Lustre® Contribution Model: Community Development

LUG 2013
Nathan Rutman (nathan_rutman@xyratex.com)
Topics

- Why we need a single Lustre tree
- Why we need custom Lustre distributions
- Code development and integration process
- Testing
- Code reunification
Why we need a single Lustre tree

- Different environments = different bugs
- Different priorities = different bug fixes / features
- Testing depth
- We all share the rewards
Why we need a custom Lustre distribution

• Different environments = different bugs
• Different priorities = different bug fixes / features
• Some features are hardware-dependent (or not "core Lustre")
  o Hardware CRCs
  o MDRAID zero-copy
  o Samba re-export
  o Tuning for scale
• External gatekeeper
  o Short I/O
  o Pin ldiskfs block bitmaps
• Stability
• Rapid turnaround
Priorities

- Customer bugs
- Sales support
- Future product plans
- Internal feature backlog
- Performance

- Work divided by teams to avoid "crisis mode"
  - Sustaining
  - New features teams
Code Development Process

• What
  o Jira tickets
  o Bug triage
  o Roadmap planning
  o Feature prioritization
• How
  o TSP for new feature development
  o Scrum for sustaining work
  o Code review
  o Automated build and test
• Landing
  o After approvals and testing, landing request
  o Gatekeepers land in maintenance or release branches
Code Review Process

- Patches are submitted to Gerritt for code review
- Gerritt initiates a Jenkins job to build Lustre for the entire coverage matrix
- Jenkins initiates standard integration testing job using Xperior
- Jenkins posts results back to Gerritt
  - Sets “Verified” flag in Gerrit ticket
- Gerritt adds appropriate git comments

- Custom branches can also be submitted for automated build and testing

MRP-691 ldiskfs: hold bitmaps to minimize reads during writes

The patch makes bitmap pages stay in correct LRUs to avoid unnecessary page eviction and possible bitmap read during bulk writes.
Change-Id: I70c2f44707a97822a2f9b13e74fae19753d50792
Signed-off-by: Alexey Lyashkov <alexey_lyashkov@xyratex.com>
Reviewed-on: http://morpheus.xus.xyratex.com:8443/gerrit/135
Tested-by: Jenkins
Tested-by: Alexander Lezhoev <Alexander_Lezhoev@xyratex.com>
Reviewed-by: Alexander Boyko <Alexander_Boyko@xyratex.com>
Reviewed-by: Alexander Zarochnyev <alexander_zarochnyev@xyratex.com>
Reviewed-by: Nathan Rutman <Nathan_Rutman@xyratex.com>
Reviewed-by: Vitaly Fertman <Vitaly_Fertman@xyratex.com>
Testing with Xperior

- Initiated via Jenkins
- System config in YAML (can be per-test)
- Set up cluster
  - VM or real
- Install any required test packages
- Test(s) description in YAML
- Results in YAML

- Open Source
Xperior plugins

- Plugins
  - code coverage tool
  - static verification tool
  - reformat after every test
  - store console output
  - store lustre-diagnostics output

- Under development
  - Random order test execution with replay
  - MDSIM test execution
  - IPMI support
Code Reunification

- Cherry-picking of bug fixes and features
  - Risk/reward of selected new features only
- Regularly integrated into our maintenance branch
- Every bug fix and feature we develop is pushed upstream
  - File a Jira ticket with Intel for each issue
  - Include Xyratex-bug-id to patches and tickets for easy reference
  - Attach patches, respond to inspections, update code
  - Sometimes there are conflicts
    - Priorities
    - Code conflicts
- We publish our tree at Github
Thank You

nathan_rutman@xyratex.com