Using the DLM as a Distributed In-Memory Database

Bart Van Assche, Ph.D.

August 21, 2015
Overview

- Introduction
- Initiator-side Clustering
- Persistent Reservation Support for HA Arrays
- Distributed Lock Manager (DLM)
- Using the DLM to Implement PR Support
Introduction

- SCSI specs require that all nodes in a multi-node array report the same state information.
- Many people build a H.A. storage array with iSCSI target software + H.A. software + data replication software.
- Complies to most SCSI requirements except:
  - Persistent reservations (PR).
  - Mode pages.
- Proper PR support is required for initiator-side clustering.
Initiator-side Clustering

- Redundant initiator and storage systems.
- Witness disk is used to decide initiator server master role.
- Witness disk ownership via SCSI persistent reservations.
Persistent Reservation Support for HA Arrays

- Multiple storage servers but ...
- Single persistent reservation state.
- Challenges:
  - Implement single PR state efficiently.
  - Handle node join, power cycle, split-brain etc. correctly.
Distributed Lock Manager (DLM)

• A distributed lock manager (DLM) provides distributed software applications with a means to synchronize their accesses to shared resources.

• Was added in 2006 to the Linux kernel as a kernel driver.

• Core component for GFS2, OCFS2, CLVM, ...

• Features
  - Supports user space and kernel applications.
  - Allows to serialize access to shared resources.
  - Provides reader/writer lock objects.
  - One lock value block per lock object.
  - Supports remote procedure calls.
DLM Lock Objects

- Each lock object exists in a DLM lockspace.
- Maximum lockspace name length: 64.
- Maximum lock object name length: 32.
- Lock Value Block (LVB)
  - Length must be a multiple of 8.
  - Max. 32 bytes for user space applications.
  - Kernel code can use larger LVB's.
# DLM Lock Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Requesting Process</th>
<th>Other Processes</th>
</tr>
</thead>
<tbody>
<tr>
<td>Null (NL)</td>
<td>No access</td>
<td>Read or write access</td>
</tr>
<tr>
<td>Concurrent Read (CR)</td>
<td>Read access only</td>
<td>Read or write access</td>
</tr>
<tr>
<td>Concurrent Write (CW)</td>
<td>Read or write access</td>
<td>Read or write access</td>
</tr>
<tr>
<td>Protected Read (PR)</td>
<td>Read access only</td>
<td>Read access only</td>
</tr>
<tr>
<td>Protected Write (PW)</td>
<td>Read or write access</td>
<td>Read access only</td>
</tr>
<tr>
<td>Exclusive (EX)</td>
<td>Read or write access</td>
<td>No access</td>
</tr>
</tbody>
</table>
DLM API

- dlm_lock()
  - Asynchronously convert to a higher or lower lock mode.
  - Optionally invoke a callback function on each other node that holds a lock that blocks this conversion (BAST = blocking AST).
  - Optionally invoke a callback function on the local node when the new lock mode is granted (AST = asynchronous system trap).
  - After the new mode has been granted, reports whether or not the LVB is valid (DLM_SBF_VALNOTVALID flag).

- dlm_cancel(): cancel a conversion requested via dlm_lock().

- dlm_unlock(): discard a DLM lock object and also its LVB.

- The LVB is invalidated:
  - If a node holding a lock object in EX or PW mode fails.
  - If a node joins the cluster as master and only NL/CR locks are left.
Lock Value Block (LVB) Propagation

- Converting to a higher mode returns the LVB to the caller (e.g. CR → EX).
- Converting to a lower mode updates the LVB on all nodes (e.g. EX → CR).
- Using the DLM as a distributed in-memory database is possible:
  - By using the namespace + lock object name as key.
  - By using the LVB as value.
  - By using e.g. the CR → EX conversion to read the LVB and the EX → CR conversion to publish a modified LVB.
DLM and Read Access

- Two approaches are possible to obtain the latest contents of the LVB:
  - Obtain read access on the lock object associated with the LVB (best for infrequent reads).
  - Use the BAST mechanism to keep all LVB copies up-to-date all the time (best for frequent reads).
DLM - Implementing Update Notifications

- Create pre.<n> and post.<n> notification lock objects where <n> is the node ID assigned by the cluster management software.
- Initialize pre.<n> to EX and post.<n> to the NL state.
- Associate a BAST with each of these lock objects.
- Send a notification and wait until remote data copies have been updated by converting lock object pre.<n> first to PR and next to NL. Subsequently convert lock object post.<n> to PR and next to NL.
- In the pre.<n> BAST, convert the post.<n> lock object from NL to EX mode and the pre.<n> lock object from EX to NL mode.
- In the post.<n> BAST, convert the pre.<n> lock object from NL to EX mode, make a local copy of the updated data and convert the post.<n> lock object from EX to NL mode.
- Advantage: cluster membership changes are handled transparently.
- Disadvantage: certain cluster membership changes result in data loss.
DLM - Implementing Update Notifications

<table>
<thead>
<tr>
<th>Updater</th>
<th>Every node i</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>Initialize pre.&lt;i&gt; to EX; post.&lt;i&gt; to NL</td>
</tr>
<tr>
<td>Update LVB</td>
<td></td>
</tr>
<tr>
<td>for i in (all other nodes):</td>
<td></td>
</tr>
<tr>
<td>request pre.&lt;i&gt; → PR</td>
<td></td>
</tr>
<tr>
<td>Convert post.&lt;i&gt; NL → EX</td>
<td></td>
</tr>
<tr>
<td>Convert pre.&lt;i&gt; EX → NL</td>
<td></td>
</tr>
<tr>
<td>pre.&lt;i&gt; PR → NL</td>
<td></td>
</tr>
<tr>
<td>request post.&lt;i&gt; → PR</td>
<td></td>
</tr>
<tr>
<td>Convert pre.&lt;i&gt; NL → EX</td>
<td></td>
</tr>
<tr>
<td>Make a local copy of the updated LVB</td>
<td></td>
</tr>
<tr>
<td>Convert post.&lt;i&gt; EX → NL</td>
<td></td>
</tr>
<tr>
<td>post.&lt;i&gt; PR → NL</td>
<td></td>
</tr>
</tbody>
</table>
Implementing Persistent Reservations (1/2)

- One DLM lockspace per LUN.
- One lock object per LUN for which the LVB contains:
  - Number of registrants
  - Whether or not the device has been reserved persistently.
  - Persistent reservation type.
  - Persistent reservation scope.
  - Value of APTPL (activate persistence through power loss).
- One lock object per LUN and per reservation key containing:
  - Reservation key.
  - Relative target ID.
  - Transport ID (up to 228 bytes for iSCSI).
Implementing Persistent Reservations (2/2)

- A lock object per LUN that serializes PR data changes.
- Propagating PR data changes via notifications.
- Each node persists APTPL reservations locally and reapplies these during boot.
- Result: an efficient and resilient SCSI PR implementation for clusters.
Cluster Node ID Discovery

- Notification mechanism needs to know node IDs of other cluster nodes.
- Node IDs are discovered by reading contents of /sys/kernel/config/dlm/cluster/comms directory.
- Node IDs are read during initialization, after an update has been published and also after an update notification has been received.
Handling Cluster Membership Changes

- If a node leaves the cluster
  - DLM handles this transparently.

- If a node joins the cluster
  - New node copies PR state from LVB's into local state.
  - If one or more of the LVB's were invalid, trigger an LVB update.

- LVB update
  - Send an LVB update notification to other nodes.
  - Each node that receives this notification copies its PR state to LVB's.
  - These LVB updates are serialized – last update takes effect.
Handling Split-Brain

- Cluster manager must stop one of the two partitions
- After the split-brain has been resolved, the same procedure is applied as when a node joins the cluster.
- If a node joins the cluster
  - New node copies PR state from LVB's into local state.
  - If one or more of the LVB's were invalid, trigger an LVB update.
- LVB update
  - Send an LVB update notification to other nodes.
  - Each node that receives this notification copies its PR state to LVB's.
  - These LVB updates are serialized – last update takes effect.
Possible Alternatives for the DLM

- Google's Chubby Lock Service
  - Userspace only
  - userspace / kernel barrier would have to be crossed twice for each distributed lock operation.
  - Strict ordering of updates is not needed for PR.
  - *design emphasis is on availability and reliability, as opposed to high performance* [Bu06].

- Zookeeper
  - Closely related to Chubby according to [Vi11].
References

Any questions or comments?
Legal Notice

©2015 SanDisk Corporation. All rights reserved. SanDisk is a trademark of SanDisk Corporation, registered in the US and other countries.