

# Lustre File System Monitoring on Franklin DRAFT

Andrew Uselton

April 29, 2008

## Abstract

This document proposes a means of monitoring Lustre file system activity on the Cray XT at NERSC known as “Franklin”. Subjects presented include the acquisition, transport, archiving, and presentation of a variety of file system measurements and statistics. Some of what is presented is speculative and will be cast as experiments to determine the feasibility of an approach along with alternatives to that approach. Early efforts will focus on the test and development companion cluster named “Silence”.

## 1 Introduction

This first brief note is meant to cover the early stages of our Lustre monitoring effort. Feel free to skip to Section 6 for an overview of next steps. Please let me know of errors and omissions in this document.

### 1.1 The Network Layout of Franklin

Figure 1 is an abstract representation of the various pieces of Franklin that will participate in file system monitoring. On the far left are the *compute nodes* (CNs), which will run the file system benchmark tests used to analyze I/O performance. Other than acting as a source of I/O activity the CNs will not participate in monitoring. Communicating with the CNs via a high speed interconnect (the *torus*) are the following systems:

- boot** The *boot* node provides a system-wide root file system for all the other nodes in the cluster. This file space is served from a fibre channel (FC) attached DDN RAID system.
- sdb** The *sdb* node maintains system state (the service database) for the cluster and will not participate in file system monitoring activities. Alone among the cluster nodes the *boot* and *sdb* nodes have (FC-attached) disks.
- login** The *login* nodes are the one user-accessible point of contact on the cluster. There are sixteen login nodes, *login01* to *login16*, and each has a

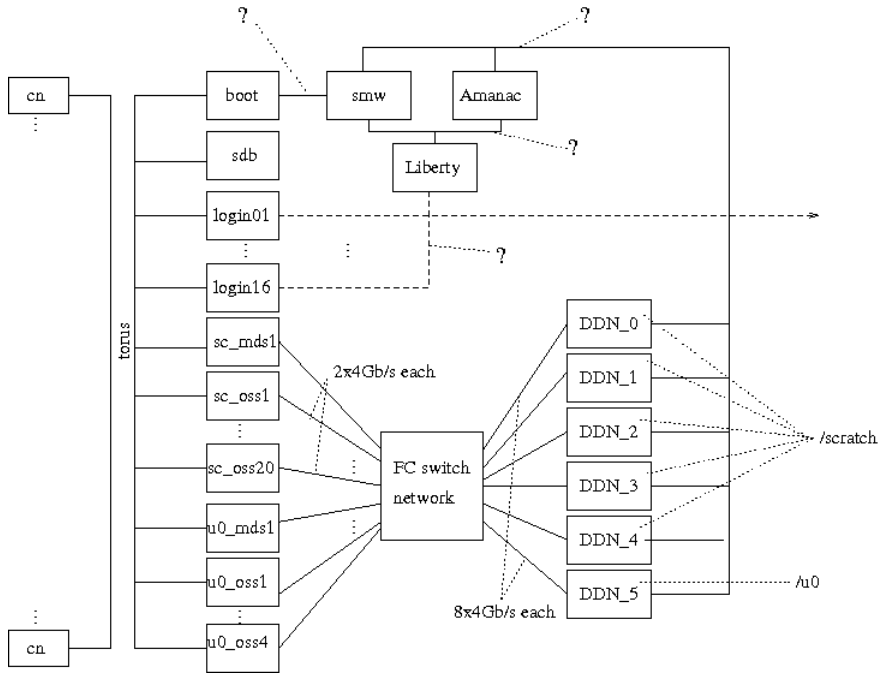


Figure 1: The layout of the various pieces of Franklin

high-speed link to the external network. The *nid*-based hostnames for the login nodes are  $\{login01:nid04100, login02:nid04103, login03:nid04104, login04:nid04105, login05:nid04106, login06:nid04111, login07:nid04112, login08:nid04115, login09:nid04119, login10:nid04580, login11:nid04583, login12:nid04584, login13:nid04587, login14:nid04588, login15:nid04591, login16:nid04595\}$  Section 2.1 below details the specific data items we plan to monitor on the login nodes. Appendix A.1 lists, in detail, the many data items that could be monitored on the login nodes.

**MDS** The Lustre MDS is the *metadata* server for a file system. Franklin has two Lustre-based files systems: */u0* and */scratch*. There are two MDS nodes for */scratch* (*sc\_mds1* and *sc\_mds2*) and two for */u0* (*u0\_mds1* and *u0\_mds2*). The *nid*-based hostname for each MDS is  $\{sc\_mds1:nid04116, sc\_mds2:nid04592, u0\_mds1:nid04140, u0\_mds2:nid04623\}$ . Each pair of MDSs is in a failover arrangement with only one actively serving metadata operations at a given time. The active MDS for */scratch* is *sc\_mds1* and for */u0* is *u0\_mds1*. Section 2.2 details the specific data items we plan to monitor on the MDSs. Appendix A.2 lists, in detail, the many data items that could be monitored.

**OSS** The Lustre OSSs are the *object storage* servers for the Lustre file system. Franklin has 24 OSSs, with 20 OSSs dedicated to the */scratch*

file system and 4 OSSs dedicated to the */u0* file system. The */scratch* OSSs are identified as *sc\_oss1* to *sc\_oss20* and the */u0* OSSs are identified as *u0\_1* to *u0\_4*. Their *nid*-based hostnames are: {*sc\_oss1:nid00024*, *sc\_oss2:nid00027*, *sc\_oss3:nid04124*, *sc\_oss4:nid04127*, *sc\_oss5:nid00032*, *sc\_oss6:nid00035*, *sc\_oss7:nid04132*, *sc\_oss8:nid04135*, *sc\_oss9:nid00040*, *sc\_oss10:nid00043*, *sc\_oss11:nid04596*, *sc\_oss12:nid04599*, *sc\_oss13:nid00504*, *sc\_oss14:nid00507*, *sc\_oss15:nid04604*, *sc\_oss16:nid04607*, *sc\_oss17:nid00512*, *sc\_oss18:nid00515*, *sc\_oss19:nid04612*, *sc\_oss20:nid04615*, *u0\_oss1:nid04143*, *u0\_oss2:nid04616*, *u0\_oss3:nid04619*, *u0\_oss4:nid04620* } Each OSS hosts one or more OSTs, which are the (software) service provided by the server. Each OST mediates the connection to one logical disk (a *LUN*) provided by the DDNs. Section 2.3 details the specific data items we plan to monitor on the OSSs and OSTs. Appendix A.3 lists, in detail, the many data items that could be monitored.

The *boot* node has a connection to the system maintenance workstation *smw*, which runs the CRMS system responsible for starting, stopping, and monitoring the cluster. No other hosts are on this “point-to-point” network. “Almanac” is the name of a system that acts as a data server for monitoring data. “Liberty” is the name of a node that has a connection to the external network. The *smw*, *Liberty*, and *Almanac* are on a shared network that has no other hosts. *Almanac* and the *smw* are also on a separate, isolated network that connects to the control ports of the DDNs.

Each of the MDSs and OSSs has one or more high-speed links to a *fibrenchannel* (FC) network composed of two high capacity switch frames. The servers connect via this FC network to a set of DDN RAID controllers. Each of the 20 OSSs supporting the */scratch* file system connects to the network via two *4Gb/s* FC (FC4) links, and the five DDNs supporting */scratch* each have eight FC4 links. There are also various links from the MDSs and links supporting the */u0* file system.

There is a test/development cluster named “Silence” that has a configuration resembling the foregoing except with fewer login nodes, CNs, OSSs, and MDSs. All of the infrastructure proposed in this document will be evaluated on *Silence* before being deployed on *Franklin*.

## 2 Data Acquisition

There is a wealth of information that can be monitored on the cluster. It is not practical to gather detailed operational data from the CNs, nor do we plan to do so on the service nodes (*boot*, *sdb*), since they do not participate directly in the Lustre file system. The login nodes, the OSSs, and the MDS all merit attention to their dynamic behavior. The appendix details the specific information available on the various nodes that could be collected. What follows is a quick overview of the values we plan to monitor initially. Section 2.5 concludes with a specific proposal for how to actually acquire the data.

## 2.1 Login Nodes

The login nodes mount both Lustre file systems, */scratch* and */u0*. The responsiveness of these nodes to user interactions is one of the two main ways that users perceive whether the file system is fast or slow. Our initial data acquisition plan does not include login node monitoring. That will come later.

## 2.2 MDS

As with the login nodes, a slow MDS can be directly perceived by the user. Acquiring relevant information about the load on the MDS and the performance of its various subsystems can help identify if the MDS itself is unable to keep up with the load, if there is a tuning problem, or if some subsystem is misbehaving. Our initial monitoring efforts will focus on values that show the rate of metadata operations and their latency:

1. From */proc/fs/lustre/mdt/MDT/mds/stats*

```
snapshot_time          1208806073.460654 secs.usecs
req_waitempty          387895 samples [usec] 1 3647 2528191 2007317551
req_qdepth             387895 samples [reqs] 0 60 16560 499738
req_active             387895 samples [reqs] 1 32 406122 929856
...
mds_reint              18350 samples [usec] 21 34335 1726917 1945854633
mds_readpage           0 samples [usec]
mds_statfs             83 samples [usec] 6 63 1181 42677
mds_sync               42 samples [usec] 14 5233 43371 141673119
```

The above data needs some interpretation which will be deferred until Section 5 on data presentation has been more fully developed.

## 2.3 OSS

Poor performance, tuning issues, or excessive load on the OSSs tends to be more visible to at-scale I/O operations from large MPP jobs engaging in bulk I/O. Some monitoring is relevant to the server as a whole and some relevant to each individual OST service. Early efforts at monitoring will focus on the bulk I/O rate seen by each OST:

1. OST stats - For each directory **dir** in */proc/fs/lustre/obdfilter/*,
2. */proc/fs/lustre/obdfilter/dir/stats*

```
snapshot_time          1208806427.826684 secs.usecs
read_bytes             37483 samples [bytes] 0 1048576 38618047029
write_bytes            92565 samples [bytes] 4 1048576 41013702098
...
```

The first value in the line of data is the amount read or written since boot time. The other values need not concern us initially.

## 2.4 DDNs

The Cacti monitoring infrastructure in place on Almanac already collects some DDN statistics. This monitoring proposal does not seek to duplicate or extend that work. In production the DDN-based data will be a valuable resource to compare with the server-side data and the CN-side data.

## 2.5 Acquiring the Data

The Cerebro data collection daemon *cerebrod* runs continuously on each of the nodes. It can be extended to collect data of any sort if the appropriate module is installed. There already exist modules to collect the foregoing data. The Lustre related values can be gathered by including the *lmt* module for Cerebro. There is some question whether it would be appropriate or desirable to install many shared libraries on the server nodes (which is what Cerebro would do by default). Cerebro can be compiled with statically linked libraries, and doing so is currently under investigation.

## 3 Data Transport

Once the data has been collected it needs to be transported off the server nodes where it can be gathered and used. In early testing (on the Silence test cluster) we plan to transport the data to a login node for further processing. In production, the eventual target will be a dedicated monitoring server that has a database and other configuration details in support of system wide monitoring. Almanac in Figure 1 is one candidate. One challenge will be that the servers (MDS and OSSs) do not route directly to any systems off the cluster. Furthermore, Almanac does not route directly to the login nodes. It may be necessary to tunnel connections through the login nodes and the Liberty gateway node. Alternatively, a daemon running on one or more login nodes and on Liberty could relay the data to Almanac.

The Cerebro monitoring infrastructure includes data communication, and the *cerebrod* daemon can act in the roles of data collector, data relay, and data destination. Thus it is compatible with either tunneling or forwarding the data through the intervening hosts.

## 4 Data Archiving

All of the data collected should be gathered in a database. The volume of data gathered amounts to a few KB per sample period. If that data is sampled once per minute the load on the database would be a few MB a day. For intensive dedicated I/O performance testing it may be valuable to sample at higher frequency, which would increase the load on the database.

The Lustre Monitoring Tool infrastructure (of which the *lmt* Cerebro modules are a part) includes a MySQL database schema for Lustre data. The *lmt*

Cerebro modules include methods for sending the data to that database.

The Cacti monitoring infrastructure also maintains an round robin database for detailed, recently observed data and for summaries of older data. That infrastructure is able to acquire data directly from a local database, so the two integrate easily.

## 5 Data Presentation

There are many uses for the acquired data and many ways to display it. The following notes detail a few ways the data can and should be used.

The Lustre Monitoring Tool infrastructure includes a near-real-time GUI display that runs on a user's remote workstation and is populated directly from the MySQL database. In order to access the data there may need to be some relay or tunneling functionality at the Liberty gateway node, or the MySQL database may need to reside somewhere other than on the Almanac server.

There is also a command line tool for gathering near-real-time data from the database.

The Cacti monitoring infrastructure includes a rich web-based data presentation infrastructure. Current detailed information and historical summaries can be made available to any web browser.

When an I/O test produces interesting or anomalous results data queries can be used to gather specific information in support of after-the-fact analysis.

## 6 Conclusion and Recommendations

There are many exploratory steps required before it is reasonable to establish a full proposal for the production configuration of Lustre monitoring. Listed here are some of the early activities that will lead to a detailed plan of action:

- Figure out if Cerebro will compile for and run on a Silence OSS. Figure out where the executable will reside and, if shared libraries can be used, where they will reside. Figure out where the configuration file will reside.
- Set up a toy instance of the Cerebro/LMT target MySQL database on a Silence login node. Establish communication between the cerebrod on the OSS and that on the login node.
- Configure Cerebro for some minimal, sensible monitoring task and verify the infrastructure works.
- Extend the monitored data to include all OSS/OST data values, and gather all login node values while we're at it.
- Configure the LMT data presentation tools, and begin early monitoring of the data is a proof of concept and a sanity check.
- Extend monitoring to all OSSs and the MDS on Silence.

- Evaluate alternatives for final siting of the MySQL database and for the data forwarding needed.
- Initiate extensive file system testing on Silence, and develop an understanding of the utility, managability, and robustness of the system.
- Evaluate the effectiveness of the pilot project and develop a proposal for a production instance of the infrastructure.

## A Detailed Reference for Data Sources

In many of the files mentioned below there will be information in the following format:

```
text      val1 samples [bytes] val2 val3 val4
```

The *text* gives the name of the data item. *val1* is the value of data item. the remaining values on the line are less important.

### A.1 Login Nodes

Here is a list of the many things available for monitoring on the login nodes:

1. */proc/loadavg*
2. */proc/vmstat*
3. Interface statistics
4. For each file system **fs**:
  - (a) */proc/fs/lustre/llite/fs/stats*:

```
snapshot_time      1208709631.618797 secs.usecs
dirty_pages_hits   85898 samples [regs]
dirty_pages_misses 856416 samples [regs]
writeback_from_writepage 0 samples [pages]
writeback_from_pressure 0 samples [pages]
writeback_ok_pages 0 samples [pages]
writeback_failed_pages 0 samples [pages]
read_bytes         193974 samples [bytes] 5 4194304 16299934239
write_bytes        172997 samples [bytes] 9 1241144 3505736517
brw_read           0 samples [pages]
brw_write          0 samples [pages]
ioctl              83 samples [regs]
open               4605 samples [regs]
close              4602 samples [regs]
mmap               0 samples [regs]
seek               36 samples [regs]
```

```

fsync                1 samples [regs]
setattr              35 samples [regs]
punch                0 samples [regs]
getattr             38164 samples [regs]
statfs               21 samples [regs]
alloc_inode          6958 samples [regs]
setxattr             4 samples [regs]
getxattr            51791 samples [regs]
direct_read          0 samples [pages]
direct_write         0 samples [pages]

```

5. For each OST `ost`:

(a) `/proc/fs/lustre/osc/ost/rpc_stats`:

```

snapshot_time:      1208710937.457208 (secs.usecs)
read RPCs in flight: 0
write RPCs in flight: 0
pending write pages: 0
pending read pages: 0

```

pages per rpc	read				write		
	rpcs	%	cum %		rpcs	%	cum %
1:	0	0	0		0	0	0
2:	0	0	0		0	0	0
4:	0	0	0		0	0	0
8:	0	0	0		0	0	0
16:	0	0	0		0	0	0
32:	16	35	35		10	25	25
64:	10	22	57		10	25	51
128:	0	0	57		2	5	56
256:	19	42	100		17	43	100

rpcs in flight	read				write		
	rpcs	%	cum %		rpcs	%	cum %
0:	45	100	100		39	100	100

offset	read				write		
	rpcs	%	cum %		rpcs	%	cum %
0:	45	100	100		35	89	89
1:	0	0	100		0	0	89
2:	0	0	100		0	0	89
4:	0	0	100		0	0	89
8:	0	0	100		0	0	89
16:	0	0	100		0	0	89
32:	0	0	100		1	2	92
64:	0	0	100		1	2	94



128: 0 0 100 | 2 5 100

(b) */proc/fs/lustre/osc/ost/stats*:

snapshot_time	1208711062.600563	secs.usecs
req_waitempty	0 samples	[usec]
req_qdepth	0 samples	[reqs]
req_active	0 samples	[reqs]
reqbuf_avail	0 samples	[bufs]
ost_reply	0 samples	[usec]
ost_getattr	0 samples	[usec]
ost_setattr	0 samples	[usec]
ost_read	45 samples	[usec] 0 0 0 0
ost_write	39 samples	[usec] 0 0 0 0
ost_create	0 samples	[usec]
ost_destroy	7 samples	[usec] 0 0 0 0
ost_get_info	0 samples	[usec]
ost_connect	1 samples	[usec] 0 0 0 0
ost_disconnect	0 samples	[usec]
ost_punch	1 samples	[usec] 0 0 0 0
ost_open	0 samples	[usec]
ost_close	0 samples	[usec]
ost_statfs	32 samples	[usec] 0 0 0 0
ost_san_read	0 samples	[usec]
ost_san_write	0 samples	[usec]
ost_sync	0 samples	[usec]
ost_set_info	0 samples	[usec]
ost_quotacheck	0 samples	[usec]
ost_quotactl	0 samples	[usec]
mds_getattr	0 samples	[usec]
mds_getattr_lock	0 samples	[usec]
mds_close	0 samples	[usec]
mds_reint	0 samples	[usec]
mds_readpage	0 samples	[usec]
mds_connect	0 samples	[usec]
mds_disconnect	0 samples	[usec]
mds_getstatus	0 samples	[usec]
mds_statfs	0 samples	[usec]
mds_pin	0 samples	[usec]
mds_unpin	0 samples	[usec]
mds_sync	0 samples	[usec]
mds_done_writing	0 samples	[usec]
mds_set_info	0 samples	[usec]
mds_quotacheck	0 samples	[usec]
mds_quotactl	0 samples	[usec]
mds_getxattr	0 samples	[usec]
mds_setxattr	0 samples	[usec]

```

ldlm_enqueue          399 samples [usec] 0 0 0 0
ldlm_convert          0 samples [usec]
ldlm_cancel           43 samples [usec] 0 0 0 0
ldlm_bl_callback      0 samples [usec]
ldlm_cp_callback      0 samples [usec]
ldlm_gl_callback      0 samples [usec]
obd_ping              73 samples [usec] 0 0 0 0
llog_origin_handle_cancel 0 samples [usec]
ost_reply             0 samples [usec]

```

6. For each MDS **mds**:

- (a) */proc/fs/lustre/mdc/**mds**/blocksize:*
- (b) */proc/fs/lustre/mdc/**mds**/connect\_flags:*
- (c) */proc/fs/lustre/mdc/**mds**/filesfree:*
- (d) */proc/fs/lustre/mdc/**mds**/filestotal:*
- (e) */proc/fs/lustre/mdc/**mds**/kbytesavail:*
- (f) */proc/fs/lustre/mdc/**mds**/kbytesfree:*
- (g) */proc/fs/lustre/mdc/**mds**/kbytestotal:*
- (h) */proc/fs/lustre/mdc/**mds**/mds\_conn\_uuid:*
- (i) */proc/fs/lustre/mdc/**mds**/mds\_server\_uuid:*
- (j) */proc/fs/lustre/mdc/**mds**/ping:*
- (k) */proc/fs/lustre/mdc/**mds**/uuid:*

## A.2 MDS

Information available on the MDS includes:

- 1. */proc/loadavg*
- 2. */proc/vmstat*
- 3. Interface stats
- 4. For each directory **dir** in */proc/fs/lustre/mds*,
  - (a) */proc/fs/lustre/mds/**dir**/filesfree*
  - (b) */proc/fs/lustre/mds/**dir**/filestotal*
  - (c) */proc/fs/lustre/mds/**dir**/kbytesavail*
  - (d) */proc/fs/lustre/mds/**dir**/kbytesfree*
  - (e) */proc/fs/lustre/mds/**dir**/kbytestotal*
- 5. */proc/fs/lustre/mdt/MDT/mds/stats*

snapshot_time	1208806073.460654 secs.usecs
req_waitempty	387895 samples [usec] 1 3647 2528191 2007317551
req_qdepth	387895 samples [reqs] 0 60 16560 499738
req_active	387895 samples [reqs] 1 32 406122 929856
reqbuf_avail	432099 samples [bufs] 112 128 55281938 7072758762
ost_reply	0 samples [usec]
ost_getattr	0 samples [usec]
ost_setattr	0 samples [usec]
ost_read	0 samples [usec]
ost_write	0 samples [usec]
ost_create	0 samples [usec]
ost_destroy	0 samples [usec]
ost_get_info	0 samples [usec]
ost_connect	0 samples [usec]
ost_disconnect	0 samples [usec]
ost_punch	0 samples [usec]
ost_open	0 samples [usec]
ost_close	0 samples [usec]
ost_statfs	0 samples [usec]
ost_san_read	0 samples [usec]
ost_san_write	0 samples [usec]
ost_sync	0 samples [usec]
ost_set_info	0 samples [usec]
ost_quotacheck	0 samples [usec]
ost_quotactl	0 samples [usec]
mds_getattr	79 samples [usec] 10 23 963 12217
mds_getattr_lock	0 samples [usec]
mds_close	0 samples [usec]
mds_reint	18350 samples [usec] 21 34335 1726917 1945854633
mds_readpage	0 samples [usec]
mds_connect	156 samples [usec] 14 3933 10561 19878001
mds_disconnect	78 samples [usec] 4626 32412 899381 14064312635
mds_getstatus	78 samples [usec] 5 12 554 4184
mds_statfs	83 samples [usec] 6 63 1181 42677
mds_pin	0 samples [usec]
mds_unpin	0 samples [usec]
mds_sync	42 samples [usec] 14 5233 43371 141673119
mds_done_writing	0 samples [usec]
mds_set_info	0 samples [usec]
mds_quotacheck	0 samples [usec]
mds_quotactl	0 samples [usec]
mds_getxattr	366 samples [usec] 11 54 5867 101227
mds_setxattr	0 samples [usec]
ldlm_enqueue	115458 samples [usec] 16 22919873 226321884 2273577007368244
ldlm_convert	0 samples [usec]
ldlm_cancel	0 samples [usec]

```

ldlm_bl_callback      0 samples [usec]
ldlm_cp_callback      0 samples [usec]
ldlm_gl_callback      0 samples [usec]
obd_ping              252971 samples [usec] 5 67 2694192 30277682
llog_origin_handle_cancel 0 samples [usec]
ost_reply              0 samples [usec]

```

### A.3 OSS

Here are the OSS/OST values available for monitoring:

1. */proc/loadavg*
2. */proc/vmstat*
3. Interface stats
4. OST stats - For each directory **dir** in */proc/fs/lustre/obdfilter*,
5. RPC stats

(a) */proc/fs/lustre/obdfilter/dir/brw\_stats*

```
snapshot_time:      1208806267.315413 (secs.usecs)
```

pages per brw	read			write		
	brws	% cum	%	rpcs	% cum	%
1:	642	1	1	5534	5	5
2:	112	0	1	58	0	6
4:	61	0	2	917	0	7
8:	42	0	2	962	1	8
16:	32	0	2	1973	2	10
32:	76	0	2	4167	4	14
64:	36	0	2	46586	50	65
128:	15	0	2	2104	2	67
256:	36810	97	100	30264	32	100

discont pages	read			write		
	rpcs	% cum	%	rpcs	% cum	%
0:	37826	100	100	92560	99	99
1:	0	0	100	5	0	100

discont blocks	read			write		
	rpcs	% cum	%	rpcs	% cum	%
0:	33342	88	88	87834	94	94
1:	4478	11	99	4730	5	99
2:	5	0	99	1	0	100
3:	1	0	100	0	0	100

	read			write		
dio frags	rpcs	% cum %		rpcs	% cum %	
0:	416	1 1		0	0 0	
1:	32926	87 88		87834	94 94	
2:	4478	11 99		4730	5 99	
3:	5	0 99		1	0 100	
4:	1	0 100		0	0 100	

	read			write		
disk ios in flight	ios	% cum %		rpcs	% cum %	
0:	0	0 0		0	0 0	
1:	1221	2 2		92406	94 94	
2:	2305	5 8		4784	4 99	
3:	3031	7 15		38	0 99	
4:	3828	9 24		29	0 99	
5:	4475	10 35		13	0 99	
6:	4527	10 46		11	0 99	
7:	5693	13 59		8	0 99	
8:	6878	16 76		6	0 99	
9:	9853	23 99		2	0 100	
10:	88	0 99		0	0 100	
11:	2	0 100		0	0 100	

	read			write		
io time (1/250s)	rpcs	% cum %		rpcs	% cum %	
1:	777	2 2		61758	66 66	
2:	20	0 2		8204	8 75	
4:	1977	5 7		21780	23 99	
8:	4235	11 18		781	0 99	
16:	15570	41 59		29	0 99	
32:	14776	39 98		6	0 99	
64:	449	1 99		0	0 99	
128:	8	0 99		0	0 99	
256:	0	0 99		0	0 99	
512:	3	0 99		0	0 99	
1024:	3	0 99		7	0 100	
2048:	0	0 99		0	0 100	
4096:	8	0 100		0	0 100	

	read			write		
disk I/O size	count	% cum %		count	% cum %	
4K:	2488	5 5		7977	8 8	
8K:	656	1 7		705	0 8	
16K:	514	1 8		1394	1 10	
32K:	363	0 9		1306	1 11	

64K:	321	0	10		2288	2	14
128K:	335	0	11		4468	4	18
256K:	219	0	11		46648	47	66
512K:	196	0	12		2250	2	68
1M:	36809	87	100		30261	31	100

- (b) */proc/fs/lustre/obdfilter/***dir***/filesfree*
- (c) */proc/fs/lustre/obdfilter/***dir***/filestotal*
- (d) */proc/fs/lustre/obdfilter/***dir***/kbytesfree*
- (e) */proc/fs/lustre/obdfilter/***dir***/kbytestotal*
- (f) */proc/fs/lustre/obdfilter/***dir***/kbytestotal*
- (g) */proc/fs/lustre/obdfilter/***dir***/stats*

snapshot_time	1208806427.826684 secs.usecs
read_bytes	37483 samples [bytes] 0 1048576 38618047029
write_bytes	92565 samples [bytes] 4 1048576 41013702098
iocontrol	1 samples [reqs]
get_info	0 samples [reqs]
set_info_async	1 samples [reqs]
attach	0 samples [reqs]
detach	0 samples [reqs]
setup	0 samples [reqs]
precleanup	0 samples [reqs]
cleanup	0 samples [reqs]
process_config	0 samples [reqs]
postrecov	0 samples [reqs]
add_conn	0 samples [reqs]
del_conn	0 samples [reqs]
connect	79 samples [reqs]
reconnect	0 samples [reqs]
disconnect	75 samples [reqs]
statfs	5 samples [reqs]
statfs_async	0 samples [reqs]
packmd	0 samples [reqs]
unpackmd	0 samples [reqs]
checkmd	0 samples [reqs]
preallocate	0 samples [reqs]
precreate	0 samples [reqs]
create	546 samples [reqs]
destroy	16803 samples [reqs]
setattr	1003 samples [reqs]
setattr_async	0 samples [reqs]
getattr	0 samples [reqs]
getattr_async	0 samples [reqs]
brw	0 samples [reqs]

brw_async	0 samples [reqs]
prep_async_page	0 samples [reqs]
queue_async_io	0 samples [reqs]
queue_group_io	0 samples [reqs]
trigger_group_io	0 samples [reqs]
set_async_flags	0 samples [reqs]
teardown_async_page	0 samples [reqs]
merge_lvb	0 samples [reqs]
adjust_kms	0 samples [reqs]
punch	1809 samples [reqs]
sync	0 samples [reqs]
migrate	0 samples [reqs]
copy	0 samples [reqs]
iterate	0 samples [reqs]
preprw	130048 samples [reqs]
commitrw	130048 samples [reqs]
enqueue	0 samples [reqs]
match	0 samples [reqs]
change_cbdata	0 samples [reqs]
cancel	0 samples [reqs]
cancel_unused	0 samples [reqs]
join_lru	0 samples [reqs]
san_preprw	0 samples [reqs]
init_export	0 samples [reqs]
destroy_export	0 samples [reqs]
llog_init	0 samples [reqs]
llog_finish	0 samples [reqs]
pin	0 samples [reqs]
unpin	0 samples [reqs]
import_event	0 samples [reqs]
notify	0 samples [reqs]
health_check	0 samples [reqs]
quotacheck	0 samples [reqs]
quotactl	3 samples [reqs]
ping	256937 samples [reqs]