Benchmark different parts of a computer system

Veereshwaran Rangasamy Chettiar Ramamoorthi

vrangasa@hawk.iit.edu

Abstract: A benchmark is a standard or point of reference against which things may be compared or assessed. This helps in knowing the maximum practical performance of a system. In this, three different parts of a computer such as CPU, Memory, Disk are being benchmarked.

1. Introduction

1.1 CPU Benchmarking

A central processing unit (CPU) is the electronic circuitry within a computer that carries out the instructions of a computer program by performing the basic arithmetic, logical, control and input/output (I/O) operations specified by the instructions.

Here benchmarking is done for two types of instructions. They are Floating-point Operations Per Second (in GFlops) and Integer Operations Per Second (in GIops).

CPU benchmarking program is designed for measuring GFlops and GIops of the CPU of Amazon EC2. The benchmarking program is developed in Java, in which threads are being implemented for measuring GFlops and GIops separately. ExecutorService API is being used for handling threads.

GFlops is computed by performing floating point operations for a 2^{31}-1 times for x seconds and then it is calculated as $[(2^{31}-1)/x]*1e-9$ GFlops.

GIops is computed by performing integer operations for a 2^{31}-1 times for x seconds and then it is calculated as $[(2^{31}-1)/x]*1e-9$ GIops.

GFlops and GIops are being calculated by running single thread, double thread and four threads.

Improvements: Thread handling and some other processes will consume CPU which impacts on GFlops and GIops of a CPU. So this consumption of CPU should be brought down.

1.2 Memory Benchmarking

Memory is the electronic holding place for instructions and data that the computer's microprocessor can reach quickly. When the computer is in normal operation, its memory usually contains the main parts of the operating system and some or all of the application programs and related data that are being used.

Here benchmarking is done for measuring throughput (in MB/s) as well as latency (in ms) of the memory.

Memory benchmarking program is designed for accessing memory in a sequential and random manner by performing “read+write” operation with varying block sizes (1B, 1KB, 1MB) using single thread and double threads. The benchmarking program is developed in C, in which threads are being implemented for measuring throughput and latency of memory.

Throughput is being computed by performing two steps:

1. Allocating a character array (say, *original*) of size X and then copying characters into that character array.
   Allocating another character array (say, *copy*)
2. And then copying *original* character array to *copy* character array.
   During this action, a timer is being run for calculating the time taken for reading *original* and writing into *copy*.

The above is done in a sequential manner. The same is implemented in a random manner by accessing random address calculated by rand()%blocksize.

Throughput is computed by accessing *X* bytes of memory as *Y* block in *A* time and then it is calculated as *X/A* MB/s.

Latency for *Y* block of memory is the time taken for accessing *Y* blocks of memory.

Improvements: Random accessing needs some random memory address calculation, which is also included in throughput and latency calculation. Implementation can be done in such a way to eliminate this overhead.

1.3 Disk Benchmarking

A hard disk drive (HDD), hard disk is a data storage device used for storing and retrieving digital information using one or more rigid ("hard") rapidly rotating disks (platters).

Here benchmarking is done for measuring throughput (in MB/s) as well as latency (in ms) of the disk.

Memory benchmarking program is designed for accessing memory in a sequential and random manner by performing read, write operation with varying block sizes (1B, 1KB, 1MB) using single thread and double threads. The benchmarking program is developed in Java, in which threads are being implemented for measuring throughput and latency of disk. ExecutorService API is being used for handling threads.

Throughput is being computed by performing the following steps:

1. Using Java's RandomAccessFile API and FileChannel API, bytes are written to a file on the disk in a sequential manner and then the written bytes are read from the file. Separate timers are being set for measuring the time taken for reading bytes and writing bytes.
2. The above is done in a sequential manner. The same is implemented in a random manner by accessing starting position of a file and ending position of file in an incremental manner.

Throughput is computed by accessing *X* bytes of file as *Y* bytes in *A* time and then it is calculated as *X/A* MB/s.

Latency for *Y* block of memory is the time taken for accessing *Y* bytes from disk.

Improvements: Random accessing needs some random memory address calculation, which is also included in throughput and latency calculation. Implementation can be done in such a way to eliminate this overhead.

2. Conclusion

The above experiments gave the benchmark result of CPU, memory, disk and those are being recorded for evaluation.