Objectives of the Presentation

• Introduce the new Lustre documentation wiki page to Lustre community
• Provide very brief overview of topics documented in the wiki page
• Invite contributions from the community to extend the documentation
• Provide insights on how to contribute to the documentation
Why Lustre Documentation?

• Several resources exist to help deploy and configure Lustre.
• But none exist to explain the internal workings of Lustre source code for developers.
• An ORNL tech-report published in 2009 provided summary on various Lustre subsystem operations.
• This tech report is out of date and is based on Lustre 1.6.
• Several Lustre subsystems underwent significant code changes.
• New features have been added to bring the current version up to 2.15.
Limitations with Existing Documentations

• Two main resources for Lustre developers
  1. wiki.lustre.org
     – Main community repository for Lustre
     – Major Lustre topics hosted are:
       • Testing, benchmarking, monitoring, development activities and how to guides
     – Various resources that explain Lustre architecture
     – Lustre 101: provides information on Lustre usage on administration
  2. Lustre Operations manual
     – Information and procedures to configure Lustre
     – Topics include failover, striping, troubleshooting, configuration and maintenance

• None cover Lustre source code documentation
The New Documentation Effort

• Need a comprehensive documentation for Lustre internal mechanics for developers

• To prepare the documentation, we planned to explore and study source code for different subsystems

• Scheduled Lustre code deep dive sessions within the team of Lustre developers

• Documented key data structures and APIs pertaining to each subsystem

• Importance was given to interfaces through which subsystems communicate and function call graphs

• Made sure to represent the code flow and data structures wherever possible
Published second edition of the ORNL tech-report (https://info.ornl.gov/sites/publications/Files/Pub166872.pdf)

Aims to document the internal workings of the latest version of Lustre

More complete and up-to-date information than the previous tech-report

Key data structures and APIs used for interaction among the various Lustre subsystems

Details in this document should remain relevant for the foreseeable future
Wiki page – “Understanding Lustre Internals”

• Hosted in main community repository for Lustre ‘wiki.lustre.org’
  – https://wiki.lustre.org/Understanding_Lustre_Internals

• Major sections and subtopics include,
  – Lustre architecture featuring its core components
  – Software stack
  – File layouts
  – Lustre test suite
  – User utilities
  – Core subsystems including,
    • MGC
    • Obdclass
    • Libcfs
    • Fld
    • Fid
Overview of the Topics Documented

1. Lustre Architecture
   - Lustre features and scalability and performance numbers
   - Lustre components including management, metadata and object storage servers and targets
   - Lustre file layout describing two categories: Normal and composite layouts

![Diagram of Lustre file system components in a basic cluster](image)

**Figure 3: Lustre file system components in a basic cluster**

![Diagram of Normal RAID0 file striping in Lustre](image)

**Figure 4: Normal RAID0 file striping in Lustre**
Overview of the Topics Documented

1. Lustre Architecture (cont.)
   - Synopsis on distributed namespace, file identifiers and layout attributes
   - Basic view of Lustre software stack

![Figure 5: Basic view of the Lustre software stack](image)

* Figure 5: Basic view of the Lustre software stack[^1]

[^1]: [1]

* Figure 6: Lustre I/O operation: Lustre client requesting file data

![Figure 6: Lustre I/O operation](image)

* Figure 6: Lustre I/O operation: Lustre client requesting file data[^1]

[^1]: [1]

---

[^1]: Open slide master to edit
Overview of the Topics Documented

2. Lustre Tests
   - Lustre test suite components
   - Various Lustre unit, feature and regression tests
   - Test framework options
   - Acceptance testing on Lustre
   - Lustre tests environment variables

3. Utils
   - User utilities including lfs, lfs_migrate, lctl, llog_reader
   - mkfs.lustre, mount.lustre and tunefs.lustre
Overview of the Topics Documented

4. MGC

- Starting from MGC module initialization with registering of MGC obd device
- MGC obd operations with an example of communication between llite and MGC through Obdclass
- Detailed walkthrough of mgc_setup(), mgc_precleanup(), mgc_cleanup() and mgc_import_event() methods
- Lustre log handling and log processing in MGC

![Figure 7: Communication between llite and mgc through obdclass][1]
Overview of the Topics Documented

4. **MGC (cont.)**

- `mgc_setup()` with function call graphs
- Comparison of code flow for MGC setup and cleanup processes

Figure 8: `mgc_setup()` call graph starting from Lustre file system mounting

<table>
<thead>
<tr>
<th>MGC_setup()</th>
<th>Loginode_compaction()</th>
<th>mgc_setup()</th>
<th>mgc_setup()</th>
</tr>
</thead>
<tbody>
<tr>
<td>ptrfs_addrref()</td>
<td>client_obd_setup()</td>
<td>mgc_llog_init()</td>
<td>kthread_run (mgc_requeue_thread)</td>
</tr>
</tbody>
</table>

Figure 9: `mgc_setup()` vs. `mgc_cleanup()`
Overview of the Topics Documented

5. Obdclass
   - `obd_device` structure defining an obd device
   - Detailed walkthrough of MGC lifecycle
   - Obd device lifecycle through various functions such as `class_attach()`, `class_setup()`, `class_precleanup()`, `class_cleanup()` and `class_detach()`
   - Various other data structures including `obd_type`, `lu_device_type`, `obd_export`, `obd_import` and `client_obd`
   - Concept of imports and exports through which obd device communication is established

Figure 10: Import and export pair in Lustre

Figure 11: Communication between ost and mdt server obd devices in Lustre
5. **Obdclass**
   - **class_attach()**
     - First method in the obd device life cycle
     - Major functionalities are,
       - Registers and adds the obd device to the list of obd devices
       - Creates, allocates and initializes a new obd device
       - Populates `obd_export` structure
       - Adds the device to `obd_devs` array

![Figure 13: Workflow of class_attach() function in obd device lifecycle](image-url)
Overview of the Topics Documented

5. **Obdclass**
   - **class_setup()**
     - Create hashtables and self-export
     - Sets the `obd_starting` flag from `obd_device` structure
     - Can view the function call workflow and data structures
     - Data structures involved are `client_obd` and `obd_import`

![Workflow of class_setup() function in obd device lifecycle](image)
Overview of the Topics Documented

6. Libcfs

- Data encryption support in Libcfs
  - Two types of encryption capabilities: data on the wire and data at rest

- CPU partition table management
  - Data structures and definitions for CPU and CPT defined in libcfs_cpu.c

- Debugging support and failure injection
  - Macros that report errors and warnings are defined in libcfs_debug.h

- Additional supporting software in Libcfs
  - Implementation of portable time API, resizable arrays, spin locks, atomic_t for reference counts
Overview of the Topics Documented

7. File Identifiers, FID Location Database, and Object Index

- Detailed description of Lustre file identifiers
  - FID allocation process and description FID structure (lu_fid) fields
- Concept of reserved sequence numbers and object IDs
  - Details on reserved sequence ranges used by Lustre such as IGIF, IDIF, OST_MDT0, LLOG, ECHO, and OST_MDT1
- Functionality of fid kernel module
  - Fid module initialization, sequence number allocation, cleanup routine
- Contents related to FID Location Database (FLD) and Object Index (OI)
  - Source code files pertaining to FLD and functions for interacting with OI
## Future Extensions to the Wiki Page

<table>
<thead>
<tr>
<th>Already documented</th>
<th>To be documented</th>
</tr>
</thead>
</table>
| • Lustre Architecture  
  • Subsystems –       |
|      • TESTS         |
|      • UTILS         |
|      • MGC           |
|      • OBCLASS       |
|      • LIBCFS        |
|      • FLD           |
|      • FID           |
| • LNET               |
| • Subsystems –       |
|      • LDLM          |
|      • LLITE         |
|      • LMV           |
|      • OSP           |
|      • PTLRPC        |
|      • MDC           |
|      • And a few more … |

• Plan to document all subsystems in Lustre
Inviting Community Contributions

• We are interested in making this documentation a community effort

• We invite more contributions from the Lustre community,
  – To extend the documentation by adding contents related more Lustre subsystems
  – To keep the documentation UpToDate with Lustre source code changes

• Lustre users/admins/developers are welcome to contribute towards this documentation
How to Contribute?

- Create an account in wiki.lustre.org
- wiki.lustre uses Mediawiki in the backend
- Contents can be easily added and are not syntax heavy like latex
- Various extensions are already enabled for adding images, cross referencing etc.
- Source code snippets, figures and tables can be directly added using simplified syntax
- Useful resources for documentation discussion,
  - Jira ticket - LUDOC-498
  - Slack channel - #lustre-internals-document
Acknowledgements

• Thanks to Lustre team at ORNL – James Simmons, Rick Mohr and Sarp Oral

• This research used resources of the Oak Ridge Leadership Computing Facility at the Oak Ridge National Laboratory, which is supported by the Office of Science of the U.S. Department of Energy under Contract No. DE-AC05-00OR22725.

• Great resources that helped me when I started to learn and document Lustre,
  – wiki.lustre.org
  – Lustre operations manual
  – wiki.whamcloud.com

• References
Thank you!
Questions?