Design Document

This is a document which tries to explain the methodologies used for benchmarking various components of a computing system. First, let us look at the CPU benchmark.

CPU benchmarking

There are eight C programs benchmark CPU. Each of these programs runs a test on CPU with inducing different kinds of stress on the CPU and measuring how it fares in different situations. All of these programs contain one single principle and it is a modified implementation of 2-dimensional matrix multiplication. The modification is that instead of just a simple multiplication operation done inside the nested loop it does more arithmetic operations. This method is used to avoid the potential optimization done by the compiler or the OS. This loop is repeated 3 times which involves 2 other matrices so as to avoid pipelining of the operation execution which can be detrimental to the performance. The highest performance is recorded by 1 thread floating point operation.

Now when a Single thread is executed the program calls a function that does the aforementioned function and records the time to execute the operations for 10000000 times. Once this is recorded the GFLOPS is calculated using the formula (number of flops) = (LoopCounter * NumberOfOperations * numberOfThreads)/(TimeTaken * 1000000000), this will give us a count in terms of GFLOPS. Then for multiple threads, the function which is responsible for the core operations is executed by different threads and a total time for executing the code by all the threads is calculated and then the above formula is applied to extract the GFLOP values.

The “e” part of the question requires the test to run for a total time of 10 minutes and extract the sample reading of the GFLOPS every second. This is achieved by the program by introducing another thread along with the threads running the core operations which keeps track of the time and wakes up every minute to take the readings of all the threads executing the core function and resets their respective counter and sleeps for another second. Once 10 mins are completed then the time thread sets a global flag to true which will, in turn, make all the other threads to stop their execution and return the control to the main.

Scope for improvement

Core function can be made more complex by performing heavy computations like differential equations etc.

Memory Benchmarking

There are 12 C Programs that collectively perform this benchmark. Each of these programs runs a test on the memory module which tries to apply different kinds of stress on it and measuring how it fares in different situations. All of the above operations have one single primary principle which is the implementation of a function which basically copies different sizes of data from one block of memory to another block both residing in the main memory.

The program is executed with varying number of threads and varying buffer sizes. Since the byte size copying is too small a buffer size gives a bad performance. According to the architecture of the t2.micro instance, 1KB is the most efficient when compared to the theoretical value.

There could be one major flaw in the program that gives cache speeds rather than the main memory speeds. This problem is handled by creating a chunk size which is too big to fit in the L1 L2 or L3 caches.

Disk Benchmarking

There are 2 programs to execute Disk benchmark. Each of these programs runs a test on Disk module with inducing different kinds of read/write operations on it and measuring how it fares in different situations. The basic principle this involves is that it runs fread() and fwrite() functions to the disk.

The program is executed with varying number of threads and varying buffer sizes. Since the byte size copying is too small a buffer size gives a bad performance. According to the architecture of the t2.micro instance, 1MB is most efficient when compared to the theoretical value.

The design or the flow of the program is each of the files executes the same functions with a varying number of threads. So the file starts off with creating the threads and creating a 10MB file for reading and writing and starting the functions inside which execution time is recorded and finally, the latency and throughput are calculated.

The problem of as file being accessed from the main memory is avoided by creating a new file every time before the next type of test is performed.

Scope of improvement

Run benchmark test based on the type of hard disk being used as. As SSD and DISK have totally different operations and hence different latencies for access based on the way the data is being accessed.