STEP : 1 – DOWNLOAD AND INSTALL PMBENCH

Use the following commands to install the pmbench tool from command line

- apt update
- apt install make gcc
- cd home
- git clone https://bitbucket.org/jisooy/pmbench.git
- unzip jisooy-pmbench-(enter_your_version).zip
- cd jisooy-pmbench-(enter_your_version)
- make

Note: Ignore the mingw32 error during make and rename the directory to "pmbench" for runpmbench.sh to work

STEP : 2 – SETUP BENCHMARK ENVIRONMENT

In the setup.sh script:

- Add the drives to be partitioned in "drivelist" array under Initial parameters
- Add the drives to be tuned in "tunedrives" array under Initial parameters
- The script tunes the performance of drives according to Linux performance evaluation guide. To change the defaults edit the swap specific settings section of the script
- After making necessary changes, execute the setup.sh script to setup the benchmark environment
- The script has capability to generate 28 swap partitions. If your kernel doesn't allow 28 swap partitions, the script can error about partitions not created. Check lsblk to see if other partitions were created successfully.

STEP : 3 – RUN THE BENCHMARK

In the runpmbench.sh script:

- Update the test size, read write ratio, number of CPU's/threads, duration of test under General Settings section as required.
- Execute the runpmbench.sh script to run the workload (Note: make sure the runpmbench.sh is under the extracted pmbench directory so that the script has access to pmbench executables)

STEP : 4 – RESULT ANALYSIS

- In ubuntu_results.py script update the path to the directory in which log files of benchmark are located.
- Execute command: python ubuntu_results.py to show the pages accessed per second on each test case
- To generate an access latency histogram update Ubuntu_histogram.py with the following:
    i. The path to the directory in which log files of benchmark are located.
    ii. Log files of the run
    iii. Read ratios and thread counts (in the same order as mentioned in runpmbench.sh)

iv. Execute command: python ubuntu_histogram.py. This generates different files for each workload type. This contributes to the y-axis (pages accessed) of histogram.

v. The x-axis of the histogram is time in microseconds and contains values in table below (the values below are calculated from the log file generated by benchmark).

0.256
0.512
1.024
2.048
4.096
8.192
16.384
32.768
65.536
131.072
262.144
524.288
1048.576
2097.152
4194.304
8388.608
16777.22
33554.43
67108.86
134217.7
268435.5
536870.9
1073742
4294967