QUALITY INITIATIVE

Robert Read

Quality Initiative

- What is it?
- Where do we go from here?

QE Successes

- LBATS - build automation on 4 architectures and OSs
- YALA - test automation
- Stage 2 testing automation
- Feature testing
- Found many bugs in our product

Overview

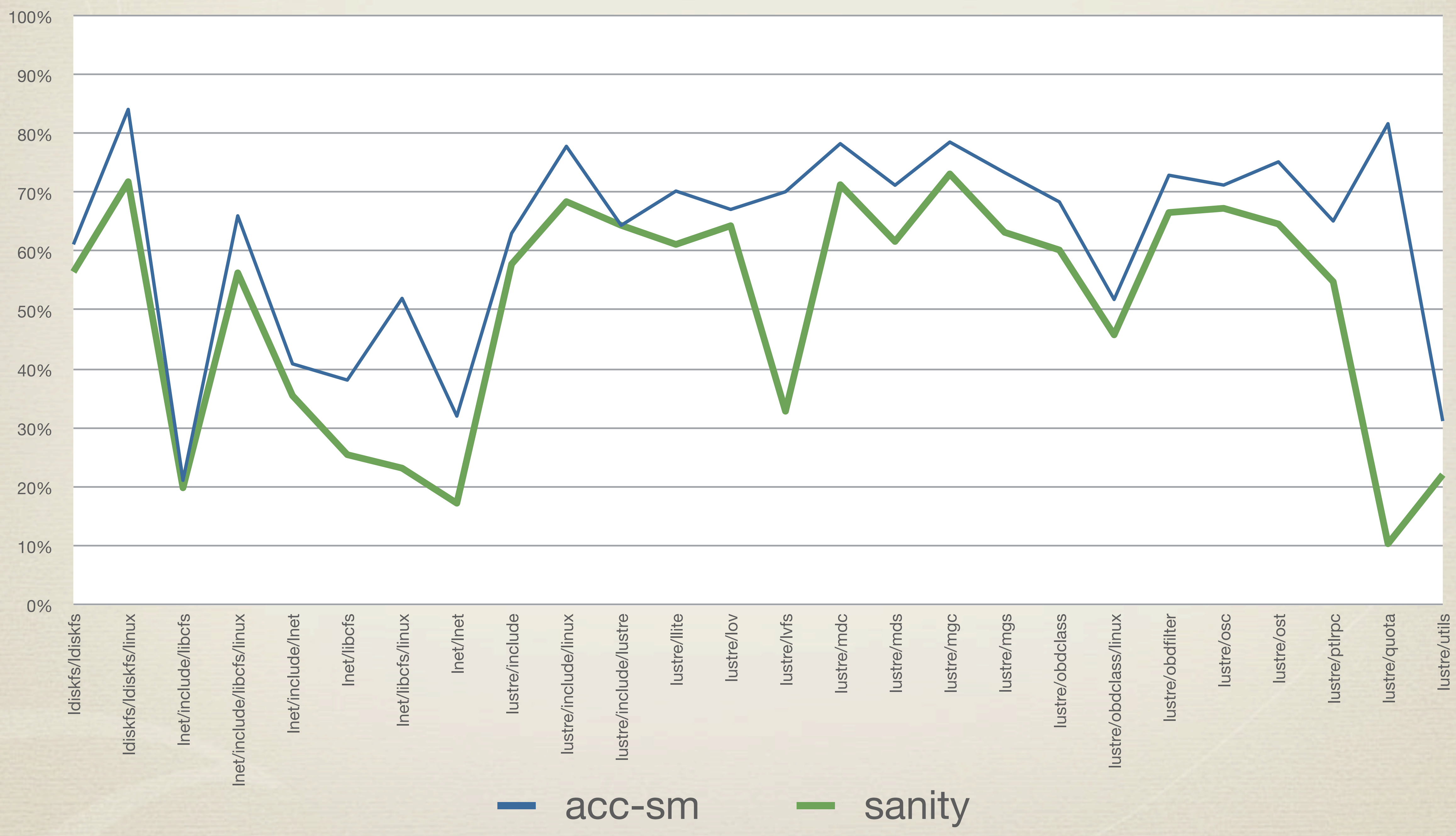
- Feedback
- Coverage
- Automation & Infrastructure

FEEDBACK

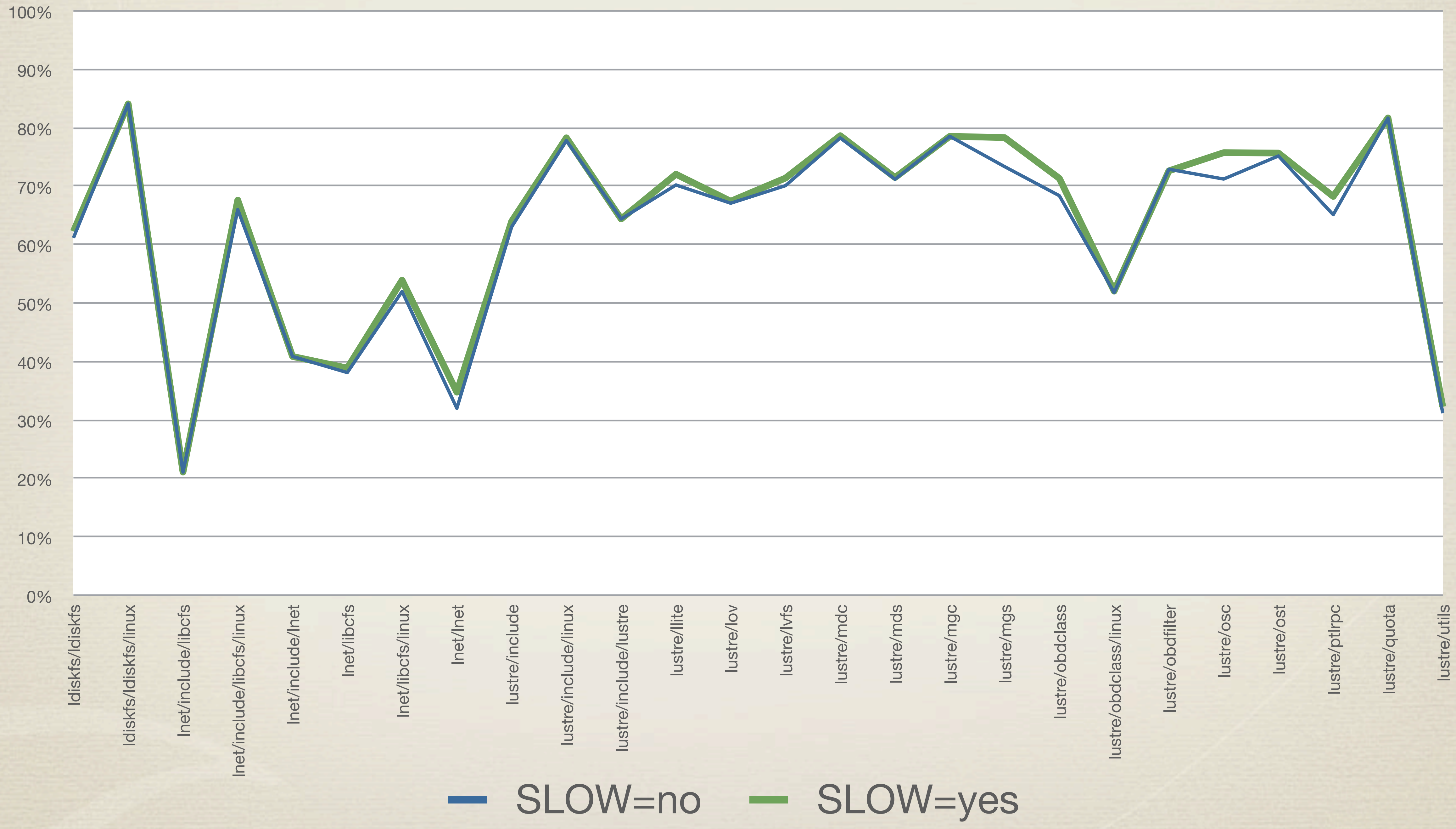
Existing coverage analysis

- Li Wei is just starting analysis
- sanity.sh on single node achieves 50% coverage overall
  - excluding liblustre, libsysio, socklnd, lnet selftest, etc
- 60-70% coverage of core Lustre modules

sanity vs. acc-sm



acc-sm vs. acc-sm



acceptance-small

SLOW=no	61.5%
SLOW=yes	63.1%

- We need to be smarter about our tests
- https://wikis.clusterfs.com/intra/index.php/Test_Coverage

Customer reported issues

* As part of QI we have been talking to customers and partners
* Understand how they hit bugs that we missed
* Share our test plans, which we are doing now with Cray

Cray

- Enable -Werror (Girish did this)
- Concurrent application mix
- Pools should not affect roll-back to pre-1.8 releases
- Interaction of OST Pools and ACLs/quotas
- Testing with failover/recovery

HP

- Run racer with at least 4 clients
  - They noticed 1.6.6 MDS hangs easily with 4 clients
- More failover/recovery testing

LLNL

- Took >6 months to stabilize 1.6.6
- Several attempts to pass on 450 node test
- They have over 50 patches on top of 1.6.6

LLNL Requests

- Large scale stress testing (1000+ clients)
- Router testing
- Multiple Lustre fs
- OSS nodes fail daily; sometimes a single OSS failure downs whole fs
- Dogfood - /home on lustre
- Stack overflow

LLNL Requests (cont.)

- Concerns about MD performance regressions
- ls -l and df perf while running jobs too slow
- 2 NICs and one NID clients don't use both of servers nic
- Memory regressions
- General reliability concerns

COVERAGE

Goals

- Smarter testing
  - test more in less time with less resources
- More comprehensive and realistic tests
- More stress testing
- Go deeper in our feature testing
  - recovery, routers, new features

Our Test Hierarchy

- Unit Tests
  - Engineers write new test cases
- Feature Tests
  - Automated feature tests (e.g. sanity-quota.sh)
  - Feature tests developed and performed by QE
- Integration Test
  - acceptance-small runs the automated feature tests

Feature Tests

- Recovery
  - Most tests - still not production ready
- Adaptive timeouts
  - Small handful of unit tests
  - Learned much more by scale testing at LLNL

Realistic Testing

- Realistic work loads
  - Real applications if possible
  - New MPI tests
- Ensure Lustre can do used “normally”
- Emphasize scale testing

Redefine Testing Levels

- Improve on SLOW=yes
- Well defined testing levels
  - Same tests always run for a given level
- All should be runnable by developers in local environment
  - And by customers

Testing Levels

- Level I - basic integration
- Level II - thorough integration, real failovers
- Level III - larger scale tests (>4 nodes), long running
- ... more as needed?

AUTOMATION & INFRASTRUCTURE

Goals

- Provide better tools for developers
- Manage information
- Better resource utilization
- Automated post check-in build and test
  - (for every commit or batch of commits)

test-framework.sh

- Original testing environment
- Fragile bash code
- Limited ability to create abstractions
- Very difficult to manage complex configurations

Lustre configuration

- Customers have difficulty running acc-sm
- Standardize how configuration is stored and used by tests
- lustre_config is current "supported" lustre configuration tool

What We Need

- MPI support
- Integrate with llapi
  - Perhaps adding more functionality
- Support diverse environments
- Provide abstractions useful for testing

Test Environment

- New environment being proposed
  - Initially focused on MPI support
  - New configuration support
  - Python or Ruby
  - Explore existing test frameworks
- Run alongside existing tests

Test results & metrics

- Detailed test tracking
  - individual tests
  - pass/fail/skip
  - duration/error message
  - other metrics would be nice
- History of individual tests (.e.g "sanity test_501g")

Autovetting

* Detect test failures when they happen
* Search bugzilla for potentially related failures
* Optionally update existing bug or create new one
* Web interface to interactively review failures and create new tickets

More data collection

- llcov (test coverage)
- rpc traces
- profiling data

Post-run Analysis

- Save detailed test info searchable format (database)
- Compare test runs
  - find new failures
  - perf regressions
- Chop search to find regressions
- Update bugzilla from autovetted data

YALA Improvements

- Need more reporting and analysis
- Perf-Pit has some of these features already
- An intern on Perf-pit team will be working on improving YALA for us

Testing on Xen

- much more efficient than VMware, esp. for kernel code
- update guest kernel from outside guest (although not the modules)
- guests boot quickly (~6s on my machine)
- supports shared virtual block devices, real failover testing is easy

Xen Usage

- Fine for Level I testing
- Developers can run Level II
- Initial feature testing by developers

SUMMARY

Areas of Improvement

- Coverage
  - Understand our existing tests
  - Focus on real-world scenarios
- Automation
  - Manage test result data
  - Easier to write and use
- Improve Reporting

Quality Initiative

Robert Read

rread@sun.com