CS553 Programming Assignment #1

Design – Benchmarking

Divya Krishnamoorthy –A2035633

Benchmarking of Amazon EC2 t2.micro instance

The benchmarking of the system is carried out with

- CPU Benchmark
- Disk Benchmark
- Network Benchmark

Design Details

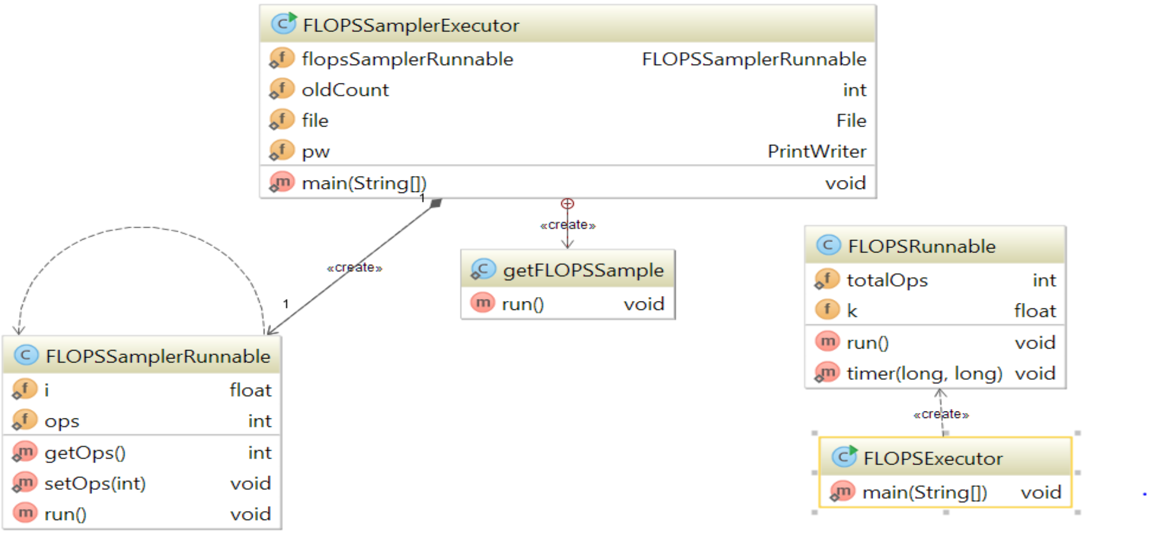
- JAVA is used as the language for implementation
- JAVA Multithreading is a key component
- JAVA.NET used for Socket programming in TCP/UDP protocols

1) CPU Benchmark

➢ GFLOPS:

The main classes are as shown below.

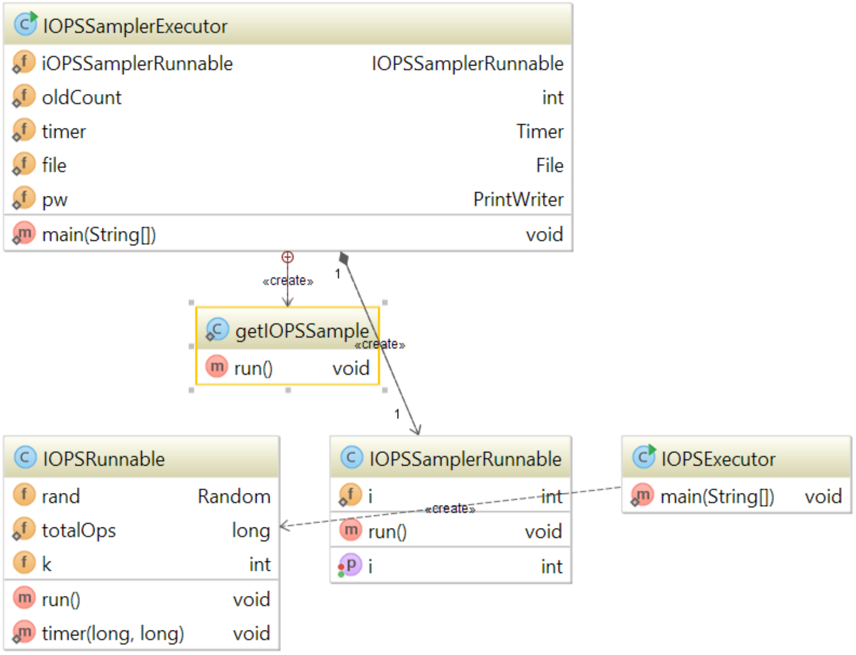
1. FLOPSRunnable is the thread implementation class for GFLOPS calculation.
2. Executor classes controls the thread creations and terminations
3. FLOPSSamplerRunnable creates the thread for sampling the FLOPS per second
4. FLOPSSamplerExecutor controls the creation and termination of the threads. It also has the TimerTask which collects the number of FLOPS executed per second



➢ GIOPS

The main classes are as shown below.

1. IOPSRunnable is the thread implementation class for GIOPS calculation.
2. Executor classes controls the thread creations and terminations
3. IOPSSamplerRunnable creates the thread for sampling the IOPS per second
4. IOPSSamplerExecutor controls the creation and termination of the threads. It also has the TimerTask which collects the number of IOPS executed per second

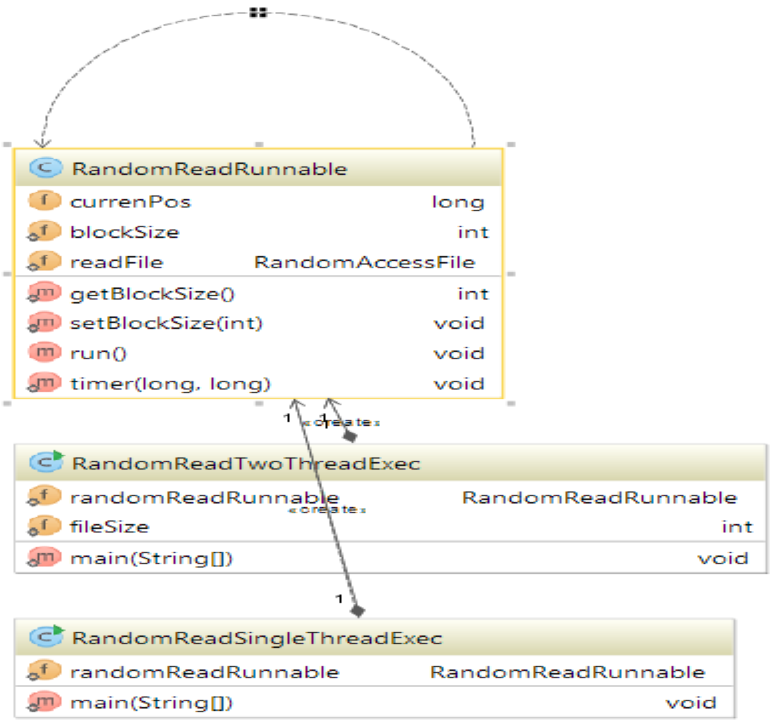


2) DISK Benchmark

- DISK RANDOM READ

The main classes are as shown below.

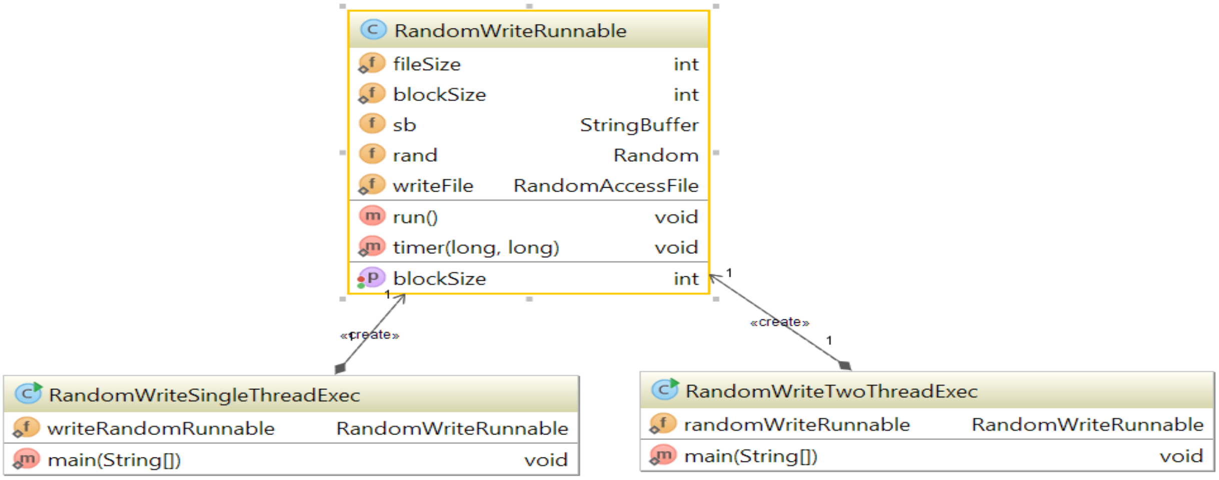
1. RandomReadRunnable is the thread implementation class for Latesncy and Mbytes/Second calculation.
2. Executor classes controls the thread creations and terminations
3. RandomReadSingleThreadExec controls 1B,1KB,1MB operations for singleThread
4. RandomReadTwoThreadExec controls the creation and termination of the threads for 1B,1KB,1MB operations for concurrent execution of two threads
5. RandomAccessFile API is used for seeking Random position in the file and proceed with the read operations.



- **DISK RANDOM WRITE**

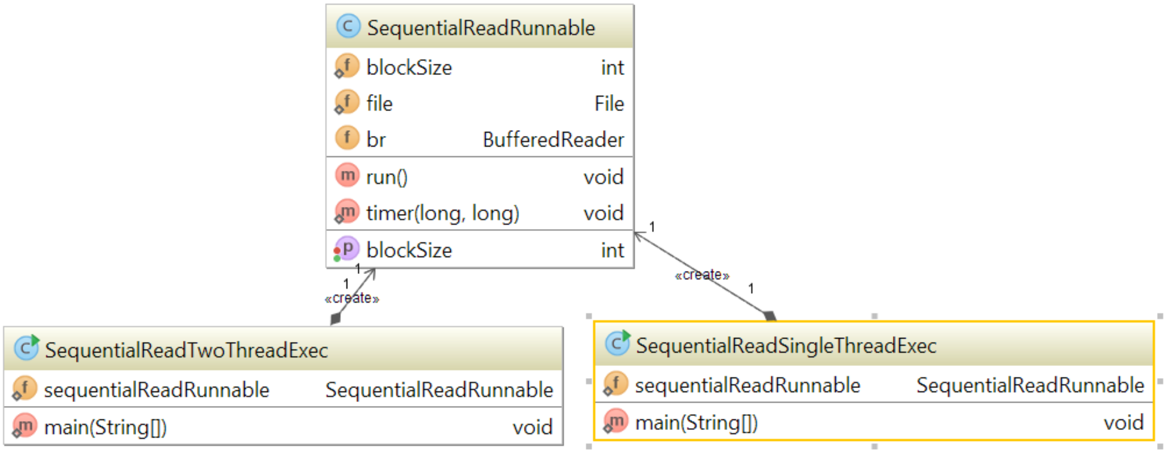
The main classes are as shown below.

2. RandomWriteRunnable is the thread implementation class for Latency and Mbytes/Second calculation.
3. Executor classes controls the thread creations and terminations
3. RandomWriteSingleThreadExec controls 1B,1KB,1MB operations for singleThread
3. RandomWriteTwoThreadExec controls the creation and termination of the threads for 1B,1KB,1MB operations on two threads concurrent execution.



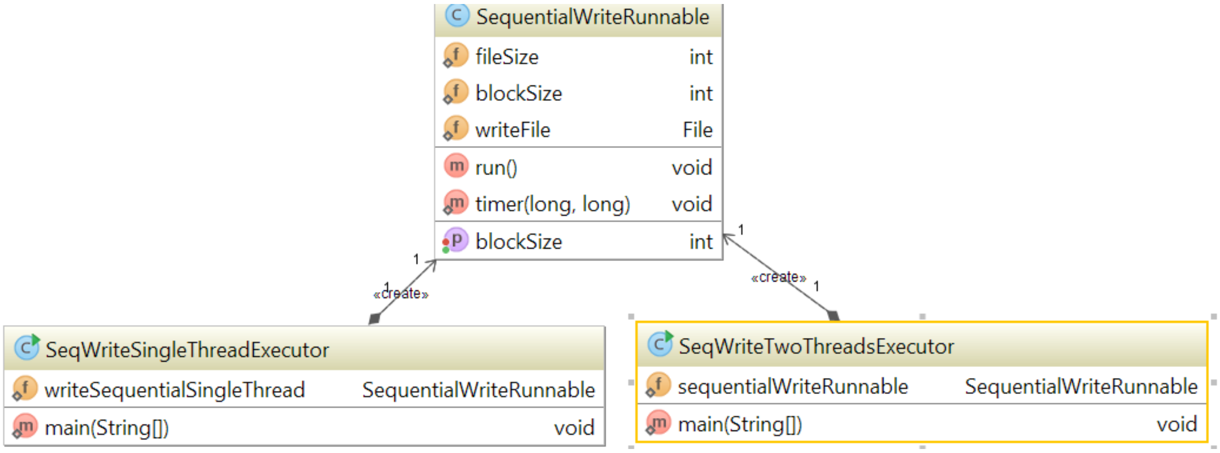
- **<u>DISK SEQUENTIAL READ</u>**

1. SequentialReadRunnable is the thread implementation class for Latency and Mbytes/Second calculation.
2. Executor classes controls the thread creations and terminations
3. SequentialReadSingleThreadExec controls 1B,1KB,1MB operations for singleThread Read.
4. SequentialReadTwoThreadExec controls the creation and termination of the threads for 1B,1KB,1MB operations on two threads concurrent execution.
5. BufferedReader is used for reading the files in different block sizes.



- **<u>DISK SEQUENTIAL WRITE</u>**
- SequentialWriteRunnable is the thread implementation class for Latency and Mbytes/Second calculation.
- Executor classes controls the thread creations and terminations
- SeqWriteSingleThreadExecutor controls 1B,1KB,1MB operations for singleThread Write.
- SeqWriteTwoThreadsExecutor controls the creation and termination of the threads for

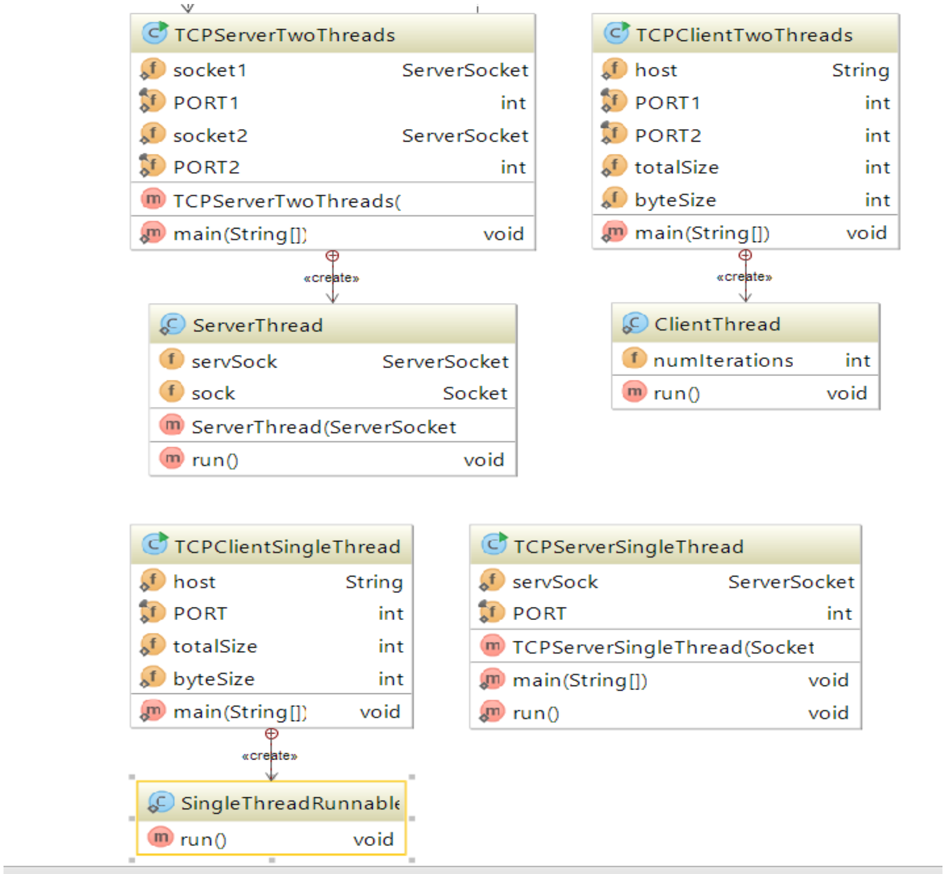
1B,1KB,1MB operations on two threads concurrent execution.



3) **NETWORK BENCHMARK**

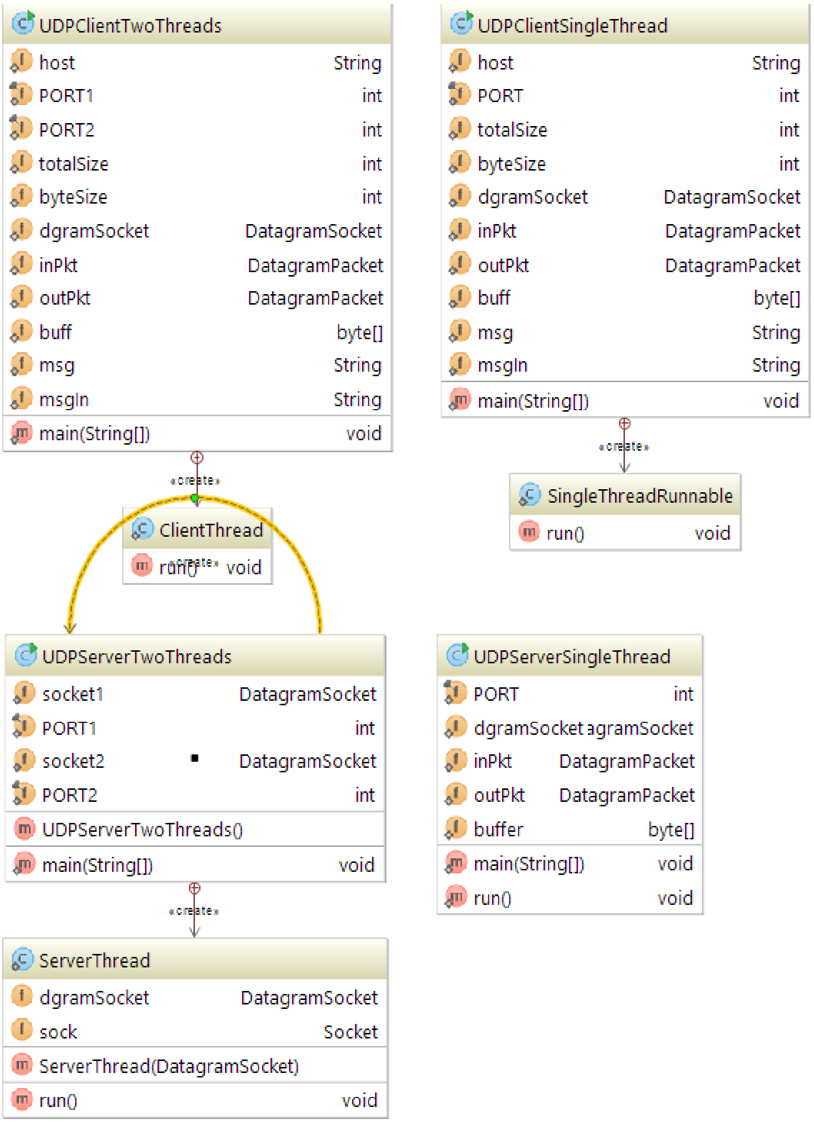
➢ TCPBENCHMARK

1) TCP Benchmarking is done on both single thread and 2 thread scenarios
2) It has a client and a server component
3) Client sends a message to the server and the server upon receipt of the message sends the message back to the client and the RTT is calculated .
4) Throughput is calculated in MB/Second
5) For TWO threads communication scenario, PORT Numbers 1234,1235 is used at the server. Each client thread communicates to the server at one port



- UDP BENCHMARKING

1. UDP Benchmarking is done on both single thread and 2 thread scenarios
2. It has a client and a server component
3. Client sends a message to the server and the server upon receipt of the message sends the message back to the client and the RTT is calculated.
4. Throughput is calculated in MB/Second
5. For TWO threads communication scenario, PORT Numbers 1234,1235 is used at the server. Each client thread communicates to the server at one port


UDPClientTwoThreads
host String
PORT1 int
PORT2 int
totalSize int
byteSize int
dgramSocket DatagramSocket
inPkt DatagramPacket
outPkt DatagramPacket
buff byte[]
msg String
msgIn String
main(String[]) void
«create»
ClientThread
run() void
«create»
UDPClientSingleThread
host String
PORT int
totalSize int
byteSize int
dgramSocket DatagramSocket
inPkt DatagramPacket
outPkt DatagramPacket
buff byte[]
msg String
msgIn String
main(String[]) void
«create»
SingleThreadRunnable
run() void
UDPServerTwoThreads
socket1 DatagramSocket
PORT1 int
socket2 DatagramSocket
PORT2 int
UDPServerTwoThreads()
main(String[]) void
«create»
ServerThread
dgramSocket DatagramSocket
sock Socket
ServerThread(DatagramSocket)
run() void
UDPServerSingleThread
PORT int
dgramSocket agramSocket
inPkt DatagramPacket
outPkt DatagramPacket
buffer byte[]
main(String[]) void
run() void