

Large Lustre File System Experiences at NICS

Troy Baer, Victor Hazlewood, Junseong Heo, Rick Mohr, John Walsh, *University of Tennessee, National Institute for Computational Science (NICS)*

ABSTRACT: *The University of Tennessee National Center for Computational Sciences (NICS), located at the Oak Ridge National Laboratory (ORNL), installed a 66,048 core Cray XT5 in December 2008. A 2.4PB lustre file system was configured as part of the XT5 system. Experiences at NICS in configuring, building, monitoring, administrating and maintaining such a large file system will be presented. Topics include, hardware configuration, Lustre component configuration, purge versus quota policies, performance tuning and reliability issues.*

1. Introduction

NICS

The National Institute for Computational Sciences (NICS) is funded by the National Science Foundation and managed by the University of Tennessee (UT). NICS staff work at the UT Joint Institute for Scientific Computing building and the NICS computing resources reside in the main computational and computer science building computer room both of which are located on campus of Oak Ridge National Laboratory (ORNL). NICS currently provides a 66,048 core (8256 dual quad-core AMD Opteron processors) Cray XT5 system called Kraken which has 99 terabytes of memory and 2.4 petabytes of dedicated formatted disk space. 4416 nodes (35328 cores) have 2 GBytes of memory per core and 3840 nodes (30720 cores) have 1 GByte of memory per core. NICS shares a 20 petabyte HPSS archival storage system with the DOE Leadership Computing Facility at ORNL. The Kraken system is designed specifically for sustained application performance, scalability, and reliability, and will incorporate key elements of the Cray Cascade system to prepare the user community for sustained, high-productivity petascale science and engineering.



Photo of NICS Cray XT5 – Kraken

Lustre File System

The Kraken resource at NICS has a 2.4 petabyte (local) lustre file system configured for use by Kraken

users. The Lustre file system is made up of 12 DataDirect Networks S2A9900 storage platforms in 6 racks. Each S2A9900 is configured with 280 one terabyte hard disk drives for a total of 3360 disk drives. There is 3.36 petabytes of unformatted space and 2.4 petabytes of formatted space using 8+2p RAID6. There are 48 Object Storage Servers (OSSs) and 336 Object Storage Targets (OSTs). See Figure 1 for a configuration of Lustre on Kraken. The default file stripe is 4 and the Kraken Lustre file system's mount point is /lustre/scratch and every user on the system has a Lustre directory of the name /lustre/scratch{username}.

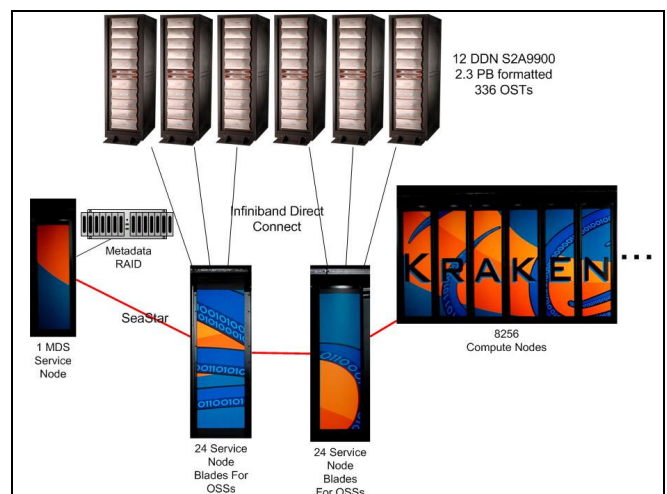


Figure 1: NICS Lustre Configuration

2. Building the Lustre File system

Our goal in configuring this Lustre file system was to maximize I/O bandwidth by minimizing contention on the SIO-DDN controller-controller port paths. The OST's were laid out in such a way that writing a file would not hit the same SIO-DDN controller-controller port unless the stripe size of the file was greater than 48. The Cray documentation for using InfiniBand with Lustre was non-existent at the time we began the install. One of the first problems that we ran into was that many of the rpm's that IB requires were not installed by default and there was no list of which rpm's were required. Thanks to Dave Dillow at ORNL and Steve Woods at Cray, a complete list of required rpm's was obtained and the rpm's were successfully installed. Here is that list of IB rpm's:

```
ibutils-1.2-1.4.x86_64.rpm
infiniband-diags-1.2.7-2.7.x86_64.rpm
libibcommon-1.0.4-0.2.x86_64.rpm
libibmad-1.0.6-0.2.x86_64.rpm
libibumad-1.0.6-0.2.x86_64.rpm
libibverbs-1.1.1-9.3.x86_64.rpm
libmlx4-0.1-0.4.x86_64.rpm
librdmacm-1.0.2-3.4.x86_64.rpm
mstflint-1.3-0.3.x86_64.rpm
ofed-1.2.5.5-0.3.x86_64.rpm
ofed-kmp-ss-1.2.5.5_2.6.16.54_0.2.8_1.0000.3800.0-
0.3.x86_64.rpm
opensm-3.0.3-2.8.x86_64.rpm
perftest-1.2-1.5.x86_64.rpm
srptools-0.04-1.9.x86_64.rpm
```

Shortly after the installation of these rpm's we learned that there was an undocumented parameter, OFED=yes, that could be inserted into the XTinstall.conf file that would install the correct IB rpm's!

Once the IB rpm's were installed we were ready to begin building a Lustre filesystem. Initially, there was only 1 cabinet of DDN drives configured. Therefore, we started by building a small lustre file system to test our build procedures. At this time, we were pausing the xtbootsys script and manually running the srp_login scripts on the OSS nodes. The srp_login script creates a connection between the OSS and the paired controller allowing the OSS to see the OST's. Once this was done then we initiated the command to build a 380TB file system. Strangely, the lustre reformat command had not completed after 36 hours over a weekend. We contacted ORNL administrators and they suggested that we change the lustre scheduling algorithm from noop to deadline. Once we did this and reran the lustre reformat command it successfully completed in 2.5 hours. If this is done it is

important to remember to set the lustre scheduling back to noop.

Now that we had a formatted Lustre file system we were ready to reboot the system with Lustre mounted. We automated the srp_login process by creating an /etc/init.d/srp_login script on each OSS nodes. We also started using scsi aliasing at this point in order to easily identify the physical location of OST's. For example, OST20 is equivalenced to /dev/scsi/r3u2l4, rack 3, controller u2, lun 4. Once these scripts were in place and running, the lustre file system would then mount successfully. After all this was done we now had a mounted and functioning 380 TB lustre file system.

Afterwards, once all the DDN cabinets were powered up and cabled, we attempted to build a 2.4PB Lustre file system. A reformat command was issued and after 2.5 hours the reformat script failed attempting to mount the OST's. We tried a second time and got the same result. We thought that the issue might be with the 7 OST's per OSS (7 OST's per OSS was new to Cray at the time). We then reconfigured the system to only use 4 OST's per OSS and reran the reformat command. This worked and completed successfully in 2.5 hours. The plan was then to try 5, 6, and 7 OST's per OSS and see where the limit was. However, because acceptance testing start time of the system was looming, we decided to use the 1.3PB file system for acceptance and rebuild a 2.4PB system after the acceptance period and before Kraken went into production.

The 1.3PB file passed all performance related acceptance tests. We were sure that we could improve the MDS performance, which at that time had a 5TB raid that consisted of five 1TB 5400 rpm Engenio LSI drives on the boot raid. NICS had available a DDN EF2915 5TB raid that consisted of fifteen 320GB 15000 rpm drives that we configured as a new MDS. This gave us a big performance gain as discussed in section 3.

During the acceptance period we learned from ORNL that 7 OST's per OSS was not a problem. What was needed to do was run the reformat command in stages, not all at once, as this overwhelmed the MDS. So we wrote a script to do this. It first reformatted the new MDS, and then reformatted 10 OSS's at a time. The script completed successfully in 12.5 hours and the 2.4PB file system mounted properly after a reboot.

3. Question of Purging vs. Quotas

With such a large Lustre file system configured as scratch space, one of the inevitable questions is whether to purge files from it periodically or to impose quotas. On Kraken-XT4, which used Lustre 1.4 and where quotas were not an option, experience had shown that walking the file system to identify files to purge was prohibitively

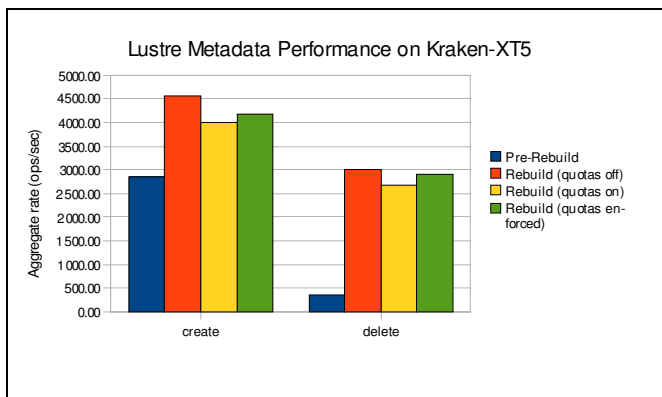
slow and degraded metadata performance on the file system to an unacceptable degree. Therefore, after rebuilding the Lustre file system on Kraken-XT5 following acceptance testing, one of the first sets of tests run was to quantify the performance impact of quotas.

Three separate tests were run:

- A threaded file creation and deletion test on a single compute node.
- A file-per-process bandwidth test, using the standard IOR benchmark.
- A shared file bandwidth test, also using IOR.

These tests were run in four situations:

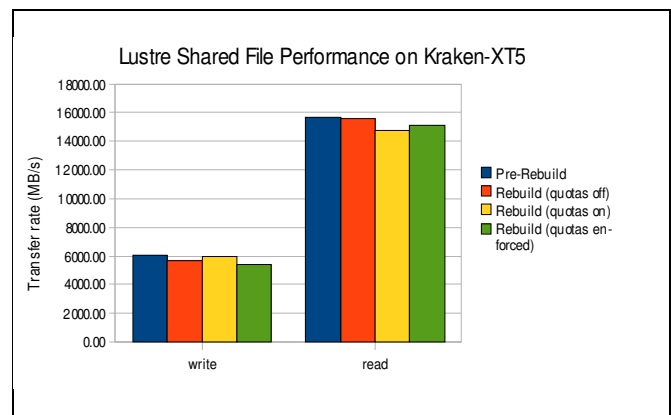
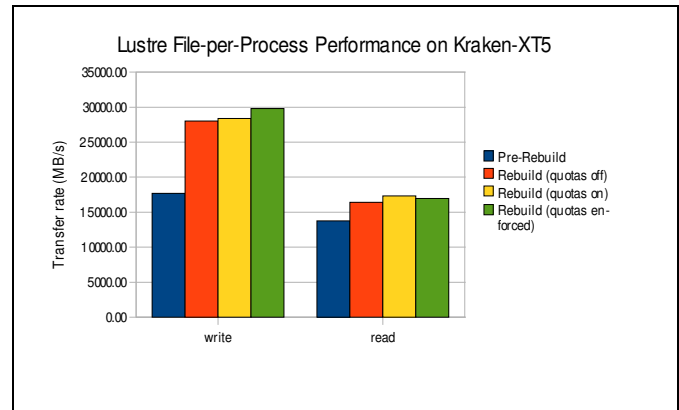
- Before the file system rebuild.
- After the file system rebuild with quotas disabled.
- After the file system rebuild with quotas enabled but not enforced.
- After the file system rebuild with quotas enforced.



The threaded file creation/deletion test showed a substantial improvement in MDS performance of the DDN EF2915 array relative to the LSI RAID boot array used before the rebuild. The sustained rate of file creation increased by 59%, while the sustained rate of file deletion increased by a surprising 718%. After enabling quotas, these rates did drop slightly, by 12% in the case of file creation and 10% in the case of file deletion.

The file-per-process I/O bandwidth on the system also saw a significant improvement due to the increased number of OSTs (and therefore disk spindles) in the file system. Surprisingly, a further *increase* in performance was observed after enabling quotas on the file system, and also the writes after enforcing quotas as well. There is no immediately obvious reason why this should be so, although it has been theorized that some OST devices were in the process of rebuilding during the first two post-rebuild sets of tests. In any case, there was no measured

performance impact on file-per-process I/O by enabling quotas.



However, shared-file I/O performance was another matter. Write and read performance to a shared file dropped by 6% and 1% respectively after the rebuild; since the maximum number of OSTs a file can be striped across on Lustre is fixed at 160, adding more OSTs did not improve performance, and since the number of Infiniband paths to the DDN 9900 controllers remained fixed, there was actually slightly increased contention across these links. Enabling quotas actually improved write performance by 5% but also decreased read performance by 5%, while enforcing quotas effectively reversed the situation. The maximum impact of quotas observed on shared-file I/O performance was 6%.

With a metadata performance penalty of 10-12% and a maximum bandwidth impact of 6%, NICS chose to move forward with quotas being enabled but not enforced on the Kraken-XT5 Lustre file system. We suggest that this small performance penalty will be largely offset by not having to traverse the Lustre file system periodically in order to generate a file purge list and, thereby, impacting file system performance significantly as the purge program runs. Our purge policy was developed

with this in mind and requires that the top Lustre file system users be notified to purge files as a file system usage threshold is reached. To date file system usage is below 50% and no file system purging has been required. We expect this to change over the next few months.

4. Configuration and Performance

Budget constraints limited our configuration to a minimum number of controllers, a high number of OSTs per OSS and with such a large system there are a large number of Lustre clients (O8300) which led to some interesting tradeoffs in performance and capability.

The DDN configuration has 6 cabinets with two S2A9900s each with four port controllers. The theoretical peak burst rate for Lustre is approximately 76.8 GB/s ($1.6 \text{ per port} * 4 \text{ ports} * 2 \text{ controllers per cabinet} * 6 \text{ cabinets}$). The estimated sustained performance is approximately 38.4 GB/s ($.8 \text{ GB/s port} * 4 \text{ ports} * 2 \text{ controllers per cabinet} * 6 \text{ cabinets}$). During our acceptance testing we were able to demonstrate about 5GB/s per cabinet as we obtained 30GB/s on portions of the IOR benchmark application. No redundant controller paths exist in our current configuration. This means no failover is possible for this configuration. Therefore, it is very important to handle controller errors with the upmost care in order to prevent loss of data. Our process for dealing with controller errors, of which we have had two, is to contact Cray that there is an issue and immediately contact DDN support. We let DDN support guide our response to any controller error in order to prevent unnecessary controller reboots which could lead to loss of data. SEC errors that indicated possible drive or controller errors usually have "Hard SCSI Error" in the error message.

Our initial OSS/OST configuration was to have 14 OSTs per OSS due to a configuration of 24 OSS service nodes. Cray told us this was not going to be supported and we eventually had to trade some compute nodes for service nodes to come up with 48 OSS service nodes and a supported configuration of 7 OSTs per OSS. This led to some interesting Lustre file system build problems discussed earlier. Eventually, the production Lustre file system was configured and preliminary tests showed we would be able to get near the maximum sustained transfer rates advertised at approximately 30GB/s. We are still limited in the number of transactions to the MDS and OSS servers which has been problematic when the portal LND layer wait queue is full due to a large job writing checkpoint files beyond the physical limit. Other users can notice the backlog in the wait queue and can notice significant slowdowns in response time to the execution of "ls" commands for example. Use of "lfs find" commands can help with this situation but is not a normal command used by users and requires user education.

Also, because of the size of the system and number of clients, Cray recommended that we increase the credit values for portals for compute nodes, OSS nodes and MDS node. The tuneable parameters were 2048 credits value on the MDS node, credits value of 512 for compute nodes and credits value of 1024 for the OSS nodes. Also the Lustre `ldlm_timeout` value was increased to 250 so that the 8000+ clients can avoid the timeout, eviction and reconnect loop. Our default strip count is 4 and the default stripe size of 1MB. Investigation on the prior XT4 Lustre filesystem showed that only two users had changed their stripe count from the default on the XT4. This indicates that more user education is needed to effectively use the Lustre filesystem.

5. Canary in the Coal Mine

While Lustre tends to provide a lot of warning messages and Error codes, it does only when a set of clearly defined conditions are met. Hardware issues affecting the portals network are not always noticed until Lustre generates errors - Lustre being the equivalent of the canary in the coal mine - followed by user complaints on a file system hang. Users are justified for deploring over the poor performance of the file system because that is the only place they notice the hidden problems. Lustre on the other hand assumes solid network hardware reliability and keeps trying to recover from hardware caused errors. Endless sequence of Lustre errors continues unless the hardware issues are resolved, mostly by a system reboot. We typically get half million to seven million lines of Lustre error messages a week. Once we separate the interconnect failure caused error messages, Lustre messages are predictable and consistent: Failed nodes are identified by the timeout and eviction sequence, heavy concurrent I/O patterns beyond the current bandwidth limits manifest themselves as a global delay.

By focusing on the onset point of volumes of Lustre errors and correlating the network log messages on the mesh link states of the HSN, we identified that a cascading chain of deadlock timeouts, also known as router error precedes the HSN collapse. The sequence of events starts with a sudden appearance of a router error on the event log file, propagation of the router error to other links, Lustre servers severed from network, nfs server boot node not responding, and stream of beer messages. By the time we see the beer (basic end-to-end reliability) messages, it is beyond the point of no return for the HSN.

The HSN sometimes recovers by itself eventually ingesting all the portal traffic. We did see the self recovery twice during last three months. But it tends to have lingering effects and job performances become unpredictable after such recovery.

5.1 RAID rebuild failure

On 2 April 2009, we observed several errors with error number -30 (EROFS) from Lustre logs on an OSS server occurring on 2 of its OSTs. We went to the machine room and noticed the LEDs on top two cages indicate error. Reporting it to the DDN support, we learned that there were hardware rebuild errors earlier on two of the 56 LUNs possibly due to interrupts on rebuilds while write-cache enabled. Lustre on the other hand did the right thing to detect it as a SCSI device error on its back end disk device and immediately remounted the affected OSTs read-only.

After warnings from DDN that we might need to reformat the two OSTs and a quick estimate on the number of files to be lost with reformat close to a million, we decided to force the rebuild option on two LUNs while performing the fsck on two OSTs. Both were successful. File system checks on the OST took about 90 minutes in parallel, and the RAID rebuild successfully completed in a day.

Guided by job schedule records, we narrowed the time windows and obtained the possibly affected user list. Extensive lfs find, debugfs stats on candidates confirmed that we avoided a massive data loss. In the end, no corrupted files were found or reported by users. Lustre prevented a massive file system corruption, and we confirmed that the Lustre did its job as it was supposed to.

5.2 Monitoring

We keep track of the following three:

- Lustre error counter: monitors the Lustre warnings, Errors, and ratio of the two. It generates the data for weekly snapshot graph, and it alerts the global system event when it happens. Thank to a large number of clients, we can tell the difference whether it is a global scale or a localized timeout-eviction sequence. It complements the SEC-rule set based warning system.
- Lustre hang sampling: random interval checks on Lustre response time and is logged continuously during production. Combined with Lustre warning message, it enabled us to identify heavy I/O jobs causing file system response hangs.
- Lustre File system state: number of files generated and total disk space used are recorded hourly. It not only provides the file usage trends but also gives extra clues to unusual Lustre warning messages.

Conclusion

Lustre is the only currently supported high performance parallel file system available for the Cray XT5 system that can be used by the compute nodes and having this distinction NICS has to take advantage of all Lustre features applicable to the NICS environment and try to manage and minimize all of Lustre's disadvantages. NICS has observed that Lustre seems to provide early warning of system failures both detected and undetected and is affectionately known as our "canary in a coal mine". Enabling but not using Lustre quotas seems to provide a decent trade-off between automated system purging and full quotas based on performance testing with and without quotas and the known performance impact of file system traversal with a script that identifies file purge targets. However, NICS purge policy will still depend on end users to take action which experience has shown can be unpredictable. NICS is iteratively improving our Lustre monitoring with a combination of log watching, Lustre file system response time and file system state

Acknowledgments

We would like to acknowledge Art Funk, the Cray support staff and the staff at ORNL's NCCS for their help with the NICS Cray XT5 install and configuration.

About the Authors

Troy Baer is an HPC systems administrator for the University of Tennessee's National Institute for Computational Sciences (NICS) at Oak Ridge National Laboratory. He can be reached by emailing <tbaer@utk.edu>.

Victor Hazlewood is a Senior HPC Systems Analyst at NICS and can be reached by email at victor@utk.edu.

Junseong Heo is a at NICS and can be reached by email at jheo6@utk.edu.

Rick Mohr is an HPC systems administrator at NICS and can be reached by email at rmohr@utk.edu

John Walsh is a Senior HPC systems administrator at NICS and can be reached by email at jwalsh3@utk.edu