INDEX

CPU Performance Benchmarking

- **Program Design**

  The experiment is implemented into two parts:

  1. Calculate FLOPS / IOSP with multi-threading
  2. Sample data for each time interval

  1. Calculate FLOPS / IOSP with multi-threading

  This part is implemented with 1, 2 and, 4 threads by running 16-18 floating / integer operations in a loop. Loop is repeated around 1 billion times.

  Formula to calculate FLOPS / IOPS:

$$\text{FLOPS / IOPS} = \frac{Total\ Number\ of\ Operations}{Time\ in\ Sec}$$

  2. Sample data for each time interval

  In this part, the loops run about 10 minutes to get number of operations performed in each seconds. This experiment is implemented by multi-threading with 4 threads. There is one timer running in main thread, which trigger event to get number of operation executed by each thread and save into an array for further. For example to draw graph and evaluate.

- **Trade offs**
  - Results are not as perfect as highly optimized benchmark algorithms like linpack.

- **Possible Improvements**
  - Data stored can be stored in form of structure. And will be easy to import into any file for further usage.
  - Optimized performance can be achieved by understanding CPU architecture and also how other benchmarking systems are doing.

Memory Performance Benchmarking

- **Program Design**
  This experiment is implemented with the help of functions like memcpy and memset. A large data is allocated into memory with pre-populated data.

  To measure memory performance, block size is varying between 1 Byte, 1 KB, and 1 MB by using 1 or 2 threads each sequential and random combination.

  Functions memcpy and memset are used to get throughput and latency.
  - **memcpy**() function copies *n* bytes from source to destinations
  - **memset()** function sets the first *n* bytes of the block of memory pointed by *ptr* to the specified *value*

  Formulas to calculate throughput and latency:

$$\text{Throughput} = \frac{Total\ Data\ accessed\ in\ MB}{Time\ in\ Sec} \quad \text{MB/Sec}$$

$$\text{Latency} = \frac{Time\ to\ in\ Milliseconds}{Total\ datat\ in\ MB} \quad \text{Milliseconds}$$

- **Possible Improvements**
  - Avoid caching to get more accurate results.

Disk Performance Benchmarking

- **Program Design**

  This experiment is implemented with help of POSIX file operation function in C like fopen, fread, fwite, fseek, fclose. This experiment has four parts –

  1. read data in sequential manner
  2. write data in sequential manner
  3. read data in random manner
  4. write data in random manner

  A large 1.5 GB file is created, before execution of program with pre-populated data for read and write operation. Use of large file helped to avoid caching file into memory.

  Disk Benchmarking is implemented with 1 and 2 threads for data size of 1 Byte, 1 KB and, 1 MB.

- **Trade offs**
  - Results are not as perfect as highly optimized benchmarking algorithms like iozone.

- **Possible Improvements**
  - Test with various data size to improve performance.
  - Avoiding memory caching to get consistent result.