Contents

[Stuff that you Need 1](#_Toc57904718)

[Files that need to be on the server 1](#_Toc57904719)

[Files needed to view the results in graphs 2](#_Toc57904720)

Things you need to know to use the benchmarking scripts :

The basic idea of the set of benchmarking scripts is to run a large number of fio tests, so tests can be run outside of work hours.

Fio (flexible i/o tester) is a program for testing i/o performance that is run in the terminal. There are a wide range of options that can be used in fio to alter how the i/o performance is tested, for example direct i/o which bypasses the RAM cache, the type of i/o (reads, writes, random reads, random writes, or a mixture of those), the block size and file size, what drives to read and write from, and the list goes on.

By hardcoding a relatively small number of variables in a python file, then repetitively generating files with all the desired combinations of these variables, a large number of tests can be created. The files containing combinations of the variables are called job files, and they are used by fio to run the tests that you want.

# Stuff that you Need

There are a set of files that need to be on the server to run the tests and a set of files that need to be on a machine that has a monitor to view the results of the tests.

## Files that need to be on the server

To run a benchmark test using the scripts, you will need to have a directory containing the following:

* benchMKscripVars.py, a file containing the order in which drives are tested, and the global sections for the job files that fio uses to run the tests.
* Make384JobFiles, a python script which generates the job files for all the fio tests that will be run, puts the job files in a directory (named using the type of i/o and the order of drives to be tested), opens a file to write a bash script, and writes a shebang and fio commands in the bash script
* createBenchMKresults, a python script which looks through all the output files from the fio tests and takes the bandwidth and IOPs results from those files.
* runTests, a bash script which makes sure the execute permission is granted for the above scripts, then runs them in sequence. Everything is inside a bash script because it is easier to ensure that the each fio test waits for the previous one to finish before starting a new test. There may be a simple way to do this in python using the subprocess module as well if need be in the future.

## Files needed to view the results in graphs

To graph the results, there are some files that need to be in the same directory on a machine that has a monitor.

* GraphResults.py, a script which reads the results from BenchMKresults and generates a directory for all the automatically saved graphs and textual results to reside in, sorted into sub directories hierarchically. The top level directory is AutoSavedResults, the next level down contains all the results for a test between changing harware and is named using the hardware setup: mobo, CPU, HBA card (identified as reported by lspci | grep SAS), the PCIe slots that are in use, and the sum of the RAM available to the CPU, in GB. The third level contains directories for every combination of i/o and drive order tested. The fourth level contains a directory for reads or writes (more than one sub directory only exists if the i/o type was varied). The fifth and final level contains the graphs (.png) and the coordinates. Coordinates are formatted as a csv with tabs as the delimiter, order of data goes number of drives in the test, IOPs (operations/s), then bandwidth (MB/s). The program saves the BenchMKresults file for the test in a directory called BenchMKresultsArchive. The BenchMKresults file is copied into the
* benchMKscripVars.py, a file containing the order in which drives are tested, and the global sections for the job files that fio uses to run the tests. Yes, it is needed on both machines

If the server that the benchmark test is being run on is a hybrid, mapping the drives may not be as straightforward as running dmap or hmap. One can use a helpful script called customConf to make the contents of vdev\_id.conf work for one's hardware setup. If hardware is changed and upon rebooting the server only the drives on a certain range of HBA cards are supposed to be tested, instead of mapping all the drives, the customConf script can be used to map only the desired drives. This is particularly useful if the test is for the purpose of comparing an old setup on a specific server model to a new one because a hybrid server can be used as that specific model: a single hybrid chassis with 60 HDD bays and 32 SSD bays can be set up as an 15 to 60 bay HDD server or a 32 drive SSD server, or any other hybrid server.

- drives mapped to have those symbolic links to use

- matplotlib

- Stuff you need to know:

-what the scripts do, at a high level, and thus what the scripts don’t do (state that other uses might break things)

-making a vdev\_id.conf using customConf

-what the benchMKscriptVars.py file is for

-what set of directories and files are created by running ./runTests and ./GraphResults

Things to know about the tests:

-test cases

-how fio is used for these tests

Things to know about bottlenecks:

-throughput of components

-possible bottlenecks

-how things were made to go fast