IML Overview and Roadmap

Joe Grund
IML Team Lead
jgrund@whamcloud.com
Agenda

• IML Background / Overview
• IML 5
• Current Work
• Where to find project / communicate with team
Background

- Integrated Manager for Lustre (IML) is an open source suite of tools for deploying, managing, and monitoring Lustre filesystems
- IML simplifies Lustre administration with intuitive interfaces and near real-time feedback
- Works with new and existing Lustre installations
- Monitors performance and system health
- Proven in production at hundreds of sites
- Used successfully in environments with over 200 OSTs
Background - Deployment

- Deploy Lustre filesystems from one centralized location (ZFS, ldiskfs)
- Near-realtime feedback
- Bring filesystem online from first principles or deploy monitoring for an existing filesystem
- Deploy specialized software, HSM
- Add more storage nodes, targets over time
Deployment scenario

• Admin needs to setup 50 servers with patchless ldiskfs 2.12.1
  • Wants HA capabilities

• Can use IML to deploy all 50 nodes in parallel
  • Uses customizable deployment profiles
  • Performs pre-flight checks
  • Installs packages
  • Configures initial HA
  • Configures NTP
  • Starts LNet

• Provides realtime feedback of entire process for all nodes
Background - Management

• Configure / change state of Lustre and related components
  • Uses state-machine to reach end state from different starting points
    • Starting LNet, state machine ensures packages are installed + kernel modules loaded before bringing LNet up

• Handle recovery situations fencing, failover
  • Automatic configuration of High Availability through Corosync, Pacemaker, and PDU / IPMI integration
Management Scenario

• Admin wants to fail all targets from a server
• Can use IML to fail targets over to secondary HA node
• Can use IML to fail targets back to host when ready
Background - Monitoring

- Holistic system metrics
  - Rich visualizations
  - Drill into filesystem, target, server
  - Find and monitor top jobs

- Aggregate logs across cluster

- HSM Copytool activity monitoring

- Alerts to cluster issues
  - GUI / Email / API

- Searchable command / event / alert log / history
Monitoring Scenarios

• Admin wants to see which OSTs are experiencing high write bandwidth
  • Uses IML’s read/write heatmap to determine OSTs
  • Clicks on OST cell in heatmap, can see which jobs are causing high write bandwidth

• Admin wants to see aggregated cluster logs to diagnose an issue
  • Uses IML to view all logs across the cluster one page
  • Searches for the particular issue and timeframe, finds the issues and can correlate with other activity through the cluster

• Admin wants to be alerted to potential issues
  • Sets up email alerts with IML, gets an email for specific events i.e. a target going offline
IML 5 - Docker

• IML 5 adds support for running within Docker stack

• Run the manager on any docker supported platform

• Continuously delivered to docker hub: https://cloud.docker.com/u/imlteam/repository/list

• Can collocate the IML manager with otherwise conflicting services
  • On lustre client / storage server
  • Alongside other admin tools
IML 5 - Libzfs / ZED integration

• IML 5 uses libzfs and ZED for ZFS monitoring + management features

• Fine grained collection of pools / datasets / props / VDEV tree

• Enables near-realtime state changes

• Works together with Udev detection to provide a holistic view of cluster devices

• Used within IML, can also be used standalone

• Results in device detection / state changes being much faster in IML 5, lower resource usage and better scaling for larger clusters
IML 5 – HA Improvements

- IML has long had its own custom Resource Agent for managing Lustre dating back to its very first versions
- At a later point, a separate Resource Agent (RA) was developed and submitted to the Lustre repo
- IML 5 has switched to using this RA plus the upstream ClusterLabs ZFS RA
- Managed mode installs will use these RAs
- Stock HA setup, closer to general usecases
IML 5 – RPM Delivery

- IML is now completely delivered via Fedora Copr, there is no tarball installer
  - [https://copr.fedorainfracloud.org/coprs/managerforlustre/manager-for-lustre-5.0/](https://copr.fedorainfracloud.org/coprs/managerforlustre/manager-for-lustre-5.0/)
- Download a .repo file and run yum install python2-iml-manager + setup command
- Components are shipped individually as separate RPMs in the repo
  - Bugfixes / non-breaking enhancements can be shipped for individual components
  - Bugfixes / non-breaking enhancements can be applied per-component, no need for full upgrade
- Switch to new repo
  - Previously: [https://copr.fedorainfracloud.org/coprs/managerforlustre/manager-for-lustre/](https://copr.fedorainfracloud.org/coprs/managerforlustre/manager-for-lustre/)
  - Will continue to function, but will no longer receive updates
- Can update from 4.x to 5.x
  - [https://whamcloud.github.io/Online-Help/docs/Upgrade_Guide/upgrade_iml-4.0-el7_to_iml-5.0-el7.md](https://whamcloud.github.io/Online-Help/docs/Upgrade_Guide/upgrade_iml-4.0-el7_to_iml-5.0-el7.md)
- More frequent releases, move towards every two months for RPMs. New features across all components get bundled in
IML 5 - Continuous Integration / Delivery

• Individual modules tested in cloud providers (Travis CI / Azure Pipelines)

• Every landing triggers a build for docker cloud and development copr repo: https://copr.fedorainfracloud.org/coprs/managerforlustre/manager-for-lustre-devel/
  • Possible to evaluate new changes before they have been promoted

• Larger integrations tested in our public Jenkins instance
  • Managed mode
  • Monitored mode
  • Upgrade testing

• All contributions run through testing / code review
IML 5 – Lustre 2.12.1 Support

- IML 5 adds support for Lustre 2.12.1 [http://lustre.org/lustre-2-12-1-released/]
- Also supports Lustre 2.10.7 [http://lustre.org/lustre-2-10-7-released/]
- Support for patchless ldiskfs / ZFS in managed mode
IML - Upgradeability

• Support upgrades from:
  • closed-source IEEL versions / older Whamcloud versions to IML 5
  • Documents describe how to upgrade from
    • 2.4.x https://whamcloud.github.io/Online-Help/docs/Upgrade_Guide/Upgrade_EE-2.4-el6_to_LU-LTS-el7.html
    • 4.0.x https://whamcloud.github.io/Online-Help/docs/Upgrade_Guide/upgrade_iml-4.0-el7_to_iml-5.0-el7.htm
Current work – IML Rust Port

• As we continue scaling to ever-larger clusters, need a general solution for next generation of enhancements

• Requirements
  • Very fast (close to C speeds)
  • Low on resources / lazy
  • Easy to deploy (minimal dependencies)

• Wants
  • Able to scale with solving difficult problems
  • Can effectively schedule tasks to many different nodes and coordinate responses
Current work – IML Rust Port

• Port IML Components to Rust + Tokio
  • Rust
    • Fast
    • Low resource usage
    • No garbage collector, RAII, memory safe, sized types stack based by default
    • Rich type system allows you to write code that is free of subtle bugs and is easy to refactor without introducing new bugs
    • Extremely thorough, eliminates need to write interface checking unit tests
    • Can write parallel code that is verified by compiler to be free of data races
  • Tokio is an event-driven, non-blocking I/O platform for writing asynchronous applications
    • Internally uses a multithreaded, work-stealing based task scheduler.
      • Work happens in parallel, all cores utilized
      • Lazy computations, do nothing until spawned
    • Fast (Zero-cost abstractions)
Current work – IML Rust Port - WebAssembly

• **WebAssembly** (*Wasm*) is a binary instruction format for a stack-based virtual machine.

• Wasm is designed as a portable target for compilation of high-level languages like C/C++/Rust, enabling deployment on the web for client and server applications.

• Write the same code, it compiles to native code on the server, and Wasm in the browser.
  • Code reuse everywhere

• Faster than JS in the browser

• First component shipped as part of 5.0
Work in Pipeline – WebAssembly Components

Module A → Native Compilation → Agent

Module A → Wasm Compilation → Manager

Module A → Browser
• CPU bound code paths will benefit from speed improvements in Rust
• IO bound code paths will benefit from Tokio multithreaded work-stealing task scheduler.
  • Especially useful at scale, many tasks can be handled in parallel while keeping resource usage low
• Goal – implement core in Rust, implement new features in Rust, port existing code into this core
  • Some Rust code already part of 5.0
  • Will be a gradual transition over the course of 5.0 lifetime
Requests / Second of “Fast” HTTP servers

- minihttp (Rust): 2,000,000.00
- rapidoid (Java): 1,500,000.00
- fasthttp (Go): 1,000,000.00
- Go: 0.00
- node.js: 0.00

Pipelined
Potential Future Work

- **Full ZFS Management**
  - IML should provide full ZFS management
    - Show all pools and datasets across a cluster
    - Provide drill-down navigation to elicit more detail on a selected target
    - Show the status of pools and datasets
      - Where imported, mounted, error conditions, configuration
  - Management
    - Create zpools / datasets
    - Support creation of various pool configurations: RAID-Z, Mirrored...
    - JBOD enclosure GUI

- **I18n Support**
  - IML text currently English, but IML is used all over the world
  - Modify/contribute *.po files consumed by services
Potential Future Work

• Enhanced Deployment
  • IML should make it even easier to setup Lustre
  • Deploy to large scale clusters with minimal operator intervention
    • Describe ideal cluster state
    • Expose variants as scalable UI widgets
    • Deploy installation in parallel with a single click
Where to find IML

• 5.0 Release (RPMS): https://copr.fedorainfracloud.org/coprs/managerforlustre/manager-for-lustre-5.0/
• 5.0 Release Docker: https://cloud.docker.com/swarm/imlteam/repository/list
• Help docs: https://whamcloud.github.io/Online-Help/
• Issues: https://github.com/whamcloud/integrated-manager-for-lustre/issues
• Direct line of communication via: https://gitter.im/whamcloud/integrated-manager-for-lustre
• Email: iml@whamcloud.com
Where to find IML - Demo Sandbox

• Easily use Vagrant + Virtualbox to spin up a VM cluster for demo / evaluation
  • https://github.com/whamcloud/Vagrantfiles/blob/master/iml-sandbox/Vagrantfile
  • vagrant up;
    • Creates a sandbox environment for running IML
    • 2 MDS, 2 OSS, 2 client nodes, iSCSI server node, admin node
    • Pre-configured networking for LNet, crossover cabling
    • VBox fence agents installed
    • Shared storage
    • Supports snapshotting
  • vagrant provision --provision-with install-iml-5;
    • Installs IML 5.0 on admin node and set’s it up
    • Docs on how to setup a fs with IML: https://whamcloud.github.io/Online-Help/docs/Contributor_Docs/cd_Installing_IML_On_Vagrant.html
• In addition to manual fs setup, sandbox has automated provisioners for creating ldiskfs / ZFS filesystems
  • Useful for evaluating monitored mode
Help Wanted

• Check Github issues for help wanted opportunities
  • https://github.com/issues?utf8=%E2%9C%93&q=is%3Aopen+is%3Aissue+archived%3Afalse+user%3Awhamcloud+label%3A%22help+wanted%22+
  • Easy to implement, team guidance

• Open an issue / submit a PR

• Use a release train model, pull in work once it’s done

• Want your feedback on useful enhancements

• Projects are public
  • https://github.com/orgs/whamcloud/projects
Closing

- IML is a project with a long history and continues advancing
  - Deployed in production at hundreds of sites since its launch in 2012
  - Open source since 2017
  - Latest release IML 5.0 is now GA

- Possible to upgrade from IEEL to IML 5
  - Upgrade docs for 2.4.x, 3.1.x, 4.0.x lines