
whamcloud

Lustre Community Development

Eric Barton
CTO
Whamcloud, Inc.

Agenda

- Community Development
  - Stability
  - Engineering
  - Roadmap
  - Collaboration
  - Distribution
- Community Testing
  - What is testing
  - Why is it uniquely hard for Lustre
  - Standardised testing

Stability Stability Stability

- Customer Perspective
  - Filesystem is a mission critical resource
    - Downtime impact can be site-wide
  - Large scale production systems
    - Downtime extremely expensive
  - Valuable Data
    - Loss / corruption unacceptable
  - Staffing to “nurse” unstable systems extremely expensive
- Developer Perspective
  - Development must tread on firm ground
    - Landing features without rigorous QA destabilizes the codebase
  - 2.0 required over 2 years of continuous stabilization effort
    - Not including some performance regressions
    - We’re not doing that again

Engineering

- The facts
  - Lustre is big (250KLoc) and complex
  - Moore's law drives requirements relentlessly
  - Instability loses customers *and* halts development
- The strategy
  - Architect for the *long term*
    - Detours and reverses are prohibitively expensive
    - Current developments must progress towards future goals
  - Develop in baby steps
    - Minimize planning/estimation uncertainty
    - Instability costs much more to fix than to avoid
  - Control technical debt
    - Unnecessary complexity invites instability
    - Restructuring is cheaper in the long run

Roadmap

- The facts
  - Customers and Vendors need a roadmap
  - Everyone is over optimistic
  - Planning uncertainty grows exponentially with range
  - Production QA adds many months to feature release schedules
- The strategy
  - Publish agreed long term direction and goals
    - No dates
  - Publish release schedule
    - Dates for long cycle feature / short cycle bugfix releases
  - Use train model of development
    - Target particular release / slip to next release when late
  - No assurances for the bleeding edge
    - Patch soup == individual responsibility

Collaboration

- The facts
  - Independent development and QA effort is expensive
  - Shared development and QA effort is hard
  - Forking will fracture the community
- The strategy
  - No "surprises"
    - Ensure short and long-term requirements are clear
    - Develop architecture ***far*** in advance of implementation
    - Prepare for landing before you start development
  - Tree of GIT repositories – mainline rooted in Oracle
    - Lustre/Oracle == linux/kernel.org
  - Trusted technical experts control landings
    - Gatekeeping implementation ***and*** architecture
  - Landings require collateral
    - Design documents, inspections, test plans, tests, test history

Landing Collateral

- No surprises
  - Start to build landing collateral in advance of development
- Documented designs
    - Demonstrate compatibility with architecture roadmap
    - Demonstrate depth of thinking / completeness
    - Enable inspection
  - Reduce risk of concept defects
  - Include test plan from the start
- Incremental changesets
  - Smaller "single idea" changesets facilitate inspection
  - Intermediate test points find defects sooner
- Test history
  - Demonstrate test coverage
    - Reduce risk of instability on landing
    - Increase confidence of future regression testing effectiveness

Distribution model

One mainline – multiple vendor distributions

- Vendor distributions
  - Commercial support providers need control
  - Quality directly affects profitability
    - Vendor has vested interested in Lustre quality
  - Product test cycles are long
- Mainline(master)
  - Primary means of sharing development
  - Stops being useful if quality is not maintained
  - Requires strict gatekeeping / continuous QA
    - Needs community resourcing
    - Needs trusted gatekeepers

What is Lustre Testing

- Developer testing
  - Locally run, manual regression
- Continuous Integration
  - Automated builds when new changes arrive
  - Automated regression testing
- Scale and load testing
  - Simulated work loads on large clusters
  - Lots of hardware required
- Performance regression testing
  - Repeated test on static hardware
  - Similar to automated regression testing
- Acceptance Testing
  - New hardware or software version

Why is testing Lustre hard

- Complete test coverage impossible
  - Huge configuration matrix
    - Version interoperation
    - Clients and servers spanning administrative domains
  - Huge multi-dimensional test space
    - Multiple servers all connected to multiple clients
    - Recovery much more complex than restart
  - Long time constants
    - Timeouts measured in minutes at scale
- Large expensive test systems required
  - Hyperion @ LLNL is a *minimum* scale test system
- Large expensive test engineers required
  - Guide the test strategy
  - Maximize use of resources

Lustre Testing Toolkit

- Standardize Lustre Testing
  - Manage configuration
  - Validate test environment
  - Drive a set of tests
  - Collect results in standard format
  - Create landing collateral
- We're building it
  - Installable package
  - Support VMware on a laptop to Hyperion and beyond
  - Suggested stages of development
    - Developer
    - Automated regression
    - Performance regression
    - Scale / load

Thank You

Eric Barton
eeb@whamcloud.com