



Architect of an Open World™

Lustre Static Code Analysis with Coverity

April 17th, 2013

Sebastien Buisson

Parallel File Systems
Extreme Computing R&D

Static Code Analysis with Coverity

- Why static code analysis is useful?
- Tool for analysis: Coverity
- Coverity applied to Lustre
- Defects found by Coverity
- Benefits for the whole Lustre Community

Why static code analysis is useful?



Architect of an Open World™

Why static code analysis is useful?

From <https://wiki.hpdd.intel.com/display/PUB/Project+Ideas>

“Run Lustre code through static analysis tools to identify potential latent bugs in the Lustre code. These are often hard to find through testing, and easily fixed once found.”

From W. S. Humphrey, "Using a Defined and Measured Personal Software Process," IEEE Software, May, 1996

“Even experienced programmers typically make a mistake for every seven to ten lines of code they develop.”

Tool for analysis: Coverity



Architect of an Open World™

Tool for analysis: Coverity

How it works



Various checkers involved:

- STRING_SIZE
- RESOURCE_LEAK
- UNINIT
- and many others...

Tool for analysis: Coverity

□ What do we get

The screenshot displays the Coverity web interface. The top navigation bar includes 'Coverity@ Connect', 'Lustre-Dev', 'Configuration', 'Help', and a user profile 'seballen buisson'. A search bar on the right contains 'Enter CID(s)'. The left sidebar has navigation menus for 'FILES', 'FUNCTIONS', 'COMPONENTS', 'CHECKERS', and 'OWNERS'. The main content area shows a table of issues:

CID	Type	Impact	Status	...	First Det...
11437	Thread deadlock	Medium	Triaged	1	01/1
11734	Copy into fixed size buffer	Low	Triaged	6	07/2
11735	Copy into fixed size buffer	Low	Triaged	2	07/2
11736	Copy into fixed size buffer	Low	Triaged	2	07/2
11737	Copy into fixed size buffer	Low	Triaged	2	07/2

Below the table, a code snippet is shown with annotations for issue CID 11437:

```
637 cfs_for_each_possible_cpu(cpu) {  
    3. Condition "cfs_trace_data[i]", taking true branch  
    4. Condition "tcd = &(*cfs_trace_data[i])[cpu].tcd", taking true branch  
    5. lock_acquire: Calling function "cfs_trace_lock_tcd(struct  
       cfs_trace_cpu_data *, int)" acquires lock "cfs_trace_cpu_data.tcd_lock".  
       [show details]  
    6. Condition "cfs_trace_lock_tcd(tcd, 1)", taking true branch  
638 cfs_tcd_for_each_type_lock(tcd, i, cpu)  
    7. lock_order: Calling "put_pages_on_tcd_daemon_list(struct  
       page_collection *, struct cfs_trace_cpu_data *)" acquires lock  
       "page_collection.pc_lock" while holding lock  
       "cfs_trace_cpu_data.tcd_lock" (count: 1 / 3). [show details]  
639 put_pages_on_tcd_daemon_list(pc, tc  
640 }
```

The detailed view on the right for issue 11437 'Thread deadlock' includes:

- Classification: Bug
- Severity: Major
- Action: Undecided
- Ext. Reference: LU-3055
- Owner: Unassigned

Additional text in the detailed view: '--- Last comment on 02/01/13 03:50 PM --- We have two other cases where the locks tcd_lock and pc_lock are taken in reverse order. So a thread deadlock could occur. Open a Jira ticket to ask.'

Buttons for 'Apply + Next', 'Apply', 'Revert', and 'Export' are visible. Below are tabs for 'Occurrences', 'History', and 'Information'. A list of events contributing to the issue is shown:

- 5. lock_acquire tracefile.c:638
- 5.4. lock linux-tracefile.c:159
- 7. lock_order tracefile.c:639

Tool for analysis: Coverity

□ What do we get

The screenshot displays the Coverity web interface. The top navigation bar includes 'Coverity: Connect', 'Lustre-Dev', 'Configuration', 'Help', and a search bar. The left sidebar contains navigation menus for 'FILES', 'FUNCTIONS', 'COMPONENTS', 'CHECKERS', and 'OWNERS'. The main content area shows a table of issues:

CID	Type	Impact	Status	...	First Det...
11437	Thread deadlock	Medium	Triaged	1	01/1
11734	Copy into fixed size buffer	Low	Triaged	6	07/2
11735	Copy into fixed size buffer	Low	Triaged	2	07/2
11736	Copy into fixed size buffer	Low	Triaged	2	07/2
11737	Copy into fixed size buffer	Low	Triaged	2	07/2

Below the table, a code snippet is shown with annotations for the selected issue (CID 11437):

```
637 cfs_for_each_possible_cpu(cpu) {
    3. Condition "cfs_trace_data[i]", taking true branch
    4. Condition "tcd = &(*cfs_trace_data[i])[cpu].tcd", taking true branch
    5. lock_acquire: Calling function "cfs_trace_lock_tcd(struct
       cfs_trace_cpu_data *, int)" acquires lock "cfs_trace_cpu_data.tcd_lock".
       [show details]
    6. Condition "cfs_trace_lock_tcd(tcd, 1)", taking true branch
638 cfs_tcd_for_each_type_lock(tcd, i, cpu)
    7. lock_order: Calling "put_pages_on_tcd_daemon_list(struct
       page_collection *, struct cfs_trace_cpu_data *)" acquires lock
       "page_collection.pc_lock" while holding lock
       "cfs_trace_cpu_data.tcd_lock" (count: 1 / 3). [show details]
639 put_pages_on_tcd_daemon_list(pc, tc
640 }
```

The right-hand panel provides details for the selected issue, '11437 Thread deadlock':

11437 Thread deadlock
In put_pages_on_daemon_list: Threads may try to acquire two locks in different orders, potentially causing deadlock (CWE-833)

Classification: Bug
Severity: Major
Action: Undecided
Ext. Reference: LU-3055
Owner: Unassigned

--- Last comment on 02/01/13 03:50 PM ---
We have two other cases where the locks tcd_lock and pc_lock are taken in reverse order. So a thread deadlock could occur. Open a Jira ticket to ask.

Buttons: Apply + Next, Apply, Revert, Export

Occurrences, History, Information

In lustre-core-buissons
Events contributing to issue:

- 5. lock_acquire tracefile.c:638
- 5.4. lock linux-tracefile.c:159
- 7. lock_order tracefile.c:639

Tool for analysis: Coverity

□ What do we get

The screenshot displays the Coverity web interface. On the left, there are navigation menus for 'FILES', 'FUNCTIONS', 'COMPONENTS', 'CHECKERS', and 'OWNERS'. The main area shows a table of issues:

CID	Type	Impact	Status	...	First Det...
11437	Thread deadlock	Medium	Triaged	1	01/1
11734	Copy into fixed size buffer	Low	Triaged	6	07/2
11735	Copy into fixed size buffer	Low	Triaged	2	07/2
11736	Copy into fixed size buffer	Low	Triaged	2	07/2
11737	Copy into fixed size buffer	Low	Triaged	2	07/2

Below the table, a code snippet is shown with a tooltip for issue CID 11437:

```
637 cfs_for_each_possible_cpu(cpu) {
    3. Condition "cfs_trace_data[i]", taking true branch
    4. Condition "tcd = &(*cfs_trace_data[i])[cpu].tcd", taking true branch
    5. lock_acquire: Calling function "cfs_trace_lock_tcd(struct
       cfs_trace_cpu_data *, int)" acquires lock "cfs_trace_cpu_data.tcd_lock".
       [show details]
    6. Condition "cfs_trace_lock_tcd(tcd, 1)", taking true branch
638 cfs_tcd_for_each_type_lock(tcd, i, cpu)
    CID 11437 (#1 of 1): Thread deadlock (ORDER_REVERSAL)
    7. lock_order: Calling "put_pages_on_tcd_daemon_list(struct
       page_collection *, struct cfs_trace_cpu_data *)" acquires lock
       "page_collection.pc_lock" while holding lock
       "cfs_trace_cpu_data.tcd_lock" (count: 1 / 3). [show details]
639 put_pages_on_tcd_daemon_list(pc, tc
640 }
```

The right-hand panel provides details for issue 11437, titled "11437 Thread deadlock". It includes a description: "In put_pages_on_daemon_list: Threads may try to acquire two locks in different orders, potentially causing deadlock (CWE-833)". It also shows classification (Bug), severity (Major), action (Undecided), and other metadata like "Ext. Reference: LU-3055" and "Owner: Unassigned". A comment section at the bottom notes: "— Last comment on 02/01/13 03:50 PM --- We have two other cases where the locks tcd_lock and pc_lock are taken in reverse order. So a thread deadlock could occur. Open a Jira ticket to ask."

Tool for analysis: Coverity

□ What do we get

The screenshot displays the Coverity web interface for analyzing a thread deadlock issue (CID 11437) in the luster-2.3.63 code. The interface is divided into several sections:

- Left Sidebar:** Contains navigation options for files, functions, components, checkers, and owners.
- Table of Issues:** A table listing issues with columns for CID, Type, Impact, Status, and First Detected. Issue 11437 is highlighted as a Thread deadlock with a Medium impact and Triaged status.
- Code View:** Shows the source code for `put_pages_on_tcd_daemon_list` with annotations explaining the conditions and lock acquisition steps leading to the deadlock. A red diamond icon indicates the specific issue location.
- Issue Details (Right Pane):** Provides a detailed description of the thread deadlock, classification (Bug), severity (Major), action (Undecided), and owner (Unassigned). It also includes a stack trace showing the sequence of events contributing to the issue.

Tool for analysis: Coverity

□ What do we get

The screenshot displays the Coverity web interface for a project named 'Lustre-Dev'. The main table lists several issues, with CID 11437 highlighted as a 'Thread deadlock' of 'Medium' impact, 'Triaged' status, and '1' occurrence. The right-hand panel provides detailed information for this issue, including its classification as a 'Bug' with 'Major' severity, and an 'Undecided' action. It also shows the extension reference 'LU-3055' and the owner 'Unassigned'. A comment from 02/01/13 03:50 PM explains the deadlock scenario involving 'tcd_lock' and 'pc_lock' acquisition order. Below the comment are buttons for 'Apply + Next', 'Apply', 'Revert', and 'Export'. At the bottom right, a list of events contributing to the issue is shown, with '7. lock_order' at 'tracefile.c:639' highlighted.

CID	Type	Impact	Status	...	First Det...
11437	Thread deadlock	Medium	Triaged	1	01/1
11734	Copy into fixed size buffer	Low	Triaged	6	07/2
11735	Copy into fixed size buffer	Low	Triaged	2	07/2
11736	Copy into fixed size buffer	Low	Triaged	2	07/2
11737	Copy into fixed size buffer	Low	Triaged	2	07/2

11437 Thread deadlock

In put_pages_on_daemon_list: Threads may try to acquire two locks in different orders, potentially causing deadlock (CWE-833)

Classification: Bug
Severity: Major
Action: Undecided
Ext. Reference: LU-3055
Owner: Unassigned

--- Last comment on 02/01/13 03:50 PM ---
We have two other cases where the locks tcd_lock and pc_lock are taken in reverse order. So a thread deadlock could occur. Open a Jira ticket to ask.

Apply + Next Apply Revert Export

Occurrences History Information

In lustre-core-buissons

Events contributing to issue:

- 5. lock_acquire tracefile.c:638
- 5.4. lock linux-tracefile.c:159
- 7. lock_order tracefile.c:639

Tool for analysis: Coverity

Defect categories

- API usage errors
- Code maintainability issues
- Concurrent data access violations
- Control flow issues
- Error handling issues
- Incorrect expression
- Integer handling issues
- Memory - corruptions
- Memory - illegal accesses
- Null pointer dereferences
- Program hangs
- Resource leaks
- Security best practices violations
- Uninitialized variables

Coverity applied to Lustre



Architect of an Open World™

Coverity applied to Lustre

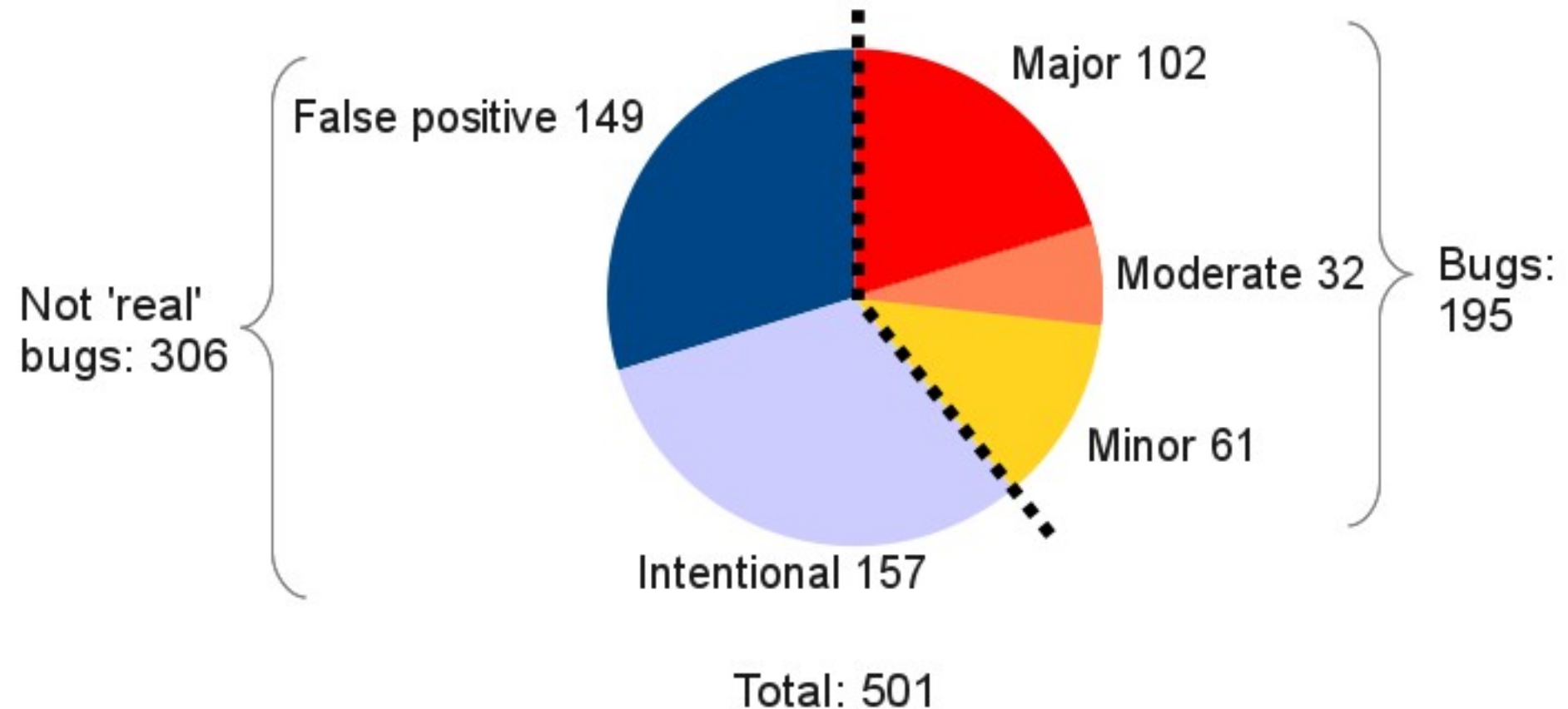
How we proceed

- Main work on Lustre Master branch + 2.1 in parallel
- Current status for master:
 - Latest tag analyzed: v2_3_63 (March 22nd)
 - Next steps: diff with new tags until 2.4 GA

Defects found by Coverity

□ Statistics on master

Defects by status



Defects found by Coverity

□ A few words on false positives and intentionals

■ How can they be avoided?

- 'fall through' in switch cases: please comment
- function pointers, like the ones set in `cfs_hash_create()`
 - Too complex path to follow for Coverity
 - Redesign code?
 - Specific Coverity comments for future analysis

Statistics on master

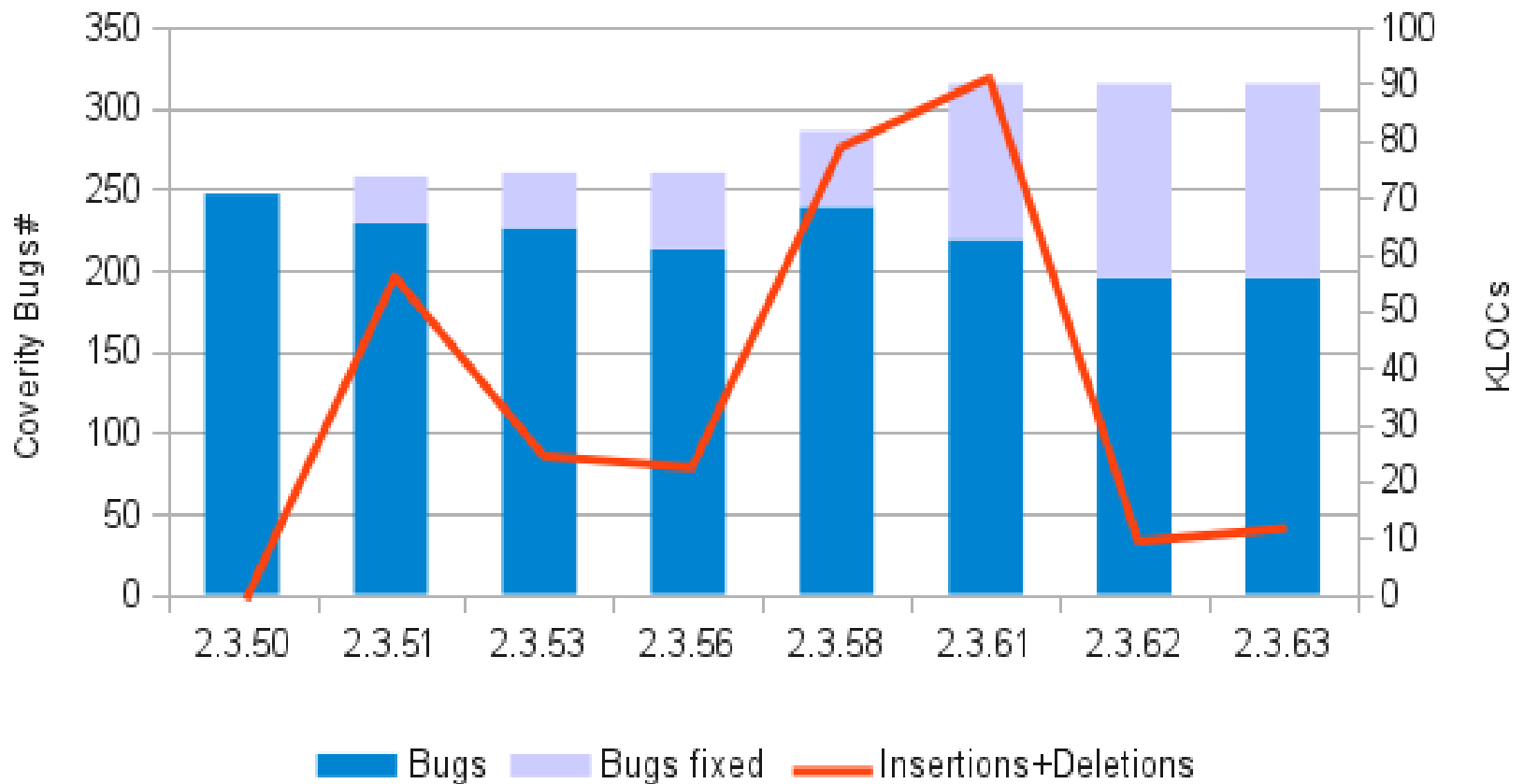
195 Bugs by category



Statistics on master

□ Historical view

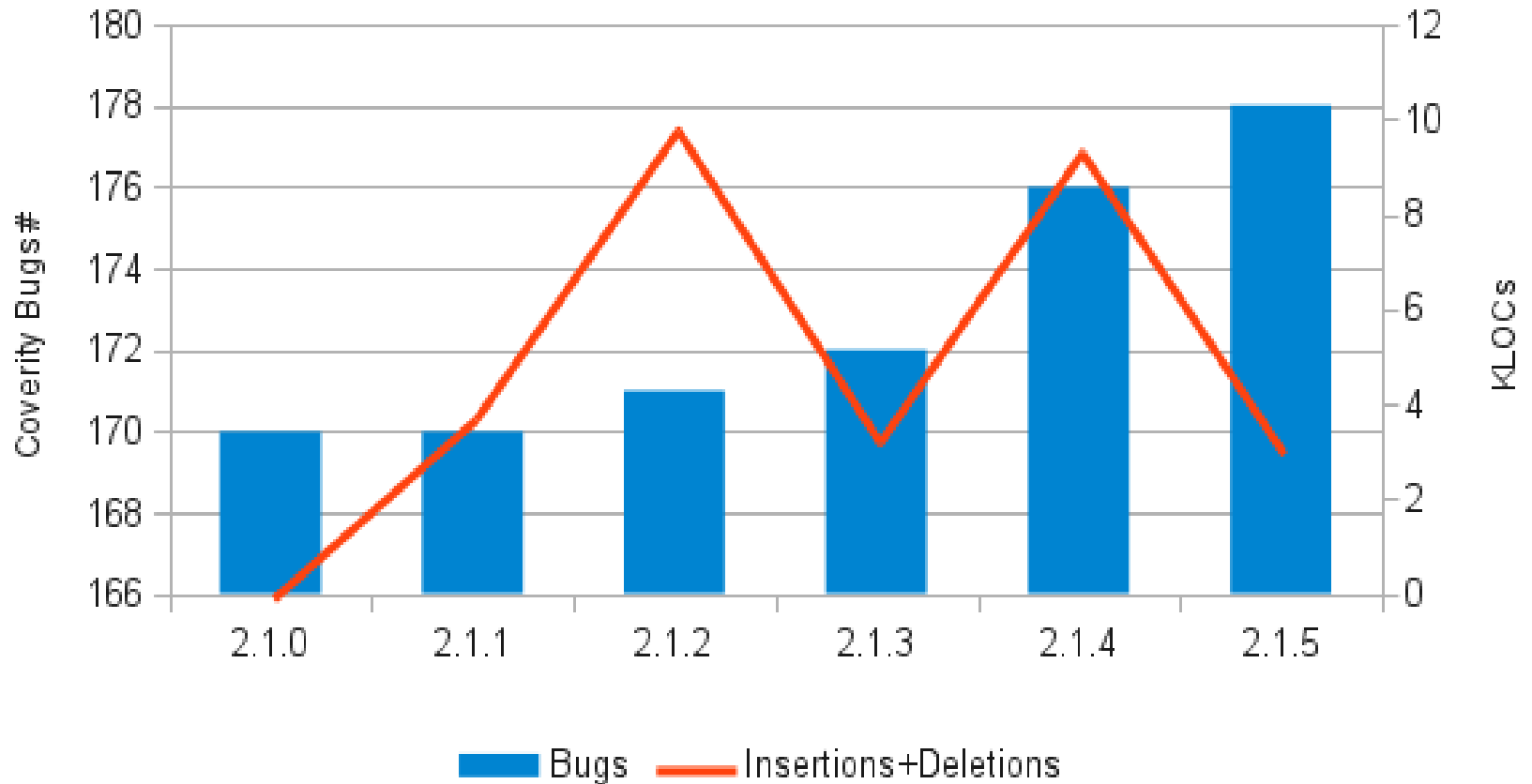
Evolution of Coverity bugs with git commits



Statistics on b2_1

Historical view

Evolution of Coverity bugs with git commits



Statistics on b2_1

Removed from 2.1.0 to 2.1.5:

- "Dereference after null check" in `lustre/obdfilter/filter.c`
 - LU-1042 "1.8 clients show wrong dates with 2.1 servers"
- "Dereference before null check" in `lustre/mdd/mdd_dir.c`
 - LU-1331 "Allow changelog to extend record"
- "Logically dead code" in `lustre/obdclass/lustre_peer.c`
 - LU-570 "Add function to find connect uuid by nid"

Introduced between 2.1.0 and 2.1.5:

- 11 new Coverity bugs

Benefits for the whole Lustre Community



Architect of an Open World™

Access to Coverity defects list

- Export only possible in CSV or XML formats

CID	Type	Class.	File	Function
11437	Thread deadlock	Bug	libcfs/libcfs/tracefile.c	put_pages_on_daemon_list
11734	Copy into fixed size buffer	Bug	libcfs/libcfs/linux/linux-tcpip.c	libcfs_ipif_query
11735	Copy into fixed size buffer	Bug	libcfs/libcfs/util/parser.c	Parser_help
11736	Copy into fixed size buffer	Bug	lnet/selftest/console.c	lstcon_batch_add
11737	Copy into fixed size buffer	Bug	lnet/selftest/console.c	lstcon_group_alloc
11738	Copy into fixed size buffer	Bug	lnet/selftest/console.c	lstcon_session_new
11739	Copy into fixed size buffer	Bug	lnet/utls/debug.c	jt_dbg_debug_kernel
11740	Copy into fixed size buffer	Bug	lustre/mgs/mgs_llog.c	mgs_modify
11741	Copy into fixed size buffer	Bug	lustre/ptlrpc/sec_config.c	sptlrpc_conf_get
11742	Copy into fixed size buffer	Bug	lustre/utls/lfs.c	lfs_find

- No remote access to web GUI
 - License considerations

Benefits for the whole Lustre Community

Jira tickets opened, patches proposed

- 30 tickets opened, covering all Coverity bugs
- Identified with 'coverity' label
- 13 already merged
- Thanks for Intel's responsiveness



Ongoing effort

- We hope as many patches as possible will be landed before 2.4 GA
- Our goal is to continue to watch Lustre code with Coverity



Architect of an Open World™
