Benchmarking Design Document

Assignment #1

Benchmarking Process:

CPU Benchmarking

- FLOPS Computation
- IOPS Computation
- Sampling FLOPS counts on regular interval for longer period
- Sampling IOPS counts on regular interval for longer period

Memory Benchmarking

Disk Benchmarking

- Write + Read
- Random Read
- Sequential Read

Network Benchmarking

- TCP – Latency and Throughput
- UDP - Latency and Throughput

Structure of coding

- This assignment is done in C Programming language
- This assignment consists of 6 C programming files
  - CPU Benchmarking
    - cpu.c
  - Network Benchmarking
    - server.c
    - client.c
  - Memory Benchmarking
    - memory.c
  - Disk benchmarking
    - diskreadwrite.c
    - diskread.c
- Each code will have 2 functions
  - One containing the benchmarking logic
  - Another one to create threads and print the output. This function will call the benchmarking logic on to the threads to produce the output
- The time duration is calculated using the "Struct timeval" structure variables

CPU Benchmarking

- The loop is defined as 10^9
- Formula used for Giga FLOPS/IOPS computation is

  (Total Operations done/(Total time taken*1000000000))

- The computations are done and the parameters are stored in an array.
- This will be the same for 1/2/4/8 threads. The next function will sleep for the sampling interval and increments the counter variable. The thread function will store the data retrieved.
- Formula for final FLOPS/IOPS count for each sampling interval is

  (Total operations done during sampling interval of all 8 threads / (Average of exact sampling time for all 8 threads*1000000000))

Memory Benchmarking:

In Memory benchmarking Throughput and Latency is calculated. They are computes using memcopy and memset functions two character pointers are moved around to perform the sequential and random operations. A block of memory (100mb) is allocated and initialized.

Total no of block= Memory/block size

Block size are 8Bytes, 8KB and 80 MB

Latency =Total time taken*1000/total used memory

Throughput= Total memory used/total time taken *1024 * 1024.

In case of multithreading:

Throughput =Total memory used by all threads/ average time taken across *1024 *1024

Latency= average time taken by all threads *1000/ total memory used by all threads.

Memcopy functions read the contents from the source point and writes in the destination.

Latency is calculated using memset function with different block size

Memory bench marking is done using sequential and random read operations with 1,4,8,10 threads.

Disk Benchmarking:

This is almost similar to memory benchmarking. API's like read, open, write are used to work on the disk. Similar to memory a large block of memory is assigned with random characters. This created file is used as a source of read and write operations in sequential and random manner.

Block size considered are 8bytes,8kb,8mb

The following operations are carried out.

- Sequential read
- Sequential write
- Random read
- Random write

No of blocks: total memory/block size

Block size are 8Bytes, 8KB and 80 MB

Formula for throughput : Total memory/total time *1024*1024

Latency=Total time*1000/Total memory read/written

Networking Benchmarking:

Client.c

- The loop is defined as 100
- The thread is defined using the struct followed by the function of TCP/UDP Client which include the sending and receiving of data, parameters from the input, define and initialize the socket related parameters, Inititalize threads and assign the related parameters.
- The start time and end time are defined using struct timeval
- The execution of multi-threads communication is done in loop
- The formula to calculate execute time is
    - (1000.0 * (end_time.tv_sec - start_time.tv_sec) + (end_time.tv_usec - start_time.tv_usec) / 1000.0)
- The formula to calculate throughput is
    - (num_thr * LOOPS * buffer_size / (1024.0 * 1024.0)) / (execute_time / 1000.0)
- The formula to calculate latency is
    - execute_time / (num_thr * LOOPS * buffer_size)

Server.c

- The loop is defined as 100
- Function for receiving data will be created for TCP and UDP where the socket will be created and binded.
- Creation of threads to process the communication is done
- The main function to select the protocol according to the input is mentioned.

Tradeoff

- The log files are not generated to keep the track of the benchmarking files
- The program can be optimized even more with global standards and all these benchmarks can be made into a single benchmarking to call all the functions- for a better execution

CPU:

- Since there was a difficulty in understanding the LINPACK benchmarking, my program's performance is no match for the LINPACK Output
- Experiment can be tested with more number of threads in different instances to attain best results
- Better results are achieveable if the logic is even more understood with practical knowledge in this field.

Networking:

- The performance can be achieved in a better hardware
- More buffer size can be provided to the client and server to get more latency and throughput data

Memory:

- Difficulty in understanding the programmatically approach of the memory.
- This will affect the coding logics and standards

Disk:

During the sequential read and write, cache used is more of a memory benchmarking than the disk benchmarking.

Improvements that can be done:

- The Threads/Block etc and all the other parameters can be coded to get it as user input
- Experiments can give us better results if tested with more number of threads
- On improved understanding LINPACK even more better results can be obtained
- Testing of memory throughput and latency can be done by not involving cache memory
- Testing of Disk throughput and latency can be done by not involving cache memory

Contribution:

The CPU and Network benchmarking are done by Sivasenthil Namachivayan (A20391478)

The memory and Disk benchmarking are done by Abinaya Janakan (A20376287)