Design Document for Benchmarking

- This is a project on benchmarking different parts of a computer system like CPU, Disk and Network.
- This benchmarking code is written in java using sockets, threads in Linux environment. The evaluation results were taken on Amazon AWS by creating t2.micro instance of EC2.

1. Benchmarking of CPU:

- **Hardware Specification**

  Amazon EC2 t2.micro instance. Intel Xeon Family, Up to 3.3 GHz Clock Speed.

- **Overview**
  - CPU benchmarking is carried in terms of GLOPS- Giga Floating Point Operations per second and IOPS – Integer Operations per second.
  - The performance of CPU is tested for 1, 2 and 4 threads.
  - To calculate the GFLOPS of CPU two classes are made in a file name CPUFloat.java. A class named CPU contains the method doCalculations(). This method contains 8 instructions on 2 floating point numbers performing total 10 add, sub, multi and division functions on each other.
  - Another class named CPUFloat extends Thread class to implement multithreads on CPU.doCalculations(). This class contains code for creating and starting threads on CPU.doCalculations().
  - Results are taken by calling at a time 1 thread then 2 threads and then finally 4 threads from the CPUFloat class.
  - To calculate the GFLOPS following formula is applied:

    (No. of Instructions * No of Iterations)/ totalTime_taken
  - Similarly, to calculate the IOPS of CPU two classes were made in a file name ThreadDemoCPUInteger. This java files contain two classes CalculationsDemo and ThreadDemoCPUInteger which extends threads to implement multithreads on CalculationsDemo.doCalculations() method.
  - doCalculations() method in CalculationsDemo class contains the code for performaing 7 instructions of add and sub.
  - Results are taken by calling at a time 1 thread then 2 threads and then finally 4 threads from the ThreadDemoCPUInteger class.
  - IOPS are calculated by the same formula as that of GFLOPS.
  - Perforamce results are shown in performance evaluation file.

- **Tradeoffs and Improvements**

2. Benchmarking for Disk:

- **Hardware Specifications:**

  Amazon EC2 t2.micro instance

- **Overview:**

- Benchmarking of disk involves calculating the throughput and latency for various disk operations like sequential read, write and random read, write.
- Performance of disk is calculated by invoking 1, 2 threads for various block sizes like 1 B, 1 KB and 1 MB for each sequential read, write and random read, write.
- Throughput is calculated by formula:

  Block-Size / Time-taken
- Disk Latency is the time delay between a request for data and the return of the data.
- To find the throughput and latency of sequential read a function named readFile is called. It reads a file of given block size by taking the block size as parameter. To sequentially read a file, File Channel and ByteBuffer APIs of java are used.
- To find the throughput and latency of sequential write a function named writeFile is called. It writes to a file a given block size data by taking the block size as parameter. To sequentially write to a file FileChannel, FileOutputStream and ByteBuffer APIs of java are used.
- To find the throughput and latency of random read a function named randReadFile is called. It reads a file of given block size by taking the block size as parameter. To randomly read a file, RandomAccessFile Channel API of java is used. Data is read randomly using the seek(pos) function. Here position is also given randomly using Math.Random() function.
- To find the throughput and latency of random write a function named randWriteFile is called. It writes to a file a given block size by taking the block size as parameter. To randomly write to a file, RandomAccessFile Channel API of java is used. Data is written at random positions by setting the pointer using the seek(pos) function. Here position is also given randomly using Math.Random() function.
- Each of these functions are called from another class which extends thread to implement 1 and 2 threads.

- **TradeOffs and Improvements:**

  Here in random read and write by calling Math.random() function causes some more time and hence shows a bit less throughput than expected. Hence by using some other technique the expected throughput can be obtained.

3. Benchmarking for Network:

- **Hardware Specification:**

  Amazon EC2 t2.micro instance.

- **Overview:**

- Here the task is to measure the throughput in Mbits/sec and latency in ms.
- There are two network connection protocols: 1) TCP and 2) UDP.
- 1) TCP: TCP is a connection-oriented protocol. In this Server receives a client's message and waits for other messages thereafter from client.
- 2) UDP: UDP is a connection-less protocol. In this Server terminates as soon as it receives one datagram packet from Client. For other packets it will have open connection every time.
- To calculate the throughput and latency various block sizes of 1 B, 1KB and 64 MB are taken.
- Following steps are performed for benchmarking network using TCP as well as UDP:
  i. A connection is created between server and client on a specified IP address and port no.
  ii. Client sends the request to the server for connection and server accepts his connection.
  iii. Client then sends a message to the Server (datagram packet in case of UDP) of a particular block size. Server replies with that message.
  iv. Time is noted when client sends a message to server and when it receives that message.
  v. Throughput is calculated by BlockSize/TimeTaken.in Mbits/sec
  vi. Network latency is the time measured by sending a packet that is returned to the sender; the round trip time is considered as latency.

- **Tradeoffs and Improvements:**

  Here the block size for each packet is hard coded for 1 B, 1 KB and 64 MB. This can be given dynamically.