GenASPEN: Agentic AI for Automatic Performance Modeling of HPC Codes

Grace McLewee[†*], Mohammad Alaul Haque Monil[‡*],
Pedro Valero-Lara[‡], Narasinga Miniskar[‡], William Godoy[‡], Allen Malony[†], Keita Teranishi[‡], Jeffrey S. Vetter[‡]
[†]University of Oregon, Eugene, Oregon, USA and [‡]Oak Ridge National Laboratory, Oak Ridge, Tennessee, USA

Introduction: The growing heterogeneity of high performance computing (HPC) programming models and hardware architectures has made automated performance modeling simultaneously more essential and more difficult to achieve. Performance models underpin critical HPC workflows—design space exploration, procurements, software optimization, and runtime scheduling. Compiler-based frameworks such as COMPASS [1] and MAPredict [2] are accurate but must be re-integrated for every new architecture; frontier LLMs are flexible but predict runtime poorly; custom LLMs buy accuracy at prohibitive cost—Omniwise required 955,000 profiled kernels [3]. No existing approach delivers accuracy, flexibility, and efficiency simultaneously. These observations **motivate** the development of GenASPEN [4].

> ***Hypothesis:*** *An agentic AI workflow combining LLM-based code translation with analytical performance modeling can achieve accurate HPC performance prediction with competitive turnaround time and without compiler-level effort.*

Our ***objective*** is to develop GenASPEN, a collaborative agentic-AI ecosystem for performance modeling that builds on the ASPEN performance modeling framework [5]. ASPEN decouples an abstract application model, a machine model, and prediction tools invoking analytical models; two steps stayed manual—translating source into an application model, and choosing the analytical model. LLMs are strong at code understanding, analytical models at quantitative prediction—GenASPEN gives each task to the component best equipped for it, with agentic AI orchestrating the LLMs, applying reasoning, and calling non-LLM tools. To our knowledge, GenASPEN is the first system to integrate generative AI, analytical modeling, and human expertise in a unified agentic workflow for automatic HPC performance modeling. The main ***contributions*** are (i) a novel collaborative agentic AI ecosystem integrating LLM and non-LLM tools with experts-in-the-loop; (ii) a code translation approach converting diverse HPC programming models into ASPEN application models without compiler-based tooling, via two LLM strategies—context-aware chain-of-thought (CA-CoT) prompting on GPT-OSS 20B/120B and fine-tuning of Code Llama 7B; (iii) new capabilities in the ASPEN framework—machine and analytical models plus tooling—to support NVIDIA and AMD GPUs and to collect application parameters; and (iv) a detailed evaluation on essential HPC and AI benchmarks and applications including BLAS routines, conjugate gradient (CG), XSBench, LULESH, and MiniMDock.

Framework: As shown in Fig. 1, GenASPEN has two agents. The ③ API takes the kernel source and the target architecture, and accepts natural-language input describing the requested prediction (e.g., "What is the runtime of this kernel when n is 1 million on an H100 GPU?"). The ① Code Translation Agent offers either ①a a CA-CoT pipeline on GPT-OSS 120B [6], or ①b a Code Llama 7B assistant fine-tuned with ChatHPC [7]—both emit ①c an ASPEN application model. The ② Performance Modeling Agent uses ②c ASPEN tools bound to GPT-OSS 20B that traverse ①c and ②a the machine model, then invoke ②b the analytical model for that GPU. Components ②a and ②b are calibrated once per architecture, offline; experts fix corner cases and saved models bypass every LLM call. Orchestration uses LangChain's LangGraph agent framework [8], [9], so GenASPEN is embeddable in applications, runtime systems, other agentic AI workflows, and libraries.

Code Translation Agent (Fine-tuning vs. CA-CoT): The fine-tuned Code Llama 7B uses LoRA adapters produced through ChatHPC and was trained on 16 CUDA-to-ASPEN pairs drawn from BLAS, deep-learning, PDE, and sparse benchmarks—completing in about 30 minutes on two H100 GPUs, with the three proxy applications and other kernels held out for testing (a 36%:64% train/test split). The CA-CoT pipeline decomposes translation into (Stage 1) kernel characteristic analysis producing an intermediate JSON representation of performance features, and (Stage 2) ASPEN application model generation through grammar-constrained synthesis, with an optional extended analysis pass—engaged only for tiled BLAS Level 3 and the three proxy applications—that mitigates context-window saturation. Fine-tuning is up to $101\times$ faster on as little as 1.74% of the tokens, but breaks down on the three proxy applications (LULESH, XSBench, MiniMDock); CA-CoT translated every one correctly, with no expert intervention.

Performance Modeling Agent: ASPEN tools—FLOP prediction, load/store prediction, and runtime prediction—are bound to GPT-OSS 20B, which selects both the correct tool and the target machine model. Machine models for NVIDIA H100 NVL and AMD MI300A are built from empirically mea-

*Grace McLewee and Mohammad Alaul Haque Monil are co-first authors.

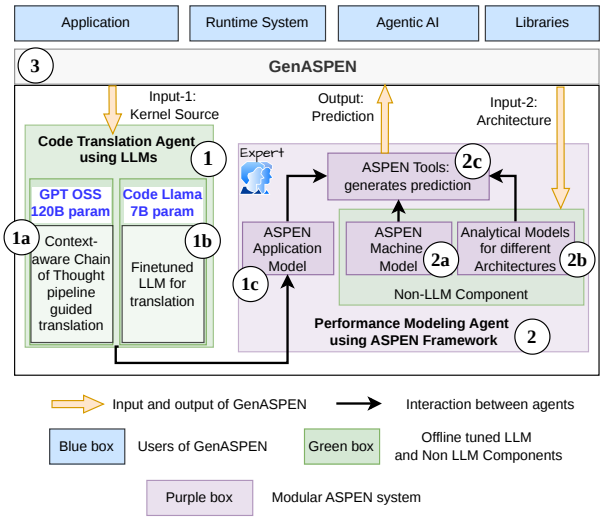


Fig. 1. GenASPEN is an LLM-enabled collaborative agentic AI ecosystem for performance modeling of HPC kernels. The (1) code translation agent offers two options—(1a) a context-aware chain-of-thought pipeline using GPT-OSS models and (1b) an AI assistant based on a fine-tuned Code Llama model—to generate (1c) ASPEN application models. The (2) performance modeling agent coordinates (2a) machine models, (2b) analytical models, and (2c) prediction tools registered with the GPT-OSS reasoning model. The (3) API is the entry point for automated performance modeling.

sured metrics: STREAM bandwidth [10] and Roofline peak FLOPs [11]. A new analytical model expresses execution time as $T_{\text{exec}} = \max(T_{\text{compute}}, T_{\text{memory}})$, with empirically derived calibration factors λ_{pattern} per access pattern (stride, stencil, sparse, shared memory) fit via linear regression—a one-time tuning step per architecture, analogous to measuring attainable bandwidth. The code translation agent annotates the access patterns in the application model, which are captured by the performance modeling agent to invoke the correct prediction model—no manual pragma annotations required.

Evaluation: A total of 22 benchmarks and applications were evaluated on two GPU systems—Hudson (2×NVIDIA H100 NVL; 3.3 TB/s attainable bandwidth, 27.3 TFLOP/s attainable peak) and Faraday (4×AMD MI300A; 3.6 TB/s, 82.1 TFLOP/s)—covering BLAS Levels 1–3, deep-learning primitives, 1D/2D/3D stencils and Laplace, SpMV/SpMM, and the CG, XSBench, MiniMDock, and LULESH proxy applications, which span memory-bound to compute-bound behavior, regular to irregular memory access, and low to very-high modeling complexity. Operation counts are exact on 19 of 22 benchmarks (98.50% FLOPs, 98.375% MOPs) because the LLM resolves both the computational pattern and its structural parameters. Execution time is predicted to **94.40%** accuracy on the H100 and **93.09%** on the MI300A. Residual error sits in the analytical model rather than the application model: MiniMDock is under-predicted because synchronization is not yet considered in modeling (future work).

Frontier LLMs vs. GenASPEN: Against four frontier LLMs (ChatGPT-5.5 with extended thinking, Claude-Opus-4.7 with adaptive thinking, Gemini-3.1-Thinking, and DeepSeek-V4-Base with DeepThink and Search), GenASPEN achieves the best execution-time accuracy on **16 of 22** kernels, averaging **94.40%** versus 43.41% (ChatGPT), 51.62% (Claude), 41.21% (Gemini), and 43.60% (DeepSeek). Median response times are 0.7 s cached, 3.0 s fine-tuning, and 69.1 s CA-CoT for GenASPEN, versus 34.8 s, 31.1 s, 30.5 s, and 211.2 s for the four LLMs. GenASPEN's fine-tuning mode uses under 1% of the tokens consumed by ChatGPT-5.5 and Claude-Opus-4.7 for model generation. Frontier LLMs performed reasonably on low-complexity kernels with regular memory access (BLAS Level 1, deep learning, 1D stencils), but their accuracy falls off as complexity grows; on the high-complexity proxy applications (XSBench, MiniMDock, and LULESH), several LLMs produced no prediction at all, whereas GenASPEN retained accuracy across every complexity tier.

Classic Approaches vs. GenASPEN: COMPASS needs up to 125 annotation lines and MAPredict up to 249 lines to model LULESH; GenASPEN needs *none* and reaches 97.04% execution-time accuracy. Against LLMPerf [12], which predicts SpMV runtime at 32.5–56.8% MAPE, GenASPEN reaches 11.16% APE. GenASPEN's near-perfect FLOP/MOP counts are consistent with Omniwise (only 2% of its predictions exceed 10% error) but require no profiled kernels.

Conclusions: GenASPEN validates the hypothesis: 94.40% (H100) and 93.09% (MI300A) mean execution-time accuracy, beating four frontier LLMs on 16 of 22 kernels on a fraction of the tokens, with zero manual annotation where classic tools need hundreds of lines. Future work: more platforms, automated recalibration, and distributed execution.

Acknowledgment

This work is funded by DOE ASCR AI for Science projects, Durban and Ellora.

References

[1] S. Lee, J. S. Meredith, and J. S. Vetter, "COMPASS: A framework for automated performance modeling and prediction," in *Proceedings of the 29th ACM on International Conference on Supercomputing*. Newport Beach, California, USA: ACM, 2015, pp. 405–414.

[2] M. A. H. Monil, S. Lee, J. S. Vetter, and A. D. Malony, "MAPredict: Static analysis driven memory access prediction framework for modern CPUs," in *International Conference on High Performance Computing*. Springer, 2022, pp. 233–255.

[3] Z. Wang, C. Ramos, M. A. Awad, and K. Lowery, "Omniwise: Predicting gpu kernels performance with llms," *arXiv preprint arXiv:2506.20886*, 2025.

[4] G. McLewee, M. A. H. Monil, P. Valero-Lara, N. Miniskar, W. Godoy, A. Malony, K. Teranishi, and J. S. Vetter, "GenASPEN: An agentic AI framework for automatic performance modeling and prediction of HPC codes," in *ACCEPTED at 2026 IEEE Int. Conf. Cluster Comput. (CLUSTER)*, Alexandria, VA, USA, 2026, (Accepted).

[5] K. L. Spafford and J. S. Vetter, "Aspen: A domain specific language for performance modeling," in *SC12: International Conference for High Performance Computing, Networking, Storage and Analysis*, Salt Lake City, 2012, pp. 1–11.

[6] OpenAI, S. Agarwal, L. Ahmad, J. Ai, and e. a. Sam Altman, "gpt-oss-120b & gpt-oss-20b model card," 2025. [Online]. Available: https://arxiv.org/abs/2508.10925

[7] P. Valero-Lara, A. R. Young, M. A. H. Monil, Z. Jin, S. Pophale, J. S. Vetter, K. Teranishi, and W. F. Godoy, "Chathpc: Building the foundations for a productive and trustworthy ai-assisted hpc ecosystem," in *Proceedings of the International Conference on High Performance Computing, Networking, Storage, and Analysis (SC '25)*. New York, NY, USA: ACM, Nov. 2025.

[8] LangChain, "Langchain: Building applications with llms through composability," 2024, accessed: 2025-09-30. [Online]. Available: https://github.com/langchain-ai/langchain

[9] J. Wang and Z. Duan, "Agent ai with langgraph: A modular framework for enhancing machine translation using large language models," *arXiv preprint arXiv:2412.03801*, 2024.

[10] J. D. McCalpin, "Stream benchmarks," 2002.

[11] S. Williams, A. Waterman, and D. Patterson, "Roofline: an insightful visual performance model for multicore architectures," *Communications of the ACM*, vol. 52, no. 4, pp. 65–76, 2009.

[12] K. N. Nguyen, H. D. N. Do, H. T. Le, and T. T. Dao, "Llmperf: Gpu performance modeling meets large language models," *arXiv preprint arXiv:2503.11244*, 2025.