CS553 Cloud Computing

Report

Homework 3 Team 11

Varun Shanbhag(A20452310), Tushar Nitave(A20444211), Talwinder Singh(A20448598)

MyDiskBench

Introduction

In this assignment we will benchmark storage systems using MyDiskBench and Iozone. We have implemented this in C++. We will be benchmarking using four different access patterns namely:

- Write Sequential
- Read Sequential
- Write Random
- Read Sequential

We will using different configurations of ***threads, record size*** and ***workload*** of ***10GB*** to benchmark using these four access patterns.

Benchmark Design

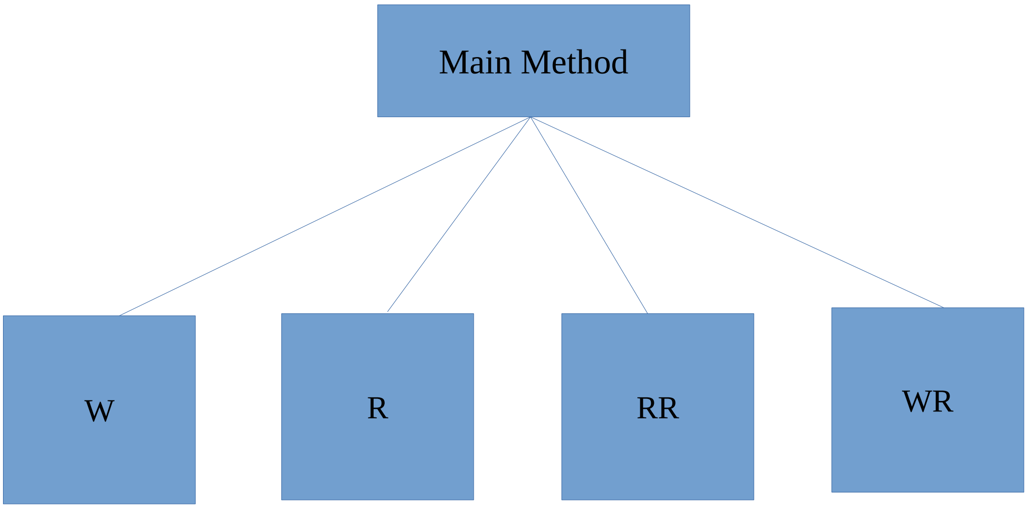

Fig 1.

Write Sequentially(W) : It Opens a file using open/2 call and opens file in O_DIRECT mode whic provides us with flexibltiy to write code directly onto disk based on given block size.

Read Sequentially(R) : It Opens a file written by write call using open/2 call and opens file in O_DIRECT mode whic provides us with flexibltiy to read code directly from disk based on given block size bypassing the cache.

Write Random(WR) : It Opens a file using open/2 call and opens file in O_DSYNC mode whic provides us with flexibltiy to write code directly onto disk based on given block size. Moreover it seeks a random point on disk within file size limit and write the record.

Read Random(RR) : It Opens a file written by write call (W) using open/2 call and opens file in O_DIRECT mode whic provides us with flexibltiy to read code directly from disk based on given block size bypassing the cache. It uses lseek function to read from a random location within file size.

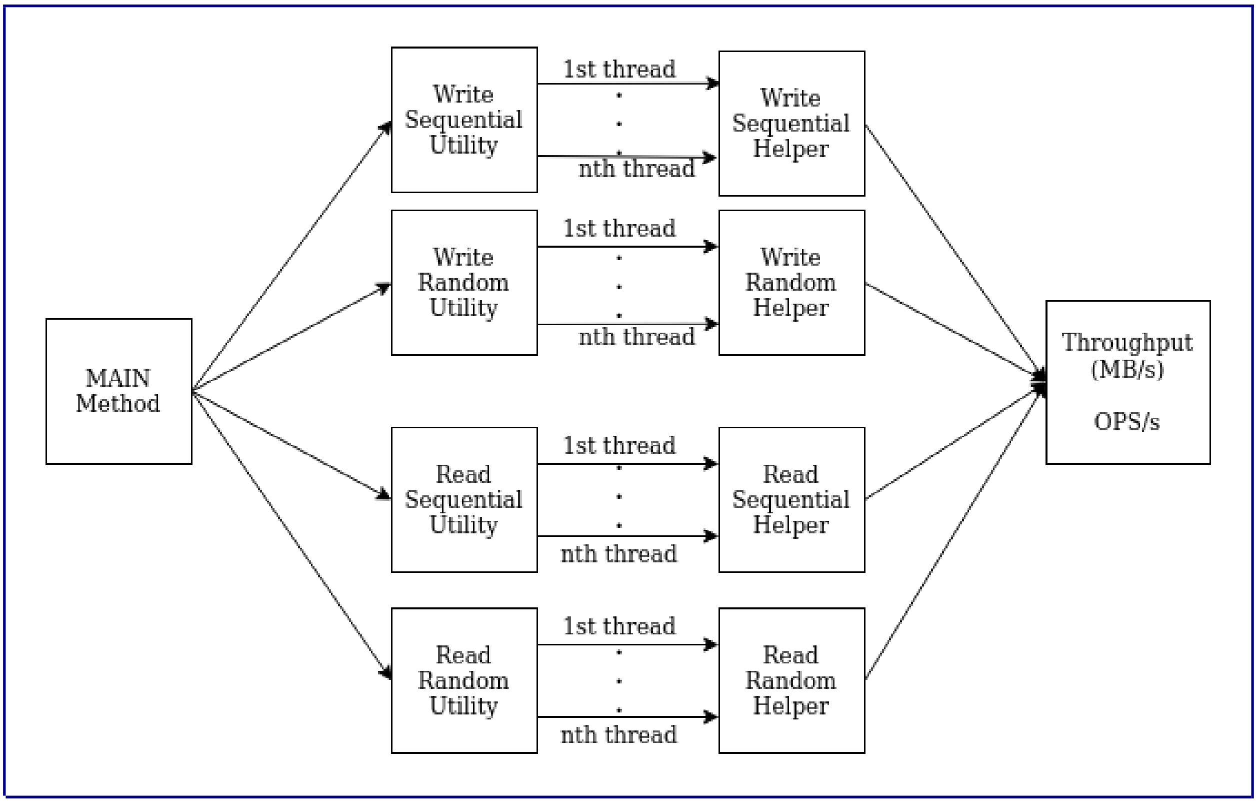

Fig.2. Program Architecture

As shown in the **Fig. 2** after running the command, main method will invoke one of the four utility methods – write sequential (WS), read sequential (RS), write random (WR), read random (RR) based on the input configuration. After this these utiltiy methods spawns N number of threads (N is number of files) which call corresponding helper method to perform the desired operation. Helper methods returns **Throughput in MB/sec** and **IOPS in OPS/sec**.

We are using **pthread** library known as POSIX threads to perform read/write operation, **O_DIRECT** flag to minimize the cache effect so that we can benchmark disk not the memory.

Design Tradeoffs

- Its advisable not to use **O_DIRECT** as it requires strict memory allignment corresponding to blocksize. If at all any record size is not divisible by disk's block size segmentation fault will happen.
- Total workload is **10GB** and cannot be changed (As workload size is fixed from this assignment point of view)
- Usage of system calls like **open/2 write/2 read/2 lseek/2** can show varied behaviour on OS which is not unix based.(However this call should work fine in any unix based systems).
- Code is not modularized and written in a single file.

Improvements

- More options can be added like (file persistance,dynamic file size defination) --> these can be acheived by adding more flags to existing program.
- UI/UX can be included (Using Python or any frontend tech)
- Provide with a provision to plot graphs if multiple tests are run using Matplotlib.
- Cross platform compatibility using libraries supported on Windows and Mac.