Blue Waters Training – Filesystems and Lustre

Christopher Heller, Manager, Storage team - NCSA/ICI
cheller@illinois.edu

Justin Davis, Field Technical Support Consultant, Cray/HPE
justin.davis@hpe.com
Agenda

- Filesystem and Data methods
- Lustre
Filesystem and Data methods (1)

- Policies/Policy Engine - purges, backup
  - Purge via external script on scratch (no longer used, now is quota based)
  - Disaster recovery via NCSA developed solution (formerly HPSS)
    - 50 data mover nodes for HPSS
    - New software will use 28 data mover nodes to move data from disk to disk for DR
Filesystem and Data methods (2)

- Robinhood
  - V3.1.3
  - Scan once and consume Lustre changelogs for modifications from start of scan time
  - Separate changelog consumer
  - PerconaDB on NVMe storage
  - 2.7 Lustre client plus scan-only modifications
  - Feeds reports and Disaster recovery software
Filesystem and Data methods (3)

- Cray View for ClusterStor™
  - Why is I/O for my job slow?
  - Why does the filesystem seem sluggish on interactive nodes?
  - Built using a mixture of open source and proprietary APIs
    - CStream (proprietary ClusterStor API)
    - Zabbix, Plex, Grafana, Zookeeper, Kibana, Monasca, Kafka
  - Utilizes agents on the Cray compute resource to relate WLM job ID and Lustre jobstats ID.
Lustre

• Blue Waters
  • 3 filesystems, home, project, and scratch
  • 24PB total usable space
  • 584 total LNET routers
  • 17,280 *2TB drives
  • Performance: Home/Project: 100 GB/s – Scratch: 1TB/s
Lustre(2)

- Our Lustre installation utilizes a single metadata target per filesystem.
- Metadata Hardware
  - Memory and Storage bandwidth intensive
  - Scratch: 128GB Memory, 8.5T on 72 spindles (raid 10)
  - Home/Project: 64GB Memory, 2.1T on 14 spindles (raid 10)
- ldiskfs based
Lustre(3)

• Kernel Raid Check
  • Default raid check mechanism
  • Checks array from start to finish serially
  • Total raid check time determined by total size of the array.

• User-level Raid Check (URC)
  • Cray/HPE proprietary
  • Checks for errors on every disk in parallel at specified rate
  • If a read error is encountered send a small offset to kernel for repair
  • Total raid check time determined by size of largest disk in the array.
Lustre(4)

- Declustered Parity (gridraid)
  - Parity for several mdraid arrays mixed and orchestrated
  - 2-stage data recovery
    - Repair missing data into spare blocks on all disks in parallel
    - Copy repaired data from spare blocks onto new disk
- Standard Parity (mdraid)
  - 1-stage recovery
    - Data repaired directly onto new disk
Lustre(5)

- Balance rebuild times and performance
  - Target is 3MB/s/disk
- OST Thresholds for deactivation
  - Deactivate at 85% full
  - Reactivate at 75% full
- OST pools
  - No active pools at this time
  - Utilizes lustre configuration log
- Failover
Lustre(6)

- Lustre supports POSIX ACLs (Access Control Lists)
  - We do not implement them here
- Standard Linux permissions (part of POSIX ACL)

<table>
<thead>
<tr>
<th>Entry type</th>
<th>Text form</th>
</tr>
</thead>
<tbody>
<tr>
<td>Owner</td>
<td>user: rwx</td>
</tr>
<tr>
<td>Named user</td>
<td>user: name: rwx</td>
</tr>
<tr>
<td>Owning group</td>
<td>group: rwx</td>
</tr>
<tr>
<td>Named group</td>
<td>group: name: rwx</td>
</tr>
<tr>
<td>Mask</td>
<td>mask: rwx</td>
</tr>
<tr>
<td>Others</td>
<td>other: rwx</td>
</tr>
</tbody>
</table>
Lustre(7)

Logging

- Inbuilt
  - Diskmonitor, zabbix, rsyslog
- External
  - SEC, cron, local site scripts
Lustre(8)

Alerts

• Inbuilt
  • Icinga, zabbix

• External
  • SEC, ESFM, cron, Telegraf, Grafana
Lustre Tuning

- max_dirty_mb: 128
- at_min: 40s
- at_max: 600s
- osc.*.max_rpcs_in_flight: 32
- osc.*.max_pages_per_rpc: 1024
- mdc.*.max_rpcs_in_flight: 8
- mdc.*.max_pages_per_rpc: 256
- lov.qos_threshold_rr=100
- llite.max_read_ahead_mb=40