From Legacy to Portable: An Agentic AI Workflow for Fortran Code Translation and Cross-Architecture Optimization

Sparsh Gupta
sgupta1@olin.edu
Los Alamos National Laboratory
Los Alamos, NM, USA
Franklin W. Olin College of Engineering
Needham, MA, USA

Kamalavasan Kamalakkannan
kamalavasan@lanl.gov
Los Alamos National Laboratory
Los Alamos, NM, USA

Maxim Moraru
moraru@lanl.gov
Los Alamos National Laboratory
Los Alamos, NM, USA

Galen Shipman
gshipman@lanl.gov
Los Alamos National Laboratory
Los Alamos, NM, USA

Patrick Diehl
diehlpk@lanl.gov
Los Alamos National Laboratory
Los Alamos, NM, USA

KEYWORDS

Agentic AI, Artificial Intelligence (AI), Fortran, Generative AI, High-Performance Computing (HPC), Large Language Models (LLMs)

ACM Reference Format:
Sparsh Gupta, Kamalavasan Kamalakkannan, Maxim Moraru, Galen Shipman, and Patrick Diehl. 2025. From Legacy to Portable: An Agentic AI Workflow for Fortran Code Translation and Cross-Architecture Optimization. In *Proceedings of SC: The International Conference for High Performance Computing, Networking, Storage, and Analysis (SC '25).* ACM, New York, NY, USA, 2 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 SUMMARY

Modern scientific applications continue to rely heavily on legacy Fortran codebases that were originally developed for homogeneous, CPU-based systems. While Fortran remains central to work across national laboratories and academia, the High-Performance Computing (HPC) landscape has been rapidly evolving towards heterogeneous, GPU-accelerated architectures. Many modern accelerators from vendors such as AMD and Intel lack native support for Fortran bindings, creating a strong need to port and optimize these legacy kernels to modern frameworks like Kokkos. The Kokkos C++ Performance Portability Ecosystem [15], originally developed under the U.S. Department of Energy's Exascale Computing Project and now part of the Linux Foundation's High Performance Software Foundation offers performance portability by allowing developers to write single-source code that runs efficiently on multiple hardware backends (CUDA [10], HIP [2], SYCL [6], OpenMP [4], HPX [8], C++ threads). Therefore, porting Fortran code to Kokkos provides an optimistic solution for performance portability; however, manual translation requires heavy domain expertise and proves to be very time-consuming.

In this work, we introduce a fully automated agentic AI workflow that leverages large language models (LLMs) [9] to translate and optimize Fortran kernels to Kokkos for diverse heterogeneous architectures. Our framework leverages Open AI models like GPT-5 [13] and o4-mini (high) [14] directly sourced from their API and we also utilize open-source LLMs, such as LLaMA 4 Maverick [1], served locally on HPC nodes via Ollama [11], with requests routed through a LiteLLM [3] proxy. These models are accessed using the OpenAI Agents SDK [12], a lightweight Python framework for building modular, multi-agent workflows. Each stage of the pipeline is managed by a specialized agent responsible for a specific task in the code transformation workflow. We also built a modular Python-based package to utilize with this workflow which includes helper functions, profiling tools functions, SLURM [7] job management scripts, and Spack-based [5] environments for CUDA, ROCm, and OpenMP. Each stage of the pipeline is managed by a specialized agent responsible for a specific task in the code transformation workflow:

- *Translator* agent, which converts Fortran kernels into standalone Kokkos-based C++ programs. The generated code includes a `main()` function, accepts problem size (`n`) and repetitions (`r`) as command-line arguments, and outputs the total kernel execution time across all repetitions for performance benchmarking.
- *Validator* agent then checks whether the code is purely C++ and contains no non-code output, and a *Fixer* agent attempts to correct syntactic or structural issues and make the code compilation-ready.
- *Build* agent submits compilation jobs in the appropriate Spack-managed environment, and a *Run* agent executes runtime sweeps across a configurable range of input sizes.
- If failures occur, error-handling agents are invoked. A *Compile Error Fixer* and a *Runtime Error Fixer* interpret SLURM error log outputs using an intermediate *Error Summarizer* agent, which refines verbose logs into concise summaries of root causes and potential fixes. These summaries are then fed back into the appropriate fixer agent to do minimal corrections to the generated code. This process continues until the code compiles and runs successfully but is bounded by configurable thresholds: `MAX_COMPILE_FIXES` and `MAX_RUNTIME_FIXES`.

SC '25, November 16–21, 2025, St. Louis, MO, USA
2025. ACM ISBN 978-x-xxxx-xxxx-x/YY/MM
https://doi.org/10.1145/nnnnnnn.nnnnnnn

- *Functionality Tester* agent checks that the output produced by the translated Kokkos program matches the Fortran program output for test cases. If discrepancies are found, the *Functionality Fixer* agent attempts to revise the code to obtain functional correctness bounded by the threshold `MAX_FUNCTIONALITY_FIXES`.
- Once a functional Kokkos baseline is established, the workflow enters a performance optimization loop. A performance summary and runtime metrics are obtained from hardware profilers such as NVIDIA Nsight Compute for CUDA and rocprofv3 for ROCm. These tools capture metrics such as L1 cache hit rates, L2 hits/misses, GPU occupancy, warp stalls, and memory throughput.
- *Optimizer* agent uses this feedback to propose structural code changes aimed at improving memory access patterns, loop restructuring, data layout changes, etc. Each new version is then recompiled, re-run, and re-profiled using the same agents. The optimization cycle continues for `MAX_OPTIMIZATION_ROUNDS`.

We evaluated this pipeline on five benchmark Fortran kernels: Conjugate Gradient (CG), Embarrassingly Parallel (EP), Multi-Grid (MG), Fourier Transform (FT) from the NAS Parallel Benchmarks, and DGEMM from OpenBLAS. The pipeline successfully produced functionally correct Kokkos implementations across heterogeneous GPU backends (NVIDIA A100, NVIDIA GH200, AMD MI250) using both proprietary and open-source LLMs. Using OpenAI's paid models, we achieved fully autonomous Fortran-to-Kokkos translation in under $3.5 per kernel (ignoring GPU runtime costs), whereas manual porting would require weeks of expert effort. Optimization rounds consistently improved performance over the baseline (v1), with GFLOPS gains observed across kernels, demonstrating that LLM-driven agents can not only achieve correctness but also drive iterative performance improvements. In contrast, open-source models such as Llama 4 Maverick frequently failed to converge within fixed thresholds.

In conclusion, this work demonstrates the feasibility of an autonomous agentic AI workflow to bridge the gap between legacy HPC codes and modern heterogeneous architectures. Future directions include: (i) expanding functionality testing beyond kernel-specific checks toward dynamic, LLM-generated correctness tests; (ii) extending optimization with architecture-specific tuning strategies; and (iii) assigning different agents to specialized LLMs (e.g., coding models for translation, lightweight models for validation, reasoning-oriented models for optimization) to increase efficiency and robustness. Together, these advances will enable scalable, fully autonomous modernization of large scientific codebases.

SUPPLEMENTARY MATERIAL

The agent prompts, the translated Kokkos source codes, and plots are available on Zenodo (https://zenodo.org/records/17064942).

ACKNOWLEDGMENTS

Research presented in this article was supported by the National Security Education Center (NSEC) Informational Science and Technology Institute (ISTI) using the Laboratory Directed Research and Development program of Los Alamos National Laboratory project number 20240479CR-IST. This research used resources of the National Energy Research Scientific Computing Center, a U.S. Department of Energy Office of Science User Facility operated under Contract No. DE-AC02-05CH11231. This work was also supported by the U.S. Department of Energy through the Los Alamos National Laboratory. Los Alamos National Laboratory is operated by Triad National Security, LLC, for the National Nuclear Security Administration of U.S. Department of Energy (Contract No. 89233218CNA000001).
LA-UR-25-27794

REFERENCES

[1] AI, M. Llama 4: The next generation of multimodal intelligence. https://ai.meta.com/blog/llama-4-multimodal-intelligence/, Apr. 2025. Accessed: 2025-08-11.
[2] AMD. Hip: C++ heterogeneous-compute interface for portability.
[3] BerriAI. Litellm: One api to run models across providers. https://github.com/BerriAI/litellm. Accessed: 2025-07-21.
[4] Dagum, L., and Menon, R. Openmp: An industry-standard api for shared-memory programming. *IEEE Comput. Sci. Eng. 5*, 1 (Jan. 1998), 46–55.
[5] Gamblin, T., LeGendre, M., Collette, M. R., Lee, G. L., Moody, A., de Supinski, B. R., and Futral, S. The Spack Package Manager: Bringing Order to HPC Software Chaos. Supercomputing 2015 (SC'15). LLNL-CONF-669890.
[6] Inc., T. K. G. Sycl: C++ programming for heterogeneous parallel computing.
[7] Jette, M., Dunlap, C., Garlick, J., and Grondona, M. Slurm: Simple linux utility for resource management.
[8] Kaiser, H., Diehl, P., Lemoine, A. S., Lelbach, B. A., Amini, P., Berge, A., Biddiscombe, J., Brandt, S. R., Gupta, N., Heller, T., Huck, K., Khatami, Z., Kheirkhahan, A., Reverdell, A., Shirzad, S., Simberg, M., Wagle, B., Wei, W., and Zhang, T. Hpx - the c++ standard library for parallelism and concurrency. *Journal of Open Source Software 5*, 53 (2020), 2352.
[9] Minaee, S., Mikolov, T., Nikzad, N., Chenaghlu, M., Socher, R., Amatriain, X., and Gao, J. Large language models: A survey, 2025.
[10] NVIDIA, Vingelmann, P., and Fitzek, F. H. Cuda, 2020.
[11] Ollama. Ollama. https://ollama.com. Accessed: 2025-07-21.
[12] OpenAI. Openai agents sdk. https://openai.github.io/openai-agents-python/, 2024. Accessed: 2025-07-21.
[13] OpenAI. GPT-5 System Card. https://cdn.openai.com/pdf/8124a3ce-ab78-4f06-96eb-49ea29ffb52f/gpt5-system-card-aug7.pdf, 2025.
[14] OpenAI. OpenAI o3 and o4-mini System Card. https://cdn.openai.com/pdf/2221c875-02dc-4789-800b-e7758f3722c1/o3-and-o4-mini-system-card.pdf, 2025.
[15] Trott, C., Berger-Vergiat, L., Poliakoff, D., Rajamanickam, S., Lebrun-Grandie, D., Madsen, J., Al Awar, N., Gligoric, M., Shipman, G., and Womeldorff, G. The kokkos ecosystem: Comprehensive performance portability for high performance computing. *Computing in Science Engineering 23*, 5 (2021), 10–18.