Fine-tuning Large Language Models with HPC data

- Select three LLMs that can run on a single consumer GPU and have a variety of pre-training data: natural language, natural language code, and code.
- Train models on the HPC dataset for next token prediction.
- Compare models by their validation perplexity after one epoch of training.
- Lower perplexity means the LLM models the language’s underlying distribution better.
- All three models train to low perplexity after one epoch.
- GPT-2 is the worst due to smaller model size (1.5 billion) and a natural language only pre-training task.
- PolyCoder performs slightly better than GPT-Neo, however, their training results are comparable.

Task 2: OpenMP Pragma Labeling

- Scrape all OpenMP-labelled for loops in dataset.
- Fine-tune models to add OpenMP pragmas to for-loops.
- Compare textual equivalence and functional correctness.
- PolyCoder fine-tuned on HPC data gets up to 97% of samples correct.
- Standard PolyCoder only gets up to 67% correct.

Task 3: Performance Modeling

- State-of-the-art LLMs that can run on consumer GPUs are bad at HPC tasks.
- Fine-tuning LLMs on HPC data can improve parallel code generation.
- LLMs understand parallel data movement enough to correctly label OpenMP data clauses.
- LLMs can be used to model code performance and perform best at this task when fine-tuned on HPC data.

In the future, we will
- study larger and better LLMs as they are released.
- study how well LLMs perform at setting more complicated parallel structures.
- train LLMs to write faster code.