JC STEM Lab of
Future Advanced
Computing Technologies

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

PF-LLM: Large Language Model Hinted Hardware Prefetching

Entropy Xu
Research Assistant Professor
JC STEM FACT Lab, Director: Prof. Yuan Xie
The Hong Kong University of Science and Technology
March 25, 2026

Outline

1. Background: The Importance of Prefetching
2. Motivation:
    1. Online Learning vs. Offline Learning
    2. Why Do LLMs Work for Prefetching Hinting?
3. Method:
    1. LLM as a code semantic learner for classifying memory access patterns.
4. Results
5. Insights for Future CPU micro-architecture design.

Background

The Importance of Prefetching

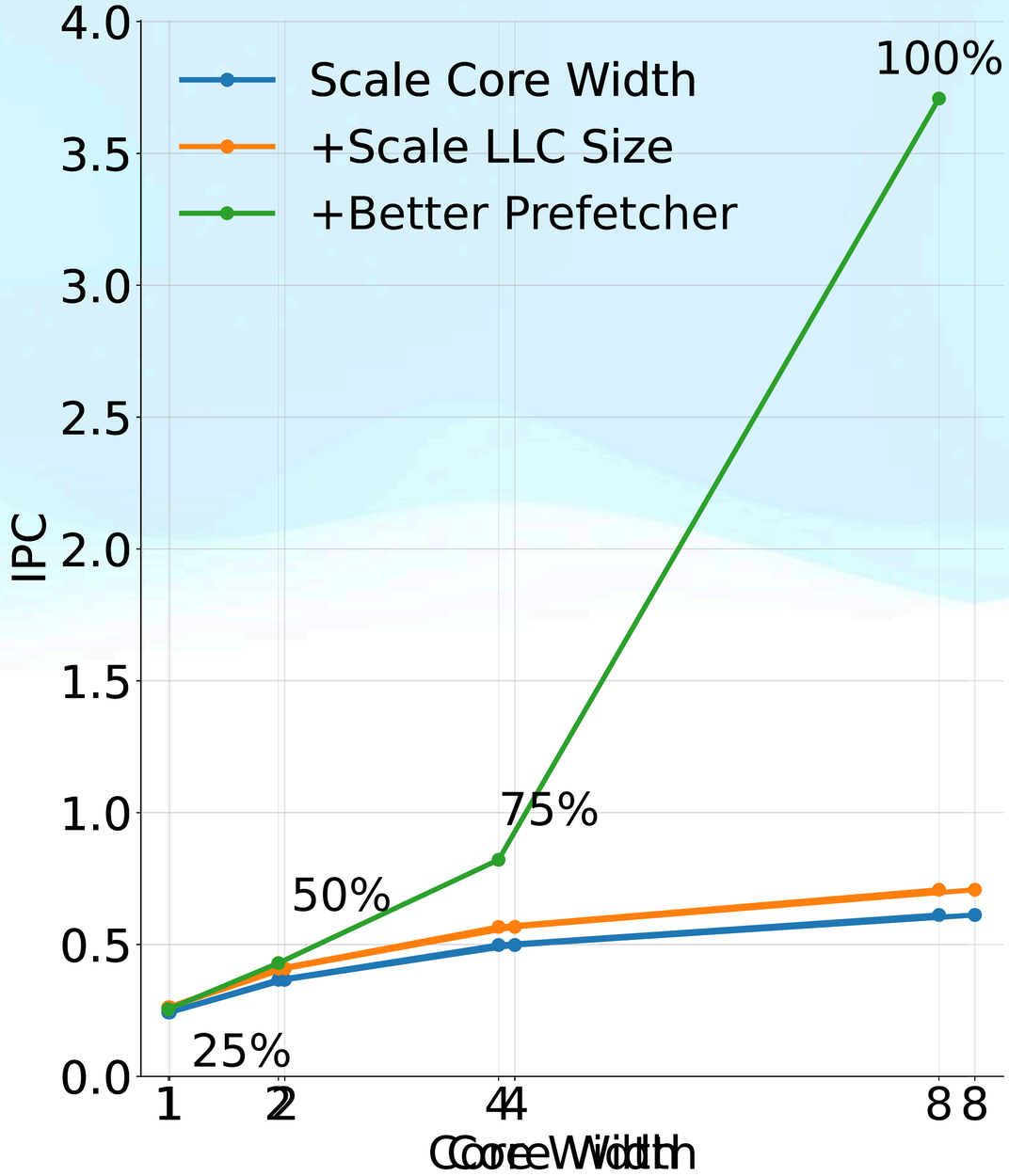
Prefetching is Effective

And Efficient

Prefetching
→
essential for single-core IPC

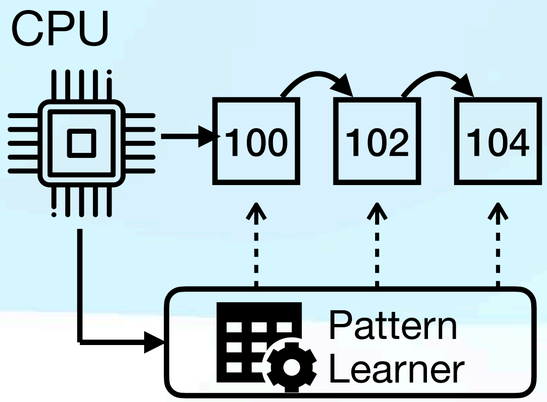
Beats spending die area on larger caches

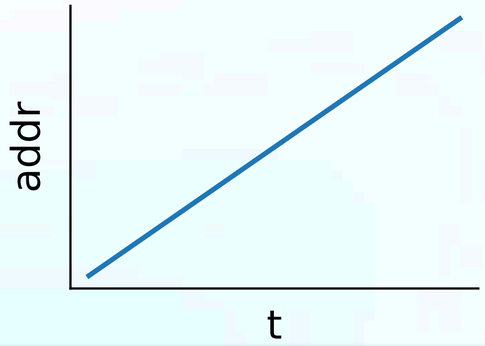
Enables wide-issue scaling



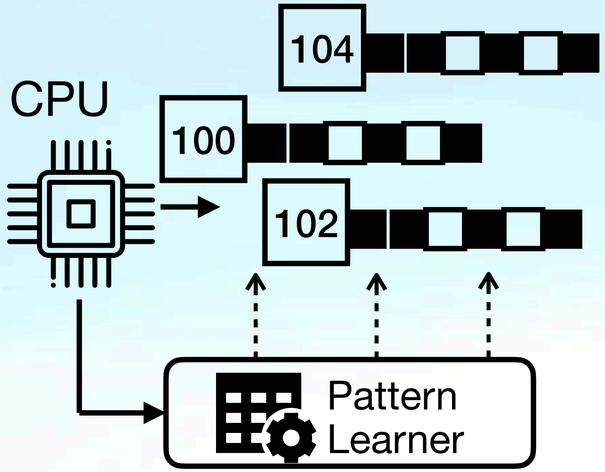
Hardware Prefetchers

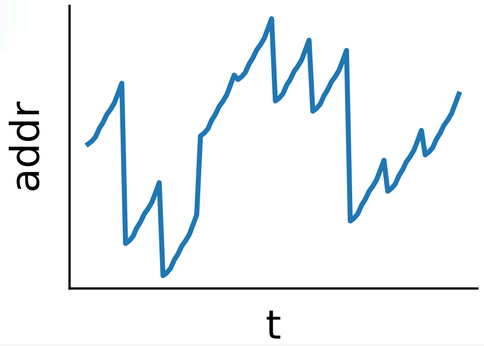
Stride/Stream Prefetcher



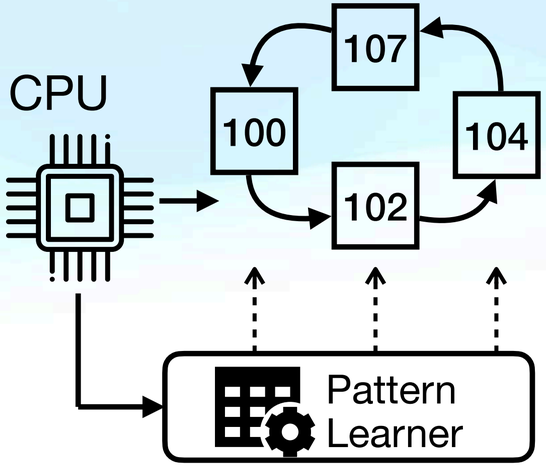


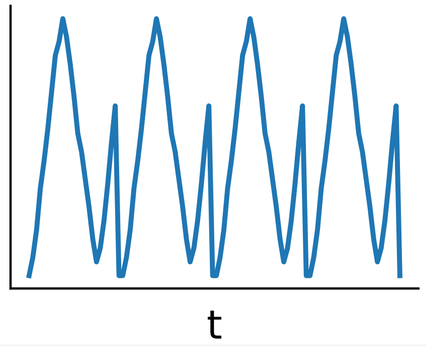
Spatial Prefetcher



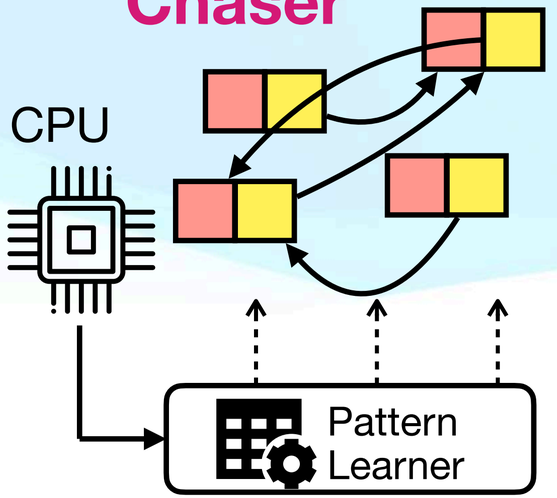


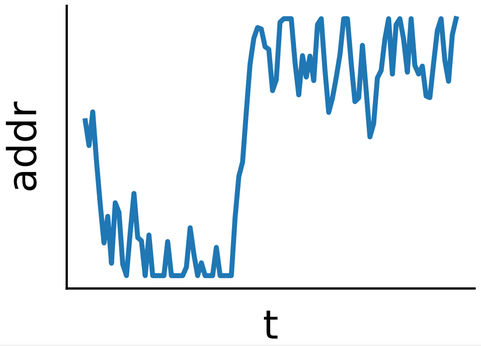
Temporal Prefetcher



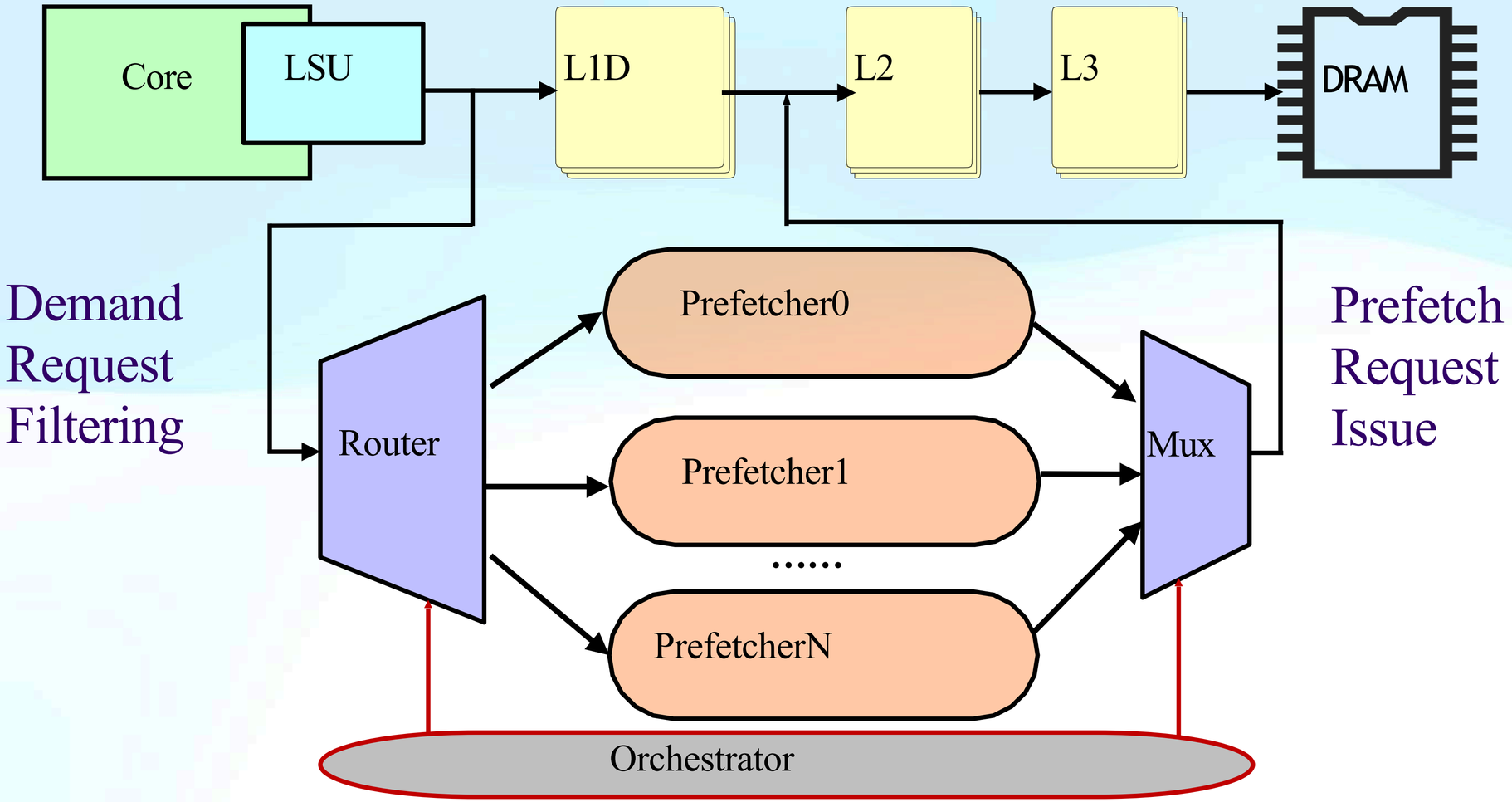


Pointer Chaser





Hardware Prefetcher Ensemble



Problems of Existing Prefetchers

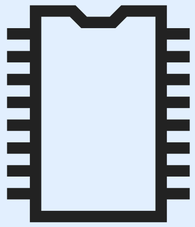
- Existing Prefetchers rely on **on-chip online learning**.

Slow Convergence

- Relies on online table-based heuristic to converge.
- Unstable and slow convergence for complex patterns.

Limited On-Chip Logic Budget

- Of course you cannot run an LLM at 3.0GHz on-chip.
- Table-based heuristics require large on-chip-storage.



Code Context Unaware

- Code context information remains unexploited for online decisions.

Motivation

Can we move the hard decision of
what, when and how to prefetch
from runtime hardware to an offline LLM-based analysis?

Human Developers Know How to Prefetch

So do LLMs?

```
typedef struct {
  uint8_t age;
  char* name;
  uint32_t country_code;
} Person;

Person people[N] = ...;
```

```
for (int i = 0; i < N; ++i){
  if (people[i].age > 20)
    printf("%d,%s,%d",
           people[i].age,
           people[i].name,
           people[i].country_code);
}
```

Initialize array of structures

Accessing an element in an array results in a **stride access pattern.**

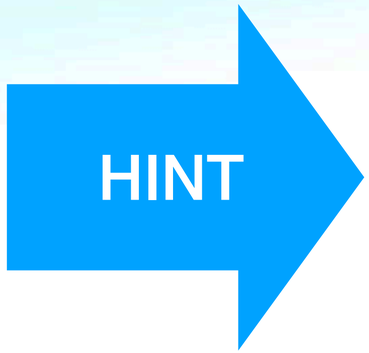
Accessing multiple elements results in a **spatial access pattern.**

Accessing the string results in a stream **access pattern.**

Main Ideas

Offline

- Fine-tune an LLM called PF-LLM to analyze code context and predict the **best prefetching policy** for each load instruction.
- Offline: expensive reasoning is allowed.
- Serves as an oracle for online decisions.



Online

- Use a lightweight hardware ensemble that **consumes** the **offline-generated** hints to perform **runtime decisions.**
- Runtime: hardware stays simple and low-latency.
- Achieve near-oracle guidance without online trial-and-error.

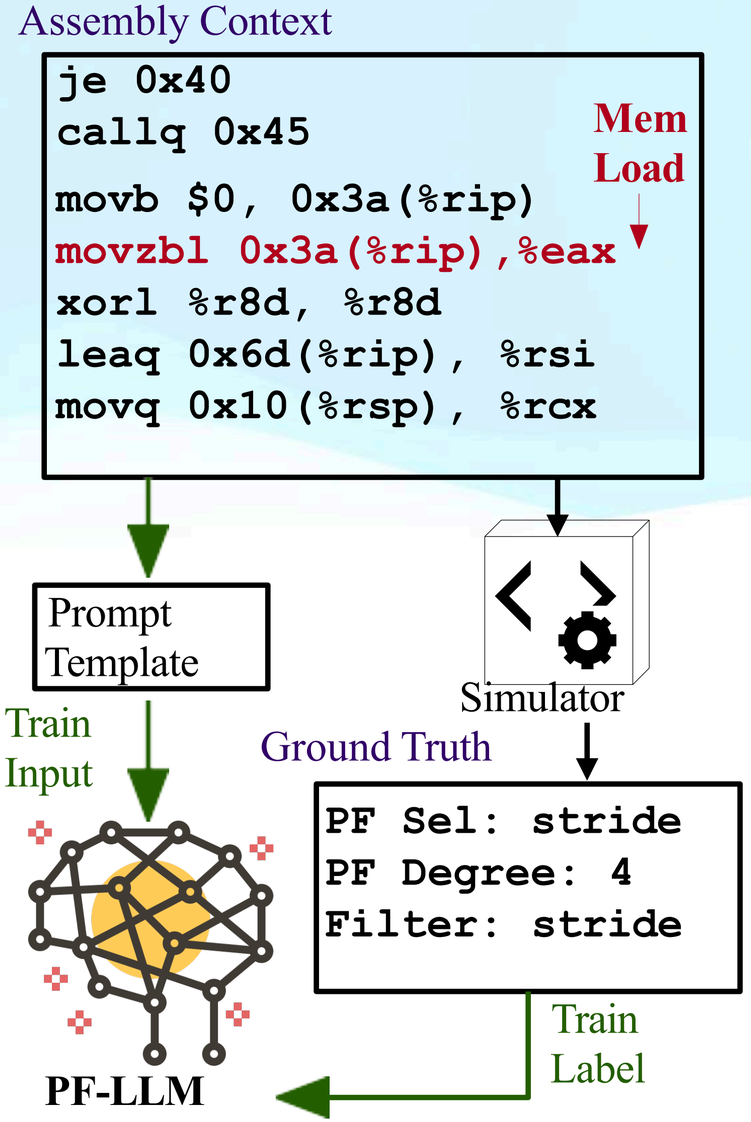
Method

Offline LLM to Hint Online Prefetching

Step 1: PF-LLM Model Training

Overview

- **Input:**
  - Assembly context formatted into a prompt (128 lines before and after)
  - Inference on each load instruction.
- **Output:**
  - Prefetch policy including:
    - Prefetcher **Selection**
    - Prefetcher Demand Request **Filter**
    - Prefetch **Degree** (aggressiveness)



Step 1: PF-LLM Model Training

Prompt Formatting

```
<|im_start|>system
You are a helpful assistant ...
<|im_end|>

<|im_start|>user
movzbl 0x3adef0(%rip), %eax
xorl %r8d, %r8d
leaq 0x6dc8d(%rip), %rsi
<load>movq 0x10(%rsp), %rcx</load>
movl 0x1c(%rsp), %edx
movq 0x3ab11d(%rip), %rdi
orl $2, %eax
<|im_end|>

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

Generic System Prompt

Assembly code as input

Input (prefill)

Output (decode)

Prefetch Policy as Label

Why Bother?

- Leverage pre-trained code-comprehension knowledge in foundation models.

Step 1: PF-LLM Model Training

Dataset Collection and Training

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	…	…	…
…	…	…	

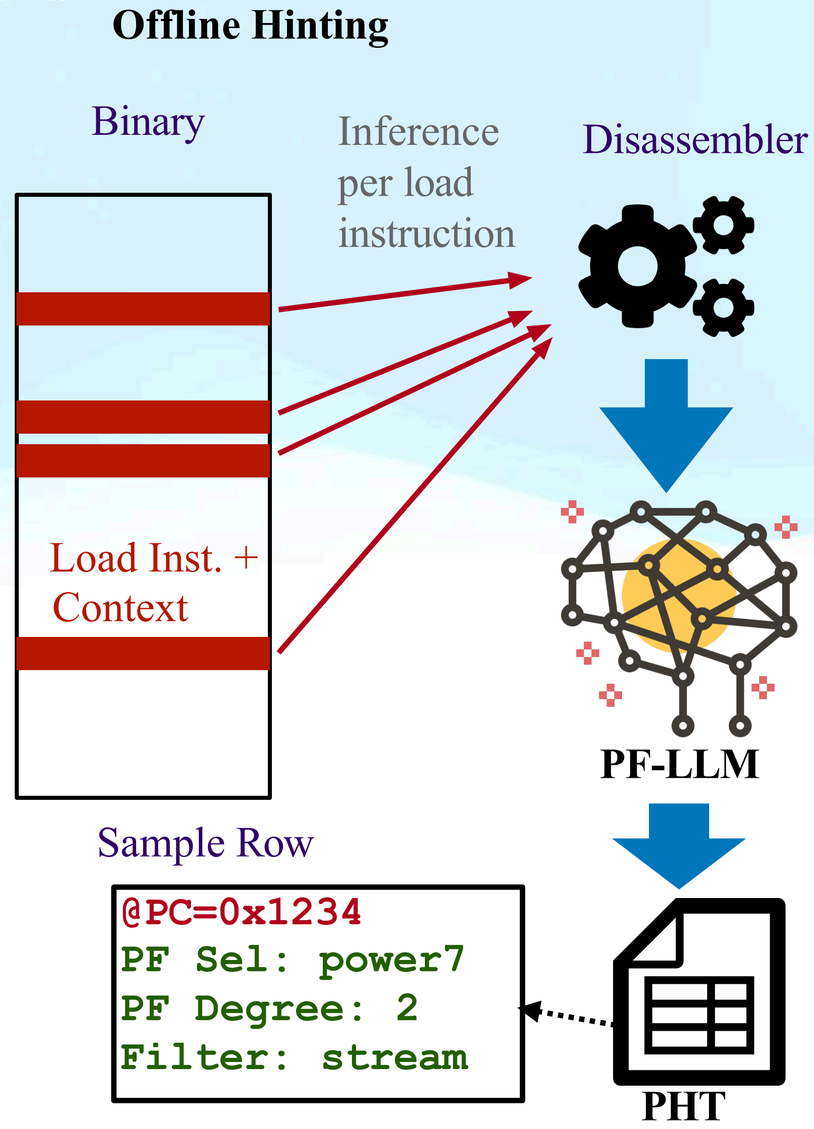
- For one benchmark in the training set, run simulations for all permutation of the prefetch policies.
- Modify ChampSim to record the AMAT of each PC with each policy.
- Select **best policy** and **worst policy** based on the AMAT.

PF Sel: stride
PF Degree: 1
Filter: stream

Step 2: PF-LLM Model Inference

Inference at compile time.

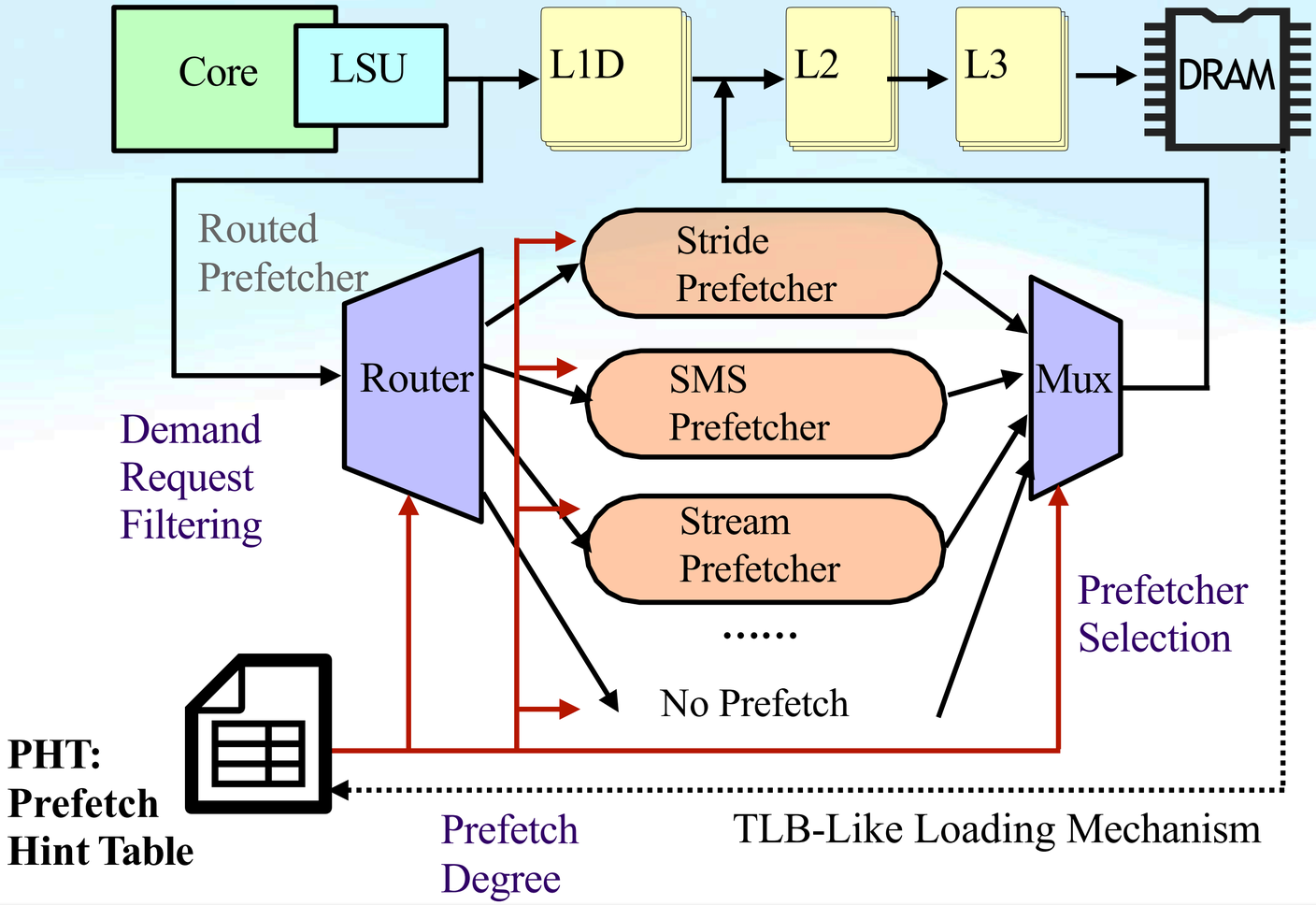
- The [Qwen-2.5-Coder-0.5B-Instruct]model is fine-tuned with our dataset.
- At compilation time, run **one model inference with the same prompt formatting for each load instruction** in the binary.
- The **per load-instruction hints** are stored in a tabular file called prefetch hint table for later use.



Step 3: Use LMHint Prefetcher at Runtime

Simple hardware but with educated HINTs.

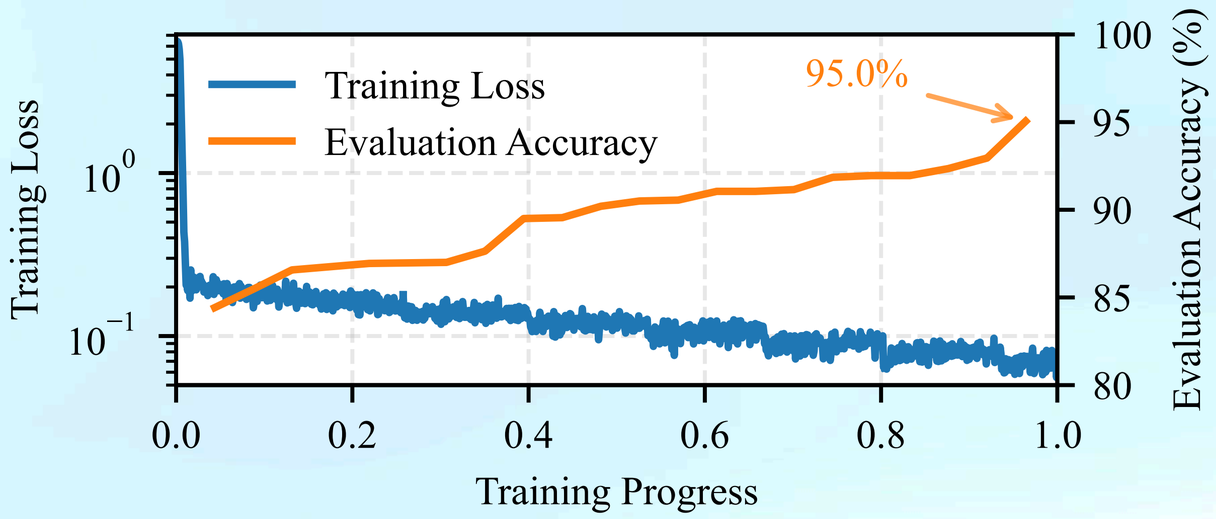
- Start with Prefetcher Ensemble Hardware
- Use a TLB-like Prefetch Hint Buffer to load hints.
- Route and Filter Prefetchers based on HINTs



Results

State-of-the-art IPC, Negligible On-chip Overhead

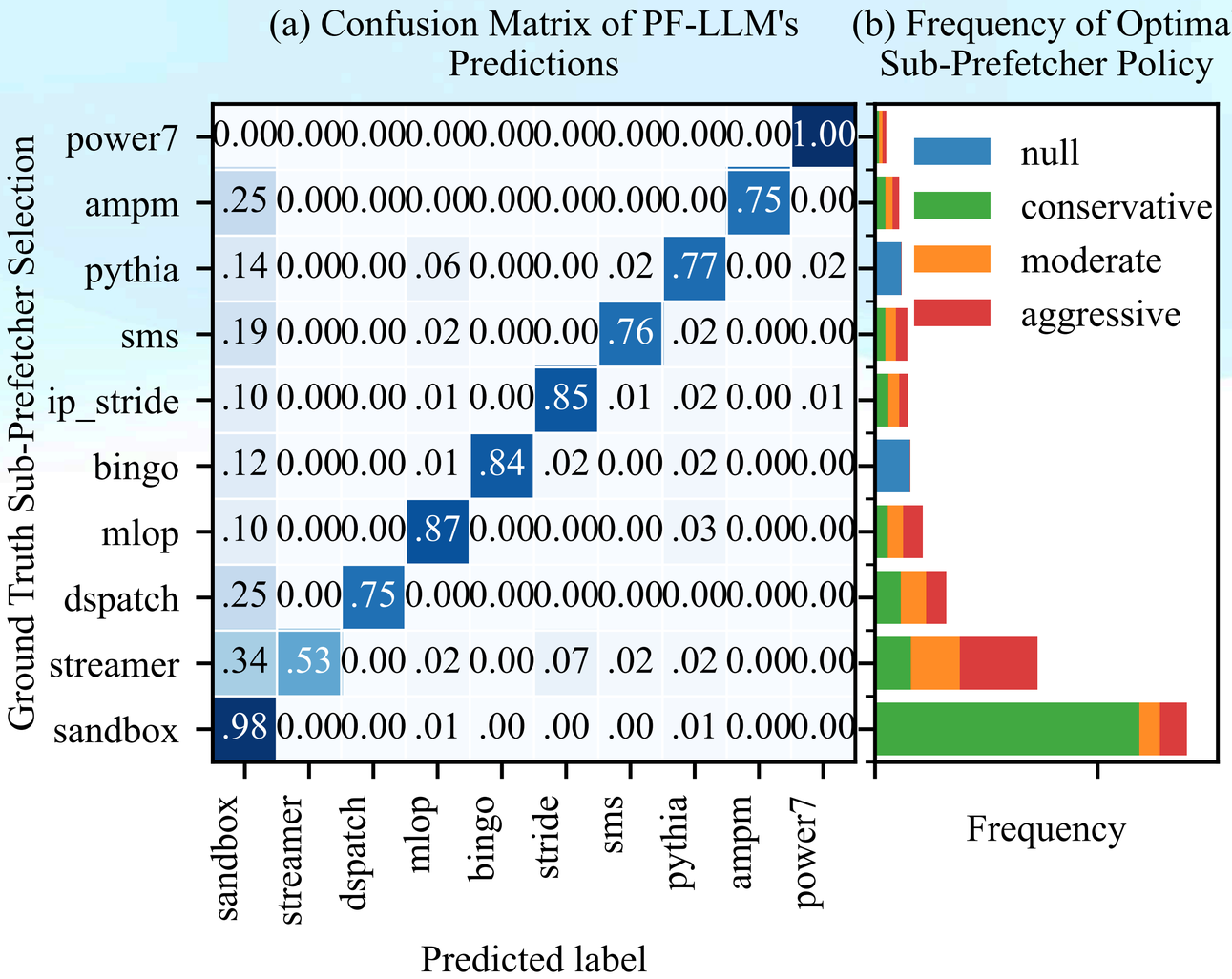
PF-LLM Training Results



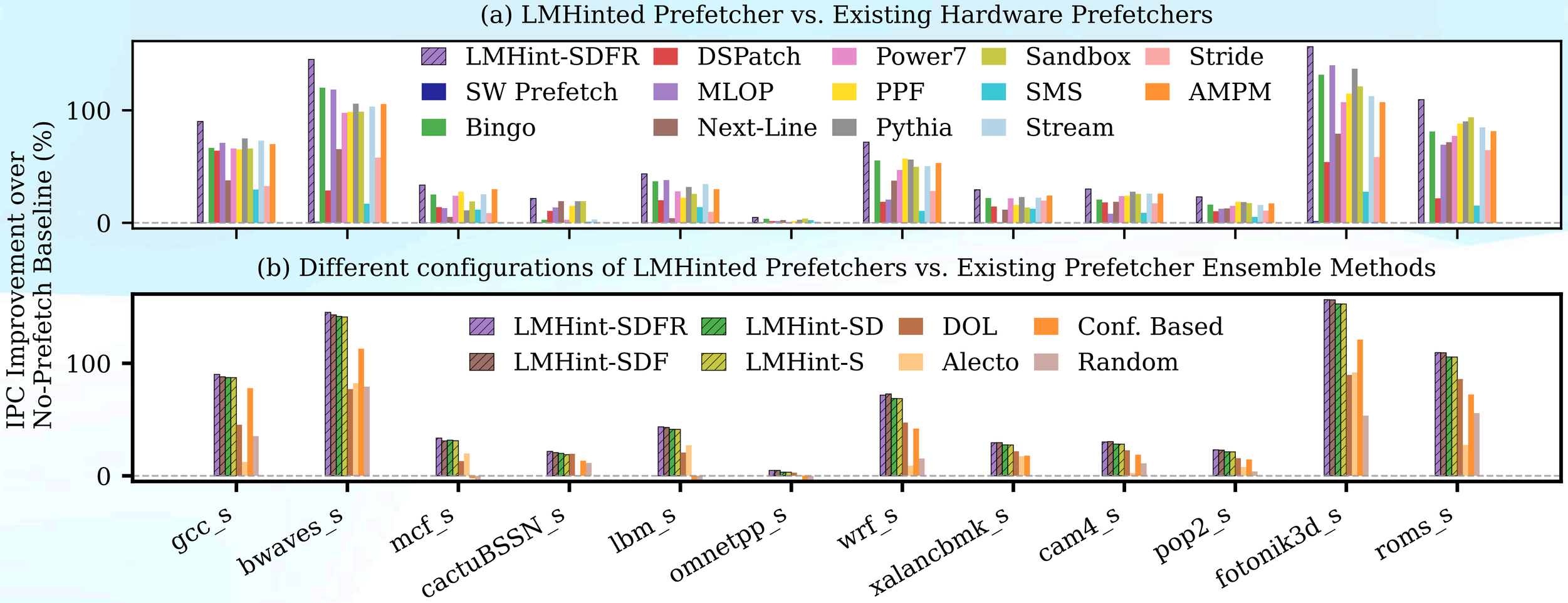
Steady convergence

Strong diagonal line

95% all-match accuracy

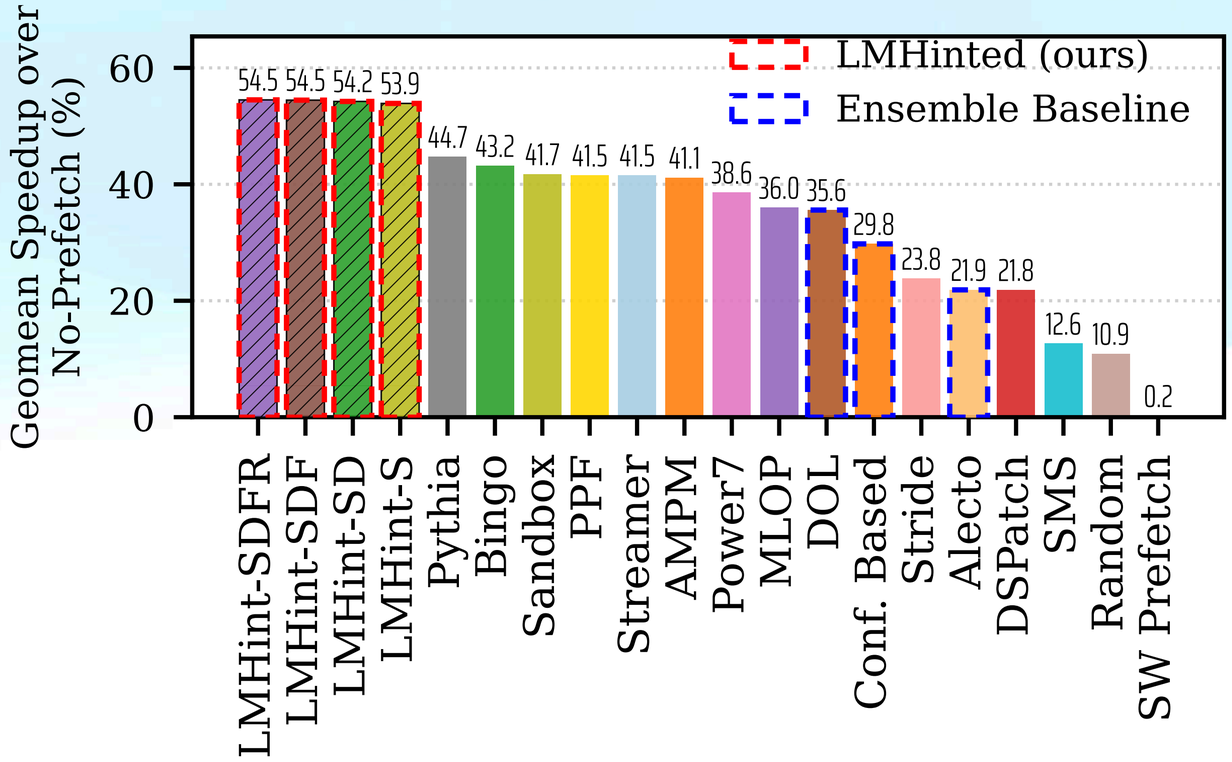


Benchmark on SPEC 2017



Benchmark on SPEC 2017

- **S**: Select Policy only
- **SD**: + Degree
- **SDF**: + Filter
- **SDFR**: + Reduce to only four most-frequently used PF



- +9.8% IPC over state-of-the-art single hardware prefetching baselines

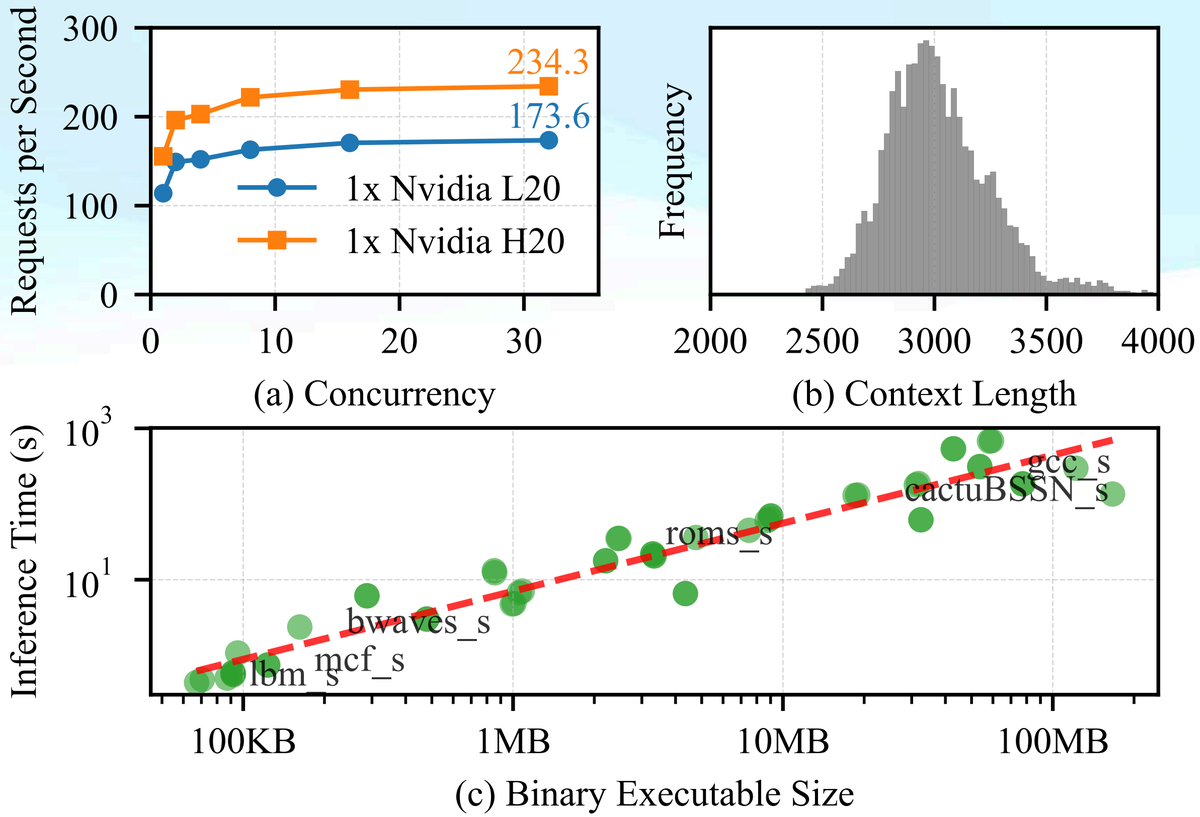
- +18.9% IPC over prior ensemble methods

Overhead Analysis of PF-LLM

Small 0.5B model;
~3k-token short context length;
high-concurrency.

With vLLM Serving, Throughput is acceptable.

Hinting speed on 8xH20 system is on-par with a 16-core compilation system.



Insights

Offload hard online decision problems to offline LLM-generated Hints

Benefits of PF-LLM

Offline intelligence for code-aware policy + simple runtime hardware for fast execution.

Good Performance

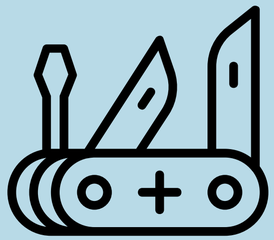
- +9.8% IPC, roughly a full generation's worth of single-core IPC improvement.

Offline Oracle

- Offline LLM instantly decides "when, how & how aggressively" to prefetch
- Eliminates slow online learning and runtime training cost

Works Anywhere

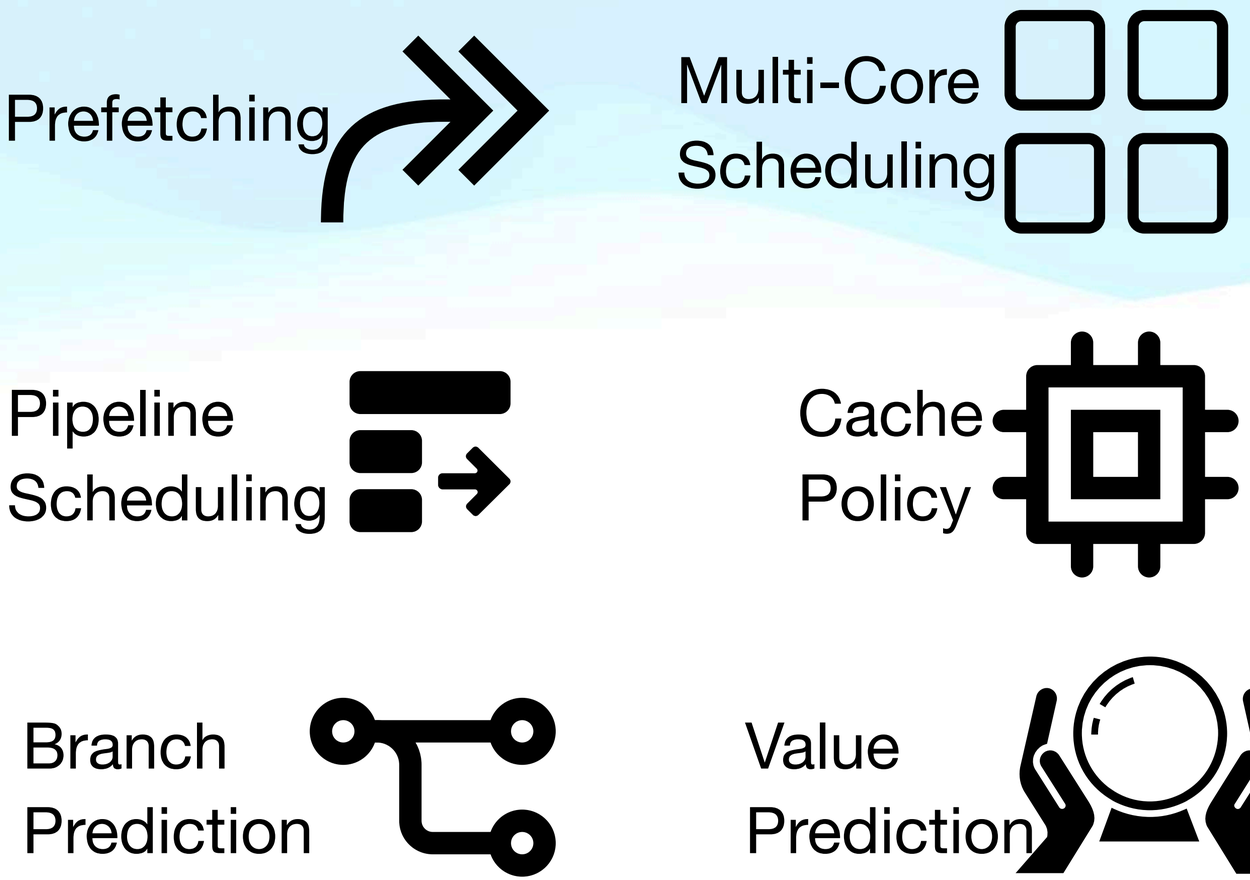
- The code context information remains un-exploited for online decisions.



LLM Hinted CPU micro-Architecture

Offline intelligence for code-aware policy + simple runtime hardware for fast execution.

- Prefetcher is just one of the important **micro-architectural decisions**.
- Using LLMs to gather offline code-context knowledge, apply to **online on-chip** policies.
- Towards a new paradigm of LLM-hinted CPU design.



Thank you!

Questions?