PURESTORAGE

AIRI POC Test Kit

V1.1 - July, 2018

Networking tests

1. airi-iozone
2. airi-libnfs

airi-iozone

Overview

Iozone is a tool for simple IO profiling of a filesystem. With **containerized iozone**, you can maximize the throughput of a single client by using docker to spawn multiple TCP connections. Each docker container will mount the NFS share from **inside** the container, which results in a new TCP connection.

Contents:

/nw-test-tools/airi-iozone/

- Dockerfile
- run_iozone.sh

Step 1: Build the iozone Dockerimage on a DGX-1

Build the above Dockerfile with the following command:

```
$ cd airi-iozone/
$ sudo docker build -t "iozone" .
```

✔ Check Step 1:

```
$ docker images | grep "iozone"
```

should return your newly-created container

Step 2: Execute the run_iozone.sh script on a DGX-1

This script requires 5 input variables:

<#-clients> <io-size> <file-size> <FlashBlade-data-VIP> <filesystem>

For example:

```
$ ./run_iozone.sh 80 512K 4G 198.18.0.100 datasets
```

✔ Check Step 2:

The FlashBlade GUI should show start showing the four test workloads that this script runs:

1. initial writes, 2. reads, 3. random reads, 4. random writes

We expect that FlashBlade shows equal performance across the two write tests and equal performance across the two read tests.

With 100 clients, a single DGX can maxes out the throughput of an 8-blade system, and the test suite takes ~ 15 min to run. (Fewer clients, less time).

Example throughput from 3:1 compressible data and 200 containers on a single DGX-1 to an 8-blade FlashBlade:

Example throughput based on number of containers for an 8-blade FlashBlade:

# clients/containers	Read (GB/s)	Write (GB/s)	io-size
50	7.1	3.2	512K
80	7.6	4.2	512K
200	8.0	5.0	512K
200	8.25	5.0	1024K

Note 1: If you abort a test, you may have to manually delete the docker containers you just created. E.g: `docker ps -a | grep 'iozone' | xargs docker rm -f`

Note 2: This script currently runs at 3:1 compression. To test with no compression, you can make this change in the script directly:

```
docker exec $MASTER iozone -I -c -e -w -C \
    $REDUCTION_3TO1 \
    …
```

→

```
docker exec $MASTER iozone -I -c -e -w -C \
    $REDUCTION_NONE \
    …
```

(For the full iozone playbook, please reach out to Joshua Robinson.)

airi-libnfs

This script only makes sense to run if you have **100,000s of files** on your FlashBlade. We shared this test because running performance tests off a customer's real-world data is often more realistic than synthetic tests, like iozone.

Overview

When we kick off a job to read a large number of files, we will invoke usage of libnfs, which opens many read threads, rather than the kernel NFS client, which reads sequentially.
This script works by feeding a list of file names stored in a "`manifest`", then using libnfs-python to read all those files with parallel streams.

Contents:

/nw-test-tools/airi-libnfs/

- libnfs-1.9.8 # used for installation of libnfs on the DGX
- run_libnfs.sh # kicks off a test job
- test.py # test script that run_libnfs.sh uses. No need to edit/read.
- manifest # **placeholder** for a file that you will need to generate
- example-manifest # example of file that your "manifest" should look like

From the FlashBlade: Create a filesystem and a data VIP.
The following steps should be performed on a DGX-1.

Step 1: Install libnfs on the DGX-1

Install libraries needed to run libnfs python scripts:

```
$ sudo apt-get install python3 python3-pip python3-dev libnfs-dev
python-pip make wget
```

Install libnfs:

```
$ cd airi-libnfs/
$ pip install libnfs
```

✔ Check Step 1:

```
$ pip list --format=legacy | grep "libnfs"
```

should return "libnfs (<version>)", confirming that libnfs-python is installed.

Step 2: Build a manifest file

Create a “manifest” file that matches your dataset. There are a couple painless ways to do it. Here’s an example that searches for files on a mounted filesystem (at “/mnt/datasets/” on this DGX) and saves the file paths to manifest. Then it uses the “sed” command to replace the file path prefix.

```
$ cd airi-libnfs/
$ find /mnt/datasets/ /b > ./manifest
$ sed -i "s|\/mnt/datasets||g" ./manifest
```

✔ Check Step 2:
Compare the file name format in your new “manifest” file to that in the “example-manifest” file. Your original mount folder should not appear in the file path anywhere - that parent directory will be specified during the test.

Step 3: Execute the run_libnfs.sh script on a DGX-1

This script will ask for two input variables.
For example:

```
$ ./run_libnfs.sh
FlashBlade data VIP? 198.18.0.100
filesystem name (where files in 'manifest' live)? datasets
```

✔ Check Step 3:
The FlashBlade GUI should show start showing read traffic.

The duration of load is based on length of the manifest file - i.e. how many files libnfs is instructed to read.

Pushing more workload:
Executing multiple copies of this script concurrently from 1 DGX will drive even higher load. For example, you could use either Docker or simply multiple ssh sessions into a single DGX to start the script multiple times. 4 concurrent copies of this script is enough to max out the bandwidth of a 7-blade FlashBlade.

Example throughput from 2 instances of the script on a single DGX-1 to an 8-blade FlashBlade: