CS553: Cloud Computing

Design Document

Benchmarking

There are 3 Benchmarks

- CPU Benchmarking
- Memory Benchmarking
- Disk Benchmarking

CPU Benchmarking:

- Used C programming language for CPU Benchmark implementation.
- In order to test the performance of CPU we have 2 criteria. One Integer Operation (GIOPS) and second Floating Operations (GFLOPS).
- To get the maximum number of operation CPU can perform, we have used threading in our experiment where we have multiple thread performing Integer or Floating operations. We have Calculated GLFOPS and GIPOS by varying concurrency of 1 Thread, 2 Threads and 4 Threads.
- We have performed Integer and floating operation in threadOperation() method, In this method we have looped the 20 arithmetic operations for (10^9) times.
- There are 4 different files each for calculating GFLOPS, GIOPS, GFLOPS (10 Minute sample outputs) and GILOPS (10 Minute sample outputs), which are CPUBenchmarkF.c, CPUBenchmarkI.c, BenchmarkF.c and BenchmarkI.c respectively.

Memory Benchmarking:

- Used C programming language for Memory Benchmark implementation.
- We conducted our experiment for Memory Benchmark on 3 different Data size/Block sizes 1 Byte, 1 Kilo Byte, 1 Mega Byte.
- We have calculated the total time for execution along with throughput and latency for Sequential Memory read and Memory write operation and similarly for read and write Memory operation.
- We have used the memcpy() method to read and write from Memory.
- This experiment has been performed by varying concurrency of 1 Thread and 2 Threads.
- We have to pass the number of threads as a command line argument for this experiment. There is only file named MemoryBenchmark.c
- Calculated Throughput and Latency will get generate in Memory_log.txt file on same location where your code is saved.
- Throughput has been calculated MB/sec and Latency has been calculated in milliseconds.

```
// this is to Read+Write a Mega byte data sequencially from Memory
void* readSeqMByte(void *arg)
{
	int i;
	int source_buffer_index,destination_buffer_index;
	source_buffer_index=0;
		destination_buffer_index=0;

	for(i=0;i<100;i++) // 100mb data
	{
		memcpy(&destination_buffer[destination_buffer_index],
&source_buffer[source_buffer_index], (1024*1024));
		source_buffer_index= source_buffer_index +(1024*1024);
		destination_buffer_index=destination_buffer_index+(1024*1024);
	}
	return NULL;
	}
```

Disk Benchmarking:

- Used C programming language for Disk Benchmark implementation.
- We conducted our experiment for Disk Benchmark on 3 different Data size/Block sizes 1 Byte, 1 Kilo Byte, 1 Mega Byte.
- We have calculated the total time for execution along with throughput and latency for Sequential Disk read and Disk write operation and similarly for read and write Disk operation.
- We have used the fprintf() and fread() method to write & read from Memory.
- This experiment has been performed by varying concurrency of 1 Thread and 2 Threads.
- We have to pass the number of threads as a command line argument for this experiment. There are different files like DiskBenchmarkR.c for Read operations and DiskBenchmarkW.c for write operations.
- Calculated Throughput and Latency will get generate in Disk_log.txt file on same location where your code is saved. Similarly Read and Write operation will get performed on DiskWrite.txt file saved on same location your source code.
- Throughput has been calculated MB/sec and Latency has been calculated in milliseconds.

```
// this is to Read a byte data sequencially from disk
void* readSeqByte(void *arg)
{
	FILE *fp;
	int i;
	fp=fopen("DiskWrite.txt", "r");
	char buffer[1024*1024];
	fseek(fp,0,SEEK_SET);
	for(i=0;i<100000000;i++) // 100mb file
	{
		fread(buffer,1,1,fp);
	}
	fclose(fp);
}
```

Possible improvements and extensions:

- CPU Benchmark – In CPU Benchmark we have to calculate the number of arithmetic operations CPU can perform. But there are some programming language barriers which cost more CPU cycles like in C or JAVA for loop has its own cost involve. So possible improvement is could be we have to remove the time factor/Cost of for loop.
- Memory Benchmark – In Memory Benchmark we have to calculate the Throughput and Latency for Read and Write Operations from memory only. Here we have problem to deal with is cache memory (L1, L2 and L3). So each time you run your experiment might get different results. So the possible improvement is we have to tackle the cache memory.
- Disk Benchmark – In Disk Benchmark we have to calculate the Throughput and Latency for Read and Write Operations from Disk only. Disk might contain the bad sectors which cost more Latency. We have to perform more number of operations in order get the reading close to actual value.