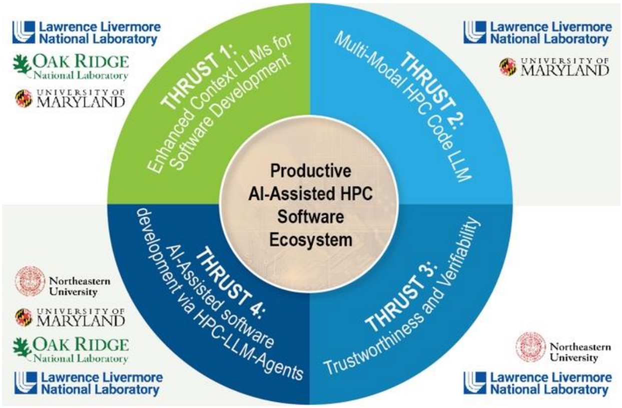
Ellora: Productive AI-Assisted HPC Software Ecosystem

Lawrence Livermore National Laboratory (LLNL), PI: Harshitha Menon (POC)
Oak Ridge National Laboratory (ORNL), Co-PI: William Godoy
University of Maryland (UMD), Co-PIs: Abhinav Bhatele and Tom Goldstein
Northeastern University (NU), Co-PIs: Arjun Guha and David Bau



Large have impressive performance on Code Generation

- Use Cases of LLMs in Code Generation
  - Code Completion
  - Refactoring
  - Document Generation
- Benefits of LLMs in Code Generation
  - Faster Development
  - Error Detection
  - Language Support

GitHub Copilot

code llama

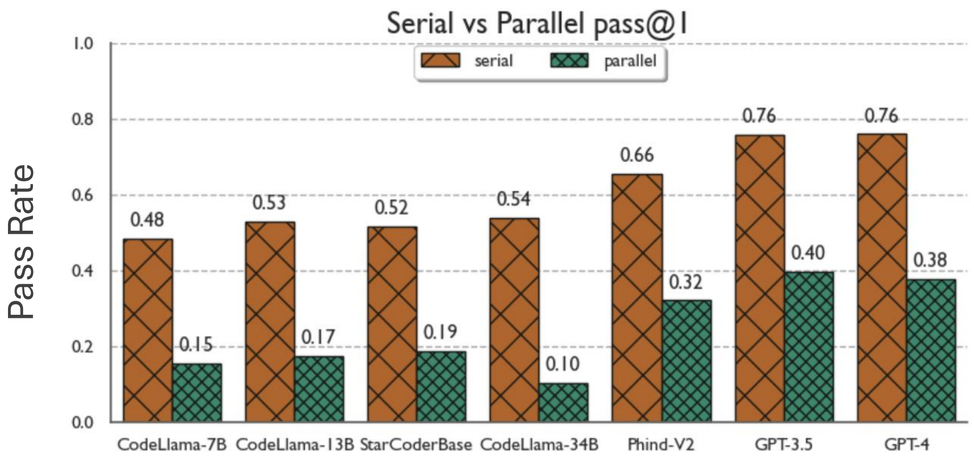
AlphaCode DeepMind

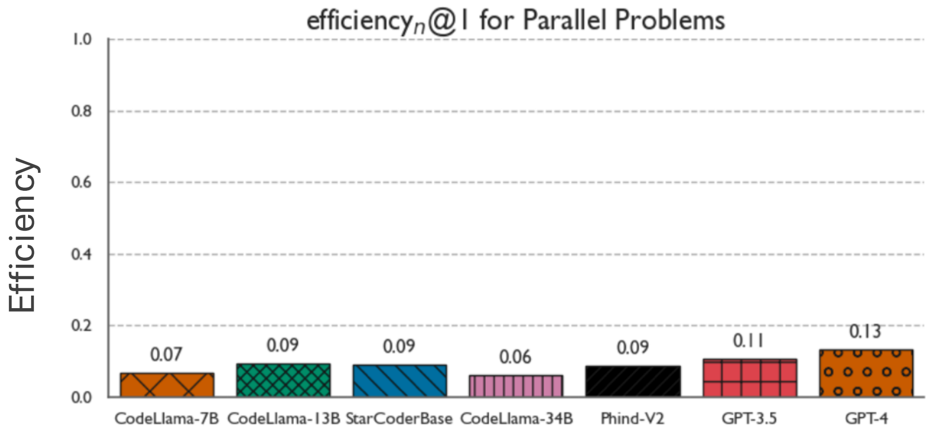
StarCoder

Codex

PolyCoder

Can Large Language Models Write Parallel Code?





Co-PI Bhatele's work — Nichols et al. 2024

- Parallel programs are more complicated
- LLMs process source code primarily as text and predicting next token conditioned on prompt and lack knowledge about the intricate details of these parallel programming model and don't consider structural aspect of code (control flow and data flow)
- LLMs struggle to generate parallel code
- Parallel code generated by LLMs has poor parallel speedup and efficiency

LLMs struggle to generate HPC code.

Ellora aims to revolutionize HPC software development

Improve LLMs' Effectiveness in HPC Domain
Enhance state-of-the-art LLMs to support large and relevant context and use data from multiple modalities for improved performance in parallel code generation.

Trustworthy and Verifiable LLM
Develop techniques to predict and explain model errors. Design models and techniques that enable explainability.

Enhance Productivity and Software Sustainability
Boost developer productivity through specialized AI-driven tools and frameworks for HPC, which will support DOE's extensive investments in ECP software ecosystem.

Thrust 1

Advance contextual capabilities of LLMs to improve LLMs' effectiveness and reduce hallucinations

Enhanced Contextual Abilities

Large Context Inference: Develop novel, efficient techniques to expand LLMs' context window size to provide access to more contextual information.

Knowledge Graph for Contextual Information: Build a knowledge graph capturing associations among code entities to provide relevant context.

Domain-Specific Retrieval Models: Implement self-supervised training for customizable retrieval models to improve the accuracy and relevance of the retrieved content.

Thrust 2

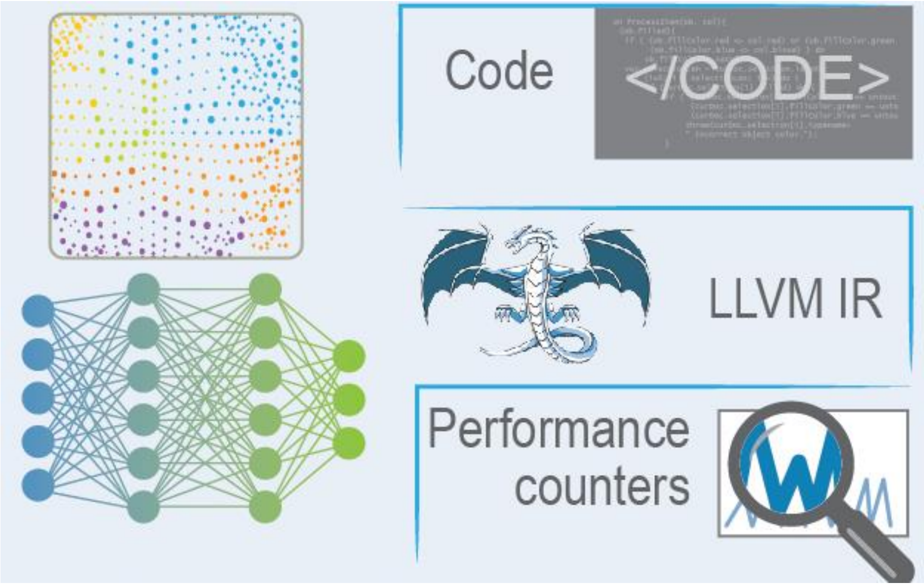
Multi-Modal LLMs for enhanced code understanding and performance

Expand Code Representations Beyond Text: Curate a multi-modal code dataset incorporating Code, LLVM IR, and performance characteristics.

Learn from Disjoint Data Across Code Modalities: Learn unified representations of many modalities using contrastive learning.

Overcome Dataset Size Challenges: Generate additional semi-synthetic data to address low-data nature of HPC domains.

Develop Multi-Modal Code LLMs: Design LLMs that learn across code modalities.



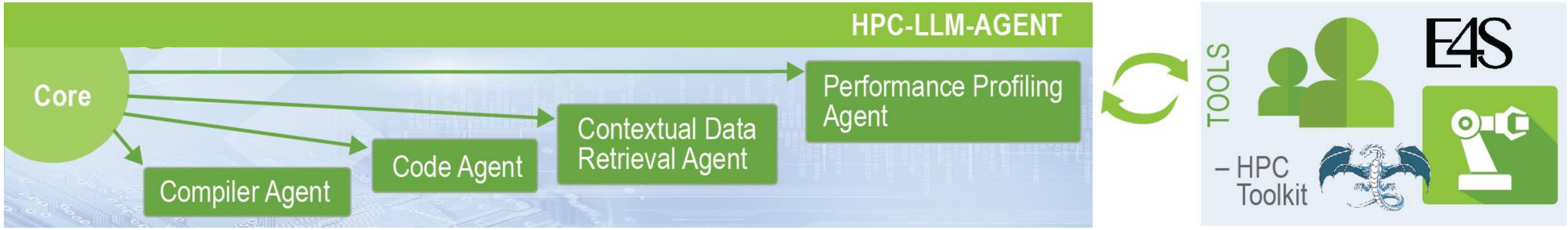
Trustworthiness and verifiability for reliable and transparent models

Attribution Methods:
Identify tasks performed by LLMs and linking model predictions to training data for improved transparency.

Model Error Prediction:
Develop techniques to predict and explain errors by analyzing LLMs to detect mispredictions.

Unlearning Techniques:
Apply unlearning methods to edit LLMs to remove erroneous behavior.

AI-Assisted software development via HPC-LLM-Agents



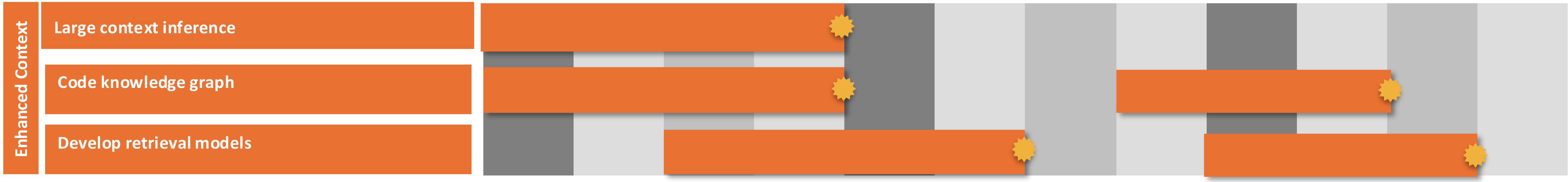
Generalize LLM agents to use HPC tools: Identify tasks performed by LLMs and linking model predictions to training data for improved transparency.

Train agents on tool-use sequence: Develop techniques to predict and explain errors by analyzing LLMs to detect mispredictions.

Enable reliable human-agent collaboration: Apply unlearning methods to edit LLMs to remove erroneous behavior.

Thrust 1

Our goal is to improve LLMs' effectiveness and reduce hallucinations by providing relevant context and enabling large context size



Tasks

- Advance LLMs for efficient large context inference
- Develop a Code Knowledge Graph for Contextual Retrieval
- Develop Retrieval Models
- Integrate with HPC-LLM-Agents

Deliverables

- Open source models
- Code knowledge graph of E4S ecosystem
- Publication on large context inference
- Publication on using knowledge graph for context
- Publication on specialized retrieval models for E4S codebase

Progress

- Train a retrieval model at scale with trillions of self-supervised tasks
- Parse code repositories and create code knowledge graph

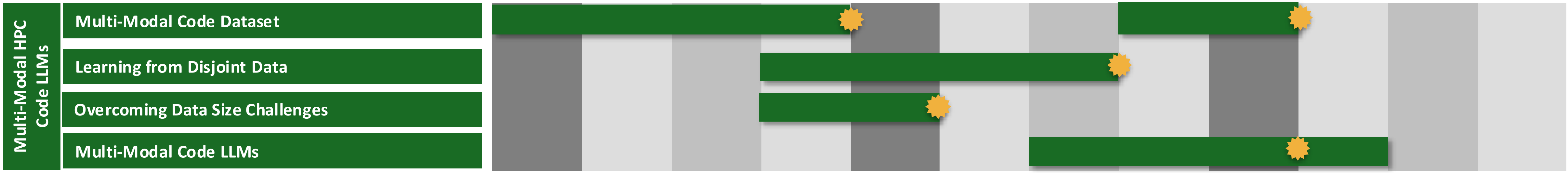
Planned Publication

- ChatHPC — fine-tune CodeLlama on a subset of E4S libraries — Q2 FY25
- Retrieval model at scale — Q3 FY25
- Code knowledge graph — Q3 FY25

Deliverable

Thrust 2

Our goal is to enable LLMs to use multiple representations of code as well as performance characteristics



Tasks

- Create Multi-Modal Source Code Data
- Learn from Disjoint Data Across Code Modalities
- Overcome Dataset Size Challenges
- Develop Multi-Modal Code LLMs
- Integrate with HPC-LLM-Agents

Deliverables

- Open source models and datasets
- Publication on Multi-Modal Code LLMs with different code representations
- Publication on low-data domain
- Publication on LLMs that support multiple code representation as well as performance characteristics

Progress

- Curated dataset of multiple code representations

Planned Publications

- Multi-Modal Code LLMs — Q2 FY25
- Dataset and model release — Q2 FY25
- LLMs for Julia — Q4 FY25

Deliverable

Thrust 3

Our goal is to develop techniques for reliable and transparent LLMs

Tasks

- Develop efficient and accurate attribution methods
- Explain and predict code LLM errors
- Edit LLMs to unlearn systematic errors

Deliverables

- Publication on attribution techniques
- Publication on unlearning bad behavior
- Release benchmarks
- Publication on how LLMs reason about code

Progress

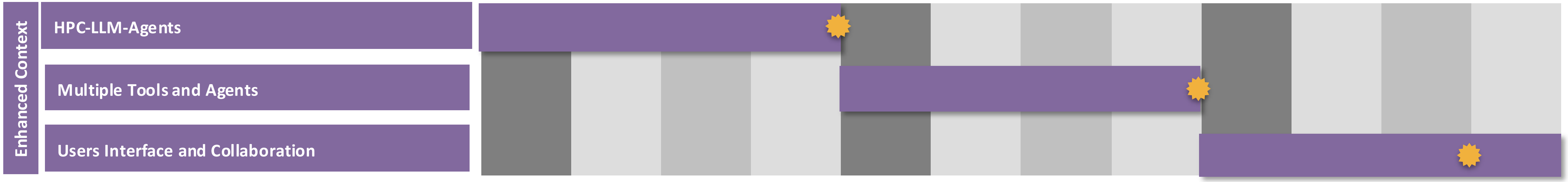
- Substance Beats Style: Why Beginning Students Fail to Code with LLMs — Submitted

<u>Planned Publication:</u>

- Understanding How CodeLLMs (Mis)Predict Types with Activation Steering — Q1 FY25
- How LLMs Reason about Pointers and Aliasing — Q3 FY25
- Attribution in Code LLMs — Q3 FY25

Deliverable

Our final thrusts enables LLMs to use HPC tools to assist in HPC software development



Applications

- Develop Specialized HPC-LLM-Agents
- Effectively Invoke Multiple Tools and Multiple Agents
- Allow Users to Collaborate with LLM Agents

Deliverables

- Publication on modular approach for HPC tasks
- Open-source specialized LLM Agents
- Publication on integration of HPC tools with LLM Agents

Progress

- Designing HPC-LLM-Agents that can generate correct and performant parallel code

<u>Planned Publication:</u>

- Modular approach for parallel code generation — Q3 FY25
- Integration of an E4S tool with LLM Agents — Q4 FY25

Deliverable

Team and Budget

Lawrence Livermore National Laboratory **$950K**

Harshitha Menon (0.35)

Giorgis Georgakoudis (0.20)

Konstantinos Parasyris (0.1)

Siu Wun Cheung (0.2)

Kshitij Bharadwaj (0.20)

Gautam Singh (New Postdoc)

Todd Gamblin (advisor)

Tal Ben-Nun (advisor)

Oak Ridge National Laboratory **$750K**

William Godoy (0.35)

Pedro Valero-Lara (0.35)

M.A.H Monil (0.35)

Aaron Young (0.2)

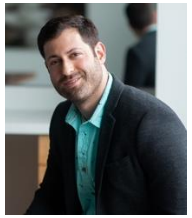
Steven Hahn (0.2)

Terry Jones (0.20)

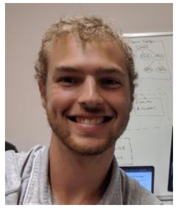
Jeff Vetter (0.1)

University of Maryland **$350K**

Abhinav Bhatele



Tom Goldstein



Daniel Nichols (PhD student)

Postdoc (TBD)

Northeastern University **$300K**

Arjun Guha

David Bau

Francesca Lucchetti (PhD student)

Alex Loftus (PhD student)

Project Challenges and Mitigation Strategies

Increasing context window size may cause memory related scaling issues: We plan to leverage Co-PI Bhatele's work on AxonNN for extreme-scale deep learning with memory optimizations.

Insufficient data for training and fine-tuning: We will generate generate synthetic data.

Reliance of HPC-LLM-Agent on various agents: We will adopt a modular approach to build Agents independently and enable individual Agents to be used.