Pooja Patel
A20396099

DESIGN DOCUMENT

PROGRAMMING ASSIGNMENT 01
CS 553 CLOUD COMPUTING

The following are the benchmarks which were asked for this assignment:

1. **CPU BENCHMARK**

- **Program Design:** For benchmarking CPU I have used C programming.
- The code is run for testing on chameleon cloud on the compute nodes.
- I have used matrix operations like adding some value to a 3*3 matrix and such 60 operations per iteration in each below functions
- I have created functions for each thread with its different precisions. Hence, I begin with function name "dp_4trd" , I have used this function for 4 threads and this particular function is for double precision, then after the function name "dp_2trd" for 2 threads and double precision , followed by "dp_1trd" for 1 thread and double precision , sp_4trd for 4 threads and single precision , "sp_2trd" for 2 threads and single precision , "sp_1trd" for 1 thread and single precision , "hp_4trd" for 4 threads and half precision , "hp_2trd" for 2 threads and half precision , "hp_1trd" for 1 thread and half precision , "qp_4trd" for 4 threads and quarter precision , "qp_2trd" for 2 threads and quarter precision , "qp_1trd" for 1 thread and quarter precision
- Also the number of iterations I have kept are 16666666667/ No. of threads. Here, the numerator is : Trillion operations / 60

Design Tradeoff and improvement:

- The usage or the utilization of CPU is increased by keeping the no. of iterations more and also each iteration has 60 operations to be perfomed
- To improve we can perform instead of matrix operations some other very complex operations like vectors and also linear equations

Pooja Patel
A20396099

2. MEMORY BENCHMARK

- **Program Design :** This is also done in C programming language
- The workload used is 1GB for different block sizes i.e. 1KB , 1MB, 10MB and have performed the benchmark for read+write for both random and sequential
- I have used one function mem_fun in which for both sequential and random read+write is performed.
- Firslty the given workload which is 1GB is divided by the size of the block to get the no. of blocks. Now this i.e. the number of blocks is divided by the no. of threads and hence each thread will have these many blocks to copy
- For sequential memcpy is used to copy from the source to destination.While for random read+write both memcpy and rand() function is used to copy from source to destination and generate the randomness repectively.
- The throughput which I got for the memory benchmark was measured in GBps

- For 1B of data we are supposed to calculate the latency and hence the latency is measured In microseconds

Design Tradeoff and improvement:

- The improvement can be that we can have used the work load of 1GB and hence we can benchmark by increasing the workload size

3. DISK BENCHMARK

- **Program Design :** For benchmarking disk I have used C language
- I have calculated throughput for block size 1MB, 10 MB, 100 MB and using 1,2,4 threads

- Also the experiment was performed both for random and sequential access patterns such as read random, write random, read sequential , write sequential
- The unit in which the throughputs were calculated was in MB/s and that of latency in ms
- I have made disk_fun which is a function in my disk benchmark code where I have written code for all access patterns
- Also for latency for 1KB blocksize for both RR and WR and for 1,2,4,8,16,32,64,128 threads are calculated in this function and latency I have calculated as the time taken * 1000 /1 million.
- The output file of the disk benchmark is generated in the output folder and also they are saved in different .out files
- The throughput was calculated using 10 GB of the data
- For disk latency the operations were limited to one million operations
- The logic which I have used is I have allocated 10GB and then I have calculated the no. of blocks which is 10GB / size of each blocks now we also have different number of threads and hence the no. of blocks each thread gets is calculated as no. of blocks / no. of threads and for loops runs for this many times (i.e. no. of block each threads gets) with fread in it for “RR” ,“RS” and
  fwrite for “WR” and “WS”
- For latency the for loop runs for 1million operations / no. of blocks per thread
- Also I have made the 10 GB file in /tmp and after the disk benchmark gets executed then I am deleting the 10 GB file

Design Tradeoff and improvement:

- For increasing the efficiency I could have first written in the disk and the performed all the read operation

4. NETWORK BENCHMARK

- This benchmark can be properly implemented in java programming language.
- For us the functionality of client and server are in the same program with both client and server having multi-threaded functionality

- The protocols which are used are UDP and TCP and also they are operated for hundred times over different sizes of the block such as 1KB , 32 KB
- The no. of threads used are 1,2,4,8 and the throughput is calculated in Mb/s and have to send 100 GB of data from client to server
- For latency have to ping pong 1 byte of data and have to calculate latency in 'ms' and have to performed for over 1 million times

Design Tradeoff and improvement:

- The improvement can be that the size of the buffer can be increased
- To increase the efficiency the packet loss can be more taken care of