OPTIMASX: An Agentic End-to-End GPU Performance Optimization Workflow in the Code Editor

[1]Yugesh Bhattarai, [2]Jessica Rubin, [1]Prajan Manoj Kumar Rekha, [1]Mohammad Zaeed, [1]Tanzima Z. Islam

yugesh@txstate.edu, jessica.rubin@bison.howard.edu, tjy33@txstate.edu, cup7@txstate.edu, tanzima@txstate.edu

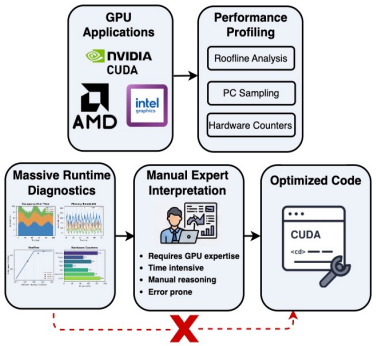
[1]Texas State University, [2]Howard University

Abstract

GPU performance optimization requires expert interpretation of runtime diagnostics and manual code transformation. OPTIMAS automates this reasoning but assumes users already possess HPC cluster access and generate their own profiling data. We present OPTIMASX, a workflow orchestration architecture that connects the developer's editor to remote HPC profiling infrastructure. OPTIMASX contributes three capabilities: (1) an autonomous profiling agent that automatically generates and submits cluster jobs, (2) a stateless orchestration model in which workflow state resides entirely in the developer's VS Code extension, enabling resilient execution across backend restarts, and (3) a modular, skill-driven agent framework that exposes optimization stages as callable tools. By preserving the optimization pipeline introduced by OPTIMAS, which improved runtime in 98.82% of 3,410 evaluated cases, OPTIMASX reduces expert-driven HPC optimization to a single interaction from within the developer's editor.

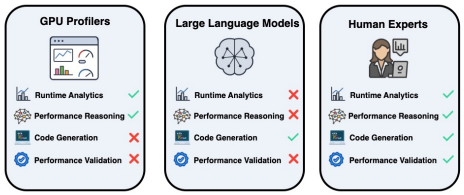
Motivation

Current GPU Optimization Workflow



No automated path exists between performance diagnostics and optimized GPU code

Limitations of Existing GPU Optimization Approaches



LLM optimizers such as CUDA-L1 [2] learn from measured speedup alone, without runtime diagnostics. No existing approach unifies analytics, reasoning, generation, and validation in one automated GPU workflow.

OPTIMAS Framework

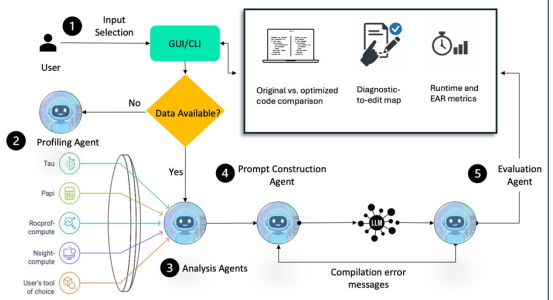


Figure 1: Original Overview of OPTIMAS End-to-End Workflow

Expanding on the above workflow, OPTIMASX is now a multi-agentic framework that automatically coordinates performance profiling, performance modeling, bottleneck detection, code optimization, and verification through specialized agents.

OPTIMASX Workflow

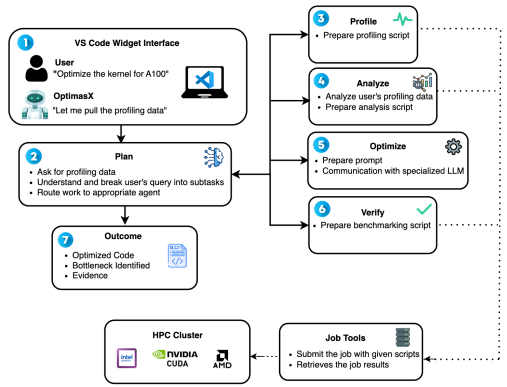


Figure 2: OPTIMASX Architecture

The user submits code and a target GPU. OPTIMASX plans, profiles, analyzes, optimizes, and verifies, then returns evidence-backed code. Existing profiling data is reused when available and collected on Lonestar6 when not. An LLM planner chooses which agent to call and when, so the order of steps can vary between runs of the same request. To limit this, the planner uses a fixed seed, skill files require profiling results before analysis can run, tool calls per turn are capped, optimized code must match the original output before it is returned, and each speedup is averaged over five runs.

Preliminary Results

Performance Results

Method	Sobol	XSBench	Cholla	HeCBench (8 apps)
Optimas	1.18 ± 0.11	4.83 ± 0.22	1.504 ± 0.001	2.282 ± 2.465
OptimasX	1.19 ± 0.11	4.80 ± 0.18	1.504 ± 0.000	3.17 ± 3.39

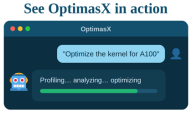
Average speedup (×) ± standard deviation over five runs. HeCBench apps: boxfilter, chemv, concat, convolutionSeparable, accuracy, bincount, amgmk, and blockAccess. Speedups are inherited from OPTIMAS. OPTIMASX automates the existing process and does not newly produce them.

Automation and Cost

Metric	Manual	Optimas	OptimasX
API cost per session	0	$0.03	$0.0019
Manual steps during profiling	15	15	0
Manual steps during analysis	15	15	0
Manual steps during optimization	2	0	0
Total manual steps	**32**	**30**	**0**

We ran the same GPU optimization task three ways: by hand, with OPTIMAS, and with OPTIMASX. Manual steps drop from 32 to zero at an API cost of $0.0019 per session. Speedups come from the OPTIMAS optimizer, rerun through OPTIMASX at a different LLM temperature, so the two rows are not directly comparable.

See OptimasX in action



Scan for the live demo

Summary & Future Work

- OPTIMASX reduces GPU kernel optimization from a multi-step manual workflow across two machines to a single request made from the code editor.
- A stateless orchestration design keeps all workflow state in the VS Code extension, so the backend can restart without losing the session.
- Extending profiling support to AMD and Intel GPUs, which the existing scripts already target but have not yet been evaluated.
- Extending beyond TACC Lonestar6 to other HPC facilities, since credentials are user-provided rather than hardcoded.

References

[1] M. Zaeed, T. Z. Islam, and V. Indic, "OPTIMAS: An intelligent analytics-informed generative AI framework for performance optimization," in Proc. 32nd ACM SIGKDD Conf. Knowledge Discovery and Data Mining (KDD '26), Jeju Island, Republic of Korea, Aug. 9-13, 2026, doi: 10.1145/3770855.3818188.

[2] X. Li, X. Sun, A. Wang, J. Li, and C. Shum, "CUDA-L1: Improving CUDA optimization via contrastive reinforcement learning," in Proc. ICLR, 2026. [Online]. Available: https://arxiv.org/abs/2507.14111

This material is based upon work supported in part by the National Science Foundation under Grant No. 2443561. The authors acknowledge the Texas Advanced Computing Center (TACC) at The University of Texas at Austin. This material is based upon work supported by the National Science Foundation under Award No. CCF-2621730. Any opinions, findings, and conclusions or recommendations expressed in this material are those of the author(s) and do not necessarily reflect the views of the National Science Foundation. The authors acknowledge that tool selection in OPTIMASX is LLM-mediated and therefore non-deterministic, so tool ordering is not guaranteed identical across repetitions of an identical prompt. Portions of the poster text were drafted and edited with assistance from Claude (Anthropic). The authors reviewed and take responsibility for all content.

NSF · TEXAS STATE UNIVERSITY · TACC TEXAS ADVANCED COMPUTING CENTER · CRA Computing Research Association · PER4MI · HOWARD UNIVERSITY