AURORALABS

LOCI - The First CPU-GPU Software Execution-Aware Model

Signal Faults and Errors
No Run, Before Tests, Logs or Production

Built for Embedded Software Engineers

Published Q1 2026

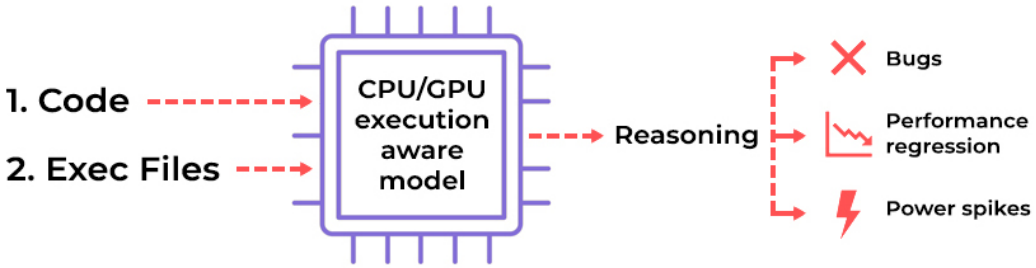
Meet LOCI:
The GPU/CPU-Execution Aware Agent that other LLMs do not have

LOCI, an MCP and S2-based solution, provides a new layer of intelligence for developers across networking, embedded systems, AI infrastructure, inference, and performance-sensitive systems. Unlike generic AI assistants, LOCI is designed to understand how code really interacts with hardware; it does not stop on syntax- it analyzes the exe files after the project build is completed, gives feedback, and finds bugs without running the SW.

How does LOCI work?

Specialized model, trained on CPU-GPU executions in different conditions

LOCI does not run or execute the software. Instead, it reads the program's existing compiled executable as static data and analyzes its structure. Even without execution, the binary already contains the full logic of how the software can behave—its functions, branches, loops, and call relationships.
LOCI breaks this static executable into execution units, maps all possible execution paths, and reconstructs a behavioral model of the program.
To make this static structure come to life, as if it were running, LOCI applies learned proprietary performance models for CPUs and GPUs that capture real hardware execution behavior.



These models estimate how long each part of the executable would take to run, how much power it would consume, and how it would stress hardware resources. LOCI composes these estimates across full execution paths and call stacks to produce predicted timelines, flame graphs, response times, power profiles, and bottleneck analysis—all without ever running the software.

LOCI analyzes from executable files without running them:

- Throughput, Bandwidth, Latency
- AI/LLM Networking: tokens-per-second drift
- CPU execution paths and branching behavior
- GPU kernels, warp divergence, and memory coalescing
- Logical bugs such as null dereferences, race conditions, and edge-case logic
- Power anomalies
- Memory pressure, allocation patterns, and concurrency hazards

AURORALABS

LOCI's outputs are constrained, physically meaningful

Large language models are generative systems optimized to produce plausible sequences of text or code, which can lead to hallucination when the model generates outputs that are syntactically valid but not grounded in execution reality. In contrast, the LOCI model is not generative and does not produce free-form outputs. It is trained in a supervised manner to predict a single, bounded numeric value - execution time, directly from assembly-level representations and measured ground-truth data. Because LOCI's outputs are constrained, physically meaningful, and tied to executable structure and hardware behavior, it cannot fabricate results or invent behaviors outside the defined execution domain.

LOCI in Action - Examples

LOCI automatically comments on pull requests, surfacing issues developers often miss:

- Performance drift detected (+12.4%) in processFrame(): nested loop increases GPU pressure.
- Power anomaly identified in computeTransform(): risk of IR-drop under peak load.
- Logical bug found: uninitialized variable 'bufferLen' and potential null pointer dereference.

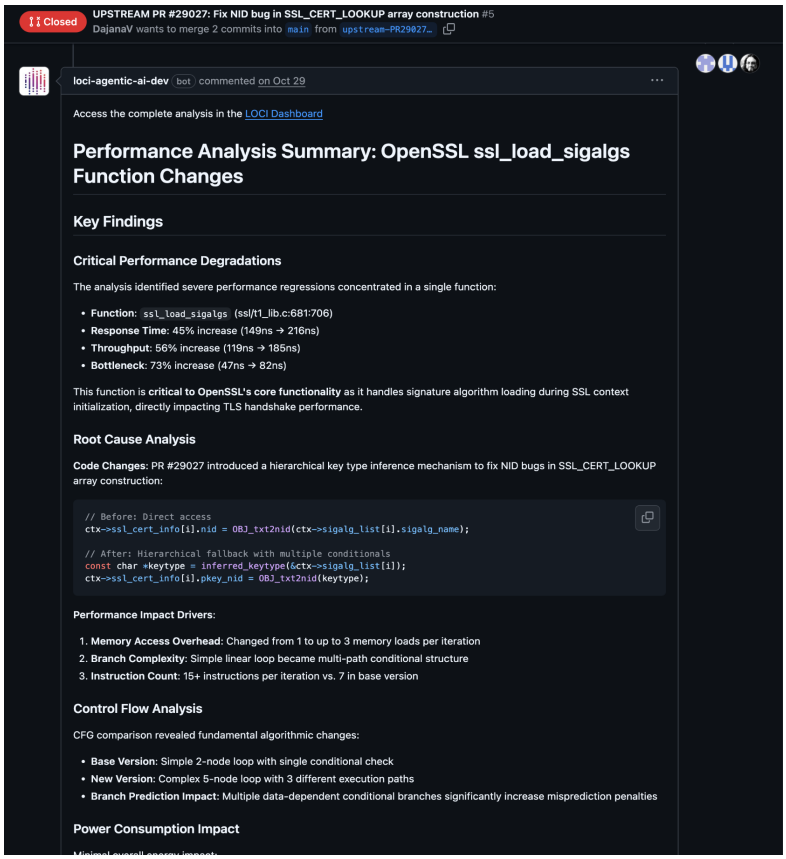
In addition, LOCI provides fixes:

1. Move memory allocation outside the inner loop to stabilize GPU timing.
2. Align GPU and CPU execution paths to avoid unpredictable divergence.
3. Ensure all variables are initialized before branching logic.

GitHub Pull Request Example

In the screenshot below, you can see a full analysis of PR in the GitHub OpenSSL project before engaging in code review or executing any tests.
Click here to enter the project on GitHub: https://github.com/auroralabs-loci/openssl/pull/5

This eliminates guesswork and provides developers with clarity before CI/CD pipelines even start.



Logic, Performance, and Power Analysis of OpenSSL PR#29027, Oct 29, 2025

AURORALABS

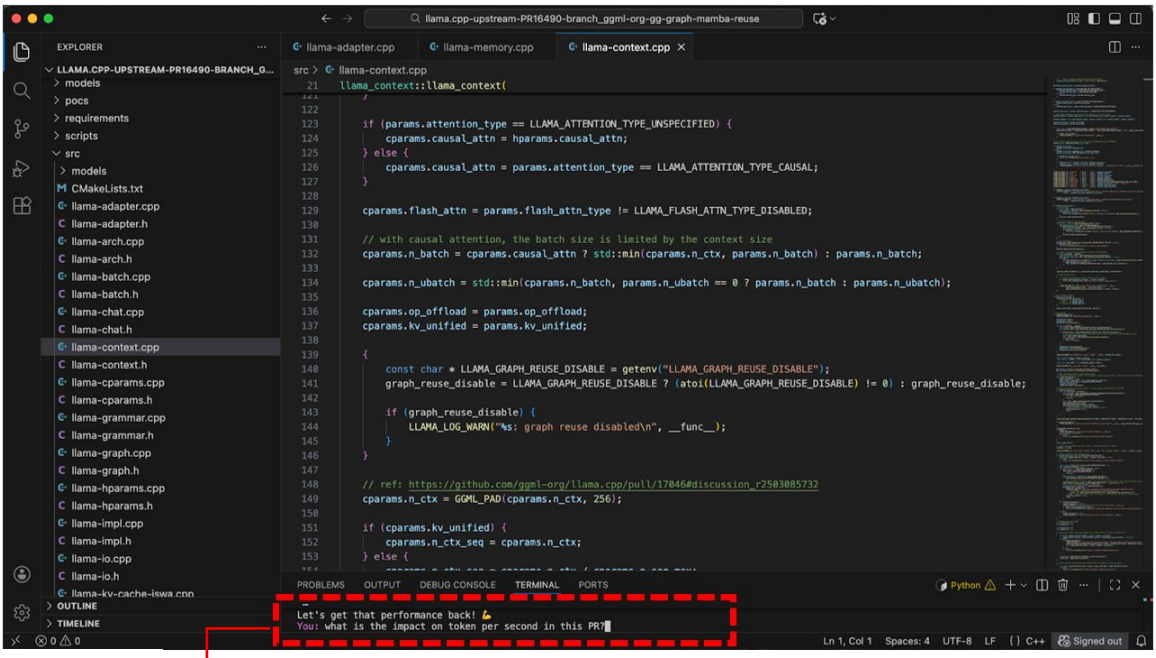
Ask LOCI Anything | Via GitHub comment @LOCI-DEV or directly from your IDE

LOCI supports interactive Q&A inside the PR. Developers can ask targeted engineering questions:

- Why did execution time increase?
- Which lines introduced performance drift?
- What caused the sudden power spike?
- Does this change introduce logical or memory safety risks?

Example of LOCI response:
"Execution drift is caused by a memory allocation moved inside a high-frequency loop. TCP_IP_Send shows a 4.7% slowdown. Variable 'packetCount' may be uninitialized. Recommended fixes: refactor allocation, initialize variable, rebalance branching logic."

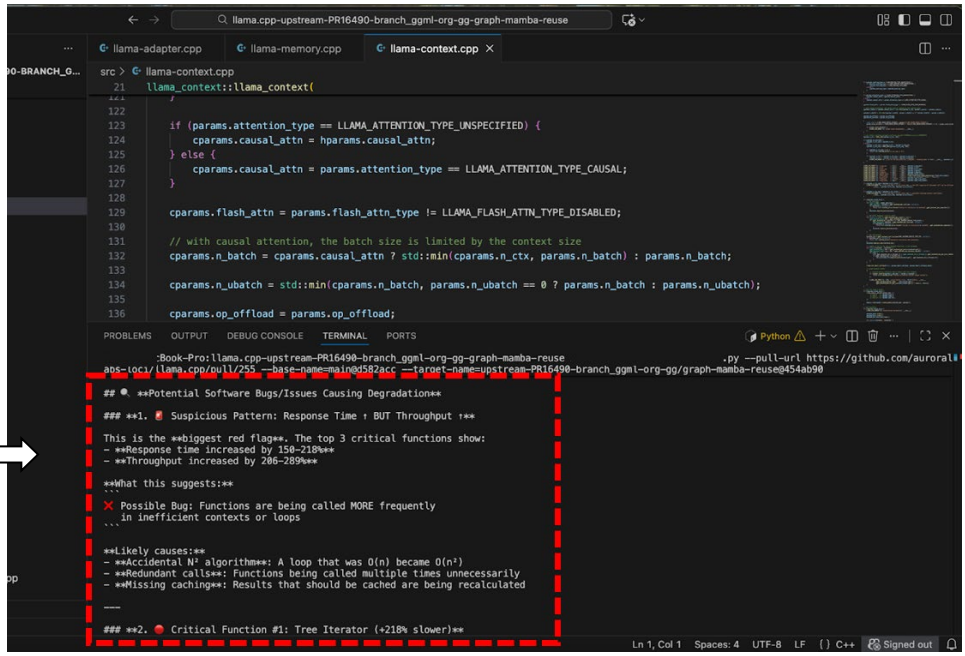
The following is LOCI's terminal App for VS Code, using its CPU/GPU execution aware modeling to find Bugs and Performance issues after project build, before running tests



LOCI Terminal

LOCI flagged the most impactful degradation issue.

The top 3 critical functions are presented



AURORALABS

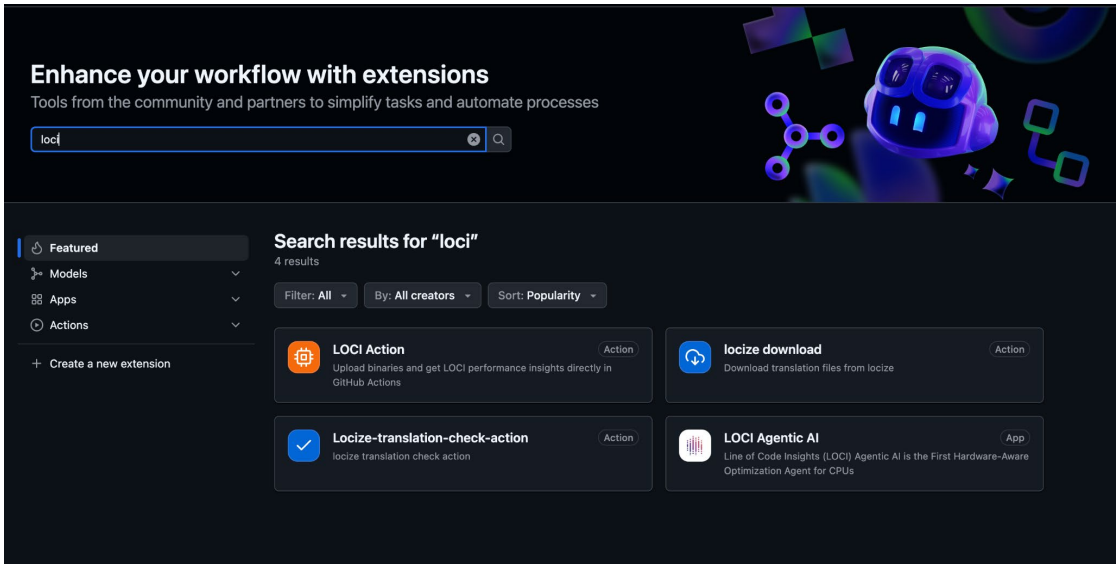
Real ROI in 2-3 Weeks

Engineering teams using LOCI consistently report measurable improvements within 2-3 weeks:

- 20%-40% reduction in test cycle time due to earlier detection
- Fewer regressions reaching integration and system-level tests
- Faster code reviews with factual, hardware-aware insights
- Stronger predictability in release cadence
- Reduced engineering cost through minimized rework

LOCI reduces the costliest factor in modern engineering: the late discovery of regressions.

Developers can see LOCI live today on open-source workflows such as OpenSSL, where LOCI interacts directly with contributors.



LOCI | Software Engineering Intelligence for Embedded Software Engineers.

Catch regressions before testing. Build with confidence

Next Steps:

1. See LOCI in action in GitHub
2. Book a discovery meeting with a demo >>
3. Contact us at info@auroralabs.com or visit www.auroralabs.com