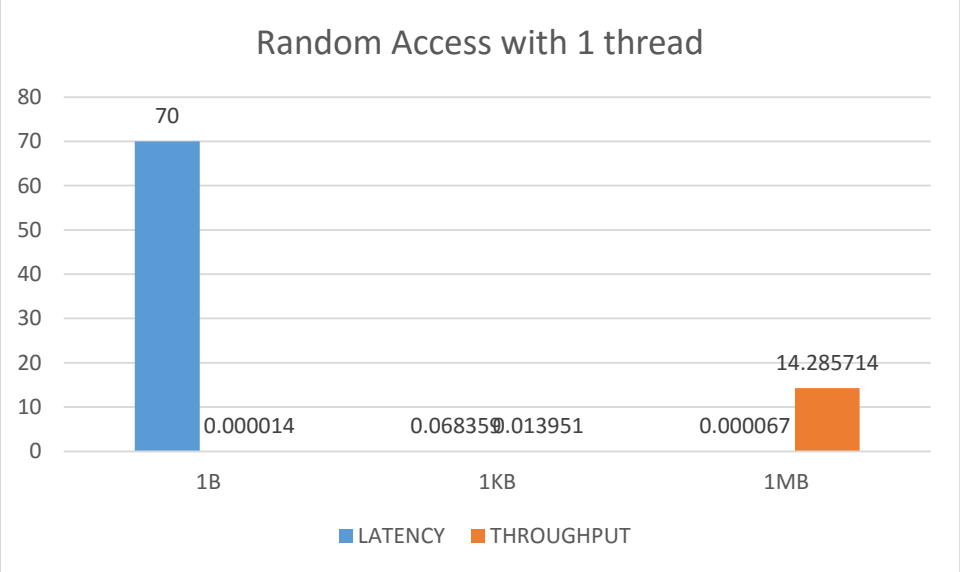
Sequential Access with 2 Threads

TEST	LATENCY	THROUGHPUT
1B	170	0.000006
1KB	0.166016	0.005744
1MB	0.000162	5.882353



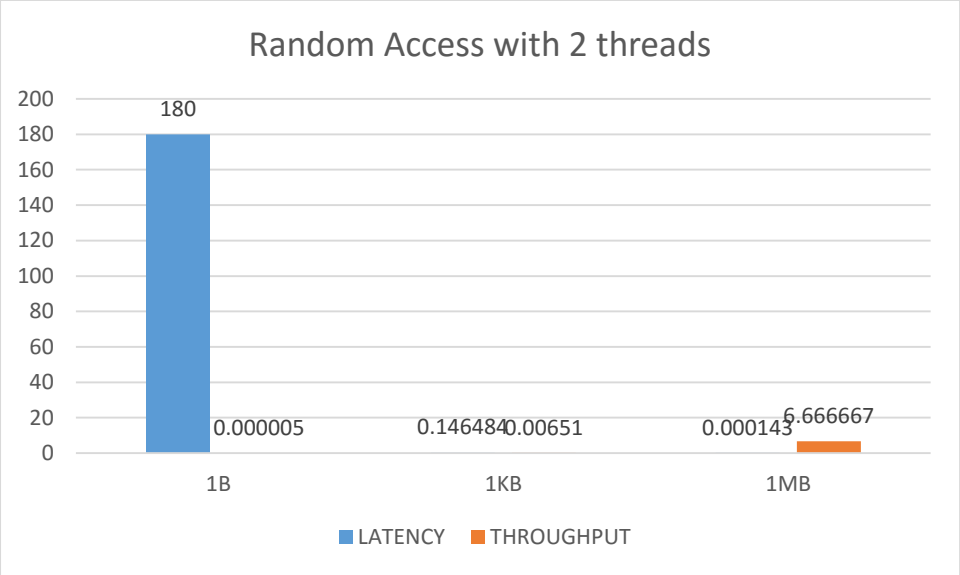
Random Access with 1 Thread

TEST	LATENCY	THROUGHPUT
1B	70	0.000014
1KB	0.068359	0.013951
1MB	0.000067	14.28571



Random Access with 2 Threads

TEST	LATENCY	THROUGHPUT
1B	180	0.000005
1KB	0.146484	0.00651
1MB	0.000143	6.666667


Random Access with 2 threads
200
180
160
140
120
100
80
60
40
20
0
180
0.000005
0.146484
0.00651
0.000143
6.666667
1B
1KB
1MB
LATENCY
THROUGHPUT

DISK BENCHMARK

Program Design DISK – Measure the disk speed

Depending upon type of access either sequential or random I read different block size from file and for write different block size read from buffer and write to file. For calculation of latency and throughput I calculate time difference between this operations for threads and average it.

Methods Explanation

main() **–** In this method threads are created and call to other methods for perform execution.
SeqRB() - In this method sequential read operation perform from diskread.txt for 1 B block.
SeqKB – In this method sequential read operation perform from diskread.txt for 1 KB block size
SeqRMB - In this method sequential read operation perform from diskread.txt for 1 MB block size
RndmRB - In this method random read operation perform from diskread.txt for 1 B block size
RndmRKB - In this method random read operation perform from diskread.txt for 1 KB block size
RndmRMB - In this method random read operation perform from diskread.txt for 1 MB block size
SeqWB() - In this method sequential write operation perform to diskwrite.txt for 1 B block size
SeqWKB – In this method sequential write operation perform to diskwrite.txt for 1 KB block size
SeqWMB - In this method sequential write operation perform to diskwrite.txt for 1 MB block size
RndmWB - In this method random write operation perform to diskwrite.txt for 1 B block size
RndmWKB - In this method random write operation perform to diskwrite.txt for 1 KB block size
RndmWMB - In this method random write operation perform to diskwrite.txt for 1 MB block size

Manual

Steps to run the code

- Open Terminal
- Connect EC2 using **ssh -i "biya812.pem" ec2-user@ec2-52-91-94-113.compute-1.amazonaws.com**

- To transfer C file to EC2 use command **scp -i /home/biya/biya812.pem /home/biya/disk.c ec2-user@54.85.180.114:/home/ec2-user/PA1**
- For compiling C file in Linux, compiler must be installed using **sudo yum group install "Development Tools"**
- Set folder where the file is located.
- Then run the command **gcc disk.c -lpthread**
  gcc command used for compiling the file and –lpthread for thread execution.
- Once, code gets compiled successfully then for execution of the program use **./a.out** command.

Source Code

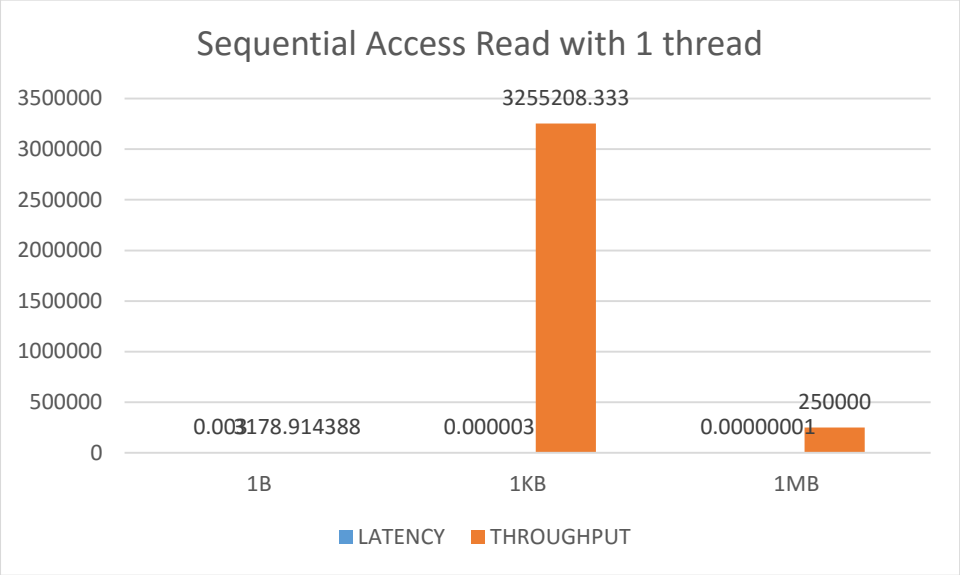
Source code can be found in disk source code folder

PERFORMANCE

Below graphs shows the throughput and latency for different threads on different data size. Throughput is inversely proportional to latency which is being depicted below in the graph. Sequential access throughput is higher than the random access throughput. As, for sequential access, pointer has to move sequentially, whereas for the random access, pointer has to move to the random

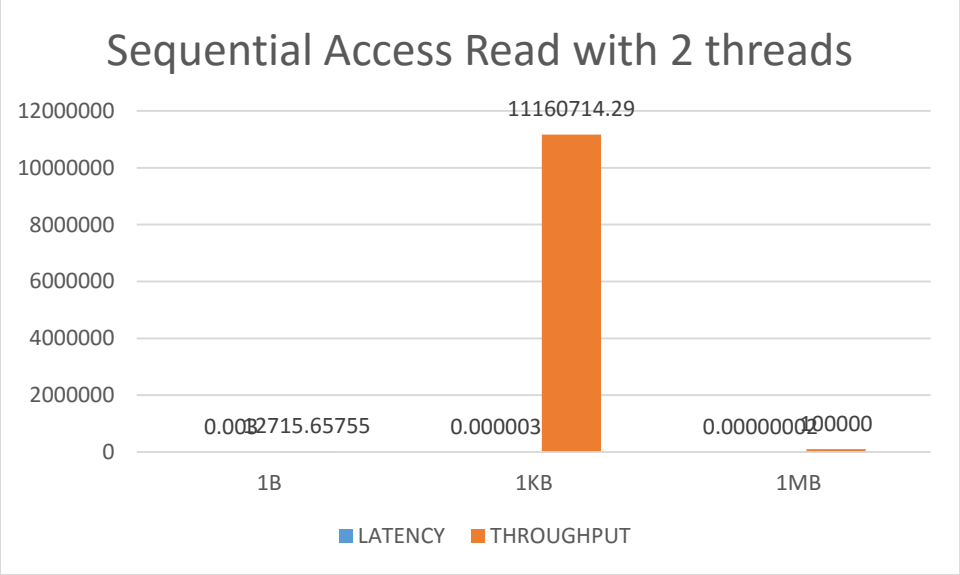
Sequential Access READ with 1 Thread

TEST	LATENCY	THROUGHPUT
1B	0.003	3178.914
1KB	0.000003	3255208
1MB	0.00000001	250000



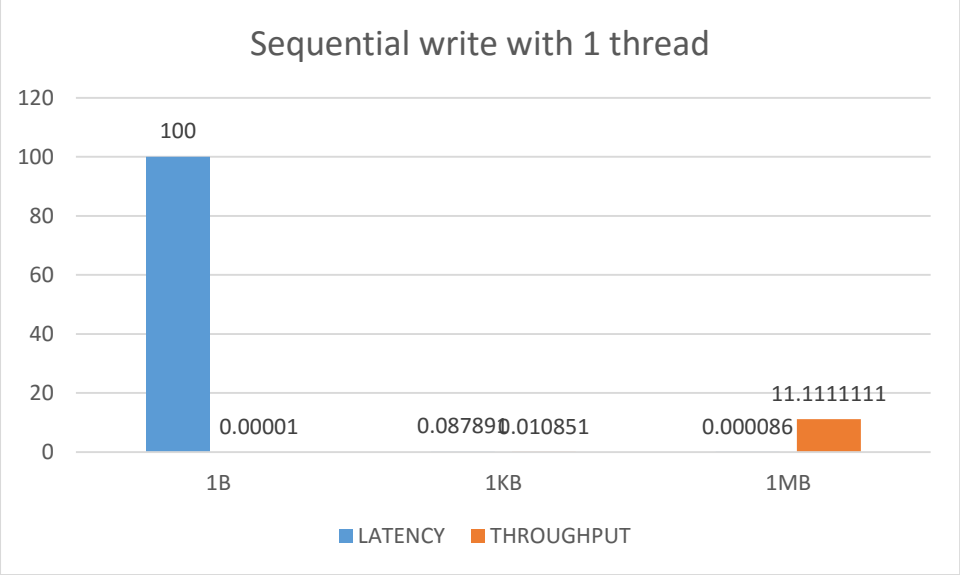
Sequential Access READ with 2 Thread

TEST	LATENCY	THROUGHPUT
1B	0.003	12715.66
1KB	0.000003	11160714
1MB	0.00000002	100000



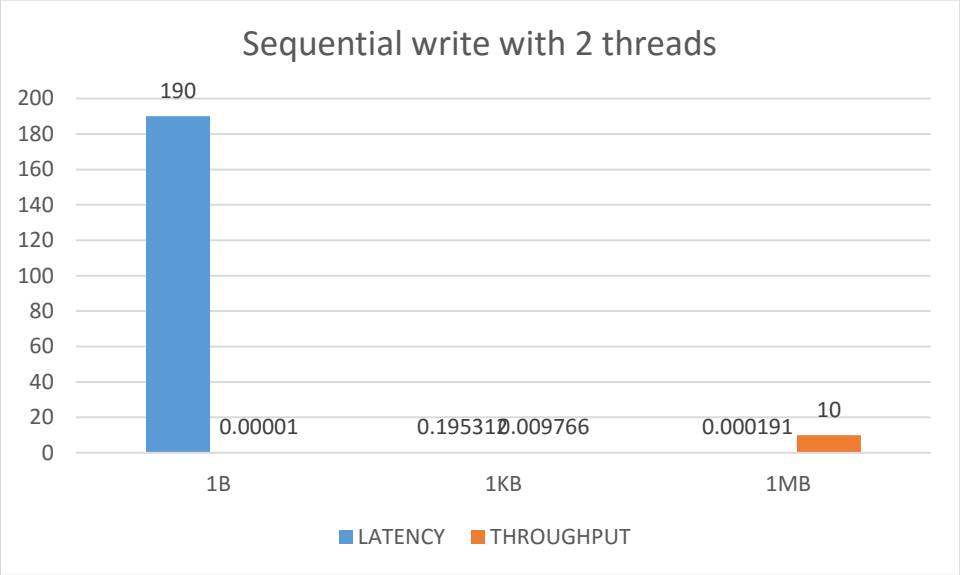
Sequential Access WRITE with 1 Thread

TEST	LATENCY	THROUGHPUT
1B	100	0.00001
1KB	0.087891	0.010851
1MB	0.000086	11.11111



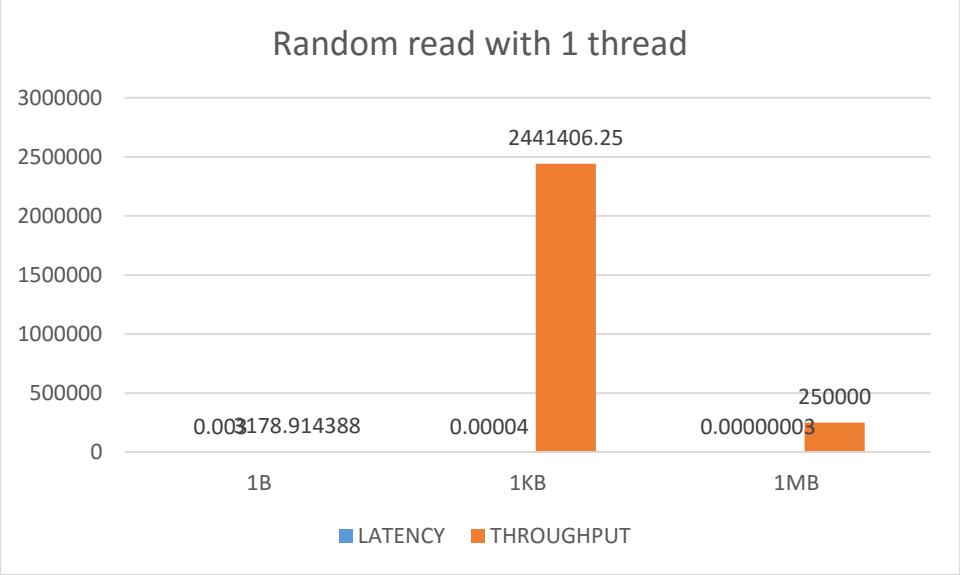
Sequential Access WRITE with 2 ThreadS

TEST	LATENCY	THROUGHPUT
1B	190	0.00001
1KB	0.195312	0.009766
1MB	0.000191	10



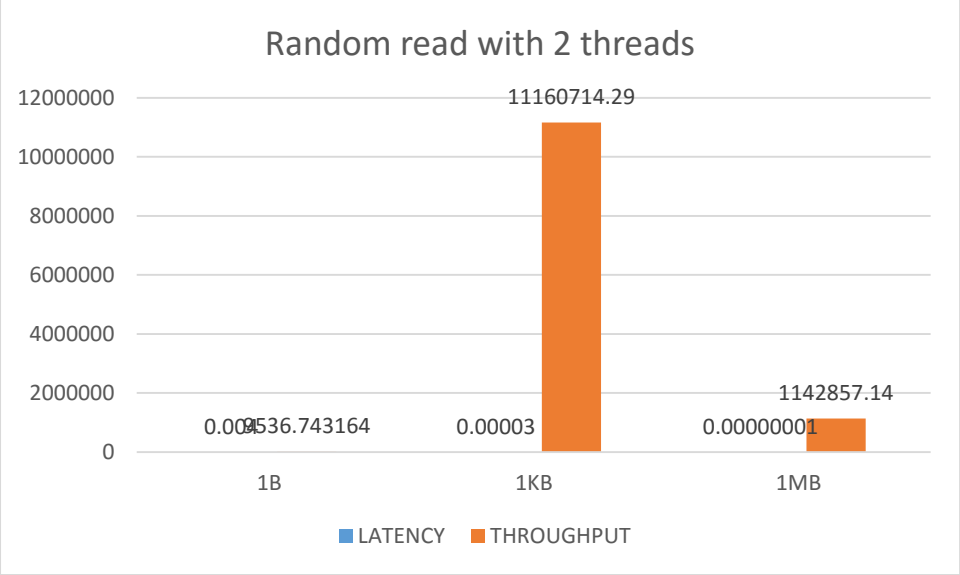
Random Access read with 1 Thread

TEST	LATENCY	THROUGHPUT
1B	0.003	3178.914
1KB	0.00004	2441406
1MB	0.00000003	250000



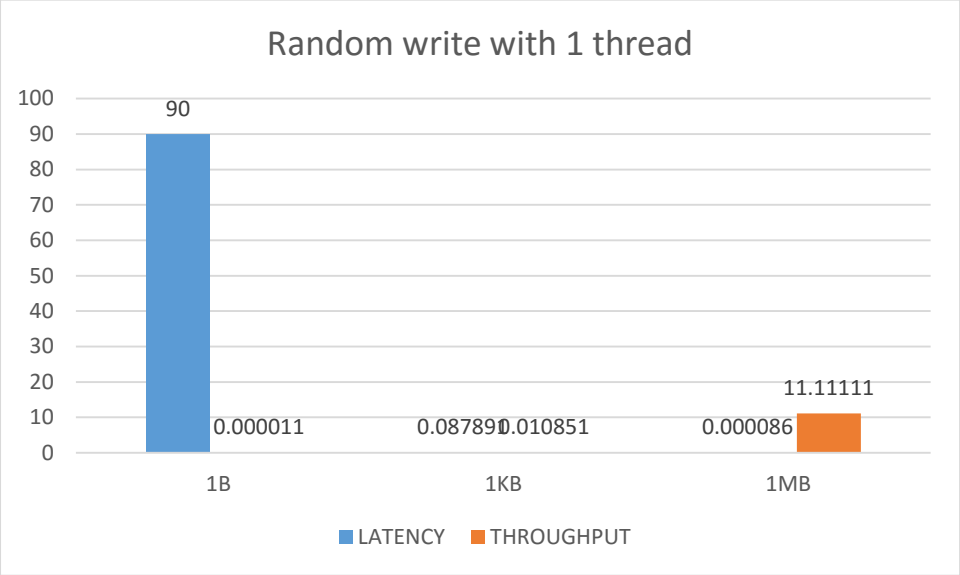
Random Access read with 2 Threads

TEST	LATENCY	THROUGHPUT
1B	0.004	9536.743
1KB	0.00003	11160714
1MB	0.00000001	1142857



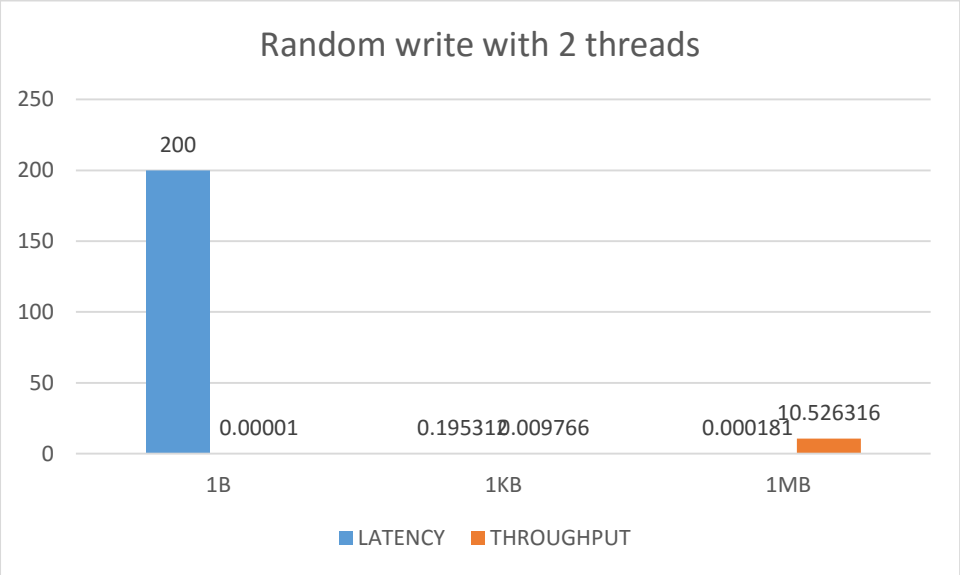
Random Access write with 1 Thread

TEST	LATENCY	THROUGHPUT
1B	90	0.000011
1KB	0.087891	0.010851
1MB	0.000086	11.11111



Random Access write with 2 Threads

TEST	LATENCY	THROUGHPUT
1B	200	0.00001
1KB	0.195312	0.009766
1MB	0.000181	10.52632



System Specifications used for Testing:

- Processor: Intel (R) core (TM) i5 -5200U CPU @2.20GHZ 2.20GHZ
- Installed memory (RAM): 4GB
- System Type: 64 bit Operating System, x64-based processor
- Cores : 2, Logical processors : 4