

Installation of an External Lustre Filesystem using Cray esMS management and Lustre 1.8.6

Patrick Webb, Cray Inc.

https://www.usenix.org/conference/lisa13/technical-sessions/papers/webb

This paper is included in the Proceedings of the 27th Large Installation System Administration Conference (LISA '13).

November 3–8, 2013 • Washington, D.C., USA

ISBN 978-1-931971-05-8

Open access to the Proceedings of the 27th Large Installation System Administration Conference (LISA '13) is sponsored by USENIX.

Installation of an External Lustre Filesystem using Cray esMS management and Lustre 1.8.6

Patrick Webb
webb@cray.com

May 1, 2013

Abstract

High performance computing systems need a similarly large scale storage system in order to manage the massive quantities of data that are produced. The unique aspects of each customer's site means that the on-site configuration and creation of the filesystem will be unique. In this paper we will look at the installation of multiple separate Lustre 1.8.6 filesystems attached to the Los Alamos National Laboratory ACES systems and their management back-end. We will examine the structure of the filesystem and the choices made during the installation and configuration as well the obstacles that we encountered along the way and the methods used to overcome them.

1. Introduction

Every high performance computing system requires an equally high performance filesystem in order to properly manage the massive quantities of data that is produced by the computations ongoing on the machine. The physical installation of our system was performed by trained Cray hardware engineers. The unique challenges of our installation arose with the software portion of the installation. Software is usually the domain of the on-site system analyst team to install and customize to their needs, and in this case Cray has permanent on-site system analysts as part of that team providing the software expertise to install, test, configure and operate the filesystem software.

The installation is designed to be built as an externally connected filesystem that is mounted by the Cielo supercomputer[1], a Cray XE6 system operated by the Los Alamos National Labs, and one of their major HPC resources. Lustre was chosen as a solution due to the experience that Cray has with integrating Lustre into their computational environment, as well being able to provide extensive support for the filesystem.

Lustre is a parallel distributed filesystem, consisting of a series of metadata servers (MDS) which keep track of metadata objects, storage servers (OSS) which manage data storage objects, and object storage targets (OST) which physically store the data objects, arranged in a hierarchical format to allow the distribution of data across many devices. Clients first contact the MDS to begin their transaction, then communicate directly with the appropriate OSS nodes to read/write to an OST. The installed filesystem is connected to the mainframe via an LNet (Lustre Networking) network protocol which provides the communication infrastructure.

The system uses specialized LNet router nodes to translate traffic between the Cray Gemini network (the proprietary Cray interconnect) and Infiniband using the LNet protocol.

In this paper we will explore the methods used to install, test, configure and operate three Lustre 1.8.6 filesystems from the perspective of the permanent Cray on-site system analyst. The filesystems discussed consists of two 2PB systems, one 4PB system, and two 350TB testbed systems. The PB filesystems are attached via fibre-channel to 12, 12 and 24 racks of disk arrays respectively, configured in a RAID6 8+2 format. Management is by a single Dell rack-mount server providing boot images and configuration management to the filesystem nodes. The focus will remain on the Cielo portion of the installation, since many of the unique challenges we encountered manifested within Cielo's environment and scale.

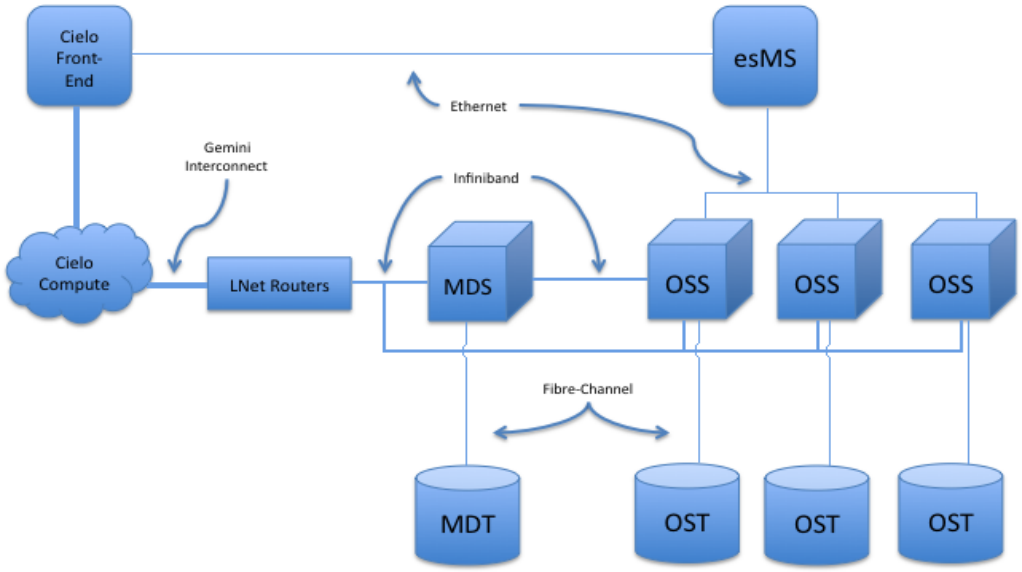


Fig. 1 A simplified diagram of Cielo's Lustre filesystem

2. System Capabilities & Overview

The Cielo Lustre filesystem (dubbed an esFS or external service filesystem in Cray parlance) is a 96 OSS, 6 MDS system connected to 48 storage racks with a total storage capacity of 8PB, managed by a single external service management server (esMS). All of the blades (OSS, MDS and esMS) nodes are Dell R710 blades. The storage racks consist of 128 2TB hard drives apiece configured into an 8+2 RAID controlled by a redundant LSI controller. The network routing on the Cray system side is handled by 104 service nodes configured as LNet routers. The interconnect between the storage racks and the Lustre servers is a fibre channel connection, and between the Lustre servers and the Cielo system is an Infiniband network. The

Infiniband network on Cielo makes use of two Director class Infiniband switches to manage the network. The management network between the esMS and the OSS nodes consists of basic 1GigE Ethernet.

The sum of resources are then split into three different filesystems managed by the single esMS blade: two 2PB filesystems and one 4PB filesystem. Each of the two 2PB filesystems are assigned 12 racks of disks, and the 4PB filesystem is assigned 24 racks.

The Infiniband network is shared between all three filesystems, and connects the Lustre components to the compute portion via an LNet network managed by the Cray LNet router nodes. The LNet routers are pooled together and shared by all three filesystems instead of separating them into smaller groups.

The software stack consists of three separate portions. On the Cielo side, the LNet routers use the Cray Linux Environment (CLE) OS customized with the necessary LNet and Lustre kernel modules. The esMS uses a SLES11 base OS. The OSS and MDS nodes are managed using Bright Cluster Manager (BCM) software running on the esMS. BCM is used to manage the different boot images and configuration options for the OSS and MDS nodes, which PXE boot their OS. The OSS and MDS nodes run a CentOS 5.4 base system customized by Cray with Lustre 1.8.6 software.

The performance of the filesystem is measured across several dimensions, and is described in detail in section 3.4.

3. Challenges

3.1 Initial Setup Challenges

The setup of the esFS system would be the responsibility of the Cray on-site system engineers and system analysts to install, test, and operate the filesystems. The first challenges manifested at the Cray factory where the initial test and development systems would be constructed and configured before shipment. These test systems would be the template for the larger Lustre filesystems, as well as platforms for test and development. One of the site analysts travelled to the Cray factory in order to participate in the construction and learn directly from the engineers assembling the system.

The following elements were constructed at the factory for the test and development system: OSS & MDS hardware configuration, Infiniband network, fiber connections to disk racks, esMS hardware configuration, LUN (a type of storage object) creation, esMS software stack, and the OSS & MDS software stack. The actual Lustre filesystem was not created, and the LNet network that connects the Cray compute hardware to the Lustre filesystem was also not assembled at the factory. The security stance of the LANL site is such that it requires incoming systems to be built up from bare metal, meaning that any assembly at the Cray factory would be useful only for testing purposes. Thus it was critical for the on-site system analysts to learn as much as possible from the Cray engineers. The task of building the entire filesystem and its management node (the esMS) from the ground up would be their responsibility.

3.2 Site Set-up Challenges

The first steps in bringing up the Lustre filesystems was to first build and configure the esMS node which would provision and monitor the OSS and MDS nodes. Despite the fact that the project was on schedule, there was significant pressure to stand up the filesystem as quickly as possible and to not deviate from the setup of the test & development system. However, there was a key critical difference between the test & development system and the full-scale production filesystem: the full-scale system was meant to have a backup esMS node with automatic failover configured. The test and development system had no such esMS backup system configured. The consequence was that the full-scale system was initially configured with only a single esMS node instead of the intended (and required by contract) primary/secondary esMS configuration. Cray documentation for adding a secondary esMS to an already configured and running single esMS didn't exist. We would be the first site to execute this task.

Building a single esMS was a straightforward procedure. It uses the SLES11 operating system as its basis, modified to add Cray Lustre control packages. BCM uses its own installation tool that requires inputting necessary configuration options (network, etc.) and allowing it to set up the entire OS under BCM management. Custom Cray scripts for monitoring and managing automatic failover were also installed at this time.

Once the esMS was fully built and configured it was time to power on and set up the OSS and MDS nodes. During power-up each of the physical nodes were checked in order to confirm that the BIOS settings had been set properly at the factory. A small number of nodes had been overlooked and needed to be reconfigured on-site. Finally, the MDS/OSS node boot images were configured into BCM.

3.3 Configuration Challenges

We decided that we would use the configuration from another Cray installation site, the National Energy Research Scientific Computing (NERSC) Center, as the basis of our own configuration. This met with a few obstacles from a managerial perspective. The desire to have as safe and stable system as possible meant that there was a great deal of pushback against any sort of deviation from a known quantity, namely the NERSC configuration. However, we faced a few issues that made duplicating NERSC unreasonable. First, the scale of the LANL filesystem was much larger than NERSC. Second, the LNet and Infiniband network at LANL used a very different set of hardware. Finally the software stack at LANL, unlike NERSC, was productized into a cohesive package managed by BCM.

3.4 Testing & Acceptance Challenges

The testing plan for the Lustre filesystem measured the baseline hardware performance, the ability to meet a minimum level of filesystem performance, and the ability of the system to ride through an interruption of one or more of the hardware components. Each Infiniband link between MDS/OSS nodes and LNet nodes were tested at ~2.7GB/s average per link. Aggregated, the system saw a

maximum raw throughput of ~70.3GB/s between 52 LNet and 48 OSS nodes. Under load, the system saw a peak of 77.4GB/s for a 2k core job (65.5GB/s required). Metadata operations showed ~22k-24k creates/11k-18k deletes per second (10k/s each required) when each core operated on its own file. All performance tests passed with only minor adjustments to meet requirements.

The fault injection tests tested for events such as a power failure, node crash, or network failure. The deliverables required automatic component failover and stated that the system would be able to automatically failover an ailing component in the following circumstances: A normal shutdown of an LSI Controller, MDS, or OSS node; An unexpected power failure of an LSI Controller, MDS, or OSS node; A loss of an LNet router; The loss of network connectivity between the Infiniband switch and an MDS, OSS, or LNet router; Loss of one or both fibre channel connection between an OSS node and an LSI controller. Of these tests, all had to either continue to serve data albeit at a degraded performance, or signal an IO error that would unambiguously indicate that IO was the fault of the job failing.

Tested failures degraded performance during recovery from no measurable impact (LNet router failure) to as much as 87% of peak, and/or caused an acceptable IO error (OSS, LSI Controller, etc.). Lustre attempts to rescue transactions from the failed components, and transactions that don't recover are discarded to avoid storing corrupted data. After recovery, performance degrades roughly proportional to the amount of filesystem resources made unavailable.

Despite these requirements, the monitoring and failover scripts were released to the customer capable only of automatically failing over a node if network connectivity was lost, or if the node panic'd and froze but remained powered on.

The orderly shutdowns of the various hardware components were not designed to initiate a failover on the assumption that if an orderly shutdown were taking place, that the responsible administrator would have either quiesced the system or manually instigated a failover in order to power off a node. A node simply being "off" meant that the monitoring system would not know if it had already performed a failover (A failing node is "STONITHed", or powered off, in order to ensure that it will not interfere with its backup.) or if that node had freshly failed. Erring towards safety, the monitoring software would not initiate a failover for a node that was simply turned off. This behavior also affected how the system responded to an unexpected power loss, namely that it did not initiate a failover.

Other fault injection tests were never designed to initiate an automatic failover, or even interrupt operations of the filesystem. The LSI controllers used a shared power supply that was internally redundant and powered pairs of controllers, so a power loss would always affect both, but never a single controller. Fibre-channel connections were not designed to be monitored by the esMS or the OSS/MDS nodes, and their redundant connection meant that losing one connection meant there were still routes available to connect to the disk racks. The fault injection testing proved as much, with minimal impact on performance.

The LNet network had another set of challenges that only arose at scale. The LNet network check that ran on each of the OSS and MDS nodes would ping a randomly chosen peer somewhere out on the Infiniband network, and if that ping were successful it would report back that it had passed. If that ping timed out, then

it would report a failure and the esMS would initiate a failover. Internally, BCM executes these checks serially every few minutes. At scale, we found ourselves monitoring 96 nodes spread across three different filesystems. The check executed every 60s, but it took as much as 90s for a failed node to report that its ping had timed out and failed. Due to the serial nature of BCM's testing, this meant that if a node near the end of the list of nodes to check were to fail, the timeout for the ping (and thus the affirmative "failed" condition) would not complete and notify the esMS. The esMS assumes a 'pass' if not explicitly notified that a node had failed, and would have already moved on to the next iteration of checks and discarded the results of the previous pass. We needed to change the behavior of the monitoring scripts dramatically.

The solutions to our mismatched expectations of our monitoring and failover scripts are described in section 5 below. It caught the management team off guard, and required close collaboration between the developers and field personnel to effect a solution in the field.

3.5 Operational Challenges

Few operational challenges arose. The stability of the filesystem was such that its popularity among the users rose to the point of the system beginning to show signs of strain due to heavy load. Despite users doing their utmost to eke every last bit of performance out of the filesystem, it remained, and still remains, incredibly stable.

Once the system was up and tested and released to users, we began to see a series of false-positive events triggered by the network checks in our monitoring scripts. The first check to throw out false-positives and cause unintended automatic failovers was the LNet network connectivity check. We had already tinkered with the timing during the initial fault injection testing to validate the check. Now the check was too sensitive. Lustre uses only one transaction credit allocated to pings, and prioritizes that very low. High traffic on the system meant that a ping could easily end up timing out if its wait in the queue took longer than 90 seconds (the test timeout parameter) to complete. Subsequent LNet pings could and would succeed, but the health check relied on a single ping to initiate a failover event.

Even checks such as TCP ping and power status checks began to see events such as these as the system load increased and the responsiveness of the OSS and MDS nodes became sluggish. Since all of these checks relied on a single ping or poll, it became more and more likely that one of those checks would time out. Without retries of these checks, a healthy yet busy node would be considered unhealthy. Again, the design of our health checks had serious flaws.

4. Resolutions

4.1 Initial Set-up

Education of the site system analysts was critical in this phase in order to ensure that the proper expertise would be on-hand when the system would be built on-site. This was accomplished by sending one of the site analysts to the Cray

factory for a week to shadow the system construction and spend face-to-face time with the developers. By having the site analyst in the factory, that analyst was also able to get hands-on experience with building up the filesystem while having the Cray development team on hand to instruct them through the process. Valuable to the developers was the ability to closely watch how an admin who had not been involved in the design of the system would follow the installation documentation, and thus improve the quality of the documentation.

4.2 Site set-up

Arguably the biggest obstacle during the set-up was the installation of the backup esMS. Lacking Cray documentation, the admins performing the installation found themselves in a difficult position. The solution was to bring in direct assistance from the developers to bypass and fix issues in the procedure that prevented moving forward. Little troubleshooting was needed, as this was fresh ground. The process involved repartitioning an in-use disk to make partitions that would be mounted by the backup esMS, then migrating data to the new partitions. Next, the backup esMS would mount those portions and make an initial copy. From there, the backup esMS would monitor the primary for failure, and make periodic incremental updates from the primary. The process of adding the backup esMS highlighted many weaknesses in the documentation and initial setup configuration that needed clarification and correction, and instigated the improvements to the Cray documentation. Overall, despite the problems it introduced, the delayed inclusion of the backup esMS improved the quality of the entire esFS installation procedure, which can then be shared with other Cray sites.

4.3 Configuration

The NERSC configuration served as an excellent starting point for the initial setup and configuration. The main resolution to this particular point of the installation was to make effective arguments for the necessity of changing the configuration to better match our hardware. The integrated software stack meant that configuration for the OSS and MDS nodes could be managed from a central location. Scaling was larger, so certain parameters in the LNet configuration in terms of numbers of transfer credits and length of timeouts had to be adjusted upwards in order to handle the additional load. Finally, the biggest difference was the configuration of the LNet routers into a single pool shared between all three filesystems rather than dividing them up into separate networks or even down to a fine-grained routing. Pooling the routers has potential loss of performance due to needing to switch traffic, and risks of instabilities if an LNet router fails spectacularly. However, the Director-class Infiniband switches provide plenty of horsepower to allow a pool configuration to work without a performance impact. With a pool of LNet routers, the set-up and configuration was much simpler (simply place them all into the same network), and it provided a great deal of redundancy in that if any LNet router failed, the traffic that router was serving could easily be sent through any other router on the network.

4.4 Testing & Acceptance

The Cray development team quickly provided an updated rpm that enabled failover for the contractually required failover triggers. The scripts were in fact already capable of performing failover actions in all required cases, but the tests simply had not yet included the code to initiate those actions. The updated RPM simply empowered those tests to carry out failovers.

In-field rewrites of the monitoring and failover scripts were the solution to the problem of LNet network checks not completing and bypassing themselves. We first monitored the return values from the nodes. Noting that nodes at the end of the node list weren't reporting back before a new health check started we then compared timing values. Noting the mismatch between LNet ping timeout, we then wrote into the check script a progressive timeout logic that checked to see if the test passed immediately, within 5 seconds, 10 seconds, etc. until ultimately the test failed and a failure was reported. The code sped up the checks on a healthy system, and left plenty of time available for a failed check to fully timeout. The modifications were fed back to the development team, who integrated them into the code base. However, the new code did not yet address the issue of an otherwise healthy but heavily loaded system from failing a single lnet ping check when a re-try would confirm that the lnet network is perfectly functional.

Poorly understood fault injection tests, namely the LSI controller tests, were solved through frank and earnest discussions the engineers and management staff. The previously existing trust between the two parties made it easy to explain the technical realities, and agree on the necessary reinterpretation of the results. All the people working were fully invested in putting forth their very best work.

4.5 Operations

Once again, in-field changes to the monitoring scripts were necessary to check the status of the networks without failing due to a mere single TCP ping, or LNet ping, timing out. We were able to discover the false positives examining internal Lustre stats, and discovering that the system would periodically oversubscribe its available credits, including the ping credit. The Cray development team took a proactive approach, and added into the code base retries for all appropriate health checks. The system analysts implemented a field fix of disabling active failure in favor of notifying via pager the analysts in the event of specific health checks failing. They kept field fixes in place while waiting for the next polished version of the esFS monitoring scripts were released.

5. Lessons Learned

Recognize and react to the differences between the test and production systems. – The difficulty of adding the backup esMS after the full installation was a troublesome and dangerous procedure that was forced by prioritizing the deadline and slavishly sticking to mirroring the test & development system. If the production and test systems will differ by design, prepare for the installation plan between the two to differ as well.

Documentation of the underlying structure is incredibly valuable. – Knowledge of the underlying structure of the various parts of the esMS/esFS systems was critical to solving many of the build problems, namely the esMS backup.

Embrace the fact that your installation will be unique. – A great deal of discomfort was felt over the fact that the actual configuration parameters differed from the model. Realizing that we must differ smoothed out the decision making and allowed for more rational choices in configuration.

Test all of the contractual requirements as early as possible. – We came very close to having real problems with contractual obligations in our failover scripts. While we were able to add in the required aspects, had we tested them earlier there would have been less pain involved.

Empower the local site analysts to create and implement fixes in the field. – The fact that the local analysts were not only able, but encouraged to implement their own fixes led to quick and effective solutions. It gave the site analysts a sense of ownership of the system, and gave the developers a short-cut to improving the overall code base.

6. Conclusions

The installation of a new filesystem is a complex task with many moving parts, that was only complicated by the fact that many tasks that could have been performed and tested in a factory setting were required to be completed in the field. In addition, the entire product was one of the first releases of the actual productization of the Cray esFS filesystem. The challenges of building such a large installation were met with a great deal of dedication and expertise on the part of the developers and site system analysts. The expected challenges of configuring the different aspects of the network, formatting the filesystem, installing the management software, testing performance, etc. were all present and expediently dealt with.

We were able to respond to the various unexpected challenges with in-field fixes that were later integrated into the release products and made available for other sites to use. Additionally, we were able to keep to the timetable due to the proactive nature of the implementation of these fixes in the field rather than waiting on a development cycle to provide a patch. This kind of dynamic relationship with the home office based developers proved to be an exceptionally strong one that produced effective solutions very quickly.

The final result of this work is an exceptionally stable and popular filesystem that has exceeded the users expectations for availability, stability, and performance. While improvements can always be made, the efforts made during the initial set up will, in my opinion, pay off in terms of the long-term health of the filesystem.

References:

[1] – C. Lueninghoener *et al.*, "Bringing Up Cielo: Experiences with a Cray XE6 System", in Proceedings Large Installation System Administration Conf., 2011.