Microsoft

Tips and Tricks for Empowering LLMs with Lustre Knowledge

Ellis Wilson

AI/LLMs and Software Engineering

Show of Hands: How many people use AI:

- At least once per month?
- At least once per week?
- Every day?

AI has reshaped my job

- I barely write code now, but my code output is 5-10x
- LLMs are reshaping every stage of my work: design docs, code development, code review, testing, performance analysis, debugging in the field, documentation writing…

The Myth about LLMs being bad with Old Code

- *Old codebases (e.g., Lustre) have as much opportunity to benefit from LLMs as greenfield code*
  - A ton of historical data can be a benefit, not a risk, **if managed correctly**
- This talk: A first stab at how to best leverage LLMs with Lustre
  - Administration, documentation, forward dev, performance analysis, etc.
- Let's trade notes after the talk!

Outline for Today's Talk

- Common Lustre-related tasks for:
  - Administrators
  - Forward developers
- Founts of Lustre Knowledge LLMs can draw from
- Controlling Chaos – guiding a Lustre LLM agent through instructions
- Examples:
  - Field debugging
  - Forward development in Lustre
  - Documentation review
- Get the code

Common Tasks of Lustre Administrators, Developers, and Integrators Ready for LLMs

Lustre Administration: Example Questions

"Why am I getting -ENOSPC when lfs df still shows free OSTs? Should I, and if so how do I, adjust the QOS allocation threshold?"

Lustre Administration: Example Questions

"My copytool is registered but hsm_archive requests aren't completing — how do I check active_request_timeout and agent status?"

Lustre Administration: Example Questions

"An OST went read-only after a transient SCSI error — how do I identify affected files with lfs find and safely deactivate it?"

Lustre Administration: Example Questions

"The ping evictor keeps evicting healthy clients — how do I tune ping_evictor timeout to stop false evictions?"

Lustre Development: Example Questions/Tasks

"I hit a kernel panic with stack X – is it already tracked in JIRA?"

Lustre Development: Example Questions/Tasks

"Feature Y is still not fully stable in master. What remaining fixes languish in review in Gerrit that need to be landed?"

Lustre Development: Example Questions/Tasks

"There's a race in OSC max-RPCs-in-flight tracking — how does the RPC credit accounting work in lustre/osc/ and where's the unprotected window?"

Lustre Development: Example Questions/Tasks

"Gerrit change LU-Z needs to be reviewed carefully for logical correctness, code consistency with surrounding code, and high test coverage"

Lustre Founts of Knowledge LLMs can draw from

Undirected LLMs Tend to Fail the Smell Test

- Modern LLMs have some amount of native knowledge “baked-in”
  - You can ask a Lustre question and you might get an answer
  - It also might just completely fabricate something, or draw from very old information
  - Probably all of us have experienced this in general AI searches

- Better Idea:
  - Point the LLM at existing sources of knowledge for Lustre

Numerous Lustre Sources of Knowledge

	Lustre codebase	Lustre Manual	Lustre Wiki	JIRA	Gerrit
What is it?	***Full kernel source*** tree including all server, client, and networking modules	***Official operations guide*** for configuring, tuning, and troubleshooting	**Community-contributed knowledge base** of guides and architecture notes	**Bug tracker** with ~20,000 issues, comments, and resolution history	**Code review system** hosting every patch, review comment, and test result
Who uses it?	Kernel developers and contributors	System administrators and storage architects	Administrators, integrators, and new contributors	Developers, QA engineers, and support teams	Developers, reviewers, and release engineers
Correctness	**High** — it is the implementation	**High** — maintained alongside releases	**Medium** — community-edited, not always verified	**High** — first-hand bug reports with stack traces	**High** — exact patches tied to specific JIRA fixes
Freshness	**High** — tracks master branch directly	**Medium** — may lag behind bleeding-edge features	**Low** — many pages are outdated or unmaintained	**Medium** — brand new and ancient issues existent	**Medium** — some recent patches, some ancient
Completeness	**High** — every subsystem included	**Medium** — covers operations, not internals	**Medium** — broad but uneven topic coverage	**Low** — only covers reported issues, not normal behavior	**Medium** — covers changes submitted, not design rationale

Challenges

Task Specificity

Certain data sources are better for certain questions

Data is Incorrect

Some data sources are effective to get a quick answer, but need to be validated

Data is Stale

Some data sources document commands or state of the code at some point in the past

Data is Incomplete

Some information is correct and fresh, but incomplete

Multiple sources required

One data source may be necessary, but insufficient, to fully answer the question or complete the task

Controlling Chaos – guiding a Lustre LLM agent through data sources with instructions

Solution Proposal

Download Locally: Downloading the datasets greatly reduces latency and avoids DDoSing the sources of the datasets

Normalize: The data may be plain text or code, but frequently isn't (xml, docbook, etc). Converting to plaintext makes them far more LLM-friendly

Build Indexes: Some data is extremely voluminous (e.g., JIRA 20K issues) and has metadata associated with it. Building an index allows LLMs to get quick and normalized answers cheaply

Create Instruction Files: LLMs frequently have known instruction file paths that they will source on startup. Creating these avoids individual users having to teach the LLM about the data source natures, organizations, etc each time.

Make Public: Upload these processed datasets and instructions so people aren't also DDoSing the data sources trying to get their own datasets

Quick Aside: Why not MCP servers?

- Model Context Protocol (MCP) servers allow for standardized interaction with data sources like these
  - Downloading an entire dataset is a bit of an anti-pattern in AI usage today
- However, the interactions with these data sources are read-only
  - Not create a new bug in JIRA, respond to a comment in gerrit, commit something to Lustre
  - Keeping everything local and normalized to plaintext/code avoids MCP clients from all over Lustre-verse hitting the already slow WC JIRA, for example
- Future: Possible to craft local MCP servers to standardize agent interaction with data sources

Examples: What We've Used it For

Root-causing Lustre Panics in the Field

Saw the following panic in our fleet:

```
[ 168.964919] LustreError: 4494:0:(fld_handler.c:262:fld_server_lookup()) srv-lustrefs-MDT0000: Cannot find sequence
0x50916999dec80000: rc = -2
[ 168.971524] LustreError: 4494:0:(fld_handler.c:262:fld_server_lookup()) Skipped 99179 previous similar messages
[ 183.245416] BUG: kernel NULL pointer dereference, address: 0000000000000000
[ 183.249138] #PF: supervisor read access in kernel mode
[ 183.251641] #PF: error_code(0x0000) - not-present page
[ 183.254042] PGD 204e202067 P4D 204e202067 PUD 2032ac8067 PMD 0
[ 183.256755] Oops: 0000 [#1] SMP NOPTI
[ 183.258496] CPU: 10 PID: 2584 Comm: ptlrpcd_02_00 Kdump: loaded Tainted: G           OE     5.4.0-1155-azure-fips
#162+fips1-Ubuntu
[ 183.264015] Hardware name: Microsoft Corporation Virtual Machine/Virtual Machine, BIOS 090008  12/07/2018
[ 183.268298] RIP: 0010:osp_update_request_destroy+0x467/0x510 [osp]
[ 183.271157] Code: 60 47 c0 48 8b 45 d0 65 48 33 04 25 28 00 00 00 0f 85 b8 00 00 00 48 83 c4 58 5b 41 5c 41 5d
41 5e 41 5f 5d c3 49 8b 44 24 30 <48> 8b 08 48 8d 98 b8 fe ff ff 49 89 c6 48 8d 45 98 48 89 45 80 4c
[ 183.279695] RSP: 0018:ffffa0e7048ffc60 EFLAGS: 00010282
[ 183.282077] RAX: 0000000000000000 RBX: ffff93529554f6f8 RCX: ffff93529554f730
[ 183.285329] RDX: ffff93529554f708 RSI: 0000000000000001 RDI: 0000000000000000
[ 183.288557] RBP: ffffa0e7048ffce0 R08: 0000000000000000 R09: ffffffffc1215d9c
[ 183.291791] R10: ffff9352941b08e0 R11: 0000000000000001 R12: ffff93529554f700
[ 183.295316] R13: ffff93529554f6f8 R14: dead000000000122 R15: dead000000000100
[ 183.298590] FS:  0000000000000000(0000) GS:ffff9354bfa80000(0000) knlGS:0000000000000000
[ 183.302263] CS:  0010 DS: 0000 ES: 0000 CR0: 0000000080050033
[ 183.304952] CR2: 0000000000000000 CR3: 0000002031474001 CR4: 0000000000370ee0
[ 183.308203] DR0: 0000000000000000 DR1: 0000000000000000 DR2: 0000000000000000
[ 183.311489] DR3: 0000000000000000 DR6: 00000000fffe0ff0 DR7: 0000000000000400
[ 183.314731] Call Trace:
[ 183.315892]  ? update_buffer_get_update+0x23b/0x5b0 [osp]
[ 183.318346]  ? kfree+0x231/0x250
[ 183.319842]  osp_thandle_destroy+0x80b/0x8e0 [osp]
[ 183.323157]  ptlrpc_check_set+0x445/0x21a0 [ptlrpc]
[ 183.326541]  ptlrpcd_wake+0xa71/0xd10 [ptlrpc]
[ 183.329700]  ? do_wait_intr_irq+0x90/0x90
[ 183.332648]  kthread+0x104/0x140
[ 183.335242]  ? ptlrpcd_wake+0x250/0xd10 [ptlrpc]
[ 183.338434]  ? kthread_park+0x90/0x90
[ 183.341157]  ret_from_fork+0x35/0x40
```

Prompt: "*We are seeing the following stack. Please review this corpus of data and advise*".

- Searched source code for crashing functions
- Queried JIRA index for matching bugs
- Read dozens of related JIRA issues
- Analyzed OSP transaction code for race conditions
- Correlated FLD errors with corrupted FID sequences
- Identified LIST_POISON values confirming UAF
- Root cause: race in osp_invalidate_request matching LU-13494
- Found second bug: Prevents a subsequent NULL deref in same area during teardown (LU-15988)
- *Recommended backporting LU-13494 and LU-15988 fixes since user is on 2.15.7; fixes landed in 2.16*

Took 2-3 minutes to root cause (correctly!)

Implementing Lustre Client-side Compression

- **Prompts:**
- 1: “Design a feature for Lustre where a client can compress data before sending it to the OSSs/OSTs“
- 2. “Use the Enhanced Adaptive Compression wiki page and LUG2023 PDF on CSDC as a basic for the design”
- 3. “Focus on lz4 and lzo compression, prioritize ldiskfs as the on-disk format, familiarize yourself with the codebase at 2.15.7, and produce a design plan.
- 4. “The plan looks good. Go ahead and implement the plan.“

Took a weekend of prompting to get to a working prototype -- another week to get to something pushed upstream. Rebase on master in-progress.

Agent (Claude Opus 4.6):

- Checked out 2.15.7-laaso
- Read three external design specs on adaptive compression
- Studied Lustre I/O path codebase and produced design plan
- Implemented lz4/lzo compression in kernel C across llite, lov, osp
- Wrote sanity-compr test suite with dozens of test cases
- Built Lustre on dev host, set up single-node test environment
- Deployed across six-VM clusters, iterated on test failures
- Sent code to GPT Codex for review targeting C-specific bugs
- Implemented performance optimizations for non-compression paths
- Reorganized into six clean commits and ported to Gerrit under LU-20178
- Ported feature to lustre-master, adapting for hybrid I/O changes
- Addressed code review feedback: struct layout, naming, coding style, compression-before-encryption ordering

Lustre Wiki Update

Prompt Summary/Rules:

- Modernize Lustre wiki into accurate, usable guide
- Prioritize stale pages over already current content
- Preserve historical pages unless misleading
- Verify facts against manual first, then source code
- Use JIRA only when manual and code disagree
- Review manual for areas the wiki is completely lacking, and create new pages without replicating manual content needlessly
- Avoid mass scripted rewrites; review and edit everything individually as a human would
- Merge redundant pages
- Leave clear deprecation notes on culled pages
- Edit wikitext sources, never generated plaintext files
- Document every single changed page in a list for later human review

Agent steps:

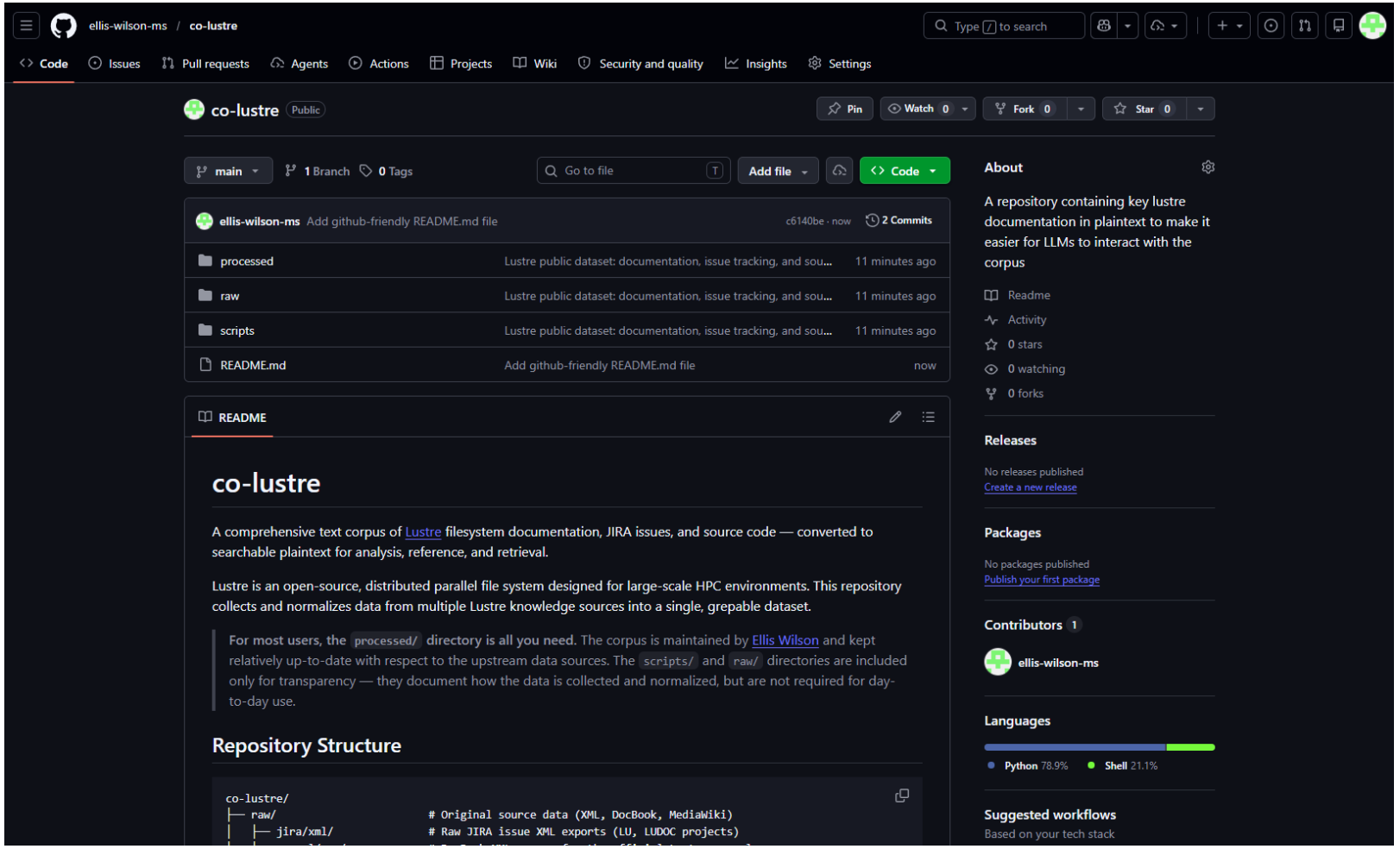
- Inventoried corpus
- Planned phased execution milestones clearly
- Organized pages by topic using action labels
- Prioritized onboarding gaps
- Expanded planned new pages
- Rewrote critical stale pages needing full modernization
- Applied surgical updates across many existing pages
- Wrote new operator guides for onboarding, ops, monitoring, security, HSM, recovery

Overall it:

- *Ran for about 4 hours nearly continuously*
- 51 pages deprecated/culled
- 22 pages merged
- 34 pages revised
- 30 new pages created
- 137 total pages touched
- ~36,500 lines changed

Get the code!

Get and Try it Yourself!



https://github.com/ellis-wilson-ms/co-lustre

Thank you! Questions?