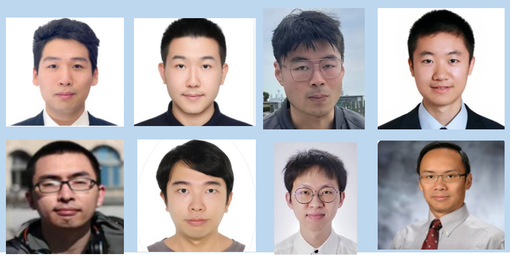
# PF-LLM: Large Language Model Hinted Hardware Prefetching

Ceyu Xu*†, Xiangfeng Sun*, Weihang Li, Chen Bai, Bangyan Wang, Mengming Li†,
Zhiyao Xie, Yuan Xie



香港科技大學
THE HONG KONG UNIVERSITY OF SCIENCE AND TECHNOLOGY

JC STEM Lab of Future Advanced Computing Technologies

## PROBLEM STATEMENT

- **Memory Wall Crisis:** Processor speed outpaces memory latency. A single DRAM access takes hundreds of cycles.
- **Hardware Limitations:** On-chip prefetchers are constrained by strict area and power budgets, restricting them to simple heuristics.
- **Ensemble Challenge:** Combining multiple prefetchers requires costly online training and struggles with dynamic phase changes.

## KEY CONTRIBUTIONS

- **PF-LLM Model:** A fine-tuned LLM that analyzes assembly code context to predict optimal prefetching strategies offline.
- **LMHint Prefetcher:** A lightweight hardware ensemble that consumes LLM-generated hints at runtime.
- **Performance:** Achieves **9.8% IPC improvement** over state-of-the-art methods on SPEC 2017.
- To our knowledge, this is the **first work** to leverage an LLM's code understanding capabilities to provide hints for a dynamic microarchitectural mechanism.

## DATASET AND PROMPT

- **Dataset:** Exhaustive testing of different prefetchers and aggressiveness levels (Degrees) for each PC using the ChampSim simulator.

| PC | PF Type | PF Degree | AMAT |
|---|---|---|---|
| 0x1234 | Stride | 1 | 25.2 |
| 0x1234 | Stride | 2 | 28.3 |
| 0x1234 | Stream | 1 | 101.0 |
| 0x1234 | Stream | 2 | 103.9 |
| 0x1235 | ... | ... | ... |
| ... | ... | ... | ... |

Best Policy → `PF Sel: stride` `PF Degree: 1` `Filter: stream`

Worst Policy

- **Prompt Template** for PF-LLM Training

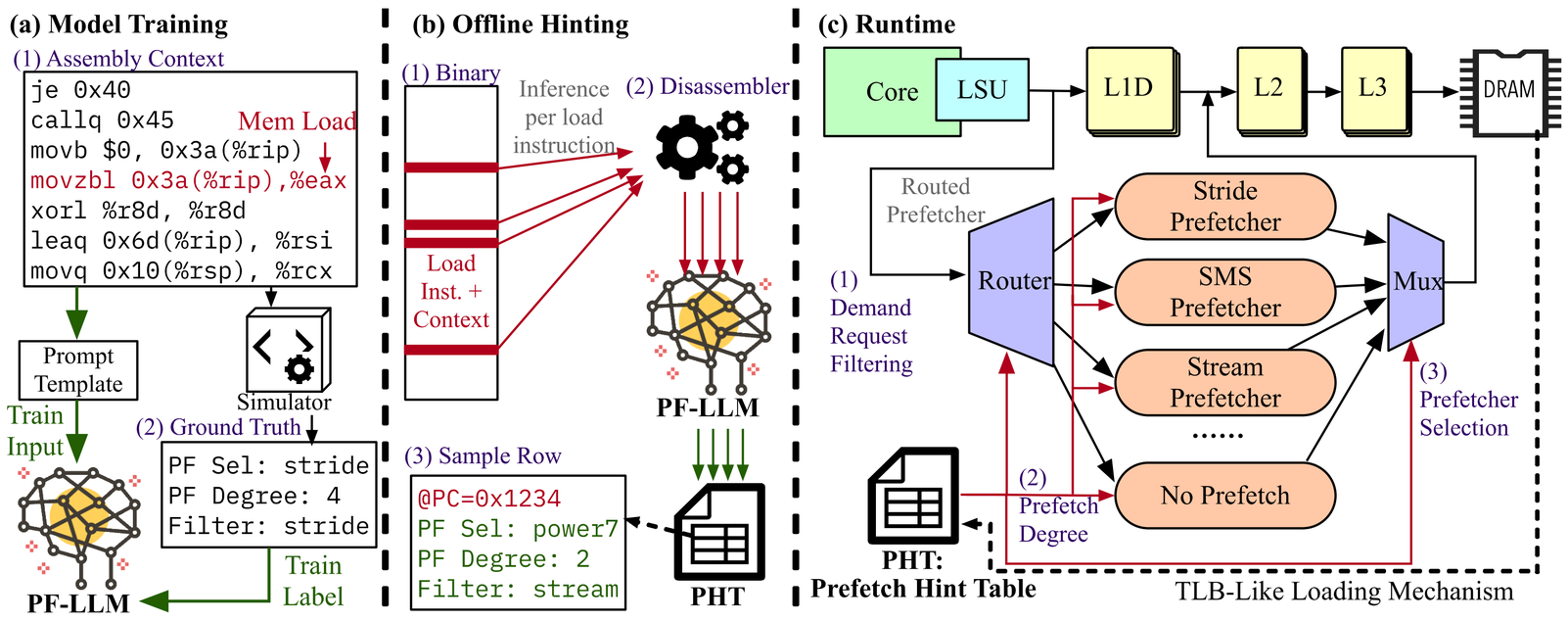
```
<im_start>system
You are a helpful assistant that generates prefetch hints for a given load instruction in an assembly code snippet. The load instruction is marked with <load> and </load>. Your output should be a JSON object with "PF Sel", "PF Degree" and "Filter" as fields.
<im_end>

<|im_start|>user
leaq    0x6dc8d(%rip), %rsi
<load> movq 0x10(%rsp), %rcx </load>
movl 0x1c(%rsp), %edx
.....
<im_end>

<|im_start|>assistant{
    "PF Sel": "stride",
    "PF Degree": 2,
    "Filter": "stream"
}<|im_end|>
```

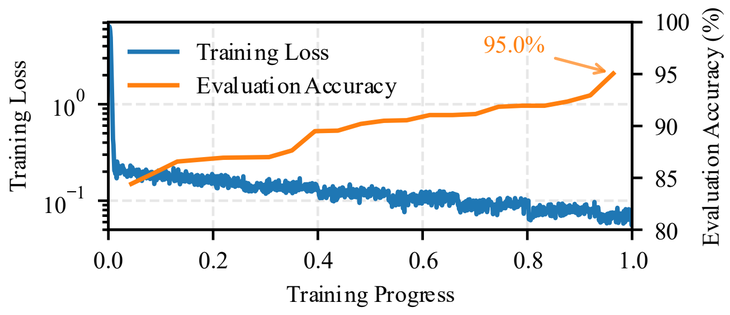
## SYSTEM OVERVIEW

**(a) Training:** Fine-tune Qwen-2.5-Coder. **(b) Offline Hinting:** Generate hints for binary. **(c) Runtime:** Hardware consumes hints.

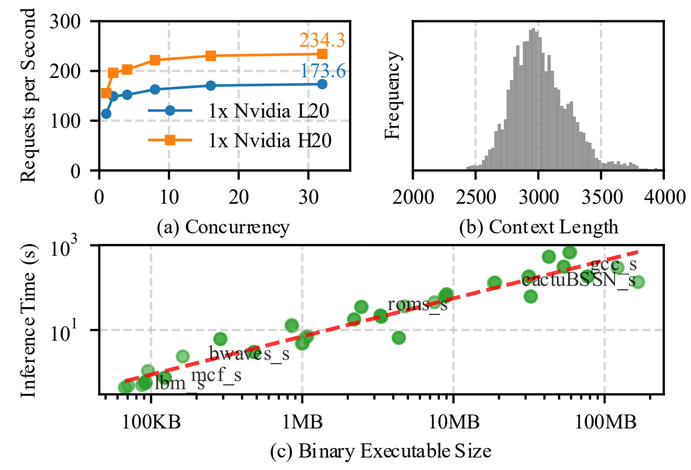



## PF-LLM TRAINING AND INFERENCE

- **Base Model:** Qwen-2.5-Coder-0.5B-Instruct
- **Input:** 257 lines of assembly code context (128 before + target load + 128 after).
- **Output:** JSON with three hint types: Prefetcher Selection, Prefetch Degree, and Demand Request Filter.
- Model achieves **95.0% prediction accuracy** on the held-out test set.




- Model interference **overhead** is low

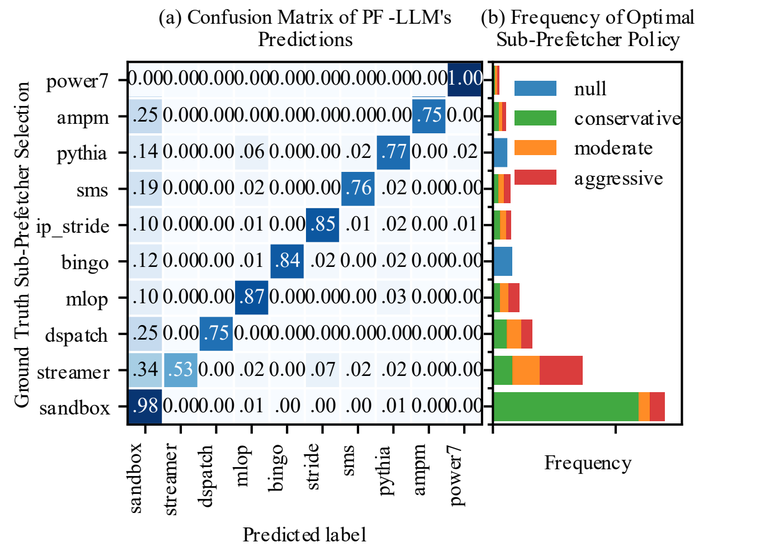



## PF-LLM PREDICTION ACCURACY

- PF-LLM aligns static assembly context with the optimal sub-prefetcher and degree for diverse memory patterns.

| Prefetcher | Target Pattern | Degree |
|---|---|---|
| Pythia | Complex-Spatial | No |
| SMS | Region-Sparse | Yes |
| Sandbox | Strided-Stream | Yes |
| ..... | ..... | ..... |

- The model achieves **near-oracle prediction accuracy,** successfully replicating the complex distribution of ground-truth prefetching policies.




## PERFORMANCE COMPARISON

- **vs. Single Prefetchers (Top):** Our framework outperforms the state-of-the-art hardware prefetcher, Pythia (MICRO'2021), **by 9.8%** across memory-intensive SPEC 2017 benchmarks. This IPC improvement stems from our per-instruction adaptive policy.
- **vs. Ensembles (Bottom):** Traditional online ensemble methods Alecto (M Li. HPCA'2025) struggle because their trial-and-error learning disrupts advanced prefetchers. By using static, offline LLM hints, LMHint achieves **an 18.9% average IPC improvement** over the best prior ensemble methods without any runtime training overhead