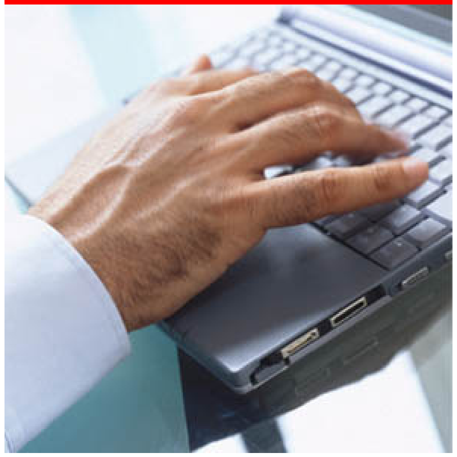
ORACLE®

Lustre Development

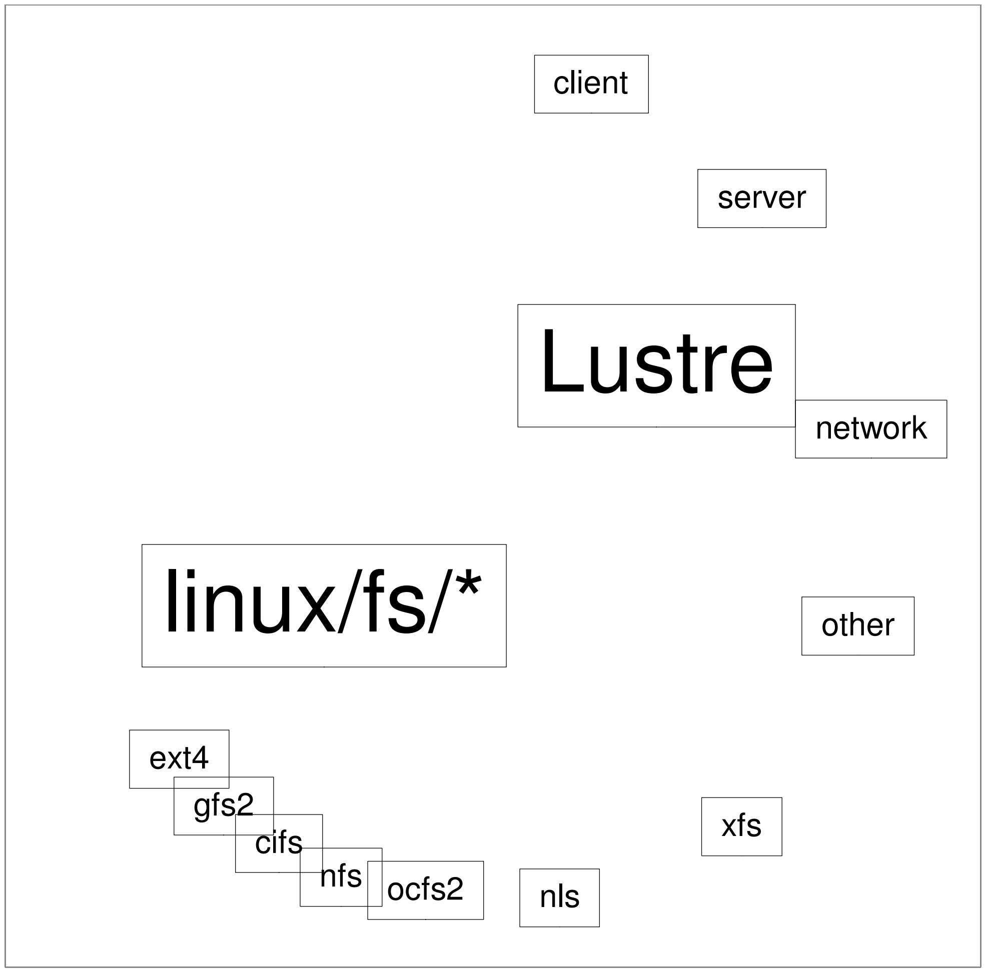
Eric Barton
Lead Engineer, Lustre Group

Lustre Development
Agenda

- Engineering
  - Improving stability
  - Sustaining innovation
- Development
  - Scaling and performance
  - Ldiskfs and DMU
- Research
  - Scaling
  - Performance
  - Resilience

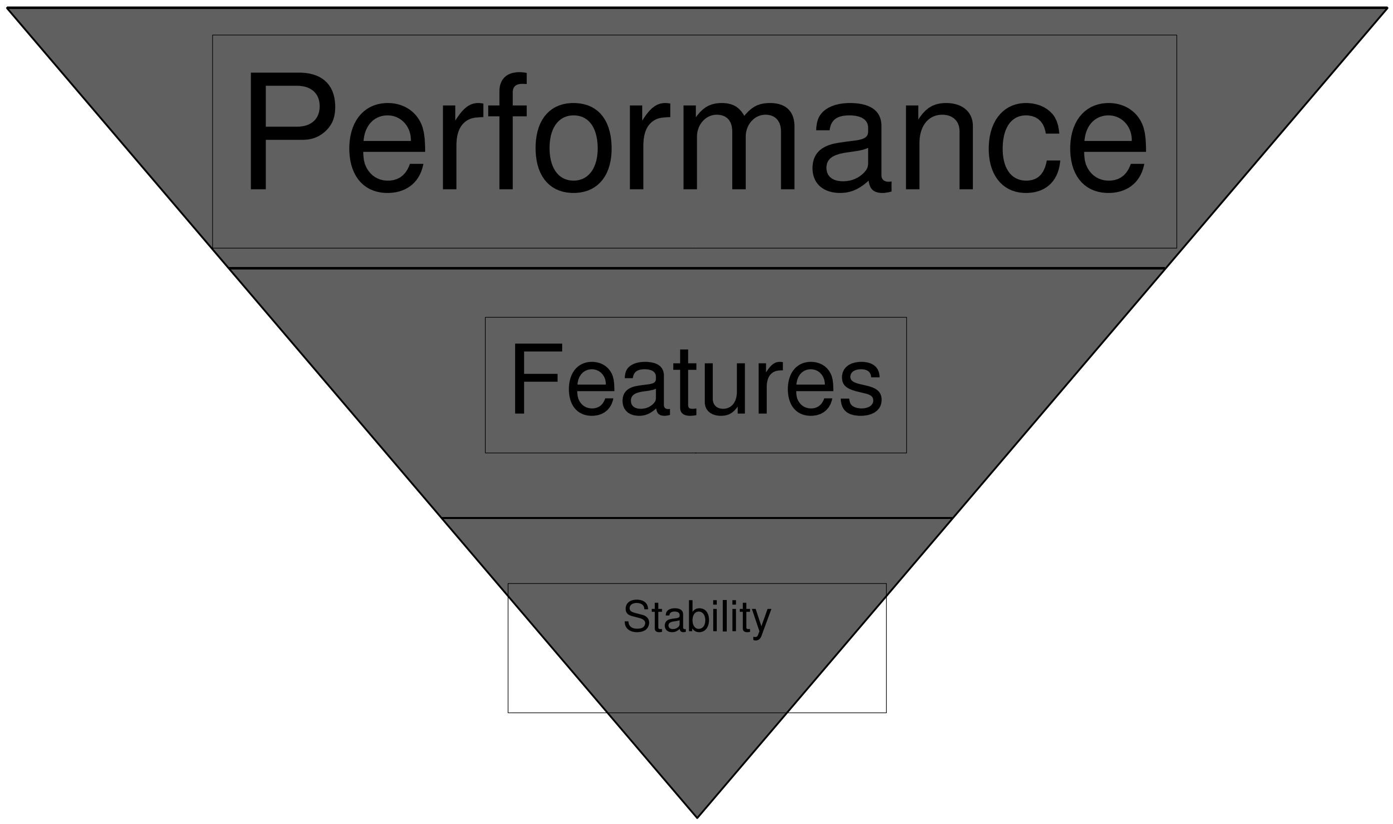


Engineering
Lines of Code



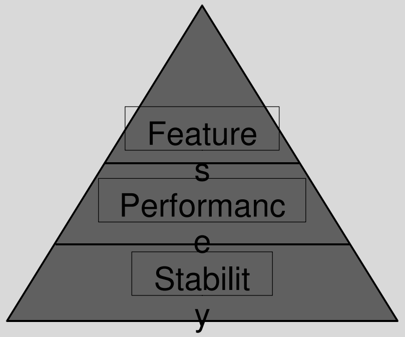
- Lustre – 257 KLOC
- Total of all in-tree linux filesystems – 471 KLOC

Engineering
Historical Priorities



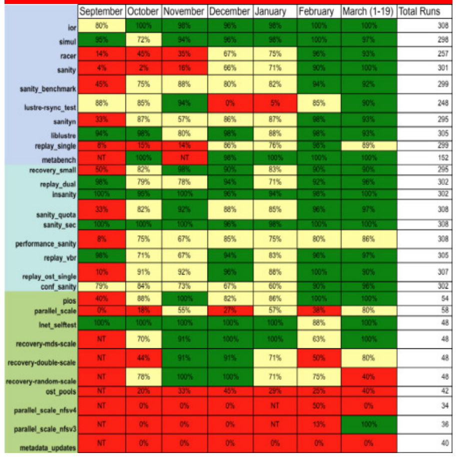
Engineering Priorities

- Stability
  - Reduce support incident rate
  - Reliable / predictable development
  - Address technical debt
- Performance & Scaling
  - Prevent performance regression
  - Exploit hardware improvements
- Features
  - Improve fault tolerance / recovery
  - Improve manageability



Engineering Knowledge

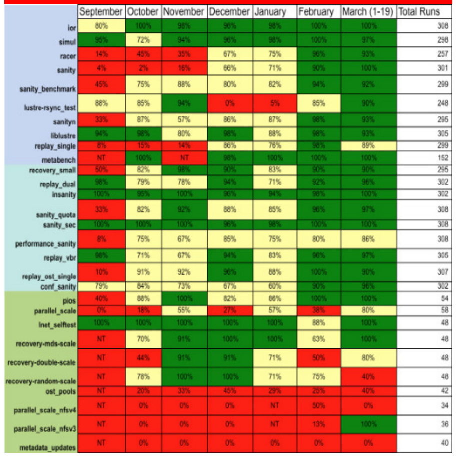
- ORNL
  - “Understanding Lustre Filesystem Internals”
- Lustre internals documentation project
  - Work in progress
  - Continuously maintained
- Subsystem map
- Narrative documentation
  - Asciidoc
- Api documentation
  - Doxygen



	September	October	November	December	January	February	March (1-19)	Total Runs
ior	80%	100%	98%	96%	98%	100%	100%	308
simul	95%	72%	94%	96%	98%	100%	97%	298
racer	14%	45%	35%	67%	75%	96%	93%	257
sanity	4%	2%	16%	66%	71%	90%	100%	301
sanity_benchmark	45%	75%	88%	80%	82%	94%	92%	299
lustre-rsync_test	88%	85%	94%	0%	5%	85%	90%	248
sanityn	33%	87%	57%	86%	87%	98%	93%	295
liblustre	94%	98%	80%	98%	88%	98%	93%	305
replay_single	8%	15%	14%	86%	76%	98%	89%	299
metabench	NT	100%	NT	98%	100%	100%	100%	152
recovery_small	50%	82%	98%	90%	83%	90%	90%	295
replay_dual	98%	79%	78%	94%	71%	92%	96%	302
insanity	100%	95%	100%	96%	94%	98%	100%	302
sanity_quota	33%	82%	92%	88%	85%	96%	97%	308
sanity_sec	100%	100%	100%	96%	98%	100%	100%	308
performance_sanity	8%	75%	67%	85%	75%	80%	86%	308
replay_vbr	98%	71%	67%	94%	83%	96%	97%	305
replay_ost_single	10%	91%	92%	96%	88%	100%	90%	307
conf_sanity	79%	84%	73%	67%	60%	90%	96%	302
pios	40%	88%	100%	82%	86%	100%	100%	54
parallel_scale	0%	18%	55%	27%	57%	38%	80%	58
lnet_selftest	100%	100%	100%	100%	100%	88%	100%	48
recovery-mds-scale	NT	70%	91%	100%	100%	63%	100%	48
recovery-double-scale	NT	44%	91%	91%	71%	50%	80%	48
recovery-random-scale	NT	78%	100%	100%	71%	75%	40%	48
ost_pools	NT	20%	33%	45%	29%	25%	40%	42
parallel_scale_nfsv4	NT	0%	0%	0%	NT	50%	0%	34
parallel_scale_nfsv3	NT	0%	0%	0%	NT	13%	100%	36
metadata_updates	NT	0%	0%	0%	0%	0%	0%	40

Engineering
Branch management

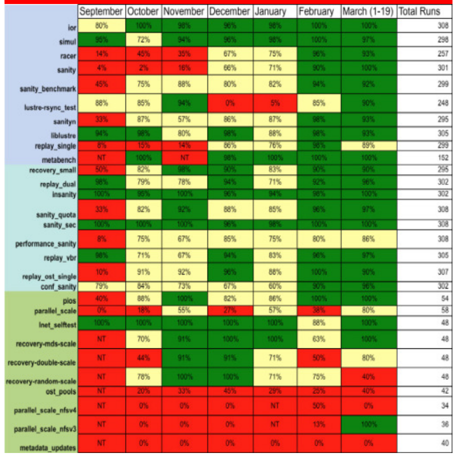
- Prioritize major development branch stability
  - Solid foundation
  - Reliable / early regression detection
  - Predictable / sustainable development
- Gatekeeper
  - Control landing schedule
  - Enforce defective patch backout
  - Influence patch size for inspection / test
- Git
  - Retained all significant CVS history
  - Single repository covers everything
  - Much easier backouts



	September	October	November	December	January	February	March (1-19)	Total Runs
ior	80%	100%	98%	96%	98%	100%	100%	308
simul	95%	72%	94%	96%	98%	100%	97%	298
racer	14%	45%	35%	67%	75%	96%	93%	257
sanity	4%	2%	16%	66%	71%	90%	100%	301
sanity_benchmark	45%	75%	88%	80%	82%	94%	92%	299
lustre-rsync_test	88%	85%	94%	0%	5%	85%	90%	248
sanityn	33%	87%	57%	86%	87%	98%	93%	295
liblustre	94%	98%	80%	98%	88%	98%	93%	305
replay_single	8%	15%	14%	86%	76%	98%	89%	299
metabench	NT	100%	NT	98%	100%	100%	100%	152
recovery_small	50%	82%	98%	90%	83%	90%	90%	295
replay_dual	98%	79%	78%	94%	71%	92%	96%	302
insanity	100%	95%	100%	96%	94%	98%	100%	302
sanity_quota	33%	82%	92%	88%	85%	96%	97%	308
sanity_sec	100%	100%	100%	96%	98%	100%	100%	308
performance_sanity	8%	75%	67%	85%	75%	80%	86%	308
replay_vbr	98%	71%	67%	94%	83%	96%	97%	305
replay_ost_single	10%	91%	92%	96%	88%	100%	90%	307
conf_sanity	79%	84%	73%	67%	60%	90%	96%	302
pios	40%	88%	100%	82%	86%	100%	100%	54
parallel_scale	0%	18%	55%	27%	57%	38%	80%	58
lnet_selftest	100%	100%	100%	100%	100%	88%	100%	48
recovery-mds-scale	NT	70%	91%	100%	100%	63%	100%	48
recovery-double-scale	NT	44%	91%	91%	71%	50%	80%	48
recovery-random-scale	NT	78%	100%	100%	71%	75%	40%	48
ost_pools	NT	20%	33%	45%	29%	25%	40%	42
parallel_scale_nfsv4	NT	0%	0%	0%	NT	50%	0%	34
parallel_scale_nfsv3	NT	0%	0%	0%	NT	13%	100%	36
metadata_updates	NT	0%	0%	0%	0%	0%	0%	40

Engineering
Test

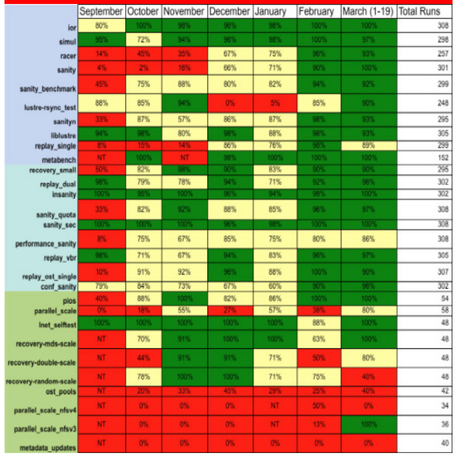
- Hyperion
  - 100s of client nodes
    - Multimount – simulate 1000s of clients
  - Multiple test runs weekly
  - Leverage much earlier in development cycle
- Daily automated testing
  - Results vetting
- Improved defect observability
  - See trends
  - Discern regular v. intermittent issues
  - Early regression detection



	September	October	November	December	January	February	March (1-19)	Total Runs
ior	80%	100%	98%	96%	98%	100%	100%	308
simul	95%	72%	94%	96%	98%	100%	97%	298
racer	14%	45%	35%	67%	75%	96%	93%	257
sanity	4%	2%	16%	66%	71%	90%	100%	301
sanity_benchmark	45%	75%	88%	80%	82%	94%	92%	299
lustre-rsync_test	88%	85%	94%	0%	5%	85%	90%	248
sanityn	33%	87%	57%	86%	87%	98%	93%	295
liblustre	94%	98%	80%	98%	88%	98%	93%	305
replay_single	8%	15%	14%	86%	76%	98%	89%	299
metabench	NT	100%	NT	98%	100%	100%	100%	152
recovery_small	50%	82%	98%	90%	83%	90%	90%	295
replay_dual	98%	79%	78%	94%	71%	92%	96%	302
insanity	100%	95%	100%	96%	94%	98%	100%	302
sanity_quota	33%	82%	92%	88%	85%	96%	97%	308
sanity_sec	100%	100%	100%	96%	98%	100%	100%	308
performance_sanity	8%	75%	67%	85%	75%	80%	86%	308
replay_vbr	98%	71%	67%	94%	83%	96%	97%	305
replay_ost_single	10%	91%	92%	96%	88%	100%	90%	307
conf_sanity	79%	84%	73%	67%	60%	90%	96%	302
pios	40%	88%	100%	82%	86%	100%	100%	54
parallel_scale	0%	18%	55%	27%	57%	38%	80%	58
lnet_selftest	100%	100%	100%	100%	100%	88%	100%	48
recovery-mds-scale	NT	70%	91%	100%	100%	63%	100%	48
recovery-double-scale	NT	44%	91%	91%	71%	50%	80%	48
recovery-random-scale	NT	78%	100%	100%	71%	75%	40%	48
ost_pools	NT	20%	33%	45%	29%	25%	40%	42
parallel_scale_nfsv4	NT	0%	0%	0%	NT	50%	0%	34
parallel_scale_nfsv3	NT	0%	0%	0%	NT	13%	100%	36
metadata_updates	NT	0%	0%	0%	0%	0%	0%	40

Engineering Process

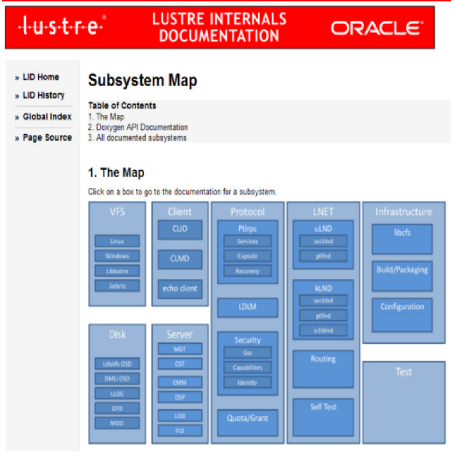
- Clear release objectives
  - Manage risk – stability / schedule uncertainty
  - Release blockers defined by bug priority
- Bi-weekly builds
  - Formal test plans
  - Prioritize test issues
- Daily review
  - Engineering progress
  - Testing results
  - Issue priorities



	September	October	November	December	January	February	March (1-19)	Total Runs
ior	80%	100%	98%	96%	98%	100%	100%	308
simul	95%	72%	94%	96%	98%	100%	97%	298
racer	14%	45%	35%	67%	75%	96%	93%	257
sanity	4%	2%	16%	66%	71%	90%	100%	301
sanity_benchmark	45%	75%	88%	80%	82%	94%	92%	299
lustre-rsync_test	88%	85%	94%	0%	5%	85%	90%	248
sanityn	33%	87%	57%	86%	87%	98%	93%	295
liblustre	94%	98%	80%	98%	88%	98%	93%	305
replay_single	8%	15%	14%	86%	76%	98%	89%	299
metabench	NT	100%	NT	98%	100%	100%	100%	152
recovery_small	50%	82%	98%	90%	83%	90%	90%	295
replay_dual	98%	79%	78%	94%	71%	92%	96%	302
insanity	100%	95%	100%	96%	94%	98%	100%	302
sanity_quota	33%	82%	92%	88%	85%	96%	97%	308
sanity_sec	100%	100%	100%	96%	98%	100%	100%	308
performance_sanity	8%	75%	67%	85%	75%	80%	86%	308
replay_vbr	98%	71%	67%	94%	83%	96%	97%	305
replay_ost_single	10%	91%	92%	96%	88%	100%	90%	307
conf_sanity	79%	84%	73%	67%	60%	90%	96%	302
pios	40%	88%	100%	82%	86%	100%	100%	54
parallel_scale	0%	18%	55%	27%	57%	38%	80%	58
lnet_selftest	100%	100%	100%	100%	100%	88%	100%	48
recovery-mds-scale	NT	70%	91%	100%	100%	63%	100%	48
recovery-double-scale	NT	44%	91%	91%	71%	50%	80%	48
recovery-random-scale	NT	78%	100%	100%	71%	75%	40%	48
ost_pools	NT	20%	33%	45%	29%	25%	40%	42
parallel_scale_nfsv4	NT	0%	0%	0%	NT	50%	0%	34
parallel_scale_nfsv3	NT	0%	0%	0%	NT	13%	100%	36
metadata_updates	NT	0%	0%	0%	0%	0%	0%	40

Development Priorities

- Lustre 1
  - Maintenance
- Lustre 2
  - Stabilization
  - Performance
    - Eliminate regressions
    - Land improvements
  - Features


LUSTRE INTERNALS DOCUMENTATION

» LID Home
» LID History
» Global Index
» Page Source

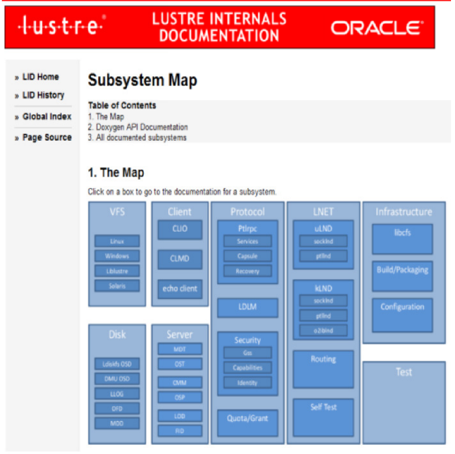
Subsystem Map

Table of Contents
1. The Map
2. Doxygen API Documentation
3. All documented subsystems

1. The Map

Click on a box to go to the documentation for a subsystem.

Development
Projects


LUSTRE INTERNALS DOCUMENTATION

» LID Home
» LID History
» Global Index
» Page Source

Subsystem Map

Table of Contents
1. The Map
2. Doxygen API Documentation
3. All documented subsystems

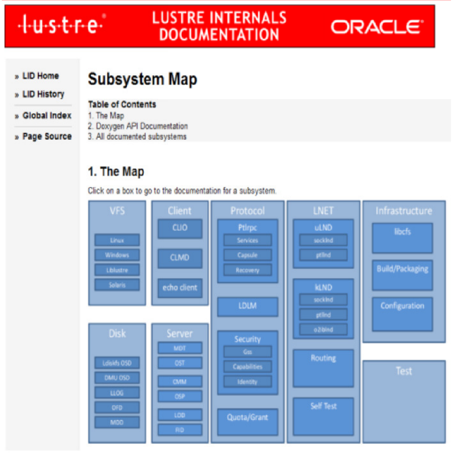
1. The Map

Click on a box to go to the documentation for a subsystem.

- SMP scaling
  - Exploit multicore servers
  - Improve metadata throughput
- Platform portability
  - Extend OS-specific / portable layering to metadata
  - Formalize porting primitives
- Ldiskfs / DMU(ZFS) OSD
  - Pluggable storage subsystem
- HSM
- Clean server shutdown / restart
  - Simplify version interoperation / rolling upgrade

Development
Imperative Recovery

- Explicit client notification on server restart



LUSTRE INTERNALS DOCUMENTATION

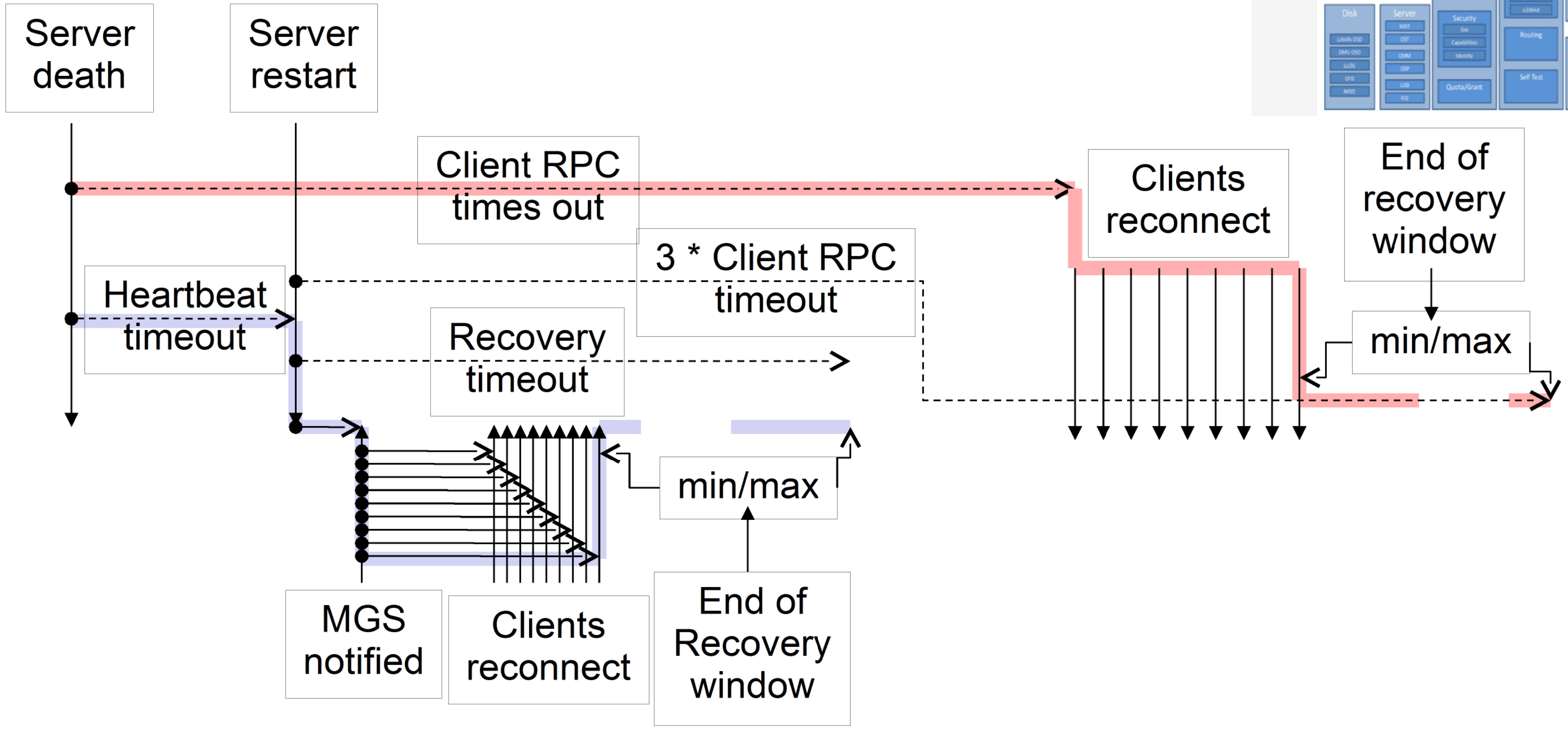
» LID Home
» LID History
» Global Index
» Page Source

Subsystem Map

Table of Contents
1. The Map
2. Doxygen API Documentation
3. All documented subsystems

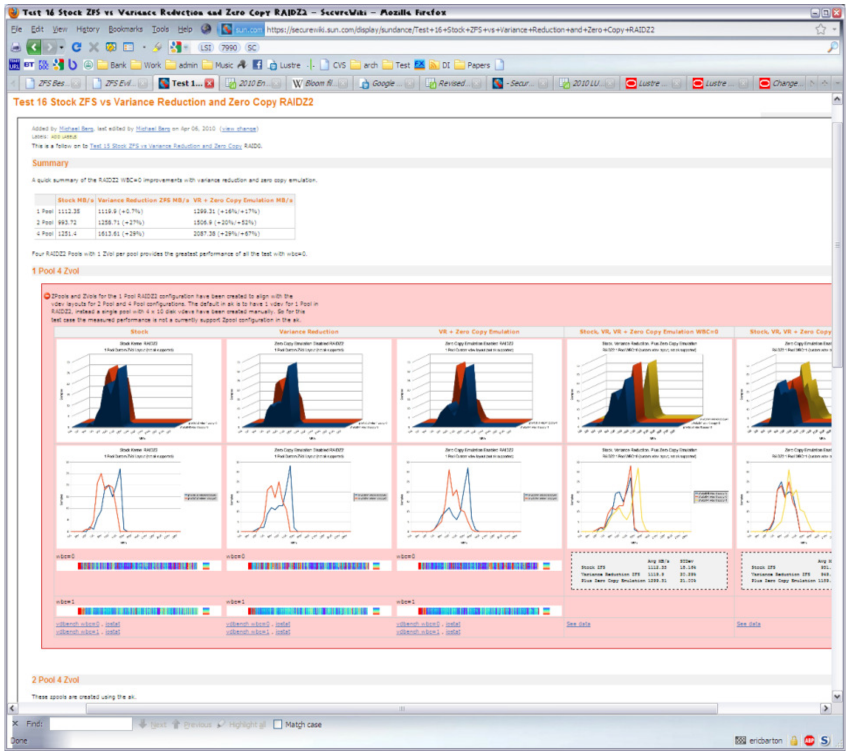
1. The Map

Click on a box to go to the documentation for a subsystem.



Development
DMU performance

- Continued comprehensive benchmarking
- ZFS enhancements
  - Zero copy
  - Improved disk utilization
- Close cooperation with ZFS development team



Test 16 Stock ZFS vs Variance Reduction and Zero Copy RAIDZ2

Added by Michael Berg, last edited by Michael Berg on Apr 06, 2010 (view change)

This is a follow on to Test 15 Stock ZFS vs Variance Reduction and Zero Copy RAID0.

Summary

A quick summary of the RAIDZ2 WBC=0 improvements with variance reduction and zero copy emulation.

	Stock MB/s	Variance Reduction ZFS MB/s	VR + Zero Copy Emulation MB/s
1 Pool	1112.35	1119.9 (+0.7%)	1299.31 (+16%/+17%)
2 Pool	993.72	1258.71 (+27%)	1506.9 (+20%/+52%)
4 Pool	1251.4	1613.61 (+29%)	2087.38 (+29%/+67%)

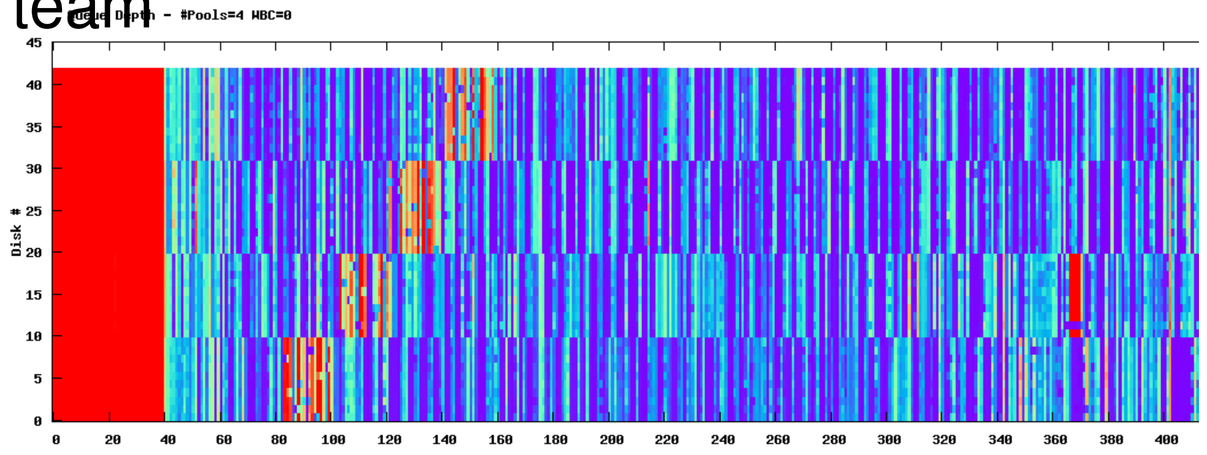
Four RAIDZ2 Pools with 1 Zvol per pool provides the greatest performance of all the test with wbc=0.

1 Pool 4 Zvol

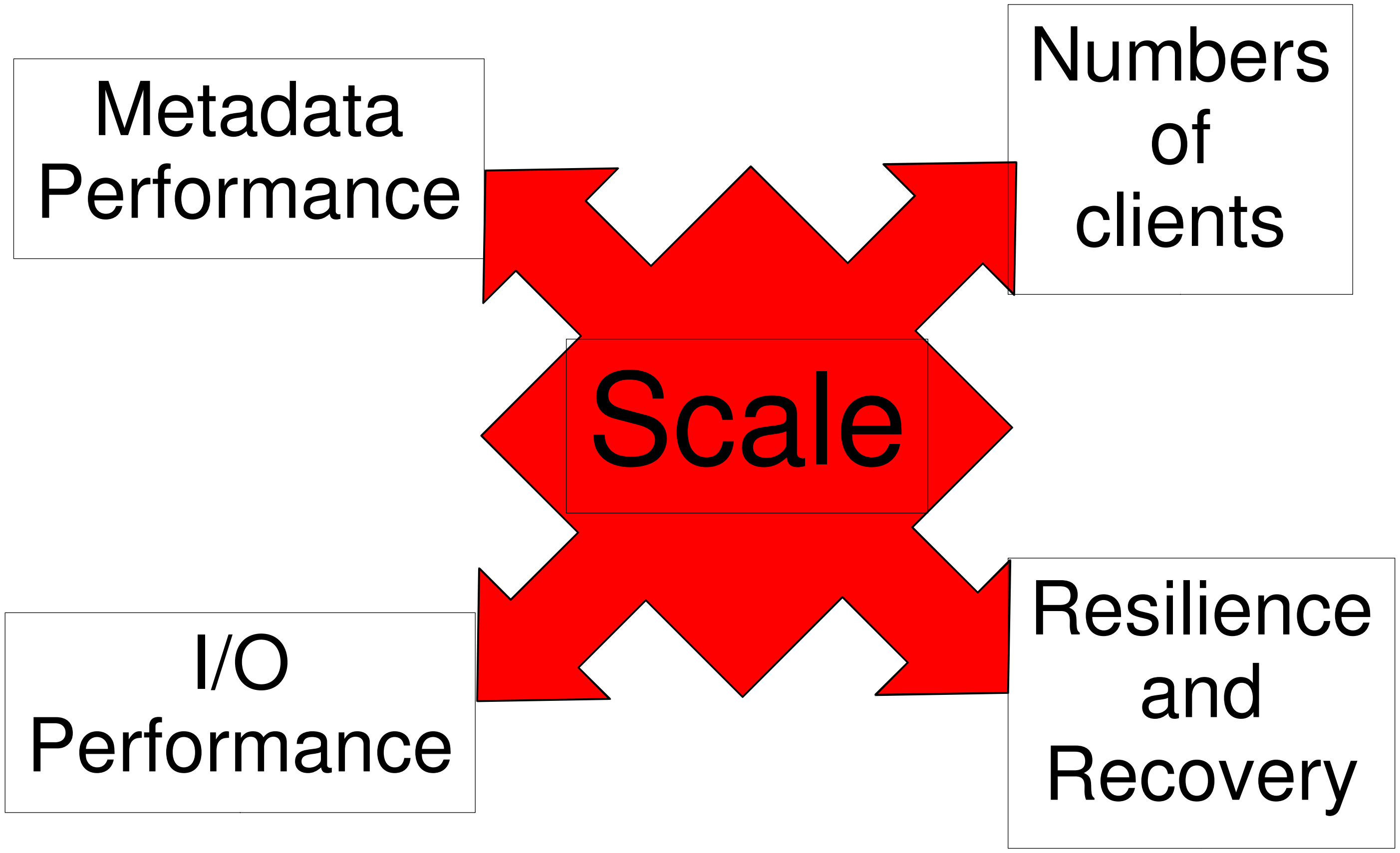
ZPools and ZVols for the 1 Pool RAIDZ2 configuration have been created to align with the vdev layouts for 2 Pool and 4 Pool configurations. The default in ak is to have 1 vdev for 1 Pool in RAIDZ2, instead a single pool with 4 x 10 disk vdevs have been created manually. So for this test case the measured performance is not a currently support Zpool configuration in the ak.

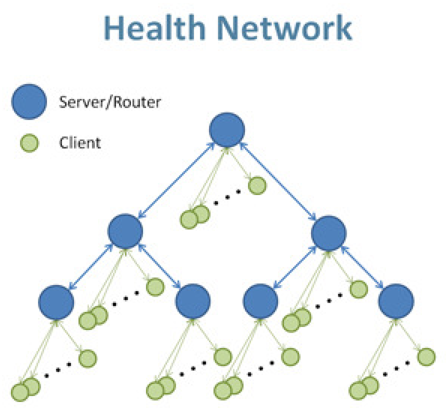
2 Pool 4 Zvol

These zpools are created using the ak.



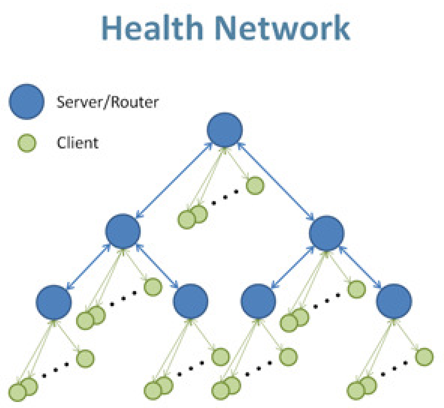
Research **Priorities**





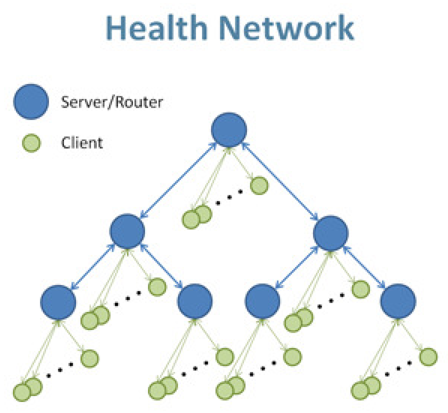
Research
Numbers of Clients

- Currently able to accommodate 10,000s
- Next steps
  - System call forwarders - 10-100x
- Further steps
  - Caching proxies
  - Subtree locking



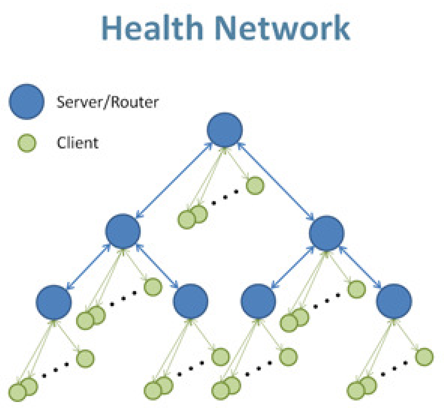
Research
I/O

- Initial NRS experiments encouraging
  - 40% Read improvement
  - 60% Write improvement
- Next steps
  - Larger scale prototype benchmarking
  - Exploit synergy with SMP scaling work
- Further steps
  - Global NRS policies
  - Quality of service



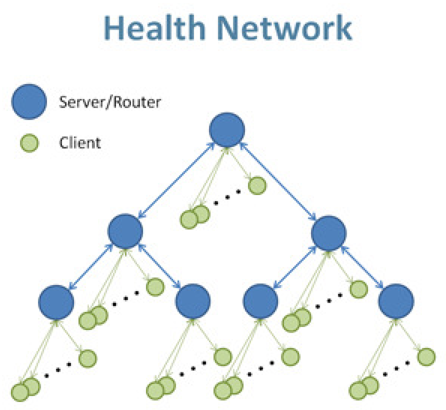
Research
Metadata

- SMP scaling
  - Deeper locking / CPU affinity issues
- CMD Preview
  - Sequenced / synched distributed updates
  - Characterise performance
- Next Steps
  - Productize CMD Preview
- Further Steps
  - CMD based on epochs



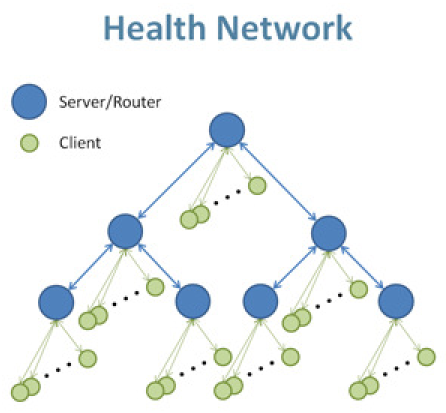
ORACLE

Research
Resilience & Recovery



- O(n) pinger overhead / detection latency
- Overreliance on client timeouts
  - O(n) to distinguish server congestion from death
  - Include disk latency
  - Required to detect LNET router failure
- Over-eager server timeouts
  - Can't distinguish LNET router failure from client death
- Recovery affects everyone
  - Transparency not guaranteed after recovery window expires
    - COS/VBR only partial solution
  - MDT outage disconnects namespace

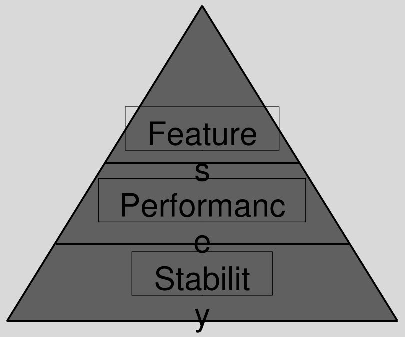
Research
Resilience & Recovery



- Scalable health network design
  - Out-of-band communications
  - Low latency global notifications
  - Collectives: Census, LOVE reduction etc
  - Clear completion & network partition semantics
  - Self-healing
- Next steps
  - HN prototype
  - OST mirroring
- Further steps
  - Epoch based SNS

Lustre Development
Summary

- Prioritize stability
  - Continued product quality improvements
  - Predictable release schedule
  - Sustainable development
- Continued innovation
  - Prioritized development schedule
  - Planned product evolution



ORACLE®