Cray Lustre Update

CUG 2014

Cory Spitz
Storage and Data Management
Cray Inc.
St. Paul, MN
spitzcor@cray.com

Tom Sherman
Storage and Data Management
Cray Inc.
St. Paul, MN
tsherman@cray.com

***Abstract*—Cray deploys 'value-added' Lustre offerings with modifications and enhancement over base Lustre releases. In order to supply the best technology, Cray has chosen to integrate upstream feature releases that may not have a community supported maintenance branch. With that, we carefully plan integration of upstream community features. This paper will discuss how the community features will map to Cray releases and how we facilitate them via our participation with the community and OpenSFS. It will also highlight Cray specific features and enhancements that will allow our Lustre offerings to fully exploit the scale and performance inherent to Cray HPC compute products. In addition, this paper will discuss the upgrade paths from prior Cray Lustre software releases. That discussion will explain the migration process between 1.8.x based software to Cray's current 2.x-based offerings, including details of support for 'legacy' so-called direct-attached Lustre for XE/XK that had previously been announced as EOL. Details for upgrading Sonexion file-systems will be included as well.**

Keywords: Lustre, OpenSFS, Sonexion, CLFS, Direct-Attached Lustre, roadmap, file-systems

I. Introduction

At Cray, we believe that we've reached an inflection point with Lustre in terms of performance, features, and stability. In our view, Lustre 2.5.x represents a solid foundation to build on, unlike prior 2.x releases. Therefore, we plan to deploy a Lustre maintenance stream based on the upstream Intel maintenance branch plus Cray patches *and* a feature stream also with patches that is based on the OpenSFS Lustre releases for those deployments that need new functionality or for those customers that want the latest version.

How we carry out our releases is rooted in our participation with Lustre as a technology, our involvement with the Lustre community, and our commitment to excellence in HPC. This paper will not have a technical focus, but it will explain our approach with Lustre and then cover some practical information associated with moving to the improved quality and stability of Lustre 2.5.x.

II. The Role of Lustre at Cray

Cray is focused on whole system performance, not just demonstrating performance with synthetic benchmarks. That kind of performance needs to be repeatable and dependable in many HPC environments. The overall system should handle changing loads and increased demands without ill effect. With that comes a view that I/O subsystem performance and compute capability ought to be robust and balanced. Further, the best overall system balance is achieved by balancing throughput within the I/O subsystem. Backend disk bandwidth not ought to be limited by server capabilities and server bandwidth ought not to dramatically exceed network bandwidth (including routing bandwidth) and so on. The best parallel file-system that can preserve the performance of the underlying HW architecture while meeting these goals is Lustre. Lustre is best because its architecture allows it to easily scale-up and linearly scale-out simply by adding more storage and servers.

III. Lustre Products at Cray

Because Lustre is such a great fit for HPC, Cray deploys Lustre on **every** XE and XC class system. Cray first provided only so-called direct-attached Lustre (DAL) with storage that is directly connected to the compute mainframe via service and I/O nodes that host the Lustre servers. With the ability to route LNET, the transport mechanism that Lustre uses, we've added so-called external Lustre file-systems. First known by the name esFS, these are now called Lustre File System by Cray, or CLFS. And Cray's new flagship Lustre storage product is Sonexion. Sonexion offers a modular, compact design and keeps costs low by eliminating HW based RAID and additional associated cables.

Cray sells and deploys our Sonexion and CLFS storage products into 3rd-party systems. The Cray Cluster Connect (C3) client provides connectivity to the Lustre storage, which is based from the same common source that we use for our CLE clients. C3 clients can be distributed onto any number of Linux cluster types.

IV. Cray's Internal Lustre Strategy

Because Lustre has been a critical component to Cray's systems offerings for such a long time, we've built up a strong Lustre storage team and we now have over 10 years of institutional experience. In addition to the SDM File-System team, whose primary job is to integrate new Lustre features into Cray products, we've got storage architects, performance analysts, and service analysts well versed in

Lustre with all of its complexity. We realize that we can't do it all on our own, so we augment the virtual team with support help from leading support vendors.

Our goal is to enable our customers to in turn let their users accomplish goals. To do that, we generate and gather requirements, take requests for enhancements from the community and directly from our customers, and then we prioritize these requirements and best see how we can provide the needed capabilities be it through our engineering efforts or guiding OpenSFS to fund technology enhancements.

Through it all we keep close ties with our customers to ensure that they continue to meet their goals and the products meet their needs. This customer intimacy is part of our corporate values.

V. Cray's Community Lustre Strategy

Cray demonstrates leadership in the Lustre community through our involvement in OpenSFS. OpenSFS is a non-profit technical organization focused on high-end open-source scalable file system technologies, primarily Lustre. OpenSFS aims to foster collaboration among entities developing and deploying Lustre, drive technologies for future requirements into open, scalable file systems, and deliver Lustre file system releases with a roadmap designed to meet those goals.

Cray is an original founder and promoter of OpenSFS. Promoter is a technical term inside OpenSFS. As a promoter Cray pays $500k annual dues and has voting rights on the board. These voting privileges ensure that Cray can help steer OpenSFS funding to meet our customer's needs.

In addition to funding popular improvements such as DNE and LFSCK, OpenSFS funds feature releases of Lustre through the Community Tree maintenance Contract, which is important for Cray's release strategy because Cray predicates our releases on the upstream OpenSFS releases. By contract, OpenSFS funds Intel (née Whamcloud) to create Lustre 'feature' releases every six months. Intel also chooses to fund its own maintenance branch, which it creates with input form the community. They've chosen to create a maintenance branch nominally every 18 months. This would mean that as a branch lives for 18 months it would skip two feature releases before the next branch is established. However, Intel responds to community demand and they recently created a maintenance branch for both 2.4.x and 2.5.x releases. The next maintenance branch is undecided by the community, but it will likely be b2_8, from which Intel will release quarterly (nominally) maintenance releases.

VI. Cray's Lustre Release Stategy

Cray SDM would like to continuously demonstrate excellence in Lustre storage for HPC by constantly improving quality, performance, and scalability. Cray's test process and procedures are among the best and before we release any software we establish and meet stringent quality metrics. Our quality standards are rigorous and evolving to further improve dependability and reliability as we continue releases. We're doing a number of different things to improve quality along development.

Cray also considers performance an integral part of our quality metrics. As such we test a wide gamut of use cases and measure performance, tracking any slip from build to build and release to release. We've developed a test framework called *LATE_perf*, which is an extension to *LATE*, our Lustre Automated Test Environment. It allows us to easily deploy performance regressions tests onto systems, which we run on a regular basis to catch any performance regression at the most ideal time, before it is integrated into the source code.

Cray has worked very hard on our test process, especially scale and load testing and we're working with OpenSFS's Community Development Working Group (CDWG) to share our know-how and in turn improve the community test plans and release criteria for the OpenSFS funded feature releases. As such, Cray is working to share our scale and load test mix with the community.

Cray also continuously tests *master*, the canonical Lustre source branch in order to provide feedback soon enough in the process to help improve quality for the next feature release. We test release branches and release candidates as well. We provide the test feedback through the CDWG and by opening tickets in the public community JIRA tracker.

To ensure that we contribute back to the community as well, we work code fixes to the *master* code line. Cray works with our support vendor to fix bugs and we encourage them to supply fixes directly to the *master* line as well. In fact, Cray does not consider an issue resolved with our support vendor until the patch has been resolved for Cray ***and*** lands to *master*.

Despite this, there is risk that the feature releases can destabilize the base and so we take care that core functionality is not hindered before we release. If a feature can be disabled then Cray may choose to release our Lustre with the limitation declared. We try to turn around feature releases inside of six months, but we shall release only when ready.

The turnaround time on Intel's maintenance releases are much faster due to the inherent limited scope and amount of change in any given maintenance release. In fact, it is more or less continuously releasable. We are able to gauge the potential release quality by constantly and automatically testing changes to the upstream maintenance branch.

We have a *cron* job that pulls upstream changes, applies them to our in-house version of the maintenance branch, builds, and finally deploys the RPMs into a location where another automated process installs onto a running system that is reserved each night to run automated regression and load tests. A human then assesses the results in the morning and if they are acceptable, the person signals to the script that the tests passes and it automatically integrates the changes into the tree. And if other changes are landed in the meantime, the patches will be rebased and retested the following test period. This process allows us to continuously develop and test our maintenance branch while never having to merge upstream releases with our local changes en masse or batch up smaller changes for integration in a bunch, which might make it harder for us to measure the effects of a single patch or find a specific regression. This has been a fantastic

tool for continuously improving the quality of our trees by adding upstream bug fixes on a regular basis.

What we name the release is also part of our strategy. Although Cray begins testing OpenSFS feature release candidates when they are available, there isn't enough time in the community release cycle to bring the software up to rigorous Cray standards. As such, we continue to enhance and stabilize the upstream sources before we release, yet we don't reversion the code. We call this model Lustre 'plus patches' because we don't re-brand the upstream release name and it implies that we carry improvements to the base Lustre release.

VII. Lustre 2.5 as an Inflection Point

With the advent of Lustre 2.5 we see an inflection point in Lustre in terms of features, performance, and stability. With the landing of a number of OpenSFS funded projects, there are an almost irresistible number of attractive features in the current release. HSM seems to be the killer feature. In fact, Intel responded to community demand to dub b2_5 a new maintenance branch because HSM support didn't arrive in 2.4 and no one wanted to wait until 2.7 to be released from b2_7, which would have been the next maintenance branch to include HSM.

Cray had been continuously deploying each Lustre feature release beginning with Lustre 2.2. We couldn't choose to deploy Lustre 2.1, despite the fact that it had a maintenance train associated with it, because it lacked support for SLES 11 SP3, the base kernel for our compute node OS. We went to 2.3 because it significantly raised the bar in terms of performance and quality. Lustre 2.4 did the same and had the added benefit that it was going to have a maintenance train associated with it from Intel. In our opinion, Lustre 2.5.x represents another jump in desirability and it too has another maintenance train from Intel. While we might expect 2.6 to raise the bar again, while also delivering LFSCK enhancements and striped directories to DNE, we feel that 2.5.x finally presents a point where we my slow down and offer a Cray maintenance stream based on Intel's maintenance releases.

We'll slowly integrate Lustre 2.5.x into all of our Lustre storage products over the course of 2014 and early 2015. We're starting with the release of 2.5.0 in CLE 5.2UP00, although 2.4.1 will remain as the default client choice in that release. Our flagship storage product, Sonexion, will see 2.5.x based Lustre servers come with the Neo 1.6 release, which will first be available early 2015, but we hope to pull that in to late 2014, perhaps in an limited availability fashion. The Sonexion product will stay on the 2.5.x code line for the foreseeable future precisely because we view 2.5.x as a good foundation for top-tier Lustre storage.

CLFS will use 2.5.x as well, but with more attractive server features like enhanced online file-system checking from LFSCK phase 2 & 3 and striped directories, Cray will put Lustre 2.6 on our CLFS roadmap, although it may arrive as late as Q4, 2014.

Cray SDM plans to create a release feature stream and a maintenance feature stream to support our storage products. But the Cray HPCS team will retain control of what they put in their distribution. CLE will include regular maintenance releases, but it may only include new feature releases as needed. For example, it may be necessary to choose a newer client to pair with SLES 12 due to kernel support issues similar to why 2.2 and later were chosen for CLE based on SLES 11 SP3.

VIII. Upgrades to CLE 5.2 UP00

Previously Cray had announced end-of-life for CLE support of the XE platform and CLE 4.2 UP02 was to be the last release. However, due to response in demand for continued support of newer kernel releases for XE systems, Cray had decided to continue XE platform support again in CLE beginning with CLE 5.2UP00. The release of this software for XE platforms is now planned for May 2014.

This change of plans presented a challenge to retain support for direct-attached Lustre systems. That's because the DAL systems ran Lustre 1.8.6 and they would have to be upgraded and converted to Lustre 2.5. We couldn't retain the capability to use Lustre 1.8.6 because neither the client nor server can operate on SLES 11 SP3 kernels. Further, we couldn't keep the servers at 1.8.6 while moving the clients ahead, even if we moved the server to a supported kernel like CentOS 5.x because 2.x clients cannot interoperate with 1.8.x servers. Therefore, we were required to move both clients and servers forward. This was challenging because there was a lot more work needed than simply upgrading. We call this out as a migration step because the on-disk metadata format changed for Lustre 2.x systems.

Fortunately, we had previously produced upgrade capability for the CLFS system beginning with its GA release in November 2013. We could then parlay our efforts into doing the same for the CLE environment, which we've now done for CLE 5.2 UP00. Because both Lustre 2.x doesn't support SLES server kernels (at least in initial 2.x releases) and ESF is based on CentOS, we needed to and chose to switch the Lustre servers in CLE to a CentOS base. This was a difficult proposition due to the CLE system management framework, which relies on a shared root, and which despite the name cannot be easily shared among desperate OS types. Therefore, we needed to develop a method to manage the Lustre server node roots. The DAL capability is the first user of the CLE IMPS facility, which provides improved image management such that we can boot CentOS based Lustre servers and SLES based login and compute nodes side-by-side in the same installation.

The upgrade steps are completely documented and easy to follow as we've employed and extended *lustre_control* to carry out much of the work. The following presents an overview of the steps to help set expectations. Mostly these are comments around the special migration steps needed to update to a newer quota format and update the file-system

metadata with additional information including the File IDentifier, which is used in 2.x to identify files in lieu of inode numbers.

The upgrade begins by installing the file system definitions from the *fs_defs* file via *lustre_control* and performing a *write_conf* operation. Lustre 2.x also adds an object-index (OI). The OI exists to map inodes to FIDs and is created by scrubbing the file-system. It is automatically executed when first upgrading to 2.x. Other scrubbing procedures will update the new linkEA, which is an extended attribute the links the inode to it's parent FID, and FID-in-dirent, which adds the FIDs to directory entries so that the MDS doesn't need to read each inode to list the files in the directory. Since the FID-in-dirent is a process that prevents downgrading the file-system back to 1.8.x, it is documented as a manual step.

Both OI scrub and adding FID-in-dirent (via scrub) process quickly, but the time that it takes to finish depends on the number of files in the file-system. Both operations are fast and can handle around 100k objects/scanned and updates per second.

Despite the ability to downgrade before FID-in-dirent is enabled, Cray recommends enabling it as documented in the upgrade procedures. Because we'd have to downgrade the compute side as well to maintain client-server interoperability, and CLE cannot be downgraded in this manner, Cray won't support fallback to CLE 4.x.

The quota changes bring various improvements including the ability to change quota limits while targets are offline, add or remove OSTs without corrupting space usage information, and no need to run a *quotacheck* operation. In addition, quota accounting and enforcement are now controlled separately. As part of the upgrade process, old quotas limits are retained. Note that if you didn't use quotas in 1.8.x then quota accounting will be enabled upon upgrade. Remember, enforcement is separate. Enabling quota accounting will take time and the speed is similar to the old *quotacheck* operation.

Finally, as a part of release testing for upgrades and migrations, Cray has worked to benchmark systems before and after upgrade to ensure that there are not any drastic performance issues. We have found one or two significant issues, which we've localized to the client. We, along with our support vendor and the community, have a good handle on the issues and we expect both future SDM Lustre feature and maintenance release to resolve the remaining issues. Please read the CLE 5.2UP00 release README for the performance details.

IX. Preview of Cray Lustre Enhancements

Cray has begun work on some exciting new features including pingless clients with imperative eviction and improved LNET RAS and re-routing. Because these features require node-health data, they'll initially only be available for Cray CLE clients. Also, if CLFS or Sonexion has non-CLE clients both features can still be used.

Cray plans to stop the Lustre ping evictor function on servers by running pingless clients. Pingless clients will allow us to eliminate OS noise and jitter contributed by the interrupt to ping the servers. We plan to incorporate Cray node health data to inform servers when clients are dead so that they don't need to wait for the lack of ping before carrying out an eviction. This will also improve the responsiveness of the Lustre file-system. A node-health agent will instruct servers to imperatively evict clients. The concept is similar to the existing imperative recovery feature.

We also plan to leverage CLE node-health data to stop pinging LNET routers in order to detect router health. A node-health agent will inform LNET peers of failed routers so that they can modify their routes accordingly. This will allow us to stop queuing messages to nodes the system knows to be down. To add to the improved responsiveness and another subsequent reduction in OS noise and jitter, we are also re-evaluating various LNET timeouts along with the improvements to see if they can also be reduced or eliminated.

X. Summary

Cray is ever improving Lustre and we've reached a milestone with Lustre 2.5.x. The software foundation is good enough to build upon, so Cray SDM will begin offering a maintenance stream based on it while we continue a feature stream based upon the OpenSFS Lustre feature releases. And with all of our storage products offering Lustre 2.5.x versions over the next sixth to nine months, all of our customers will be able to benefit from the hard work of Cray, our support vendor, and the community of an ever improving Lustre.

Acknowledgment

Thank you is owed to OpenSFS and its community members for maintaining and improving Lustre as a vendor-neutral parallel file-system. We'd also like to thank our Lustre support vendor, Xyratex, and Intel HPDD as well as they have both helped us solve so many earlier Lustre 2.x bugs, which now allows us to offer a quality maintenance stream.

About The Authors

Cory Spitz leads the Storage and Data Management File-system team at Cray. He has worked at Cray for 13 years and has been involved with Lustre since 2009. He is an active participant in OpenSFS, particularly the CDWG, TWG, and the contract Project Approval Committees.

Tom Sherman is the Storage and Data Management Director of Product Management at Cray and has responsibility for all Cray storage products including Sonexion, CLFS, and TAS.