ChatHPC: Building the Foundations for a Productive and Trustworthy AI-Assisted HPC Ecosystem

Pedro Valero-Lara, Aaron R. Young, Mohammad Alaul Haque Monil, Swaroop Pophale, Zheming Jin, Jeffrey S. Vetter, Keita Teranishi, William F. Godoy
Oak Ridge National Laboratory
Oak Ridge, TN, USA
{valerolarap,youngar,monilm,pophaless,jinz,vetter,teranishik,godoywf}@ornl.gov

ABSTRACT

We introduce ChatHPC: a tool chain for developing AI assistants fine-tuned from Code Llama and specifically designed to boost productivity in HPC software development. Our initial ChatHPC models target four critical areas: parallelizing sequential code, optimizing existing parallel implementations, porting applications across heterogeneous platforms, and tooling. By tailoring Code Llama—a robust foundation model for code generation—to the nuances of the HPC software stack (e.g., programming systems, math libraries, I/O, tools), we enable developers to generate accurate, high-performance, and platform-specific code while requiring significantly fewer manual tasks. Our approach leverages domain-specific fine-tuning and uses datasets that include HPC benchmarks and real-world applications to ensure practical utility. These efforts aim to create a more cost-effective and productive AI-assisted HPC software ecosystem. The ChatHPC ecosystem is composed of the ChatHPC library and a collection of fine-tuned AI assistants created by the ChatHPC library for the specific tasks that constitute the essential parts of the HPC software stack. The creation of a new AI assistant in ChatHPC consists of a three-step iterative process: **(1) fine-tuning, (2) testing, and (3) refinement.** The **fine-tuning** process creates a parameter-efficient adapter for each AI assistant using the *training data* supervised by experts and a JavaScript Object Notation (JSON) format file with a set of prompt-context-answer tuples and data about the capabilities to be created. In our case, this data usually occupies only a few KB of memory. The *adapters* are small and trainable modules added to the base LLM to enable lightweight and efficient fine-tuning for specific tasks, thereby preserving the original model knowledge while adapting to new scenarios. Each AI assistant consumes about 100 MB of memory. The Base Model (Code Llama 7 billion [13 GB] in our case) does not need to be replicated. This is a scalable approach that allows the HPC community to work on multiple AI assistants simultaneously with relatively modest resources. **Testing** focuses on checking if the model is built correctly according to its design specifications and ensures that the model actually meets the intended user needs and performs well. This occurs after fine-tuning when a functional AI assistant is available for testing using *testing data* that is different from that used during fine-tuning (training data). The testing data may contain prompts about capabilities not related to the capabilities created **Refinements** are made after testing when *learning gaps* are identified. This expert-guided refinement consists of complementing the training data with additional information called *refinement data*, which covers the deficiencies found during validation for a new fine-tuning process. This iterative process allows AI assistants to dynamically learn and adapt to complex tasks over time as more learning gaps are identified. Although a user can interact with each AI assistant (adapters) individually, these adapters can also be merged with the base model's weights, if necessary, to create a single LLM model file. The ChatHPC library relies on well-known, robust, efficient, and widely used tools and languages, including Meta Code Llama LLM, the LoRA (low-rank adaptation) tool used for fine-tuning pretrained LLMs, and the open-source PyTorch Python framework used for building machine learning models. We also implemented a Python command line interface library in ChatHPC for ease of use. Specifically, this work offers the following:

1. A lightweight software infrastructure (i.e., the ChatHPC library) that enables HPC experts to efficiently create and supervise trustworthy AI capabilities for critical HPC software tasks, thereby lowering the barrier to AI integration in HPC systems while maintaining reasonable computational requirements.

2. A demonstration and evaluation of ChatHPC on important components of the HPC software stack, including programming systems (e.g., Kokkos, IRIS, SYCL, OpenMP, CUDA), scientific libraries (e.g., MAGMA), tools (e.g., TAU), and I/O (e.g., ADIOS2). We also show the challenges for the unique patterns and characteristics of each product.

KEYWORDS

Large Language Models, Productivity, Trustworthiness, High Performance Computing.

ACM Reference Format:
Pedro Valero-Lara, Aaron R. Young, Mohammad Alaul Haque Monil, Swaroop Pophale, Zheming Jin, Jeffrey S. Vetter, Keita Teranishi, William F. Godoy. 2025. ChatHPC: Building the Foundations for a Productive and Trustworthy AI-Assisted HPC Ecosystem. In *The International Conference on High Performance Computing, Network, Storage, and Analysis (SC 2025), November 17–22, 2025, St.Louis, MO, USA.* ACM, New York, NY, USA, 1 page. https://doi.org/10.XXXX/XXXX.XXXX

This manuscript has been authored by UT-Battelle LLC under contract DE-AC05-00OR22725 with the US Department of Energy (DOE). The publisher acknowledges the US government license to provide public access under the DOE Public Access Plan (https://energy.gov/downloads/doe-public-access-plan).

SC 2025, November 17–22, 2025, St. Louis, MO, USA

ACM ISBN XXXXX…$15.00
https://doi.org/10.XXXX/XXXX.XXXX