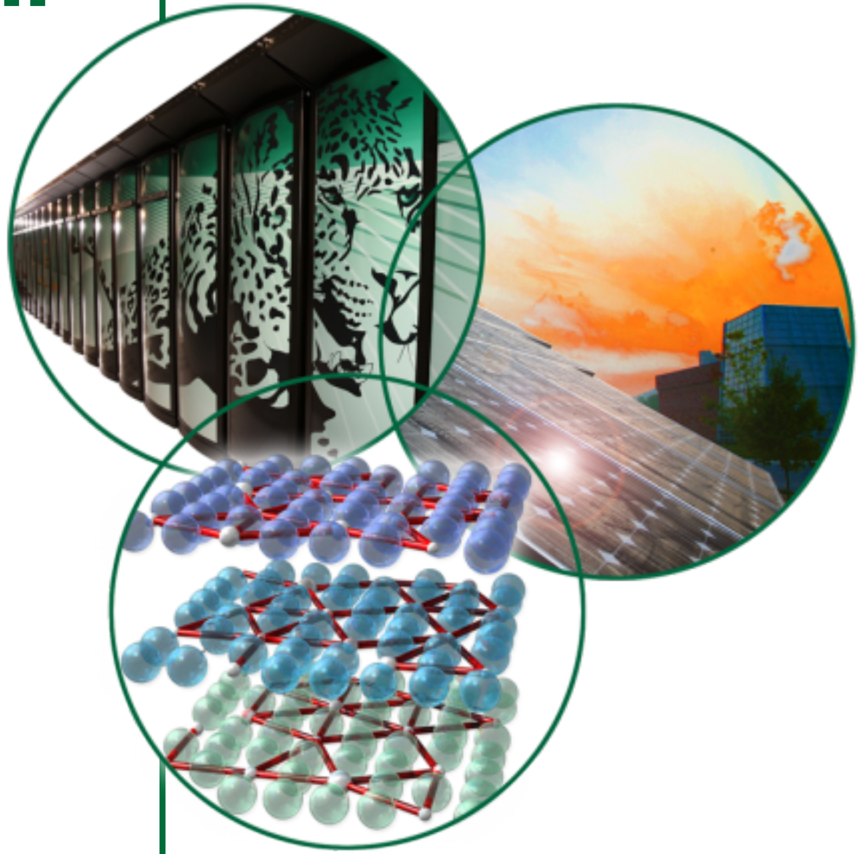
Best Practices for Scalable Administration of Lustre



Blake Caldwell
National Center for Computation Sciences

April 25, 2012
LUG 2012 – Austin, TX

U.S. DEPARTMENT OF ENERGY

OAK RIDGE NATIONAL LABORATORY
MANAGED BY UT-BATTELLE FOR THE DEPARTMENT OF ENERGY

What's different at scale?

- What we expect:
  - Overhead in administering more nodes
  - More frequent failures and new failure modes
- How we deal with them:
  - Redundancy
  - Automated monitoring and alerting
  - Scalable administration tools
  - Testing

Scale-out over time

- Deployments get staged/split/repurposed and entirely new deployments come along
  - Heterogeneous environment: hardware, software stacks, infrastructure, security policies, availability and performance requirements
- NCCS now manages 11 production Lustre filesystems
  - 272 Lustre servers (198 for Widow)
  - 5 Infiniband fabrics with 1458 HCAs
    - Different OFED stacks

Commonality of Best Practices: Consistency

- Ideal – single shared OS image
  - Capture differences within configuration management
- Reality – different hardware, maintenance procedures and timelines prevents this
- Choose flexible cluster management tools that support this abstraction
  - May still need custom tools

Best Practice 1:
Common Image for Lustre Servers

- GeDI (Generic Diskless Installer) for image creation and provisioning
  - Images built from RPMs
  - Combines read-only NFS mount with ramdisks
  - Handles creation of host specific scripts that run before init
- Benefits
  - Manage image by chroot on management server
    - Package management (yum) works
  - Stateless: powerman –r for a clean slate
- 7 of our filesystems share the widow image

Best Practice 2: Configuration Management

- Configuration management continually enforces consistency within a cluster
- Hierarchical structure for flexible shared configuration across clusters
- Version control provides accountability, history, workgroup coordination

Best Practice 3: Monitoring and Alerting

- Failures scale too
  - Need to be [made] aware of them
- Monitoring infrastructure needs to be extensible
  - Combination of Nagios, Splunk, SEC, scripts
- Nagios customizations
  - Hardware checks
    - RAID controllers
    - Nodes: OMSA
  - Lustre health, OSTs mounted, LNET stats
  - Network fabric

Best Practice 3a: Notifications for Diagnostics

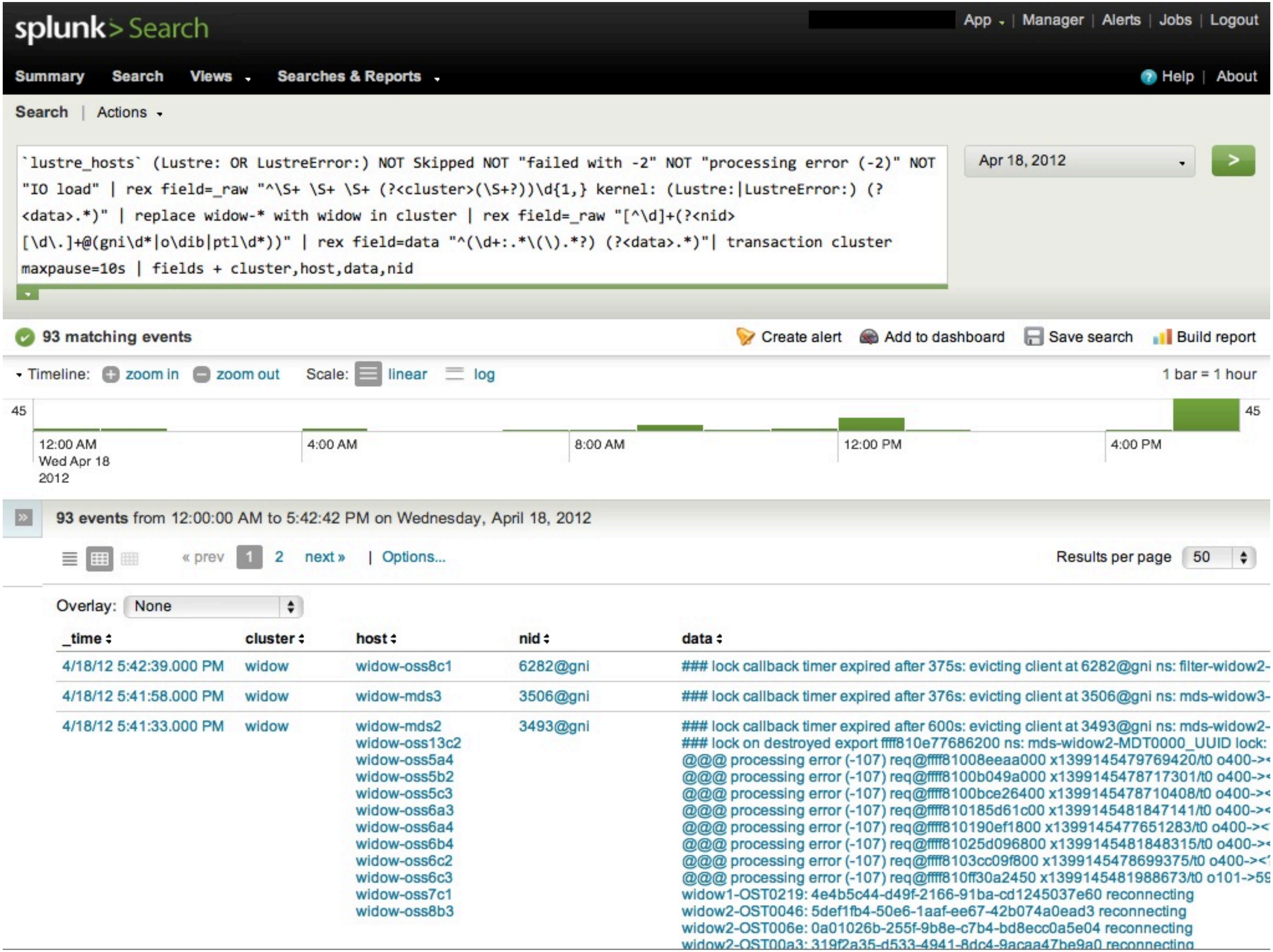
- Alerting *should* be a first diagnostic step
- Common first notifications of Lustre problems
  - Lustre health check
  - Multipath checks fail
  - Server load high or checks timeout
  - Users: “df hangs” or “a client won’t mount”
- Look at where problems slipped by without notifications for where to improve monitoring

Best Practice 3b: Monitor Storage Interconnect Health

- Any marginally functioning component could be affecting Lustre, but be masked by redundancy
- Need to address:
  - Monitor physical layer errors
    - Lost connectivity to nodes HCAs is usually obvious, Nagios checks to monitor link degradation
    - Monitor switch uplinks as well!
    - SymbolErrors make us nervous
  - Monitor IB switches (spines/line cards/fans/power supplies) just like any other network device
    - Custom Nagios plugins
  - Topology verification

Best Practice 4: Event Correlation

- Event correlation from Lustre log messages is difficult
- Splunk has SEC's functionality, but can be interactive
- Splunk alert examples:
  - Storage array logs: remove transient warnings, known bugs, and then email log
  - Storage array component failures (disk/power)
  - OSS node reboots
  - Lustre: read-only targets, symptoms of open bugs



```
`lustre_hosts` (Lustre: OR LustreError:) NOT Skipped NOT "failed with -2" NOT "processing error (-2)" NOT
"IO load" | rex field=_raw "^\S+ \S+ \S+ (?<cluster>(\S+?))\d{1,} kernel: (Lustre:|LustreError:) (?
<data>.*)" | replace widow-* with widow in cluster | rex field=_raw "[^\d]+(?<nid>
[\d\.]+@(gni\d*|o\dib|ptl\d*))" | rex field=data "^(\d+:.*\(\).*?) (?<data>.*)"| transaction cluster
maxpause=10s | fields + cluster,host,data,nid
```

93 events from 12:00:00 AM to 5:42:42 PM on Wednesday, April 18, 2012

_time	cluster	host	nid	data
4/18/12 5:42:39.000 PM	widow	widow-oss8c1	6282@gni	### lock callback timer expired after 375s: evicting client at 6282@gni ns: filter-widow2-
4/18/12 5:41:58.000 PM	widow	widow-mds3	3506@gni	### lock callback timer expired after 376s: evicting client at 3506@gni ns: mds-widow3-
4/18/12 5:41:33.000 PM	widow	widow-mds2 widow-oss13c2 widow-oss5a4 widow-oss5b2 widow-oss5c3 widow-oss6a3 widow-oss6a4 widow-oss6b4 widow-oss6c2 widow-oss6c3 widow-oss7c1 widow-oss8b3	3493@gni	### lock callback timer expired after 600s: evicting client at 3493@gni ns: mds-widow2- ### lock on destroyed export ffff810e77686200 ns: mds-widow2-MDT0000_UUID lock: @@@ processing error (-107) req@ffff81008eeaa000 x1399145479769420/t0 o400-> @@@ processing error (-107) req@ffff8100b049a000 x1399145478717301/t0 o400-> @@@ processing error (-107) req@ffff8100bce26400 x1399145478710408/t0 o400-> @@@ processing error (-107) req@ffff810185d61c00 x1399145481847141/t0 o400-> @@@ processing error (-107) req@ffff810190ef1800 x1399145477651283/t0 o400-> @@@ processing error (-107) req@ffff81025d096800 x1399145481848315/t0 o400-> @@@ processing error (-107) req@ffff8103cc09f800 x1399145478699375/t0 o400->< @@@ processing error (-107) req@ffff810ff30a2450 x1399145481988673/t0 o101->59 widow1-OST0219: 4e4b5c44-d49f-2166-91ba-cd1245037e60 reconnecting widow2-OST0046: 5def1fb4-50e6-1aaf-ee67-42b074a0ead3 reconnecting widow2-OST006e: 0a01026b-255f-9b8e-c7b4-bd8ecc0a5e04 reconnecting widow2-OST00a3: 319f2a35-d533-4941-8dc4-9acaa47be9a0 reconnecting

Best Practice 5: Diagnostic Procedures

- Collect from clients:
  - Collect crash dumps (kdump)
  - Lctl dk or debug daemon
  - Timeouts
    - `lctl get_param –n ost.*.ost_io.timeouts`
- On management server
  - Aggregate kernel/Lustre syslog messages
  - IPMI console logging (conman)

Best Practice 6: Workload Characterization

- Need to determine if slow response time an issue or expected behavior
- We have scripts that generate “MDS Trace Reports”
  - Correlate Cray XK apstat information on jobs with rpctrace from /proc/sys/lnet/debug
  - Latencies by RPC type (e.g. LDLM_ENQUEUE)
    - Email if LDLM_ENQUEUE >= 1s
  - Top RPC intensive jobs (correlated with job size)

Best Practice 7:
Fill in the gaps with custom tools

- Implement purge policy
  - We use ne2scan/genhit/purge from Nick Cardo at NERSC
- Usage by user/project
  - Lustre DU – pulls usage data from DB instead of metadata
- Performance statistics
  - DDNTool – polls DDN S2A 9900 performance and environmental stats via API, then stores in DB

Summary

- We need consistency at scale
- Administration best practices
  1. Common OS image
  2. Configuration management
  3. Monitoring and Alerting
  4. Event correlation
  5. Diagnostic procedures
  6. Workload characterization
  7. Custom tools

Resources

- DDNTool/Lustre DU
  - J. Hill, D. Leverman, S. Koch, D. Dillow. “Determining the health of Lustre filesystems at scale.” Cray User Group 2011, Fairbanks, AK. 1 May 2011. Conference Presentation.
  - http://info.ornl.gov/sites/publications/files/Pub28556.pdf
- MDS Trace Tool
  - R. Miller, J. Hill, D. Dillow, R. Gunasekaran, D. Maxwell. “Monitoring tools for large scale systems.” Cray User Group 2010. Edinburgh. Scotland. 24 May 2011. Conference Proceedings.
- GeDI
  - http://sourceforge.net/projects/gedi-tools/
- Splunk
  - http://www.splunk.com
- Linux@LLNL Software
  - https://computing.llnl.gov/linux/downloads.html