# Automated Lustre Failover on the Cray XT

**Nicholas Henke, Wally Wang, Ann Koehler**, *Cray Inc.*

**ABSTRACT:** *The increasing scale of current Cray XT systems will demand increased stability from the critical system services. Lustre failover is being deployed to allow continued operation in the face of some component failures. This paper will discuss the internal Lustre mechanisms involved during failover and our automation framework. We will also discuss the impact of failover on a system and the future enhancements that will improve Lustre failover.*

**KEYWORDS:** XT, Lustre, failover

## 1. Introduction

As production workloads ask more of Cray XT systems, the need arises for system services that can recover from failure. Failures in critical resources often require system reboots to rectify. These system restarts impact users by losing current application progress leading to lost time and productivity. To alleviate such issues, software must be capable of recovering from component failures and automated in such a manner that action is taken as quickly and efficiently as possible.

This paper will discuss the mechanisms of Lustre failover and how it is automated on the Cray XT. We will also cover hardware and software configuration requirements for Lustre failover. Finally, we will cover some known limitations and discuss exciting new enhancements that will further improve the Lustre failover offering.

## 2. Lustre Background

Building a Lustre cluster requires a Lustre MetaData Server (MDS) and Lustre Object Storage Servers (OSSs), each with disk storage. A pool of client systems access these servers through one of many supported networks [1]. The storage on each Lustre server is referred to as a target, giving us a MetaData Target (MDT) on the MDS and an Object Storage Target (OST) on the OSS. Cray XT systems are usually configured with one MDS and MDT combination along with one or more OSS nodes. Each OSS node is capable of serving requests for one or more OSTs.

Lustre clients are connected to each service in the file system over the internal Cray high speed network. The Lustre clients send parallel data transfers to each service independently. The primary method of fault detection is through network timeouts on these file system requests.

## 3. Impact of Lustre Server Failure

Lustre is the critical system resource that provides application with high speed parallel file system access. The loss of any one of the Lustre servers will stall most file system access, an event which is usually considered a system-wide outage. A typical customer machine will have many active applications, each possibly accessing one or more files within Lustre. When a Lustre server is lost, the Lustre client will transition into recovery mode where it will attempt to reconnect to the lost server and resend the pending transactions. The portion of the application that is running on a Lustre client in recovery will be stalled, preventing the application as a whole from making further progress until the service is restored.

Correcting a Lustre server outage with a system reboot is a harsh approach. Job progress is often lost, causing angst for both the system users and system administrators. The number and duration of machine interrupts are an often monitored statistic used to make determinations for Service Level Agreements, often with contractual impacts.

## 4. Overview of Lustre Failover

To alleviate problems due to Lustre server failures, we will employ the use of Lustre Failover. The primary objective of Lustre Failover is to regain system functionality after a failure while minimizing the job loss due to the interruption. Lustre Failover allows the targets from the lost server to be accessed via a backup server. After reconnecting to the targets, Lustre clients will replay any lost transactions and then resume full operation.

### *4.1. Configuration*

Lustre failover configuration uses the scheme of a primary server with one or more backup servers. Typically each OSS has a partner for failover, such that they serve as the backup for each other in what is called an “active-active” configuration. While this arrangement does result in performance loss, it simplifies hardware configuration and minimizes the extra resources needed to provide fault tolerance. Lustre currently allows one MDS server per file system, so it must make use of an idle backup node for failover. This arrangement is referred to as “active-passive”.

### *4.2. Server death*

Lustre server loss is determined by the ability to service client requests in finite time. Failure in one of the hardware components (network, processor, memory, etc) or bugs in the software will crash the node, rendering it unresponsive to requests. The node may remain nominally alive, for instance being active on the network, but clients will stop seeing responses to outstanding requests.

Lustre clients detect the loss of a server through network timeouts. When a message is sent to a server, a deadline is set for a response. When the response is not received within the deadline, the Lustre client will mark the target as down. The client will then enter into a loop that will attempt to re-establish the connection. All requests that access the down target will be blocked until the connection is returned.

Server death is visible to administrators through the Cray XT state management software. The Cray RAS and Management System (CRMS) will detect the loss of node heartbeat and change the state accordingly. Lustre clients also provide ample notification of service interruption by posting messages on the CRMS console network.

### *4.3. Restarting lost services*

The targets that were served from the down Lustre server need to be made available from an alternate node. To allow clients to reconnect, this backup server will start the services and begin accepting requests for those targets. The services are restarted in recovery mode to ensure that clients can reconnect and then reissue outstanding requests before sending any new I/O requests. This replay guarantees that all of the clients see a consistent state for the restarted target before returning to normal operation.

### *4.4. Client reconnect and replay*

When attempting to re-establish connections, the Lustre clients will first re-contact the original server in hopes the loss was a temporary network issue. If this fails, the client will attempt reconnection to each of the backup servers. The connection attempts will continue until a server responds that it is now the acting host of the stalled target. The Lustre client is informed that it should wait to send new requests and that replay mode has been invoked. Once all of the Lustre clients have reconnected to the backup server, they are instructed to resend their transactions in the order they were originally sent. Upon successful completion of replay, the client resumes normal operation. The client first resends any blocked operations and then returns to processing new requests.

## 5. Automating Lustre Failover

Automation of Lustre Failover is done via an external framework as Lustre does not provide this service natively. The configuration and state of the services must be stored in a central location to provide the automation agents with the correct information. The automation framework must then monitor the health of the services and communicate this to interested agents within the machine. Once monitoring has deemed a server as unhealthy, action must be taken to rectify the issue. The manifestation of this on the Cray XT is failover proxy named ‘xt-lustre-proxy’. This daemon runs on each Lustre server and is the active agent for the failover framework.

### *5.1. Configuration and state management*

The Cray System Database (SDB) is used as the central storage for managing the configuration and status of the various services. The relationship of primary and backup servers along with the current active node for each service is stored in the SDB tables. This information will allow the failover proxy to understand what action to invoke upon service failure. During start-up, the proxy on each Lustre server polls the SDB data to determine what services it will monitor.

To allow for seamless Lustre failover configuration, the existing Lustre command and control suite, Lustre Control, has been modified to support with changes to the file system definition file. When generating a configuration, it will produce a set of comma separated value (CSV) files formatted for the SDB tables. Lustre Control will push this data into the SDB when the file system is set and will also start and stop xt-lustre-proxy as it operates on the Lustre file system. These operations are done in such a manner that additional administrator action is rarely needed.

### *5.2. Health Monitoring*

The most difficult aspect of the failover framework is health monitoring. It must be sensitive to notice failure quickly but robust to ensure that false failures are minimized. The failover proxy uses multiple sources of data to provide a full health picture of the Lustre server components. It uses the CRMS heartbeat to determine basic hardware status and operating system health. A new heartbeat event is registered with CRMS to track the health of the Lustre software stack. The proxy utilizes an existing health check within Lustre to verify it is still functioning properly. This information is collated by the proxy while making an allowance for short temporary failures to help improve the accuracy of the data even while the machine is under high load or duress. This state is then sent out via CRMS events to partner proxies that are also monitoring this service.

### *5.3. Triggering Action*

Once xt-lustre-proxy determines a node is unhealthy, it will take action to start the services on a backup server. Care must be exercised to ensure the services are not running on two servers at once, as the file system will be corrupted if driven from multiple locations. The proxy will first send a CRMS event to shoot the primary node ensuring it does not miraculously return to life, and then the services are mounted on the backup node. Once restarted, the proxy resumes monitoring while Lustre enters recovery and begins the healing process.

## 6. Current Limitations

While Lustre Failover generally available in Cray XT 2.2, there are a few known limitations. There are some cases where MDS failover is not as robust as we would prefer and some small deficiencies are present in the management and operation infrastructure. We are addressing these issues as a priority and will be working to ensure solutions are found for all issues.

Currently there are interactions between Lustre quotas and failover that result in a non-functional file system after MDS failover. MDS failover with quotas enabled should be avoided until these issues have been resolved.

We also understand that the current duration for a failover event is not optimal. With the current release, it usually takes ten to fifteen minutes from the time of a server failure until clients can send new I/O requests. On very large configurations or under heavy load, we have seen failover instances that require thirty minutes to complete. We hope to address this issue with the improvements described in the Future Work section.

Given the duration of failover, some user job loss is inevitable. Our goal is for all applications to survive the failover; however users with tight batch time limits may result in job failure. The current Lustre architecture requirement that all clients reconnect for replay also necessitates that loss of a compute node during a failover event can cause the replay to fail. In this case, it is possible for applications to receive errors for file system operations that were active across the failover. New applications should remain unaffected.

Finally, there are operations in Lustre failover that have not been automated or exposed in comfortable interfaces. The mechanism for failback, the ability to return services from the backup server to a newly repaired primary node, is not automatic. A manual process is documented for administrator use. Status of failover progress can only be monitored by an administrator with login access to the Lustre servers.

## 7. Future work

To address the known issues and to provide an enhanced Lustre Failover offering, we are exploring several improvements. We are collaborating with Sun to develop a feature called Imperative Recovery that aims to reduce failover duration. The initial milestone is to achieve failover in less than five minutes with the ultimate goal of failover completion in the one to three minute range. The changes needed are primarily in the notification layer around failover events. It will allow xt-lustre-proxy to instruct the Lustre clients to forcefully switch their connection to the backup server, greatly reducing the amount of time it takes all of the clients to reconnect and start recovery. The failover proxy will also be able to take responses to these client reconnections and use that information to instruct the backup server to stop waiting for clients to reconnect. This should further reduce failover duration by eliminating time spent waiting for clients that will not reconnect.

Sun has also developed a failover related feature called Version Based Recovery (VBR) that should greatly improve the recovery behaviour in the face of lost clients. It allows servers to reduce the recovery quorum to those nodes that had outstanding uncommitted transactions. VBR also enables servers to minimize the replay failures to those clients who had outstanding transactions that depend on a client who failed to reconnect. The rest of the clients and transactions should proceed through recovery unaffected.

Finally, the Gemini Network will become available. The driver stack developed for this should allow for shorter Lustre Network (LNet) timeouts. Reducing the timeout will allow health detection to take place quicker through shorter wait intervals. It will also solidify the detection of unresponsive peers by returning positive feedback when the remote host is dead.

## Acknowledgments

The author would like to highlight the immense amount of help that has enabled this paper. In particular, the wonderful folks in Cray Software Product Support (SPS) have provided valuable review and feedback on failover for customers. Many folks in Cray Software Development have been instrumental in discussing whole system impact, expected behaviour and potential solutions. The engineers in the Lustre Group at Sun provided in-depth help with data analysis and problem resolution.

## References

1. Lustre File system. http://www.sun.com/software/products/lustre/features.xml

## About the Authors

Nicholas Henke, Wally Wang and Ann Koehler are a Software Engineers in the Lustre group at Cray Inc. They can be reached via email at nic@cray.com, wang@cray.com and amk@cray.com.