Perfomance&Functionality Testbed for Clustered Filesystems: LUSTRE and some of its friends.

Giuseppe Bruno[1]

[1]Economic Research and International Relations.
Bank of Italy

LUSTRE User Group, April 2013. San Diego (CA)

Outline

1. Motivation
   - Running and designing a new computing platform
   - Meeting all the users' needs
2. Employed benchmarks
   - Microbenchmarks
   - Application benchmark mimicking typical workload
3. Lustre and some if its friends
   - The file systems under scrutiny
   - The functionality test: POSIX compliance

Outline

1. Motivation
   - Running and designing a new computing platform
   - Meeting all the users' needs
2. Employed benchmarks
   - Microbenchmarks
   - Application benchmark mimicking typical workload
3. Lustre and some if its friends
   - The file systems under scrutiny
   - The functionality test: POSIX compliance

Running a computing platform.

In running a computing platform we come over functional and perfomance problems.
From the analysis of benchmarks and compliance tests we can mitigate, work around and sometimes even remove the issues.

- POSIX compliance issues
- performance issues in reading/writing data on OSTs
- performance issues in reading/writing metadata on MDT

Running a computing platform.

In running a computing platform we come over functional and perfomance problems.
From the analysis of benchmarks and compliance tests we can mitigate, work around and sometimes even remove the issues.

- POSIX compliance issues
- performance issues in reading/writing data on OSTs
- performance issues in reading/writing metadata on MDT

Running a computing platform.

In running a computing platform we come over functional and perfomance problems.
From the analysis of benchmarks and compliance tests we can mitigate, work around and sometimes even remove the issues.

- POSIX compliance issues
- performance issues in reading/writing data on OSTs
- performance issues in reading/writing metadata on MDT

Designing a new computing platform.

In designing a new computing platform we want to meet all the users' requirements.
The users' population at the Research Department of the Bank of Italy presents a very diversified I/O Workload:

- Large sequential I/O patterns with file sizes ranging from 10 to almost 100 GBytes;
- millions of files with a size of less than 4 KBytes;
- cooperative production of Windows office documents whose data are the result of some LINUX processing;
- Batch scheduled jobs interacting with windows files.

Designing a new computing platform.

In designing a new computing platform we want to meet all the users' requirements.
The users' population at the Research Department of the Bank of Italy presents a very diversified I/O Workload:

- Large sequential I/O patterns with file sizes ranging from 10 to almost 100 GBytes;
- millions of files with a size of less than 4 KBytes;
- cooperative production of Windows office documents whose data are the result of some LINUX processing;
- Batch scheduled jobs interacting with windows files.

Designing a new computing platform.

In designing a new computing platform we want to meet all the users' requirements.
The users' population at the Research Department of the Bank of Italy presents a very diversified I/O Workload:

- Large sequential I/O patterns with file sizes ranging from 10 to almost 100 GBytes;
- millions of files with a size of less than 4 KBytes;
- cooperative production of Windows office documents whose data are the result of some LINUX processing;
- Batch scheduled jobs interacting with windows files.

Designing a new computing platform.

In designing a new computing platform we want to meet all the users' requirements.
The users' population at the Research Department of the Bank of Italy presents a very diversified I/O Workload:

- Large sequential I/O patterns with file sizes ranging from 10 to almost 100 GBytes;
- millions of files with a size of less than 4 KBytes;
- cooperative production of Windows office documents whose data are the result of some LINUX processing;
- Batch scheduled jobs interacting with windows files.

Outline

1. Motivation
   - Running and designing a new computing platform
   - Meeting all the users' needs
2. Employed benchmarks
   - Microbenchmarks
   - Application benchmark mimicking typical workload
3. Lustre and some if its friends
   - The file systems under scrutiny
   - The functionality test: POSIX compliance

Meeting all the users' needs.

We deal with about 500 users asking different computing services:

- Lustre file system is a permanent storage space (home and group directory)
- Quotas are the main tool for a fair disk space allocation policy;
- Extended Access Control Lists are often required by the users.

Most of the computing is done with statistical and econometric packages. Some web applications access the Lustre filesystem.

Meeting all the users' needs.

We deal with about 500 users asking different computing services:

- Lustre file system is a permanent storage space (home and group directory)
- Quotas are the main tool for a fair disk space allocation policy;
- Extended Access Control Lists are often required by the users.

Most of the computing is done with statistical and econometric packages. Some web applications access the Lustre filesystem.

Meeting all the users' needs.

We deal with about 500 users asking different computing services:

- Lustre file system is a permanent storage space (home and group directory)
- Quotas are the main tool for a fair disk space allocation policy;
- Extended Access Control Lists are often required by the users.

Most of the computing is done with statistical and econometric packages. Some web applications access the Lustre filesystem.

Meeting all the users' needs.

We deal with about 500 users asking different computing services:

- Lustre file system is a permanent storage space (home and group directory)
- Quotas are the main tool for a fair disk space allocation policy;
- Extended Access Control Lists are often required by the users.

Most of the computing is done with statistical and econometric packages. Some web applications access the Lustre filesystem.

Outline

1. Motivation
   - Running and designing a new computing platform
   - Meeting all the users' needs
2. Employed benchmarks
   - Microbenchmarks
   - Application benchmark mimicking typical workload
3. Lustre and some if its friends
   - The file systems under scrutiny
   - The functionality test: POSIX compliance

The employed microbenchmarks for filesystem evaluations

- Bonnie++ v. 1.96
- IOZONE v. 3.4.14
- **dd** UNIX command
- Mdtest v. 1.8.3 (with home grown modifications)
- NBENCH from smbtorture 3.6.8

Benchmarks under investigation:

- metarates for metadata performances
- Flexible File System Benchmark (ffsb) seems another interesting tool

Microbenchmarks: the employed commands

1. bonnie++ -s=1000 -r=500
2. iozone -a -n1000m -g1000m
3. dd if=/dev/zero of=tmp_DD bs=1000000 count=1000 (write)
4. dd if=/dev/zero of=tmp_DD bs=4k count=256k (write)
5. dd if=tmp_DD of=/dev/null bs=1000000 count=1000 (read)
6. dd if=tmp_DD of=/dev/null bs=4k count=256k (read)
7. mdtest -z6 -I6000 -i 10

Microbenchmarks: some comparative tables

Performance measures are not comparable over file systems because of different storage, networking and computing.

Table 2.1: Peak performance measures for **data I/O**

Write in MB/s

used tool	Lustre 1.8	NFS mount	GPFS 3.4	GPFS 3.5	FEFS	Lustre 2.1
Bonnie++	180	906	1205	834	453	284
Iozone rl=4K	250	2354*	1256	1827	981	413
dd bs=4K	137	660	796	506	336	193
dd bs=1M	239	710	1900	1400	866	409

Note: starred values are counterintuitive. A possible explanation might be in the caching behaviour.

Microbenchmarks: some comparative tables

Performance measures are not comparable over file systems because of different storage, networking and computing.

Table 2.2: Peak performance measures for **data I/O**

Read in MB/s

used tool	Lustre 1.8	NFS mount	GPFS 3.4	GPFS 3.5	FEFS	Lustre 2.1
Bonnie++	847	1229	324	1598	2626	906
Iozone rl=4K	1022	6847*	2857*	2085	8049*	4581
dd bs=4K	606	4800	269	787	1600	457
dd bs=1M	1500	7900	1100	903	6800	5100
dd bs=1M write from a node and read from a diff. node	201	971	179	N.A.	712	N.A.

Note: starred values are counterintuitive. A possible explanation might be in the caching behaviour.

Microbenchmarks: comparative tables for Metadata activities

Performance measures are not comparable over file systems because of different storage, networking and computing.

Table 2.3: Peak performance for **metadata I/O**

Metadata Create/s

used tool	Lustre 1.8	NFS mount	GPFS 3.4	GPFS 3.5	FEFS	Lustre 2.1
bonnie++ dir	832	4792*	2158	3255	1583	1962
mdtest dir	1491	2895	2050	1839	4508*	3184
mdtest file	853	3706	2878	3575	1677	1570
mdtest tree	952	3940	N.A.	3417	2094	1736

Note: starred values are counterintuitive. A possible explanation might be in the caching behaviour.

Microbenchmarks: Using NBENCH for simulating Windows client activities

Sharing files between LINUX and Windows environment has been achieved with a SAMBA server on the LINUX platform. The commands employed for simulating the Windows activities are:

1. smbtorture //serv_name/gs-cifs -Uroot% -c client.txt -N 10 NBENCH
2. smbtorture //serv_name/gs-cifs -Uroot% -c client.txt -N 10 -L NBENCH

Using NBENCH for simulating Windows client activities

With the benchmark **smbtorture** we simulate multiple access to the file system from Window clients.

- the -c flag indicates the load description file. In our test we employed the file client.txt provided by DBENCH
- the -N flag sets the number of concurrent Windows clients
- the -L flag asks for the client opportunistic lock
- the NBENCH option allows to emulate the workload described in the file client.txt

Some results with NBENCH

Performance measures are not comparable over file systems because of different storage, networking and computing.

Table 2.4: performance measures with a SMB/CIFS load (Bandwidth MB/s)

used tool	Lustre 1.8	NFS mnt	GPFS 3.4	GPFS 3.5	FEFS	Lustre 2.1
NBENCH with oplock	11.5	120*	76.8	1.3*	21.6	1.3*
NBENCH wo oplock	6.6	59.1	46.8	94.5	25.6	11.5

starred values require a further investigation.

Outline

1. Motivation
   - Running and designing a new computing platform
   - Meeting all the users' needs
2. Employed benchmarks
   - Microbenchmarks
   - Application benchmark mimicking typical workload
3. Lustre and some if its friends
   - The file systems under scrutiny
   - The functionality test: POSIX compliance

Application Benchmark.

- The application benchmark is composed of a set of shell scripts;
- The benchmark simulates the operation of a tree-like file structure;
- Most of the activities are metadata intensive.

Application benchmark.

The application benchmark breaks down in the following activities:

1. empty tree creation with number of branches and depth as parameter;
2. filling each leaf of the tree with a byte sequence;
3. change the groupship for each leaf of the tree;
4. update the byte content for each leaf of the tree.

Application benchmark.

The application benchmark breaks down in the following activities:

1. empty tree creation with number of branches and depth as parameter;
2. filling each leaf of the tree with a byte sequence;
3. change the groupship for each leaf of the tree;
4. update the byte content for each leaf of the tree.

Application benchmark.

The application benchmark breaks down in the following activities:

1. empty tree creation with number of branches and depth as parameter;
2. filling each leaf of the tree with a byte sequence;
3. change the groupship for each leaf of the tree;
4. update the byte content for each leaf of the tree.

Application benchmark.

The application benchmark breaks down in the following activities:

1. empty tree creation with number of branches and depth as parameter;
2. filling each leaf of the tree with a byte sequence;
3. change the groupship for each leaf of the tree;
4. update the byte content for each leaf of the tree.

Application Benchmark.

Performance measures are not comparable over file systems because of different storage, networking and computing;
The purpose of the table is to show the amount of data collected for analysis.

Table 2.5: tree like processing performance for a typical Metadata bounded load

Execution time in seconds

used tool	Lustre 1.8	NFS mnt	GPFS 3.4	GPFS 3.5	FEFS	Lustre 2.1
tree clean up	57	10	12	45	20	21
tree creation	54	8	9	16	29	21
graph creation	57	12	12	21	30	25
change groupship	28	7	8	5	10	80*
graph update	90	19	20	56	72*	40

Note: starred values require a further investigation.

Outline

1. Motivation
   - Running and designing a new computing platform
   - Meeting all the users' needs
2. Employed benchmarks
   - Microbenchmarks
   - Application benchmark mimicking typical workload
3. Lustre and some if its friends
   - The file systems under scrutiny
   - The functionality test: POSIX compliance

The file systems considered.

In our research we have considered the following file systems:

- Lustre 1.8.7
- Lustre 2.1
- FEFS from Fujitsu
- native GPFS 3.4
- client NFS mount of GPFS 3.4
- native GPFS 3.5

The file systems considered.

In our research we have considered the following file systems:

- Lustre 1.8.7
- Lustre 2.1
- FEFS from Fujitsu
- native GPFS 3.4
- client NFS mount of GPFS 3.4
- native GPFS 3.5

Outline

POSIX compliance comparison.

Table 3.1: POSIX compliance with different file systems

Section	Filesystem	Succeded	Failed	Unresolved	Usupported	Total
ANSI.hdr	Lustre 1.8	32	7	0	203	386
	Lustre 2.1	32	7	0	203	386
	FEFS	32	7	0	203	386
	NFS/GPFS	32	7	0	203	386
	GPFS 3.4	32	7	0	203	386
	GPFS 3.5	32	7	0	203	386
ANSI.os F	Lustre 1.8	925	3	76	0	1205
	Lustre 2.1	951	9	77	0	1244
	FEFS	952	8	77	0	1244
	NFS/GPFS	943	17	77	0	1244
	GPFS 3.4	952	8	77	0	1244
	GPFS 3.5	951	8	77	1	1244

POSIX compliance comparison.

Table 3.2: POSIX compliance with different file systems

Section	Filesystem	Succeded	Failed	Unresoved	Unsupported	Total
ANSI.os M	Lustre 1.8	63	0	23	0	1244
	Lustre 2.1	66	0	20	0	1244
	FEFS	66	0	20	0	1244
	NFS/GPFS	63	3	20	0	1244
	GPFS 3.4	66	0	20	0	1244
	GPFS 3.5	66	0	20	0	1244
POSIX.hdr	Lustre 1.8	24	18	0	178	394
	Lustre 2.1	24	13	0	179	394
	FEFS	24	13	0	179	394
	NFS/GPFS	24	13	0	179	394
	GPFS 3.4	24	13	0	179	394
	GPFS 3.5	24	13	0	179	394

POSIX compliance comparison.

Table 3.3: POSIX compliance with different file systems

Section	Filesystem	Succeded	Failed	Unresolved	Unsupported	Total
POSIX.os F	Lustre 1.8	960	14	56	66	1298
	Lustre 2.1	1028	12	25	65	1253
	FEFS	1019	14	32	65	1253
	NFS/GPFS	915	52	100	65	1254
	GPFS 3.4	955	16	96	0	1254
	GPFS 3.5	1019	17	31	65	1255
Total	Lustre 1.8	2004	42	155	447	5827
	Lustre 2.1	2101	41	122	447	5776
	FEFS	2093	42	129	447	5776
	NFS/GPFS	1977	92	197	447	5777
	GPFS 3.4	2029	44	193	0	5777
	GPFS 3.5	2092	45	128	448	5778

Concluding remarks

In the designing phase of a computing platform it is of paramount importance:

- Achieving a measurable description of the user's requirements;
- pinning down the functionalities more frequently employed and/or more critical;
- finding out the more relevant bottlenecks in performances

Concluding remarks

Harnessing the available benchmarks:

- Outlook
  - Bonnie++, IOZONE and dd features a different caching behaviour;
  - Mdtest microbenchmark had to be integrated with measures on chgrp and chmod operation per seconds.

For Further Reading

R. Latham.
The Impact of File Systems on MPI-IO Scalability.
Argonne National Laboratory, IL 60439, 2011.

S. Alam, E.H. Hussein, K. Howard, N. Stringfellow and F. Verzelloni.
Parallel I/O and the Metadata Wall.
Swiss National Supercomputing Centre, 2012.