ISC High Performance

REINVENTING HPC

MAY 12 – 16, 2024 | HAMBURG, GERMANY

HPC-Coder: Modeling Parallel Programs using Large Language Models

Daniel Nichols[*], Aniruddha Marathe[†], Harshitha Menon[†], Todd Gamblin[†], Abhinav Bhatele[*]

[*] University of Maryland

[†] Lawrence Livermore National Laboratory

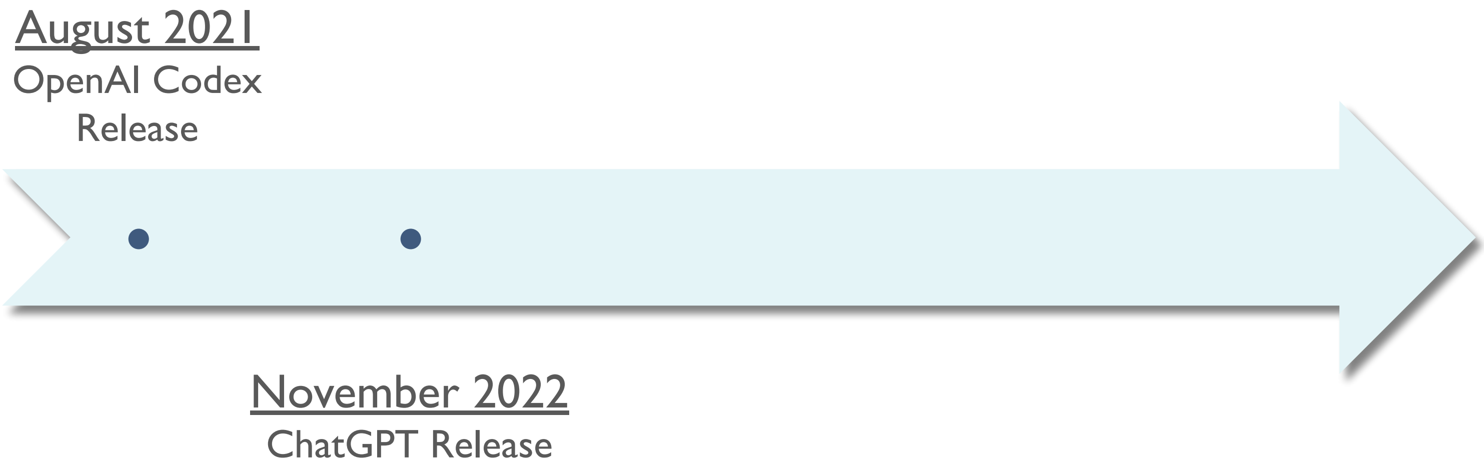
Timeline and Motivation

Timeline and Motivation

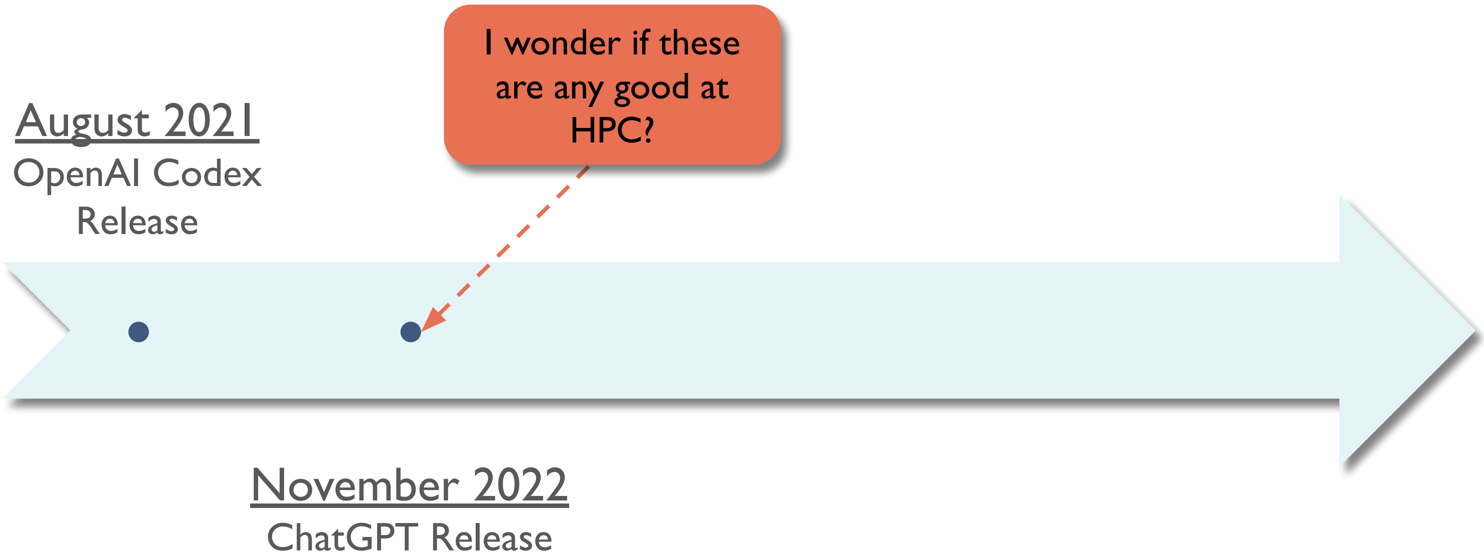
August 2021

OpenAI Codex Release

Timeline and Motivation



Timeline and Motivation



Timeline and Motivation

I wonder if these are any good at HPC?

August 2021
OpenAI Codex Release

November 2022
ChatGPT Release

June 2023
HPC-Coder First Release

Timeline and Motivation

August 2021
OpenAI Codex Release

I wonder if these are any good at HPC?

November 2022
ChatGPT Release

June 2023
HPC-Coder First Release

August 2023 - Now
CompCoder, HPC-GPT, OMP-GPT, ParEval, RLPF, Tokompiler

Timeline and Motivation

I wonder if these are any good at HPC?

August 2021
OpenAI Codex Release

November 2022
ChatGPT Release

June 2023
HPC-Coder First Release

August 2023 - Now
CompCoder, HPC-GPT, OMP-GPT, ParEval, RLPF, Tokompiler

and hopefully many more…

Code LLMs are Bad at Parallel Code

- PolyCoder
  - State-of-the-art, open-source, code LLM released in 2022
  - 2.7B parameters and outperformed OpenAI's Codex

Code LLMs are Bad at Parallel Code

- PolyCoder
  - State-of-the-art, open-sc ir
  - 2.7B parameters and out de

We prompted the LLM to write simple parallel code…

```
/*
    Use MPI to compute the average of X across ranks
    and return the result on all ranks.
    Use the default communicator.
*/
double mpiAverage(double X) {
```

Code LLMs are Bad at Parallel Code

- PolyCoder
  - State-of-the-art, open-sc
  - 2.7B parameters and out

We prompted the LLM to write simple parallel code…

and usually got incorrect output.

```c
/*
    Use MPI to compute the average of X across ranks
    and return the result on all ranks.
    Use the default communicator.
*/
double mpiAverage(double X) {
```

```c
    int rank;
    int nprocs;
    double average;
    MPI_Status status;
    MPI_Request request;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    if (rank == 0) {
        MPI_Recv(&status, 1, MPI_INT, 0, 0,
            MPI_COMM_WORLD,
                &request);
        MPI_Send(&X, 1, MPI_INT, 0, 0,
            MPI_COMM_WORLD);
        MPI_Send(&average, 1, MPI_DOUBLE, 0, 0,
                MPI_COMM_WORLD);
    }
    return average;
}
```

Code LLMs are Bad at Parallel Code

- PolyCoder
  - State-of-the-art, open-sc ... ir
  - 2.7B parameters and ou ... de

We prompted the LLM to write simple parallel code…

and usually got incorrect output.

We observed an ability to generate parallel code constructs, but not use them correctly

```
/*
    Use MPI to compute the average of X across ranks
    and return the result on all ranks.
    Use the default communicator.
*/
double mpiAverage(double X) {
```

```
    int rank;
    int nprocs;
    double average;
    MPI_Status status;
    MPI_Request request;

    MPI_Init(&argc, &argv);
    MPI_Comm_rank(MPI_COMM_WORLD, &rank);
    MPI_Comm_size(MPI_COMM_WORLD, &nprocs);

    if (rank == 0) {
        MPI_Recv(&status, 1, MPI_INT, 0, 0,
             MPI_COMM_WORLD,
                  &request);
        MPI_Send(&X, 1, MPI_INT, 0, 0,
             MPI_COMM_WORLD);
        MPI_Send(&average, 1, MPI_DOUBLE, 0, 0,
                  MPI_COMM_WORLD);
    }
    return average;
}
```

Improving LLMs for Parallel Code

Improving LLMs for Parallel Code

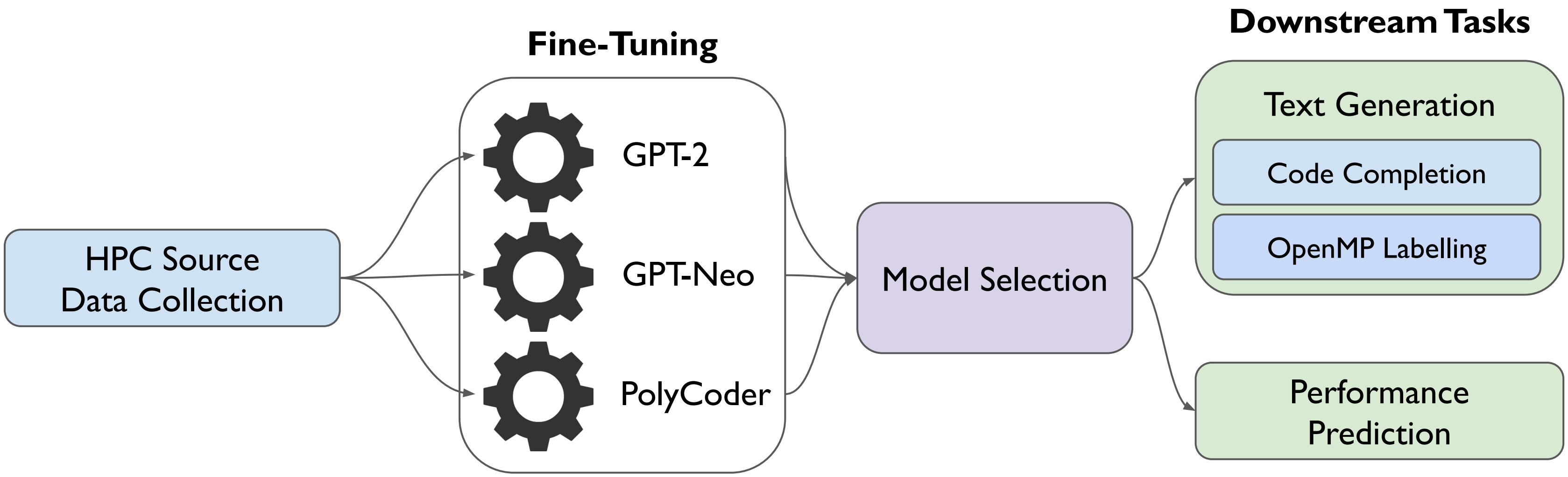
RQ 1 – *How can we train LLMs to better understand and generate parallel and HPC code?*

Improving LLMs for Parallel Code

RQ 1 – *How can we train LLMs to better understand and generate parallel and HPC code?*

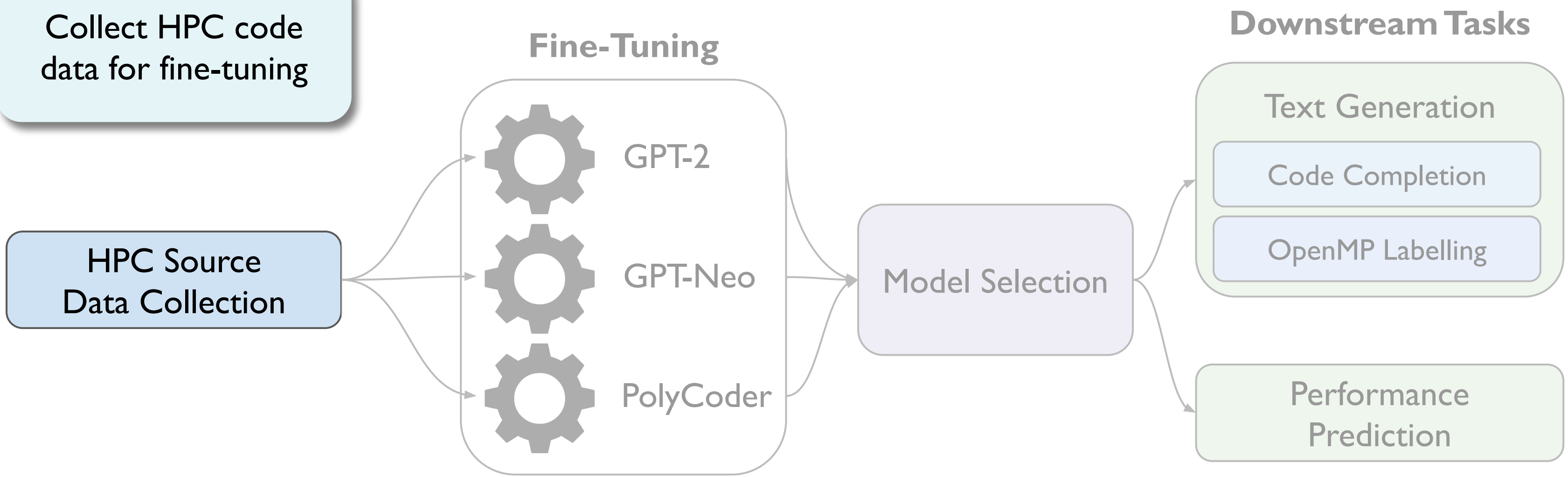
RQ 2 – *How can we effectively measure the capabilities of LLMs at modelling parallel and HPC code?*

Overview of Our Approach



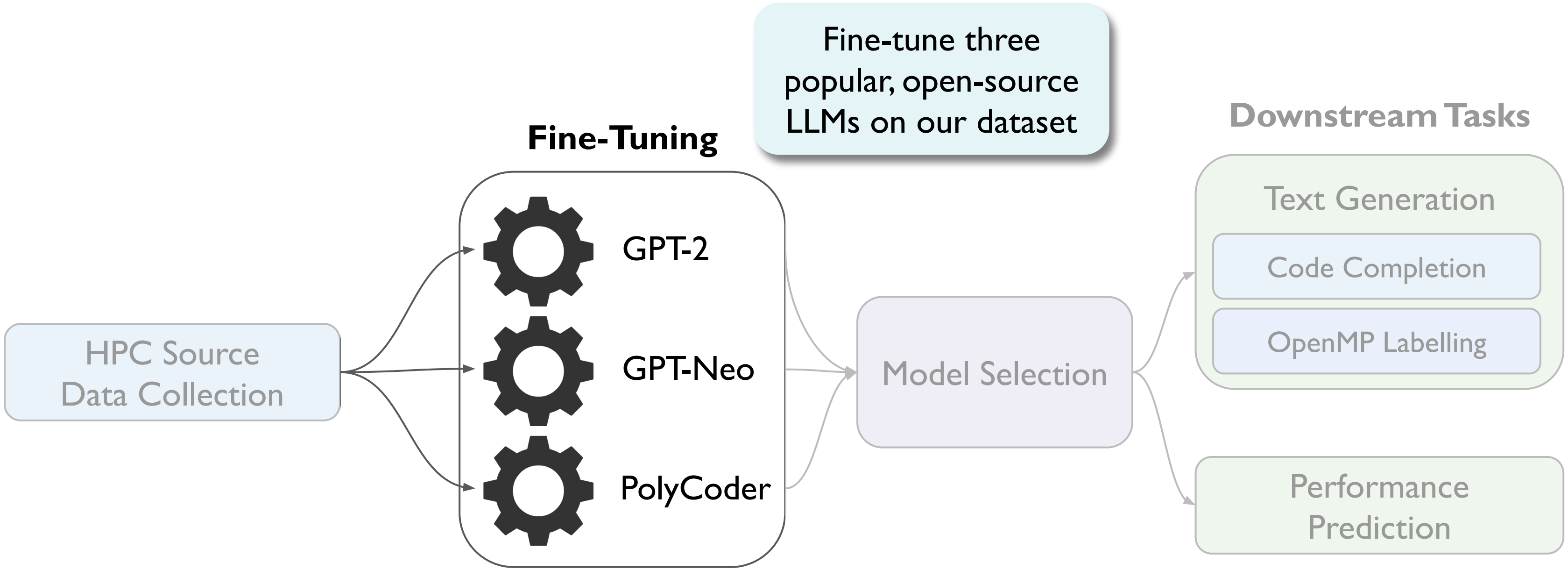
Overview of Our Approach

Collect HPC code data for fine-tuning



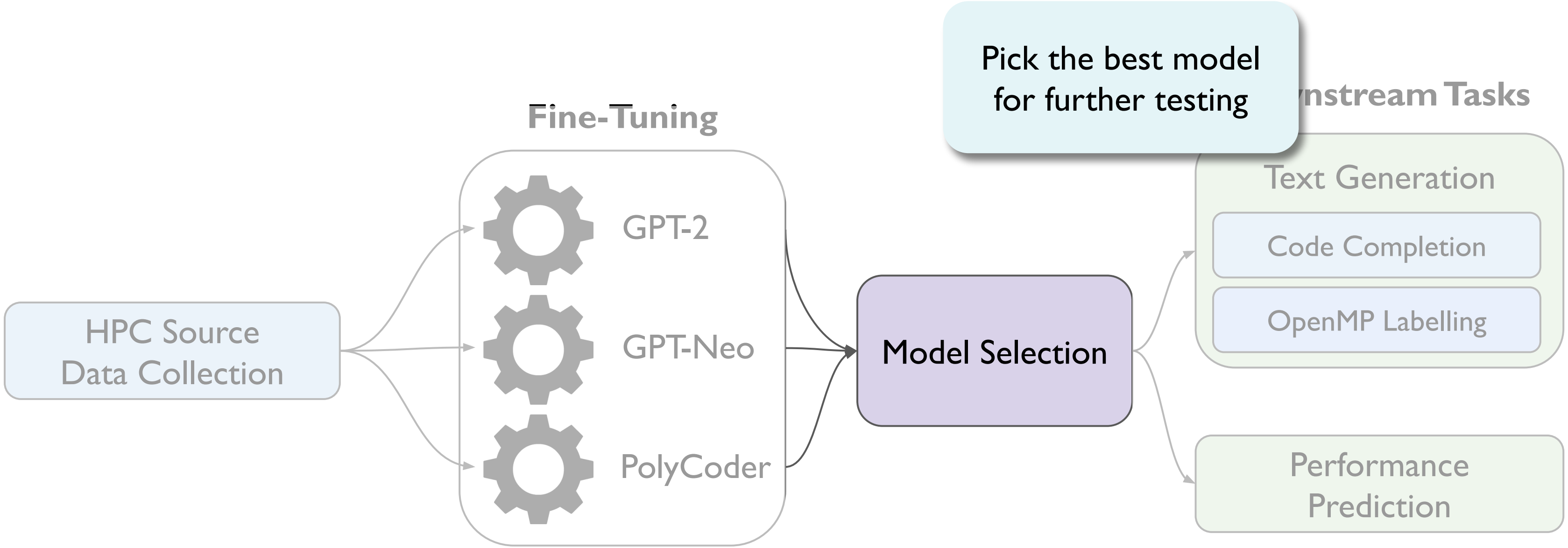
RQ 1 – *How can we train LLMs to better understand and generate parallel and HPC code?*

Overview of Our Approach



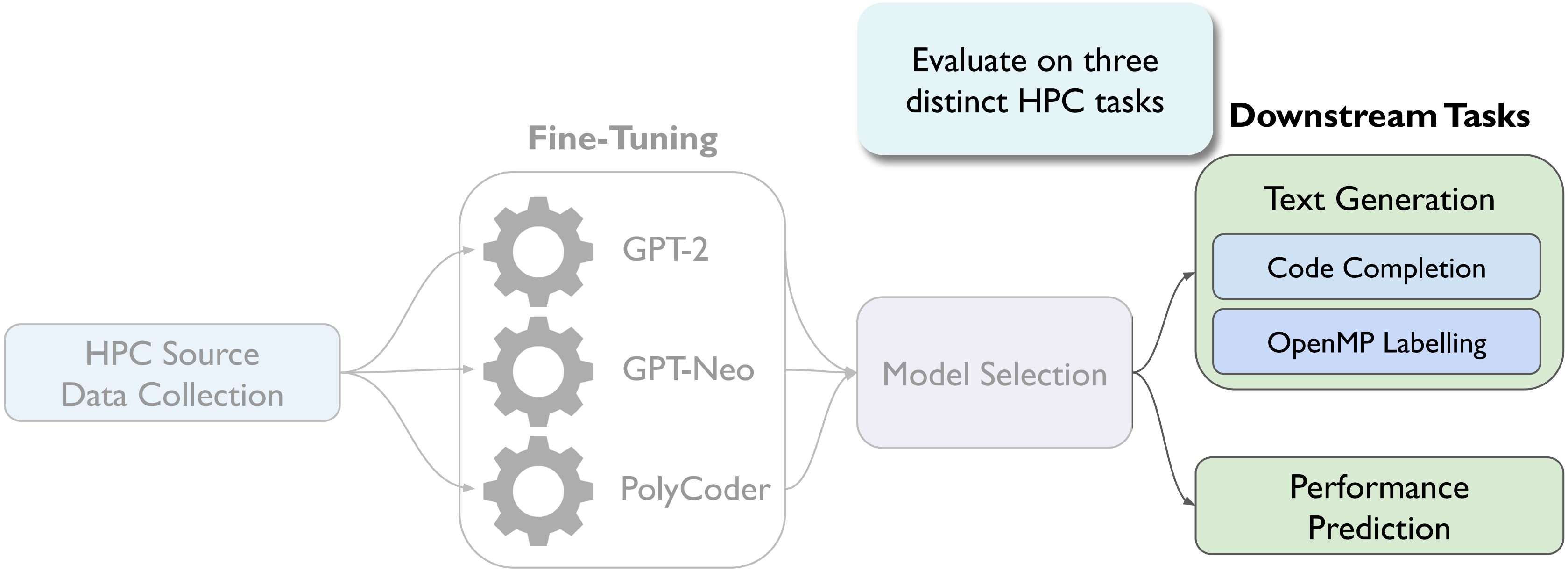
RQ 1 – *How can we train LLMs to better understand and generate parallel and HPC code?*

Overview of Our Approach



RQ 1 – *How can we train LLMs to better understand and generate parallel and HPC code?*

Overview of Our Approach



RQ 2 – *How can we effectively measure the capabilities of LLMs at modelling parallel and HPC code?*

Collecting a Parallel and HPC Code Dataset

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives
  - Large amounts of parallel and HPC code from disparate sources and projects

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives
  - Large amounts of parallel and HPC code from disparate sources and projects
  - Quality code data

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives
  - Large amounts of parallel and HPC code from disparate sources and projects
  - Quality code data
  - Clean data; no duplicate files, no auto-generated code

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives
  - Large amounts of parallel and HPC code from disparate sources and projects
  - Quality code data
  - Clean data; no duplicate files, no auto-generated code
- HPC-Coder Dataset

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives
  - Large amounts of parallel and HPC code from disparate sources and projects
  - Quality code data
  - Clean data; no duplicate files, no auto-generated code
- HPC-Coder Dataset
  - Scrape GitHub for HPC repos with ≥ 3 stars

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives
  - Large amounts of parallel and HPC code from disparate sources and projects
  - Quality code data
  - Clean data; no duplicate files, no auto-generated code
- HPC-Coder Dataset
  - Scrape GitHub for HPC repos with ≥ 3 stars
  - Filter by C/C++ source files

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives
  - Large amounts of parallel and HPC code from disparate sources and projects
  - Quality code data
  - Clean data; no duplicate files, no auto-generated code
- HPC-Coder Dataset
  - Scrape GitHub for HPC repos with ≥ 3 stars
  - Filter by C/C++ source files
  - De-duplication by SHA-256 hash

Collecting a Parallel and HPC Code Dataset

- Dataset Objectives
  - Large amounts of parallel and HPC code from disparate sources and projects
  - Quality code data
  - Clean data; no duplicate files, no auto-generated code
- HPC-Coder Dataset
  - Scrape GitHub for HPC repos with ≥ 3 stars
  - Filter by C/C++ source files
  - De-duplication by SHA-256 hash
  - Remove large (> 1MB) and small (< 15 tokens) files

Collecting a Parallel and HPC Code Dataset

Filter	# Files	# LOC	Size (GB)
None	239,469	61,585,704	2.02
Deduplicate	198,958	53,043,265	1.74
Deduplicate + remove small/large files	196,140	50,017,351	1.62

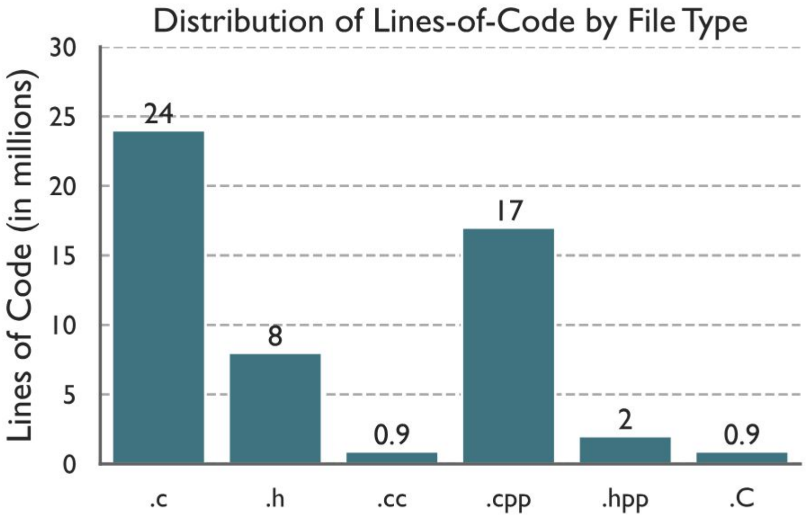
Collecting a Parallel and HPC Code Dataset

Filter	# Files	# LOC	Size (GB)
None	239,469	61,585,704	2.02
Deduplicate	198,958	53,043,265	1.74
Deduplicate + remove small/large files	196,140	50,017,351	1.62

Approximately 18% of files are removed during preprocessing.

Collecting a Parallel and HPC Code Dataset

Filter	# Files	# LOC	Size (GB)
None	239,469	61,585,704	2.02
Deduplicate	198,958	53,043,265	1.74
Deduplicate + remove small/large files	196,140	50,017,351	1.62



PSSG

UNIVERSITY OF MARYLAND

Selecting LLMs to Fine-Tune

- Focus on LLMs that fit on consumer GPUs

Selecting LLMs to Fine-Tune

- Focus on LLMs that fit on consumer GPUs
- Choose from a variety of pre-training data

Selecting LLMs to Fine-Tune

- Focus on LLMs that fit on consumer GPUs
- Choose from a variety of pre-training data
- Choose state-of-the-art LLMs in these categories (at the time)

Selecting LLMs to Fine-Tune

- Focus on LLMs that fit on consumer GPUs
- Choose from a variety of pre-training data
- Choose state-of-the-art LLMs in these categories (at the time)

Model Name	No. of Parameters	Pre-Training Data
GPT-2	1.5B	natural language
GPT-Neo	2.7B	natural language + code
PolyCoder	2.7B	code

Fine-Tuning Methodology

- Fine-tune the existing LLMs on our dataset
- Auto-regressive fine-tuning

Fine-Tuning Methodology

- Fine-tune the existing LLMs on our dataset
- Auto-regressive fine-tuning

```
int i = 0;

#pragma omp parallel ___
```

Fine-Tuning Methodology

- Fine-tune the existing LLMs on our dataset
- Auto-regressive fine-tuning

```
int i = 0;

#pragma omp parallel ___
```

tokens

PSSG

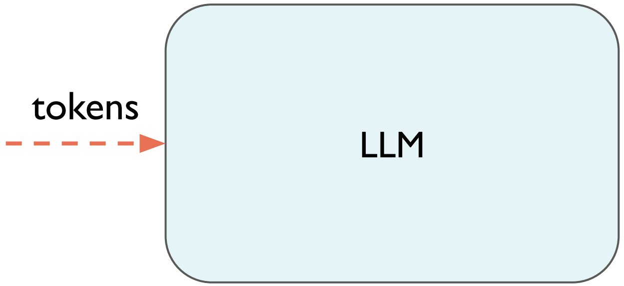
UNIVERSITY OF MARYLAND

Fine-Tuning Methodology

- Fine-tune the existing LLMs on our dataset
- Auto-regressive fine-tuning

```
int i = 0;

#pragma omp parallel ___
```

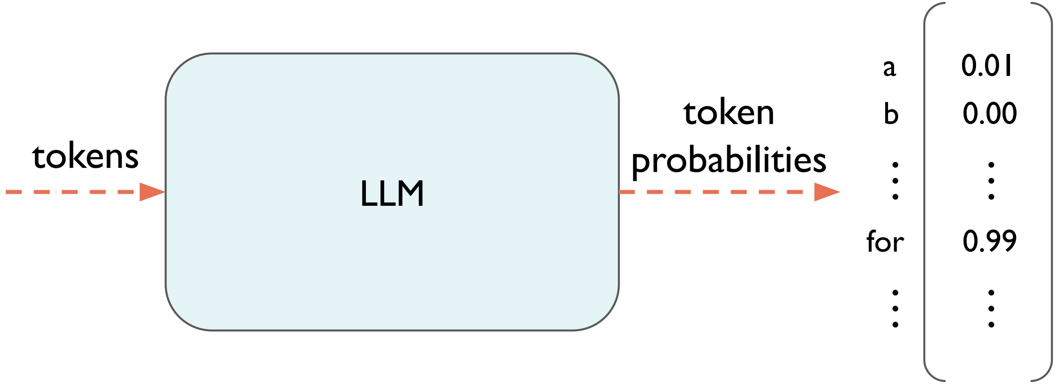


Fine-Tuning Methodology

- Fine-tune the existing LLMs on our dataset
- Auto-regressive fine-tuning

```
int i = 0;

#pragma omp parallel ___
```

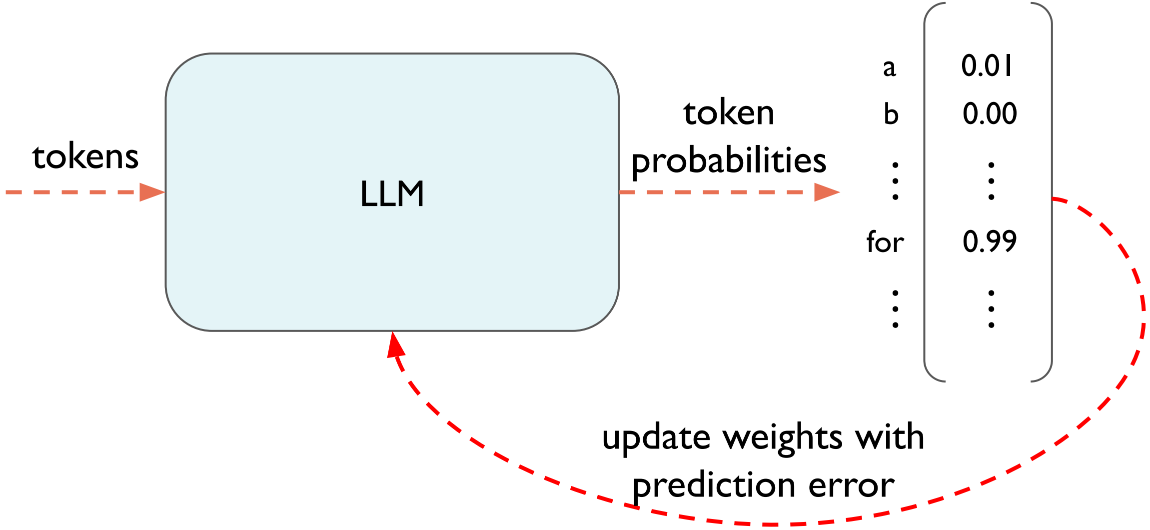


Fine-Tuning Methodology

- Fine-tune the existing LLMs on our dataset
- Auto-regressive fine-tuning

```
int i = 0;

#pragma omp parallel ___
```



Fine-Tuning Methodology

- Fine-tune the existing LLMs on our dataset
- Auto-regressive fine-tuning
- Fine-tune for 1 epoch
- Record perplexity
  - Inversely proportional to how "perplexed" the LLM is by tokens in the distribution
  - Lower is better
- Run downstream tasks every 1000 steps

Fine-Tuning Results

Model	GPT-2	GPT-Neo	PolyCoder
Final Validation Perplexity	4.47	2.23	2.24

Fine-Tuning Results

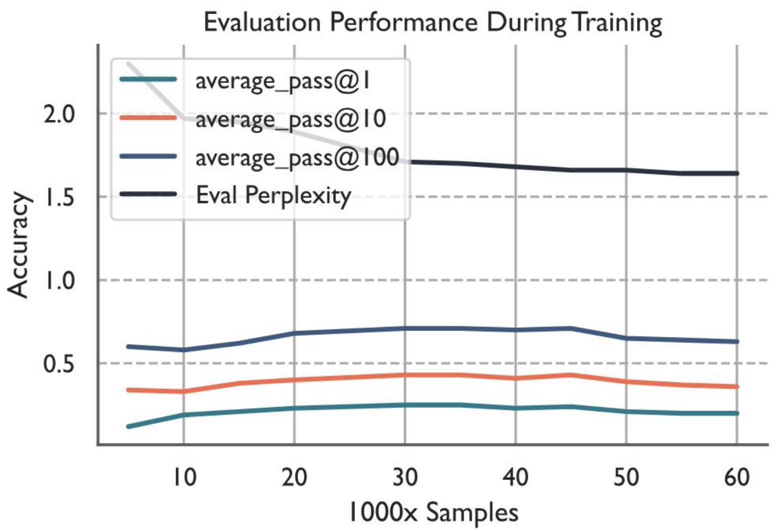
The larger models train to a lower perplexity

Model	GPT-2	GPT-Neo	PolyCoder
Final Validation Perplexity	4.47	2.23	2.24

Fine-Tuning Results

The larger models train to a lower perplexity

Model	GPT-2	GPT-Neo	PolyCoder
Final Validation Perplexity	4.47	2.23	2.24

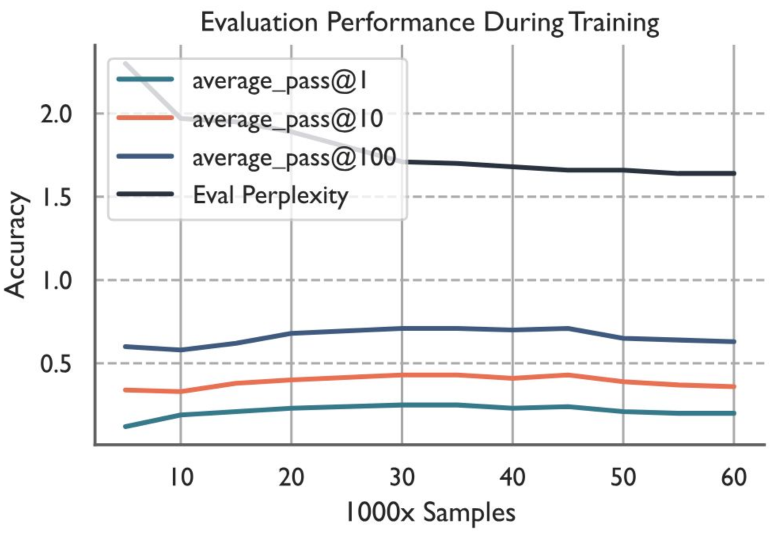


Fine-Tuning Results

The larger models train to a lower perplexity

Model	GPT-2	GPT-Neo	PolyCoder
Final Validation Perplexity	4.47	2.23	2.24

After ~45k steps the downstream performance drops, but perplexity keeps getting better



Evaluation Task 1: Code Generation

- How well can the LLMs generate code?

Evaluation Task 1: Code Generation

- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code

Evaluation Task 1: Code Generation

- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code
- Measure the pass@k

Evaluation Task 1: Code Generation

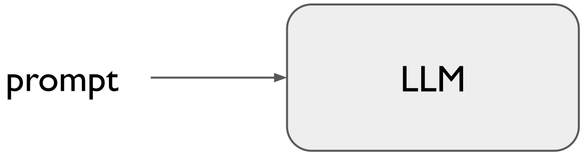
- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code
- Measure the pass@k

prompt

```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

Evaluation Task 1: Code Generation

- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code
- Measure the pass@k




```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

Evaluation Task 1: Code Generation

- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code
- Measure the pass@k


```
int sum = 0;
#pragma omp parallel for
for (int i = 0; i < N; i++) {
    sum += x[i] * y[i];
}
return sum;
```

```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

Evaluation Task 1: Code Generation

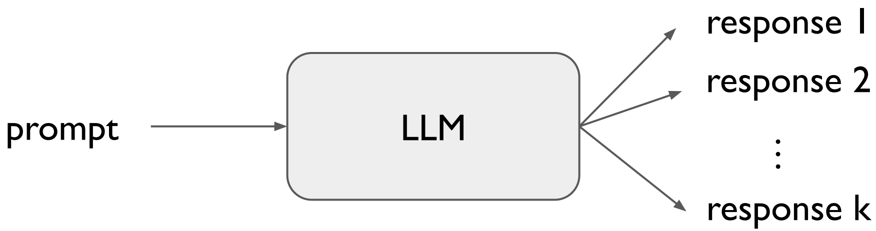
- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code
- Measure the pass@k


```
int sum = 0;
for (int i = 0; i < N; i++) {
    sum += x[i] * y[i];
}
return sum;
```

```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

Evaluation Task 1: Code Generation

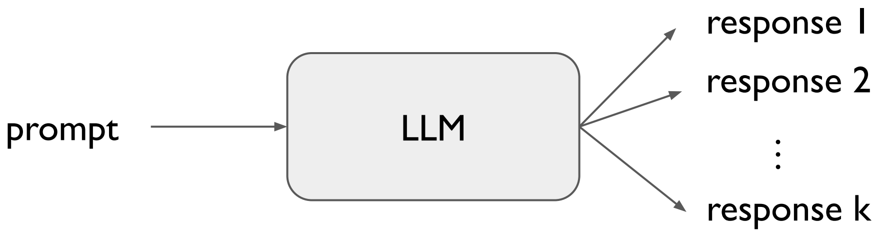
- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code
- Measure the pass@k




```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

Evaluation Task 1: Code Generation

- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code
- Measure the pass@k



What is the probability at least one of k responses is correct?

```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

Evaluation Task 1: Code Generation

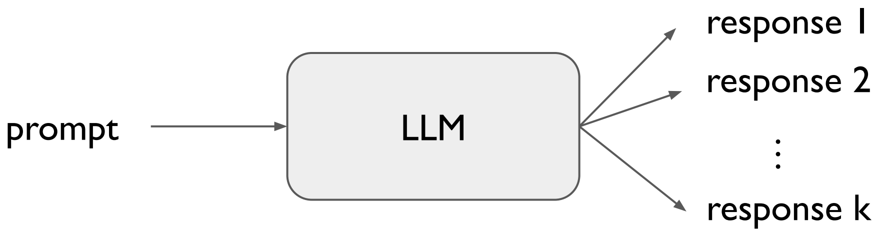
- How well can the LLMs generate code?
- 25 unique kernels spanning serial, OpenMP, and MPI code
- Measure the pass@k

Number of samples generated per prompt

$$\text{pass@}k = \frac{1}{|P|}\sum_{p \in P}\left[1 - \binom{N - c_p}{k} \Big/ \binom{N}{k}\right]$$

Set of prompts

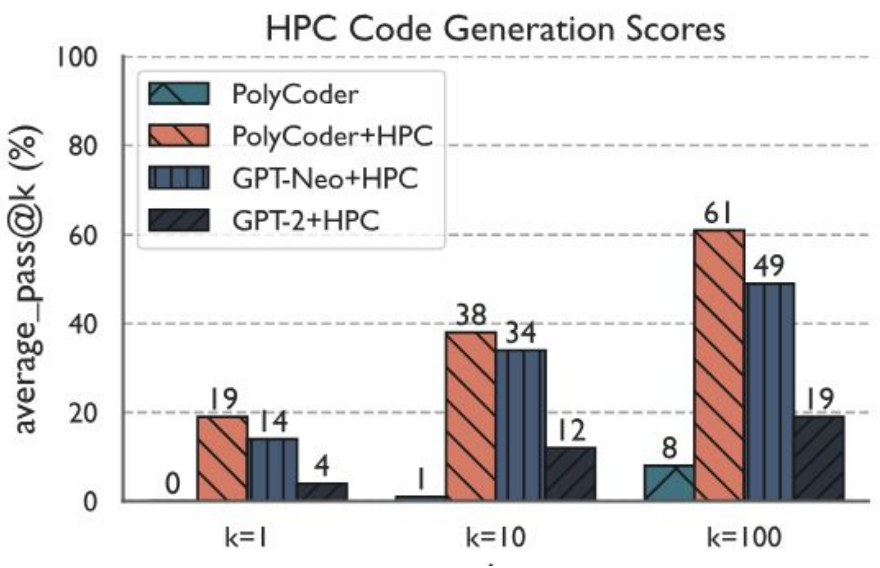
Number of correct samples for prompt p




```
/* Compute the dot product of x and y
using OpenMP */
int product(int *x, int *y, size_t N) {
```

What is the probability at least one of k responses is correct?

Evaluation Task 1: Code Generation



HPC tuned models perform much better than PolyCoder

Evaluation Task 2: OpenMP Pragma Generation

- Fine-tune LLMs to predict OpenMP pragmas
- Create dataset of 16k for loops from earlier dataset
- Fine-tune for 3 epochs

Evaluation Task 2: OpenMP Pragma Generation

- Fine-tune LLMs to predict OpenMP pragmas
- Create dataset of 16k for loops from earlier dataset
- Fine-tune for 3 epochs

```
#pragma omp parallel for
for (int i = 0; i < N; i++) {
    x[i] = foo(x[i]);
}
```

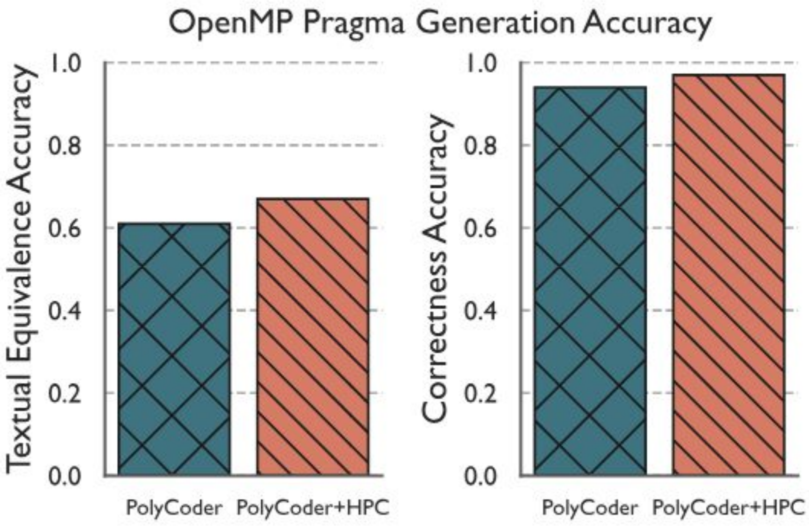
Evaluation Task 2: OpenMP Pragma Generation

- Fine-tune LLMs to predict OpenMP pragmas
- Create dataset of 16k for loops from earlier dataset
- Fine-tune for 3 epochs

```
#pragma omp parallel for
for (int i = 0; i < N; i++) {
   x[i] = foo(x[i]);
}
```

```
for (int i = 0; i < N; i++) {
   x[i] = foo(x[i]);
}
<OMP>#pragma omp parallel for<END_OMP>
```

Evaluation Task 2: OpenMP Pragma Generation

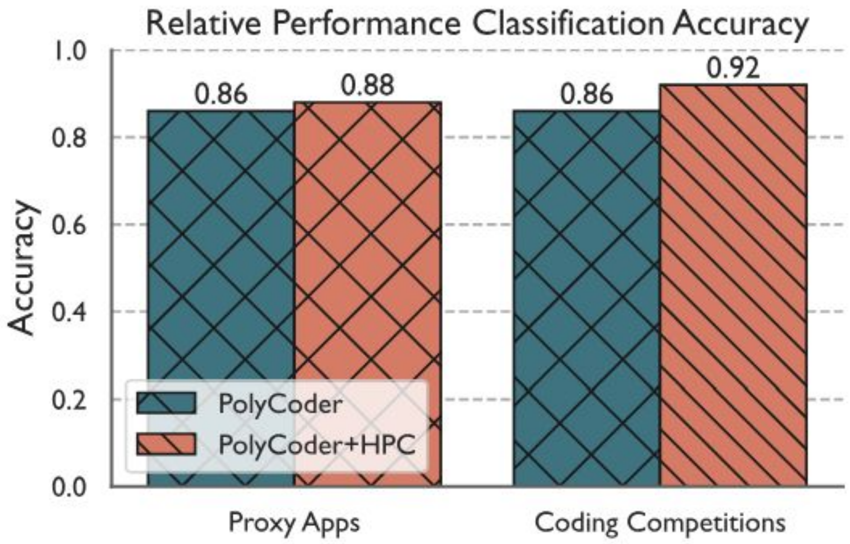


Up to 97% accuracy predicting the OpenMP pragmas.

Evaluation Task 3: Relative Performance Modeling

- Compile and run entire commit history of Kripke and Laghos
- Fine-tune LLM as classifier to predict performance degradation given commit diff
- 1 – performance improved or stayed the same; 0 – performance got worse

Evaluation Task 3: Relative Performance Modeling



Up to 92% accuracy predicting performance regressions.

Conclusion and Takeaways

- Fine-tuning can improve the performance of code LLMs on low data resource problems
- State-of-the-art LLMs are bad at parallel and HPC tasks
- We need custom evaluations on HPC and parallel tasks

Contributions and Next Steps

- A large, HPC source code dataset
- A fine-tuned HPC code LLM: HPC-Coder
- Benchmarks for evaluating LLMs on HPC tasks
- HPC-Coder-v2 in coming weeks…

"Can Large Language Models Write Parallel Code?" HPDC '24

"Performance-Aligned LLMs for Generating Fast Code" arXiv 2404.18864

dnicho@umd.edu