Design Document

Benchmarking the Amazon EC2 instance.

Pavankumar Shetty
CWID-A20354961
CS-553 Spring 2016
Main Campus

Table of Contents

Introduction

This project aims to teach you how to benchmark different parts of a computer system, from the CPU, memory, disk, and network.

Here I tried to benchmark the Amazon EC2 instance. (t2. micro instance)

Overview and Key points:

- A measure of the speed of a computer in operations per second, especially arithmetic operations involving floating-point numbers and input output operations per second, which are termed as **IOPS** and **FLOPS**. These are usually measured in GIGAFLOPS and GIGAIOPS.
- The time period between a request for a thing to perform an action and the action being carried out is termed as **LATENCY**. Usually measure in mille-seconds or seconds
- Output or production, as of a computer program, over a period of time is usually termed as **THROUGHPUT.** In this experiment it is measured as Megabytes per second.
- CPU benchmarking is done in terms of IOPS and FLOPS.
- Memory and Disk benchmarking is measured in terms of Latency and throughput.

Language & Platform

Language used:

CPU: C++ programming, which covers Multithreading (pthread library)
MEMORY: C++ programming, which covers Multithreading (pthread library).
DISK: JAVA programming, which covers Multithreading.

Platform: Built on Amazon EC2 Linux environment.
Instance type used: t2. micro

Design Principle

1. CPU

Performed experiments to benchmark the CPU of the amazon EC2 t2. micro instances. Designed in C++. The experiment covered the following key points.

1. Have designed two sets of function namely, **void* iops_benchmarking(void *) and void* flops_benchmarking(void *)** to allow the processor run multiple instructions per cycle concurrently. And used **gettimeofday()** wall clock to calculate the time elapsed to perform the set of operations in these functions and calculated the IOPS and FLOPS.
2. Used **pthread** library to implement threading for measuring the processor speed at different level of concurrency. I ran the function with 1, 2 and 4 threads and recorded each values and computed the IOPS and FLOPS out of it.
3. Based on piazza discussions and links within it, have calculated the theoretical peak performance of the processor in flops/sec.
   CPU => GHz * IPC * no of Cores => 2.5GHz * 8 * 1 => 20 GFlops.
4. As a separate experiment, ran the benchmark on floating and integer instructions and 4 threads for a 10-minute period using a separate thread just to keep track of the number of operations achieved per second and saved it accordingly to a file and plotted a graph of these 600 samples.
5. Ran the **Linpack** benchmark to report the best performance achieved and compared with the theoretical and the performance achieved by me.

2. MEMORY

Performed experiments to benchmark the MEMORY of the amazon EC2 t2. micro instances. Opted designing in C++ over java, to avoid the virtual memory conflict. The experiment covered the following key points.

1. Have designed two sets of function namely, **void *sequential_memory(void*) and void *random_memory(void*)** to allow the processor perform **read+write** operations **randomly and sequentially** in the memory. And used **gettimeofday()** wall clock to calculate the time elapsed to perform the set of operations in these functions.
2. Used **pthread** library to implement threading for measuring the memory speed at different level of concurrency. I ran the code with 1 and 2 threads as per experiment requirement. I used different block sizes viz., 1B, 1KB, 1MB.

3. Major metrics calculated out of these experiments were throughput(MBps) and latency (in msecs).
4. Based on piazza discussions and links within it, have calculated the theoretical memory bandwidth of the memory.
5. Ran the **Stream** benchmark to report the best performance achieved and compared with the theoretical and the performance achieved by me.

3. DISK

Performed experiments to benchmark the DISK of the amazon EC2 t2. micro instances. Designed in Java. The experiment covered the following key points.

6. Have designed four sets of function namely, **sequentialRead(), sequentialWrite(), randomRead() and randomWrite()** to allow the processor perform **read+write** operations **randomly and sequentially** in the DISK. And used **System.nanoTime()** to calculate the time elapsed to perform the set of operations in these functions.
   Used **RandomAccessFile API** for accessing the disk in random way.
   Used **FileInputStream** and **DataOutputStream** and **ByteArray**
7. Used **Thread and Runnable Interface** library to implement threading for measuring the disk speed at different level of concurrency. I ran the code with 1 and 2 threads as per experiment requirement. I used different block sizes viz., 1B, 1KB, 1MB.
8. Major metrics calculated out of these experiments were throughput(MBps) and latency (in msecs).
9. Based on piazza discussions and links within it, have calculated the theoretical memory bandwidth of the memory.
10. Ran the **IOZone** benchmark to report the best performance achieved and compared with the theoretical and the performance achieved by me.