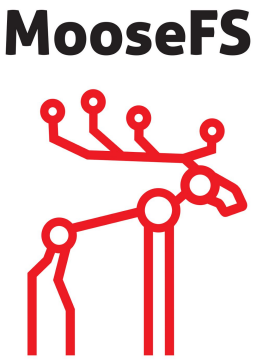

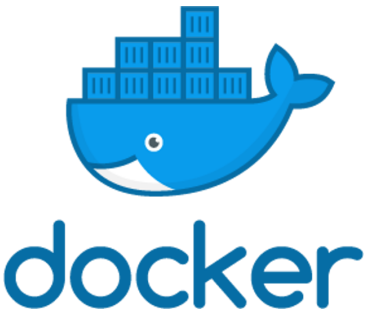

MooseFS showcases good performance on Docker!

There is one exciting news for all the MooseFS users that MooseFS showcases good performance on Docker containers. So, this means our customers can now deploy MooseFS easily using Docker containers.The tests were performed with Docker 1.13 and MooseFS 4.0 software version but the results are also achievable with MooseFS 3.0.92+ version.

We conducted a test, in which MooseFS was run on Docker to check whether rapid deployment of MooseFS can be achieved or not using Docker.The setup consisted of 10 servers, one as client, two as master servers and seven machines as chunk servers. All the machines were connected in ring topology using Intel I350 Gigabit Network Connection cards.

The experiments showed us how the use of containers affected the performance of MooseFS storage cluster. Results show us that containerised storage is 5-10% slower than bare metal setup. Although, for speed-critical systems, it is recommended to use bare metal installation.However, the Docker based setup is recommended, for first steps with MooseFS to quickly run on any operating system supported by Docker.

Such a setup, can also be used to test MooseFS features. Using this configuration one can test MooseFS POSIX compatibility, redundancy and other features.

In this blog, we will first give some information about MooseFS and Docker and then we will explain the set-up and how the tests were conducted to run MooseFS in docker. And then there will be analysis of performance of MooseFS in Docker as compared with bare metal with the help of graphs.This blog concludes with scripts and detailed results as mentioned in the appendix.

About MooseFS

MooseFS is a fault tolerant, highly available, highly performing, scaling-out, network distributed file system. It spreads data over several physical commodity servers, which are visible to the user as one resource.
For standard file operations MooseFS acts like any other Unix-like system:

- A hierarchical structure (directory tree)
- Stores POSIX file attributes (permissions, last access and modification times)
- Supports special files (block and character devices, pipes and sockets)
- Symbolic links (file names pointing to target files, not necessarily on MooseFS) and hard links (different names of files that refer to the same data on MooseFS)
- Access to the file system can be limited based on IP address and/or password

Distinctive features of MooseFS are:

- High availability (i.e. redundant meta-data servers)
- High reliability (several copies of the data can be stored on separate computers)
- Capacity is dynamically expandable by simply adding new servers or disks
- Deleted files are retained for a configurable period of time (a file system level "trash bin")
- Coherent snapshots of les, even during write/access operations

MooseFS is an Open Source software available on https://github.com/moosefs/moosefs.
For more information about MooseFS please visit: http://moosefs.com

About Docker

Docker is a container platform which provides additional layer of abstraction and automation of operating system level virtualization on Windows and Linux. It isolates software which can run on a shared operating system instead of running on full Virtual Machine. Such a solution is more and more popular due to it's really fast and reliable deployment capability.

For more information about Docker please visit: https://www.docker.com/

2 MooseFS in Docker

The following section provides description and configuration details for MooseFS in Docker containers. Only one server was dedicated as MooseFS client. Benchmark was executed inside MooseFS client mount point. Benchmark tool used in this test was IOzone software, version 3.465.

```
docker run -p 9419-9422:9419-9422 -d --privileged mfs4chunkserver
```

2.1 Test setup

We used 10 servers for the setup, one as client, two as masters and seven machines as chunk servers (Figure 1). All the machines were connected in ring topology using Intel I350 Gigabit Network Connection cards. To eliminate hard disk bottleneck, 100GB RAM disks were created on each chunk server. No kernel modifications and no additional components were required. MooseFS replication was set to goal 1. Docker instances used official Ubuntu 14.04 image from https://hub.docker.com/_/ubuntu/.

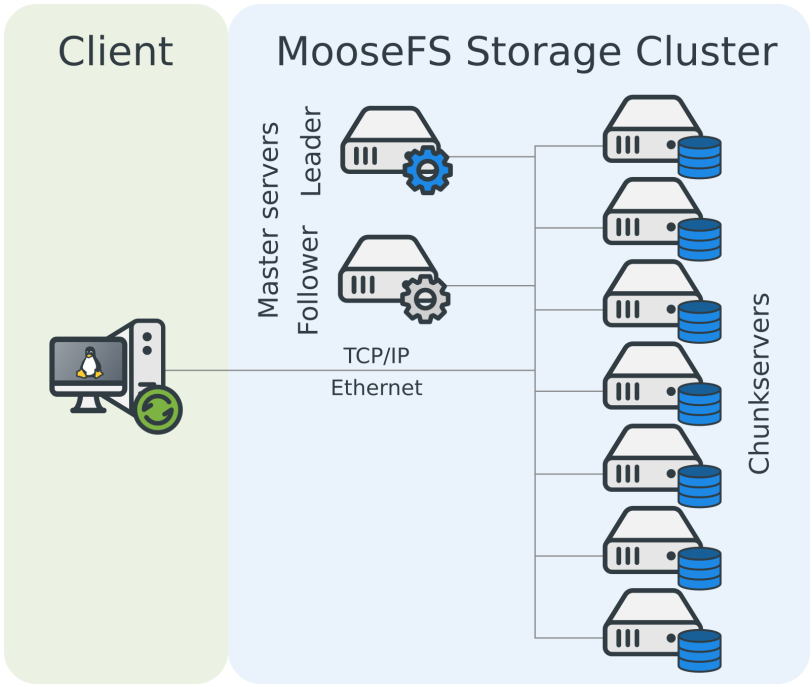

Figure 1: Storage cluster architecture

3 Results

The following graphs show read/write and random read/write operations throughput with block size 4k, 16k, 2048k and 1, 2, 8 threads. The purpose of this test was to compare the performance of Docker container with bare metal, so the Y axis is scaled such that 100% is the performance of bare metal MooseFS.

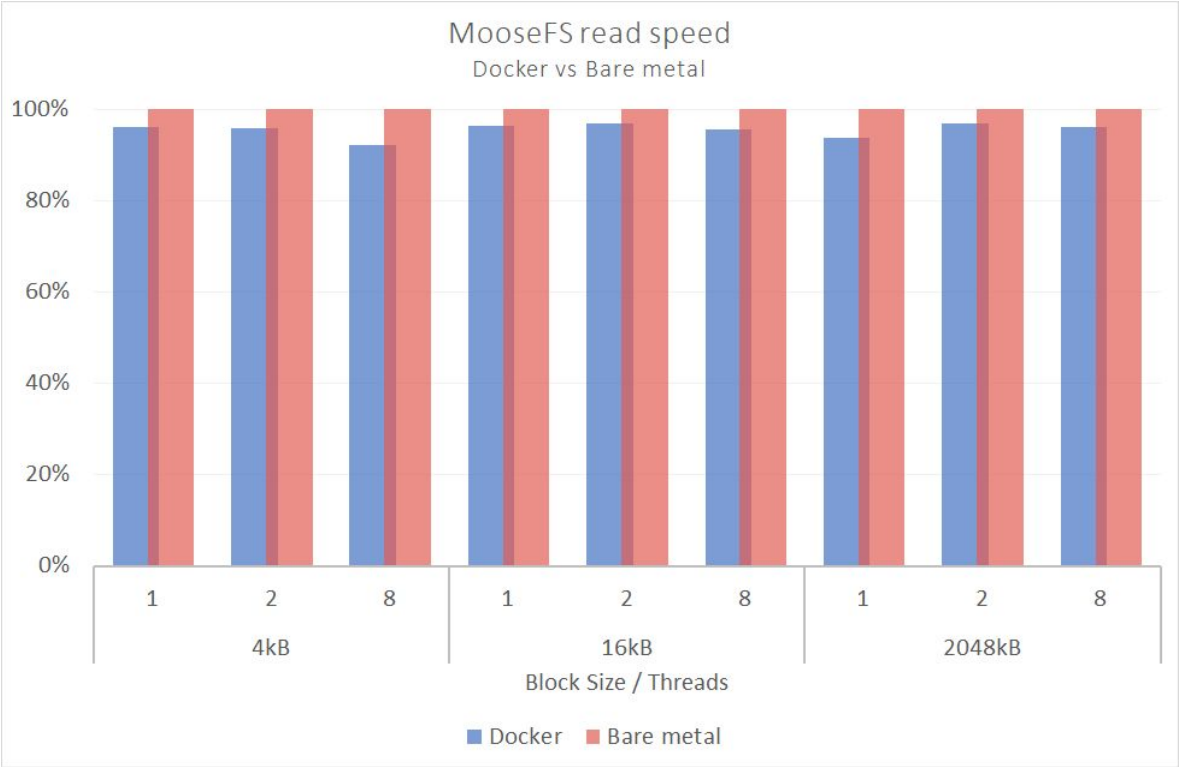

Figure 2: Read speed test results using 1, 2, 8 threads for each 4 kB, 16 kB, 2048 kB Block Size

Figure 3: Write speed test results using 1, 2, 8 threads for each 4 kB, 16 kB, 2048 kB Block Size

Figure 4: Random read speed test results using 1, 2, 8 threads for each 4 kB, 16 kB, 2048 kB Block Size

Figure 5: Random write speed test results using 1, 2, 8 threads for each 4 kB, 16 kB, 2048 kB Block Size

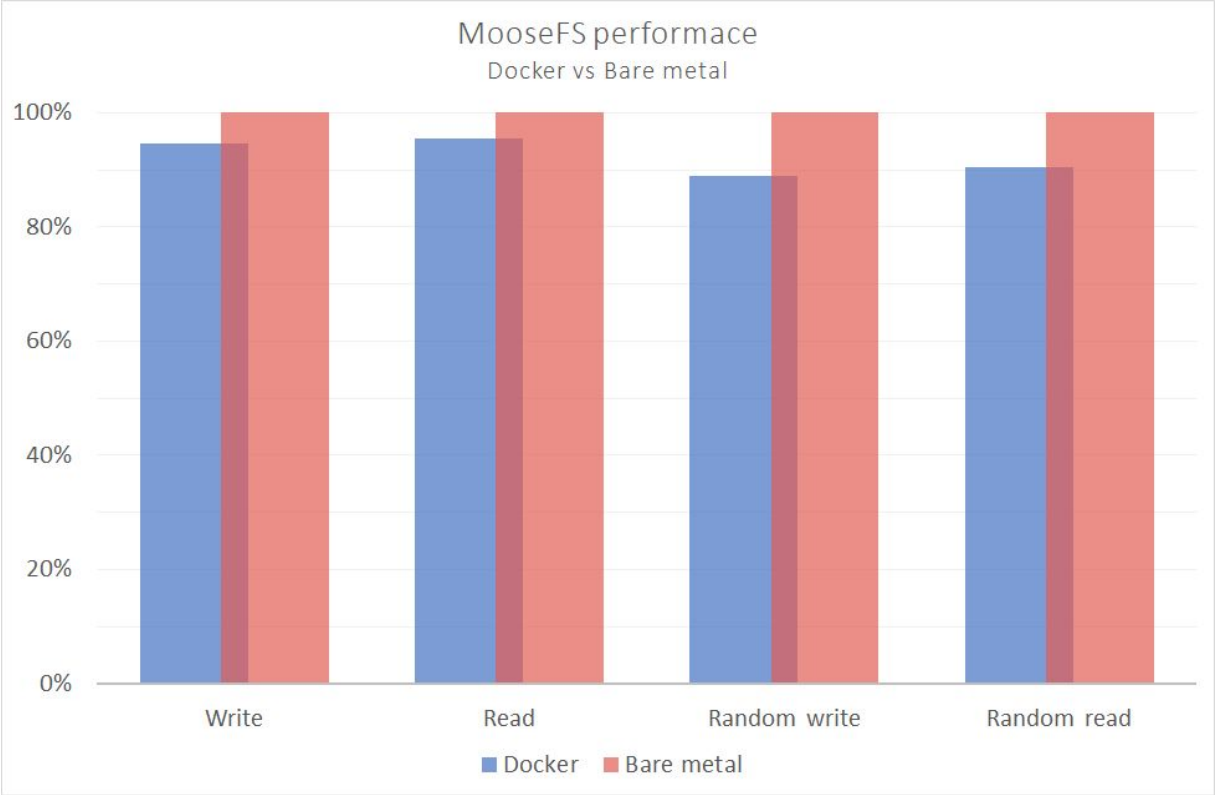

Figure 6: Performance loss when using Docker containers instead of bare metal MooseFS

Appendix

This section provides Docker scripts and detailed results gathered during all IOzone benchmark tests. The below table shows all the results for performed tests. It includes block sizes starting from 4 kB to 2048 kB and 1 to 16 threads.

A. Scripts

Master

Dockerfile

FROM ubuntu:14.04

```
# Install wget, lsb-release and curl
RUN apt-get update && apt-get upgrade -y

# Add and install master package
ADD moosefs-pro-master_4.0.7-1_amd64.deb
/home/moosefs-pro-master_4.0.7-1_amd64.deb
```

```
RUN dpkg -i /home/moosefs-pro-master_4.0.7-1_amd64.deb

# Expose ports
EXPOSE 9420 9421 9422 9423 9424 9425

# Add and run start script
ADD mfslicence.bin /home/mfslicence.bin
ADD start.sh /home/start.sh
RUN chown root:root /home/start.sh
RUN chmod 700 /home/start.sh
CMD ["/home/start.sh", "-d"]
```

---

start.sh

```
#!/bin/bash
cp /etc/mfs/mfsmaster.cfg.sample /etc/mfs/mfsmaster.cfg
cp /etc/mfs/mfsexports.cfg.sample /etc/mfs/mfsexports.cfg
cp /home/mfslicence.bin /etc/mfs/mfslicence.bin

# Add hostname to hosts
ifconfig eth0 | awk '/inet addr/{printf substr($2,6)}' >> /etc/hosts
echo "      mfsmaster" >> /etc/hosts
ifconfig eth0 | awk '/inet addr/{print substr($2,6)}'

mfsmaster start -a

if [[ $1 == "-d" ]]; then
    while true; do sleep 1000; done
fi

if [[ $1 == "-bash" ]]; then
    /bin/bash
fi
```

Chunkserver

Dockerfile

```
FROM ubuntu:14.04

# Install wget, lsb-release and curl
RUN apt-get update && apt-get upgrade -y
```

```
ADD moosefs-pro-chunkserver_4.0.7-1_amd64.deb
/home/moosefs-pro-chunkserver_4.0.7-1_amd64.deb
RUN dpkg -i /home/moosefs-pro-chunkserver_4.0.7-1_amd64.deb

# Expose ports
EXPOSE 9419 9420 9422

# Add and run start script
ADD start-chunkserver.sh /home/start-chunkserver.sh
RUN chown root:root /home/start-chunkserver.sh
RUN chmod 700 /home/start-chunkserver.sh

CMD ["/home/start-chunkserver.sh", "-d"]
```

---

start-chunkserver.sh

```
#!/bin/bash
cp /etc/mfs/mfschunkserver.cfg.sample /etc/mfs/mfschunkserver.cfg

mkdir -p /mnt/ramchunk
mount -t tmpfs -o size=17g tmpfs /mnt/ramchunk
chown -R mfs:mfs /mnt/ramchunk
echo "/mnt/ramchunk" > /etc/mfs/mfshdd.cfg

echo "10.1.1.1 mfsmaster" >> /etc/hosts

mfschunkserver start

if [[ $1 == "-d" ]]; then
    while true; do sleep 1000; done
fi

if [[ $1 == "-bash" ]]; then
    /bin/bash
fi
```

Client

Dockerfile

```
FROM ubuntu:14.04

# Add multiverse repository for iozone installation
```

```
RUN echo "deb http://pl.archive.ubuntu.com/ubuntu/ trusty main
restricted universe multiverse" > /etc/apt/sources.list

RUN apt-get update && apt-get install -y wget lsb-release curl fuse
libfuse2 tree iozone3

# Add and install moosefs 4.0 client
ADD moosefs-pro-client_4.0.7-1_amd64.deb
/home/moosefs-pro-client_4.0.7-1_amd64.deb
RUN dpkg -i /home/moosefs-pro-client_4.0.7-1_amd64.deb

# Create user for iozone test purpose
RUN useradd -ms /bin/bash core

# Add and run start script
ADD start-client.sh /home/start-client.sh
RUN chown root:root /home/start-client.sh
RUN chmod 700 /home/start-client.sh

CMD ["/home/start-client.sh", "-test"]
```

---

start-client.sh

```
#!/bin/bash

echo "10.1.1.1 mfsmaster" >> /etc/hosts
mkdir -p /mnt/mfs

# mount mfs
mfsmount /mnt/mfs -H mfsmaster
mfssettrashtime -r 0 /mnt/mfs
mfssetgoal -r 1 /mnt/mfs


if [[ $1 == "-d" ]]; then
    while true; do sleep 1000; done
fi

if [[ $1 == "-bash" ]]; then
    /bin/bash
fi


if [[ $1 == "-test" ]]; then
```

```
	set -e
	start_date=`date +%y%m%d`
	tf_name='docker-test'

	cd /mnt/mfs

	for i in 1 2 3 4 5
	do
		su -c "mkdir -p ${tf_name}-${start_date}-${i}" core
		for j in 4 8 16 32 64 128 256 512 1024 2048
		do
			for t in 1 2 4 8 16
			do

				echo "IOZONE test loop: ${i}, block size:
${j}KB, Threads number: ${t}"
				su -c "iozone -Ie -r${j} -t${t} -s1g -i0 -i1
-i2 > /mnt/mfs/${tf_name}-${start_date}-${i}/iozone-r${j}-t${t}-`date
+%y%m%d`-`date +%H-%m`.log" core

				echo "Sleeping for sustained files"
				sleep 120
			done
		done
	done
fi
```

B. Detailed Results

Table 1: IOzone test results showing how performance changes with block size and number of processing threads for Bare metal and Docker

Block Size	Threads	Docker				Bare metal			
		Write MB/s	Read MB/s	Random write MB/s	Random read MB/s	Write MB/s	Read MB/s	Random write MB/s	Random read MB/s
4k	1	217	362	112	19	222	377	138	21
	2	356	603	158	37	367	628	206	44
	4	491	825	206	81	518	876	277	93
	8	589	966	277	148	629	1 047	318	179
	16	641	974	314	235	684	1 088	376	294

8k	1	390	558	215	33	397	593	263	39
	2	621	845	303	68	650	870	394	80
	4	842	1 055	386	149	907	1 074	530	173
	8	978	1 129	503	276	1 094	1 185	606	329
	16	1 057	1 141	589	443	1 172	1 196	701	549
16k	1	635	730	392	51	681	757	464	73
	2	1 040	963	554	112	1 088	994	692	156
	4	1 148	1 116	679	240	1 212	1 153	1 191	309
	8	1 150	1 138	824	477	1 209	1 192	1 151	571
	16	1 143	1 142	978	779	1 209	1 197	1 198	988
32k	1	836	865	697	91	908	888	814	96
	2	1 146	1 049	978	202	1 210	1 133	1 204	233
	4	1 150	1 124	1 130	424	1 209	1 171	1 215	496
	8	1 151	1 138	1 131	822	1 208	1 190	1 205	947
	16	1 144	1 142	1 118	1 143	1 207	1 199	1 201	1 202
64k	1	1 032	914	1 133	160	1 117	954	1 205	171
	2	1 154	1 069	1 140	353	1 209	1 103	1 207	365
	4	1 153	1 123	1 136	694	1 214	1 167	1 211	676
	8	1 151	1 137	1 138	1 142	1 208	1 194	1 207	1 188
	16	1 144	1 142	1 131	1 146	1 209	1 199	1 204	1 206
128k	1	1 100	961	1 131	272	1 182	1 019	1 206	279
	2	1 148	1 108	1 139	548	1 208	1 116	1 208	601
	4	1 154	1 133	1 137	990	1 213	1 180	1 214	1 098
	8	1 147	1 138	1 134	1 146	1 208	1 186	1 205	1 207
	16	1 145	1 141	1 134	1 146	1 208	1 197	1 202	1 206
256k	1	1 095	980	1 132	299	1 183	1 013	1 206	298
	2	1 160	1 089	1 138	572	1 209	1 151	1 207	621
	4	1 156	1 122	1 136	1 010	1 218	1 171	1 213	1 116
	8	1 151	1 138	1 133	1 146	1 208	1 192	1 208	1 207
	16	1 145	1 142	1 137	1 145	1 208	1 199	1 205	1 206
512k	1	1 103	964	1 134	311	1 183	1 026	1 204	310
	2	1 151	1 102	1 138	581	1 209	1 134	1 213	630
	4	1 152	1 134	1 137	1 030	1 212	1 177	1 210	1 130
	8	1 151	1 137	1 135	1 146	1 208	1 193	1 209	1 207

	16	1 142	1 142	1 137	1 146	1 207	1 199	1 203	1 205
1024k	1	1 106	947	1 131	323	1 184	1 037	1 205	313
	2	1 145	1 082	1 143	583	1 213	1 121	1 207	637
	4	1 153	1 129	1 138	1 029	1 213	1 182	1 209	1 127
	8	1 151	1 138	1 134	1 146	1 207	1 189	1 209	1 207
	16	1 144	1 141	1 135	1 145	1 208	1 198	1 203	1 205
2048k	1	1 101	968	1 129	331	1 180	1 033	1 206	314
	2	1 147	1 088	1 137	591	1 209	1 123	1 209	639
	4	1 150	1 132	1 140	1 032	1 210	1 166	1 207	1 130
	8	1 149	1 138	1 136	1 145	1 205	1 185	1 209	1 206
	16	1 140	1 142	1 134	1 144	1 206	1 198	1 204	1 205