Determining the health of Lustre filesystems at scale

Jason J. Hill, Dustin B. Leverman, Scott M. Koch, David A. Dillow
Oak Ridge Leadership Computing Facility
Oak Ridge National Laboratory
{hilljj,leverman,smkoch,dillowda}@ornl.gov

Abstract

Monitoring the components of a Lustre file system is crucial to meeting mission requirements as the scale and complexity of the installation grows. Determining the health and performance of the file system becomes non-trivial, and the complexity increases faster than the size of the installation. This paper discusses the ongoing work at the Oak Ridge Leadership Computing Facility to monitor the health of its center-wide Lustre file systems.

1 Introduction

The Spider parallel filesystems at the Oak Ridge Leadership Computing Facility (OLCF) at Oak Ridge National Laboratory (ORNL), serve as center-wide filesystems for computing, visualization, and application development platforms. This design allows the computational users at the OLCF to generate data on large compute platforms and do analysis and post-processing on different compute platforms and not have to perform large data moves between the platforms. It also allows us to not build high bandwidth networks between the compute platforms and just build a Scalable I/O Network (SION) to handle data generation, analysis, and visualization. The compute, analysis, and visualization systems require a stable, high-performance storage platform to accomplish the mission of delivering breakthrough science. Verifying that the filesystems are healthy and available for use is a key problem that required solving.

The Spider parallel filesystems are made up of 192 Dell PowerEdge 1950 Lustre[2] object storage servers (OSS), 4 Dell PowerEdge R900 Lustre metadata servers (MDS), 2 Dell PowerEdge 1950 Lustre management servers (MGS), 48 couplets of DDN S2A 9900[1] storage systems, 18 management Ethernet switches, 3 Cisco SFS-7024 DDR Infiniband switches, and 48 24-port Mellanox DDR Infiniband switches. Being able to detect faults in this complex problem domain is a large task that the OLCF has attempted to solve via several mechanisms.

When beginning the task of monitoring a Lustre filesystem's health and the health of the infrastructure that provides the filesystem you first must define the service level objectives for the filesystems. Then a monitoring infrastructure can be built around those objectives and measured. These answers can vary across installations. For the purposes of the OLCF it means that all requisite nodes are available (whether in fail-over condition or not), have all of their Lustre devices available, and can communicate via SION. This includes monitoring the health of the physical hardware and all networking connections.

Next, the determination of health can be monitored from the Lustre client's perspective. Additional monitoring can come via log parsing and analysis. Finally we will discuss future work that will be undertaken at the OLCF to more closely monitor the health and performance of our Lustre filesystems.

2 Monitoring Lustre

Monitoring the health of Lustre requires the ability to query status from the hosts that serve the filesystem. At the OLCF we use Nagios[6]. Nagios gives us the ability to query standardly available information from this host as well as the ability to write custom scripts and assign custom SNMP OID's to query. We also have written SNMP Traps that will suppress the notifications if we are in a maintenance period or if we know there is a problem with a part of the filesystem. These traps work on a parent child relationship and allow us to more clearly see the current problem areas if prior problems existed.

2.1 Monitoring Infrastructure

Our Nagios setup includes a single 4 socket quad core HP DL360 G6 with 32 GB of system memory. Our current monitoring load on this box is 1078 hosts with a total of 6094 service checks. The Lustre monitoring makes up 323 of those hosts (29.96%), and 2441 service checks (40.05%). Every time we find a new piece of information that we wish to monitor on the production filesystems we add 212 service checks. For every new host we currently add to the monitoring we add 11 service checks. This will without a doubt grow as we monitor more processes on the Lustre servers.

Our monitoring network consists of 16 Dell PowerConnect 5448 48-port managed Ethernet switches, and 2 Cisco 3560-E 48 port managed Ethernet switches. The Dell switches are connected to each of the Cisco switches via a single copper gigabit Ethernet connection. The Cisco switches are connected to the OLCF routing core via single fiber gigabit Ethernet Connections. The Monitoring VLAN allows all SNMP traffic from the management Ethernet VLAN for the Lustre servers.

2.2 Backend Storage Availability

Using built in Nagios functionality we verify that the storage controllers are on the network and are at least partially available via their management interface. We do more monitoring of these by monitoring the controllers' syslog messages that will be discussed later.

2.3 Server Hardware Health

Using tools available from the Nagios Exchange[7] for monitoring Dell Baseboard Management Controllers, we can verify the status of the dual power supplies in each Lustre server, that the internal temperature is within range, and that all the chassis fans are working properly. Verifying the correct state of the hardware is the first step in reliably verifying that the filesystem is healthy.

2.4 Server Software Health

This area of monitoring is any software or operating system metric that we think is important to a server being ready to serve Lustre, but we don't monitor Lustre specific processes here. We currently check that the server is not under undue load, and that some processes for statistical collection and configuration management are running.

2.5 Lustre Health Checks

We have written several scripts to determine if certain parts of Lustre are correct. These are queried via Nagios through custom SNMP OID's.

2.5.1 Lustre Health

The first (and sometimes best) place to look to find out if the Lustre devices on a node are healthy is the health_check file located in /proc/fs/lustre. This check will get the contents and if it is not "healthy", then will return -2 (Nagios return code for critical), and will print error text to stdout as shown in Listing 1. This will get picked up by the snmpwalk and returned to Nagios to be visualized on the web dashboard. The information will also be sent in any notifications (e-mail, SMS pager, etc.) that Nagios will send about this event.

Listing 1: Bash script for checking health

```
if [[ $(cat /proc/fs/lustre/
    health_check) != "healthy" ]];
    then
  echo "CRITICAL: Lustre is unhealthy
      . Call Lustre OnCall Admin"
  exit 2
else
  echo "OK: Lustre is healthy"
  exit 0
```

2.5.2 LNET Statistics

One key area to measure in the OLCF's Lustre implementation is the status of Lustre Networking (LNET) messages on the Lustre servers and Lustre routers. This can be an indication of major problems that need attention quickly. This Nagios check will look at the contents of /proc/sys/lnet/stats. The contents of the file are described below in order:

- Messages outstanding
- Max outstanding messages
- Number of errors
- Current size of outstanding(send) messages
- Size of all sent messages (total)

- Current size of incoming (rcv) messages
- Size of all received messages (total)
- Route length
- Route count
- Size of dropped messages
- Count of dropped messages

In all cases the Lustre Operations Manual [5] should be the authoritative source for this information as it may change with newer versions of Lustre. If the number of outstanding messages is greater than 30,000 for two straight minutes, this check will return -2 and print the status of the current and previous check's outstanding messages to stdout. Which as was earlier noted will get picked up by Nagios and displayed via the web interface and in all notifications.

2.5.3 Lustre Device Check

With the release of Lustre version 1.6 a centralized filesystem configuration file that was required to start and/or stop the filesystem went away as a requirement. This means that a central location for querying information about the construction of the filesystem was not possible. At the OLCF we wrote a few tools to build, start, and stop the filesystem based on a configuration file that contained the backend storage device name (as it would appear on the OSS), OST device number, and the OSS that it should primarily reside on. The same information holds true for the MGT and MDT. Leveraging this information we developed a Nagios check to find if any device was not mounted; as this presents a large issue for users who are interacting with the filesystems. If all devices are not mounted, this check returns -2 and prints the device(s) that are not mounted to stdout.

2.5.4 Infiniband Health Monitor

At the OLCF we not only need to assure that the resources are available, but that the networks connecting the compute and storage platforms are working at their peak capacity. In that vein we developed a Nagios check that will report on the status of the Infiniband interfaces for a Lustre server or Lustre router. This check requires a configuration file that includes the device name, number of interfaces on the device, and the correct speed. While this check could be abstracted further to handle different device types, currently it only functions for Infiniband. If any interface is not in it's optimal configuration, this check will return -2 and print the devices' current configuration to stdout.

2.5.5 Monitoring Paths to backend storage

Another very important piece of information for the OLCF's Lustre installation is the use of multiple paths to the backend DDN 9900 storage. We developed a multipath monitoring Nagios check that will report the status of the paths to the storage. We currently look for two "active" and "ready" paths for every Lustre server. Our MDS nodes actually have 4 paths through two different Fibre Channel SAN's for redundancy to the MDT (two through each SAN). If we do not find at least two paths in this state, we return -2 and print the count of devices, number of good and number of bad paths to stdout.

2.6 Lustre Client-side Monitoring

While the majority of monitoring and determination of Lustre health is concentrated on the server side, checks can be written to verify that clients have the correct "view" of the filesystem. First one could determine if the filesystem is mounted on the node. This can be done via a "df" command and using grep for the filesystem, or you could cat /proc/mounts and grep for the filesystem. In either case, if the filesystem is not found (return code is not zero), the script should return -2 and print some useful text to stdout for Nagios to pick up and print in any notifications.

Secondly if a reference state is saved you could compare the output of 'lfs osts' or 'lfs check servers'. Even without a reference state you could grep for 'inactive' to get a hint that part of the filesystem (or all of it) is unavailable. This second option would be more preferable to the "df" method because the lfs commands will not "hang" if the filesystem is experiencing issues.

Listing 2: Output from lfs osts

```
# lfs osts /lustre/widow2 | more
OBDS:
0: widow2-OST0000_UUID ACTIVE
1: widow2-OST0001_UUID ACTIVE
2: widow2-OST0002_UUID ACTIVE
3: widow2-OST0003_UUID ACTIVE
4: widow2-OST0004_UUID ACTIVE
```

2.7 Monitoring headaches

In our current configuration, there may be times when the SNMP timeout results in a check failing to execute on a Lustre server. To more closely target the information we desire, we have extended the SNMP querying infrastructure in Nagios to support retrieving a “bulk get” of information and post processing that information on the Nagios server. We see issues in checking the process table on the Lustre metadata servers because of the quantity of Lustre processes. We run a local patch that increases our metadata thread pool to 2048, which causes the process table to become 7000-10,000 entries deep. The standard snmpwalk will not complete in 30 seconds (SNMP timeout). This is another example of how the monitoring framework must scale to meet future needs.

Another headache comes in the Lustre client side checking of filesystem mounts. If the filesystem is having problems the ‘df’ command will hang, causing the check to timeout. This is an early warning system for relaying filesystem problems, but can be unreliable for positive identification of filesystem issues. For example the node could have a bad connection to SION, and that would cause the ‘df’ to hang, but the filesystem would not have any problems.

3 Log Monitoring

The next step we use in monitoring the health of the filesystem relies on the logging information sent from all the components to a central syslog infrastructure. Assembling all of that information in a single place allows the OLCF to correlate events together.

3.1 Simple Event Correlator (SEC)

Using the Simple Event Correlator (SEC)[8], we can analyze log information near real time related to the backend storage. We additionally have rules in place that will alert if resources are low on a Lustre server. SEC parses a log stream in real time and if an event matches a pre-defined set of rules will trigger an action. SEC has the ability to tie multiple events together, but without replaying the log stream through SEC you cannot search through the triggered events.

For example the data in Listing 3 comes across the logstream.

Listing 3: Syslog message

```
Apr 11 09:56:12 widow-ddn1a1 ALRT:
    INT_PS    Failing Disk 21G, S/N
    GTF000PAJ8TYBF, Reason(Cmd Retry
    Error)
```

In Listing 4 there is a rule that will alert via e-mail that the disk in the DDN has been failed.

Listing 4: SEC rule for failed DDN Disk

```
type= single
continue= takenext
ptype= regexp
pattern= ([A-z0-9._]+)-(ddn[A-z0-9._
    -]+) DC_REC\s+Failing Disk (.*),
desc= failed_disk_$1_$2
action= event monitor[backup.warn] $1
    : DDN $2 Critical: failed disk $3
    ; shellcmd /bin/mailx -s "$1-$2
    failed disk $3" root@$1
```

In this ruleset we can match any DDN that sends it’s log stream to the central location, and send mail to different groups of administrators based on the hostname of the DDN. Rules like this are extremely powerful in quickly monitoring many disk arrays. It can be useful with new storage system installations as there is a higher potential for early disk failures in an installation.

Another example would be the Infiniband Connection Manager resource on a Lustre server being unable to create new queue pairs. The sample logstream is in Listing 5.

Listing 5: Syslog message

```
Jun  5 00:00:15 spider-oss30 kernel:
    ib_cm/2: page allocation failure.
     order:4, mode:0xd0
```

And the rule to alert the administrators:

Listing 6: SEC rule for ib_cm allocation failure

```
type= single
ptype= regexp
continue= dontcont
pattern= ([A-z0-9._-]+) kernel: ib_cm
    .*page allocation failure
desc= lustre_page_alloc_fail_$1
action= event monitor[kern.warn] $1:
    Lustre IB page allocation failure
    ; \
         shellcmd /bin/mailx -s "$1:
              Lustre IB page allocation
                failure" root@$1
```

Note that this issue was fixed with the introduction of FMR for the o2iblnd in Lustre 1.8 and a move from OFED 1.2 to OFED 1.3.

3.2 Splunk

The OLCF has a license for the software product Splunk[9], and we can use this product to replicate some of the functionality of SEC. Splunk cannot completely replace SEC, but has some additional features like a very nice interface, saved searches, trending, and searching indexed log data. Additionally the product is highly interactive and with the indexed data analysis and research into problems can be much quicker. This has the ability to make correlating backend storage events to Lustre filesystem events much easier. Work is currently ongoing at the OLCF to develop Splunk searches and alerts based on events triggered to syslog streams.

4 Future Work

4.1 Lustre Monitoring Toolkit

One large piece of health monitoring and availability software that the OLCF has yet to implement is the Lustre Monitoring Toolkit (LMT)[4] developed by Lawrence Livermore National Laboratory. This software has been used by several other sites with great success in monitoring performance and availability of Lustre resources. This work has been delayed several times by filesystem initial rollouts, upgrades, and problems with the production filesystems. Adding this data can help the OLCF to better understand the workload happening on the Lustre filesystems in Aggregate, and can also help to monitor usage on specific OST's which can affect perceived performance to the user or alternatively if the OST were to fill up, cause a partial outage for the filesystem.

4.2 Custom DDN 9900 Monitoring

One piece of information that is missing from monitoring the DDN 9900 storage arrays is a way to validate the configuration of the controller against a known good configuration. While this work may seem less than important, validation of configuration of luns on the storage controller, tuning parameters for the devices, network configurations for the storage controllers, and how the devices are presented to the host operating system are important groundwork to build a determination of health on. Using an expect script could allow us to gather the configuration information and do a side-by-side comparison and return an error code to Nagios if a deviation existed.

4.3 Server Side Client Statistics

With the roll out of Lustre version 1.8 on the server side the OLCF now has access to statistics about the behavior of clients with respect to the filesystem. There are significant challenges in parsing this data as the OLCF has had up to 26,000 simultaneous Lustre clients connecting to the filesystem. With 192 Lustre OSS nodes each with 26,000 directories of client statistics, parsing the aggregate data from a single client becomes quite compute intensive. Factor in most parallel jobs are running from more than one Lustre client, and the time it takes to gather the statistical information about a job at a point in time is not relevant because of skew in the time it takes to gather data. Work will have to be done to develop these features. One promising area of interest is the lltop[3] utility developed at the Texas Advanced Computing Center (TACC).

4.4 Rationalized printk

Parsing Lustre error messages is a very challenging task, made more challenging by the change in error text between released versions of Lustre. A "rationalized printk" function[10] has been developed at TACC and has some promise in being able to programmatically determine faults and failures in a Lustre filesystem. The patches are in-hand at the OLCF and we are working to integrate them into our future builds in hopes of furthering the analysis of Lustre error message analysis and reporting on those events.

4.5 Failover Awareness

Almost all of our current Lustre monitoring tools are not aware of the potential failover conditions that will allow the filesystems to remain available, but be slightly degraded in performance. Our next step is to make these tools aware of failover conditions and suppress reporting for nodes that fail checks but the failover partner has the resources.

5 Conclusion

Determining the health of a Lustre filesystem relies on the ability to detect and monitor events starting at the disk storage system layer, Lustre server hardware, site specific software that runs on Lustre servers, connections to the storage from the Lustre server, and several Lustre layer pieces. This complex puzzle can be simplified for some sites, but as the size of the installation grows, so does the complexity. As filesystems move into the Exascale era, the load of monitoring and determining the health of the filesystem will continue to grow, and so will the load on the monitoring (management) network. The current OLCF Spider monitoring load is at the tipping point of requiring a single Nagios server just to handle the load for monitoring the filesystems. As we continue to add checks, new filesystems (and thus new servers) it may be necessary to split out the filesystem monitoring from the rest of the OLCF infrastructure. This presents challenges for both the Lustre administration staff and the rest of the OLCF support staff.

The determination of health of the filesystem is important in delivering a storage platform that is highly available on many compute platforms to meet mission requirements. Quick diagnosis of problems and alerting system administrators to issues can lessen any potential outages to the central part of the OLCF's computing infrastructure. This effort is ongoing and adapting as the technologies are updated and as system knowledge evolves.

References

[1] Data Direct Networks S2A9900. `http://ddn.com/9900`.

[2] Lustre Filesystem. `http://wiki.lustre.org`.

[3] Lustre Load Monitor (lltop). `https://github.com/jhammond/lltop`.

[4] Lustre Monitoring Toolkit (LMT). `https://github.com/chaos/lmt/wiki`.

[5] Lustre Operations Manual. `http://wiki.lustre.org/index.php/Lustre_Documentation`.

[6] Nagios. `http://www.nagios.org`.

[7] NagiosExchange. `http://exchange.nagios.org`.

[8] Simple Event Correlator. `http://simple-evcorr.sourceforge.net/`.

[9] Splunk. `http://splunk.com/`.

[10] John Hammond. Rationalizing Message Logging for Lustre. In *Lustre User Group Conference*, 2010.

This research used resources of the Oak Ridge Leadership Computing Facility, located in the National Center for Computational Sciences at Oak Ridge National Laboratory, which is supported by the Office of Science of the Department of Energy under Contract DE-AC05-00OR22725.