

Lustre and AI assisted Development & Testing

Oleg Drokin, DDN

LAD, Sep 30, 2026



Pre-AI Era

- In the times past Lustre development process was pretty typical:
 - Code reviews
 - Automatic building and testing pipelines
 - Static scripting to highlight common bad patterns (checkpatch.pl)
 - Static code analysis (smatch, sporadical Coverity runs)

Advent of LLMs

- A couple years ago LLMs entered our field of view
- Limited context and primitive tooling
- Still some usefulness – documentation checking
- Patch review results unsatisfactory

Compile/build fixer

- First real useful tool: “healing” broken patches
- Give it a patch that does not compile
- Get a fixed patch (may be, for simpler breakages)
- Still held back by limited context and tooling
- Struggled to work on large files.

Here's a patch, what looks wrong?

- Patch review was big goal that was hard to meet
 - Dumping a big patch into an LLM would only provide it with limited context of surrounding code
 - Providing every diff with the source file would be limited by model max context
- The results were very shallow.
- External tooling like “only see if we touch doc files and check those” while more promising were also more limited.

Sashiko

- Several months ago I was told about Sashiko
 - the system behind Linux kernel review bot
- The system looked too heavy and full kernel centric
- It had externalized prompts that you could use with a regular coding agent!
- The prompts were pretty elaborate and split into many parts
- It turned out that adding basic “Lustre” part was easy enough
 - along with removing stuff we don’t need

First real results

- I selected a patch from the ready queue and had a manual pass on it as a baseline
- Run claude 4.6 with the prompts
 - And also some lesser models with pi.dev (qwen3.5 35B, 27B, ...)
- Results were much better than anticipated
- The elaborate prompting drove the agents to dive deep into just the sort of boring stuff people have trouble doing.
- First attempt at running the review on a master-next resulted in rejection of over half
- Though it's still not as great on architectural front

Production

- AI review is now a staple of our code review process
- But we are not dropping the “two human reviewers” requirement
 - not yet
- AI provides people additional anchors for their own reviews
- Developers mostly love it
 - There are also calls to run it earlier in the pipeline
- But there are still false positives and nitpicks
 - and room for human judgement calls
- Plenty of real bugs gets highlighted too!

Testing

- Lustre undergoes a lot of testing which produces a lot of results
 - Hundreds of hours per patch
- Few read those results
- There's also “full” testing with even more results that nobody reads
 - Other than QA people sometimes, mostly around release time

Testing with AI helpers

- AI to the rescue!
- Now it's relatively easy to just task an agent to compare two subsequent builds to get differences highlighted
- Not just highlighted, but to drill into any outliers to see if there's a trend or the finding is really new.
- This greatly expands our capability to detect regressions before they make it into the main tree
 - Previously we only mostly tracked “unique crashes”

Example new finding

● **Regression → flag in Gerrit** 3 findings

New vs master 4741 and recurring/growing across builds (995→996), so caused by the master-next delta — these go to Gerrit, never to a new LU. The first is pinned to a change; the other two are real but un-pinnable regressions where a blind -1 would be wrong.

TEST / SET	STATUS	EVIDENCE	PROPOSED ACTION
conf-sanity test_73g the patch's own new test · missing interop version guard · ×11	FAIL	• BASE does not exist in master 4741 — added by the change under test, delta commit 2b44b299dc LU-20403 (ptlrpc refresh import conn on primary migration). • ROOT fails in interop against a 2.15 server that lacks the feature: mounting the server's @tcp10 NID returns EINVAL. The adjacent test_73c already guards this (\$OST1_VERSION >= version_code 2.16.54 skip); test_73g has no guard. • 996 confirmed recurring — ×11 in 996 (995 was ×7). Same root cause, more runs hit it.	Gerrit 67017 (LU-20403, PS4, NEW) Test bug — add a server-version skip guard mirroring test_73c (\$OST1_VERSION / \$MDS1_VERSION). Recommend Code-Review -1 until guarded. No LU.
sanity-lnet test_220 / 221 (+207/223/225) route aliveness · full-part-1	FAIL	• BASE master 4741 runs these ~90× at 100% PASS — genuinely new. • NEXT Wanted "up" lctl found "down" → check_route_aliveness(). Persists across 994, 995 and 996 (test_220 ×6). • CAUSE the 994 suspect (66609/ LU-20294) is gone from the tree yet the failure persists: the only lnet	Real regression, un-attributed. Needs a bisect / test-build — do not -1 a single change on this evidence. LU-20294 (retracted suspect)

Example new finding 2

Never a ticket — these must be caught before the patch lands. Both rows are runtime, not pass/fail; the functional diff is blind to them.

ROW	WHAT MOVED	EVIDENCE	SUSPECT PATCH
R1 SLOWER	Adaptive- timeout floor rose across recovery/ replay Paired per- config medians (view B), baseline → target: - replay- dual/ test_24 18s → 27s 1.50x 101 configs - recovery- small/ test_110c 18s → 29s 1.65x, 110j 19s → 29s	The shift is additive, not multiplicative — every affected test gains roughly the same +6 to +11 s regardless of its baseline duration. That is the signature of a raised timeout floor, not of slower work. Control kills the drift explanation: lustre-master 4755 vs 4754 (same helper, same thresholds) puts the worst systemic slowdown at 1.13x (sanity/test_300c), with recovery-small/110c and 110j at 0.95x (19s → 18s) and no replay-dual subtest in view B at all. So master-to-master these are flat; the move is specific to -next.	63400 LU-12064 obdclass: auto-scale at_min based on expected clients The only delta commit that changes an AT floor. It computes at_min = clamp(ilog2(expected_clients)*3, 5, 60) and raises it automatically from tgt_update_max_clients(). Master's default is at_min = 5; anything the autotune computes above that lengthens every AT-driven wait by the difference — which is what the numbers show. The new at_min_param_set() also rejects at_min < 1 with -EINVAL and pins autotune off on the first manual write, so a test that lowers it through at_min_set() gets a silent behaviour change too.

Results

- Lustre master seems to be more stable now with all these aids
- We hope it'll result in a much better release timelines
 - Historically stabilization pre release could be lengthy.
- If these findings hold we will probably adjust the feature-freeze / code-freeze timelines

Coding

- AI coding agents are a decent help for rote stuff, changing interfaces and the like
 - Though they tend to be chatty.
 - Still a big win compared to e.g. Coccinelli
- Complicated changes on the other hand remain a challenge.
- We allow AI-assisted code contributions
 - Please use Assisted-by commit tags
- In my view many of those still fail to anticipate the depth of Lustre
 - Something that's ok in a review tool is a different matter in a coding tool.

Conclusion

- In the end humans are still in control.
- Whoever submits a patch is still responsible for it
- We (humans) still do the final vetting and approvals
- So in effect these are just more advanced tools
 - Tools that help us make a better job.

Thank You

Contact | first.last@ddn.com

