ECEN 676 - Advanced Computer Architecture Project Presentation - Group 8

Path Confidence based Lookahead Prefetching

Team Members:

Vineeth Kumar Ponugoti (836000192)
Bhavishya Kotipi Sivaramareddy (435009710)
Raveena Manasi Burrannagari (535005868)
Sai Vikhyat Pattar (936001854)

Introduction - Misses in a Cache

We incur a large performance penalty if there is a miss in the i-cache.

For the next 10-50 cycles, there will be no instructions to fetch, if there is an L2 Hit

If there is a miss in the L2, we have nothing to do for hundreds of cycles.

IPC will suffer

What is the solution?

Prefetch Memory Addresses.

Meaning: Predict memory addresses that will be accessed in the future. Fetch them from lower levels of memory hierarchy before they are actually required.

Introduction

Instead of Instructions, we can prefetch data as well

Important Distinction

- Instructions are fetched into the in-order part of the OOO pipeline.
- Data is fetched into the OOO pipeline
  - As a result, the final IPC is slightly more resilient to data cache misses
  - Nevertheless prefetching is very useful
  - I-cache hit rates are typically very high, where as d-cache hit rates are much lower
  - Hence, the margin for improvement is significant !!

Challenges with traditional prefetching

- Difficulty in Predicting Complex Access Patterns
- Page Boundary Issues
- High Hardware Complexity
- Over-Prefetching and Bandwidth Waste
- Inability to Adapt to Varying Access Patterns

These challenges highlight the need for advanced techniques like Signature Path Prefetching (SPP), which addresses these issues through dynamic adaptation and better handling of complex patterns, including those crossing page boundaries .

Background and prior work

Almost all prefetching algorithms operate on the miss sequence, not on the access sequence.

Next Line Prefetching: If a cache line X is accessed, there is a high probability of accessing the next sequential cache lines: X+1 and X+2. The reasoning is that code of most functions span multiple cache lines and we will have spatial locality

Stride Based Prefetching: Most programs often sequentially access array elements. The hardware will observe that for the same PC (Program Counter), the memory address keeps getting incremented by a certain value. This fixed increment is called is called a stride

Most studies in this area, try to work on the following extensions

How many iterations do we prefetch in advance? This is also called as prefetch depth

For a subset of prefetches, monitor the number of cycles between when a line is prefetched and it is actually used. The aim is to achieve this delay to be as minimum as possible. Negative delay indicates the block is being prefetched too late and very small positive delay indicates block is fetched just before it is needed.

Recent Advances

Best Offset Prefetcher: It evaluates several offsets during execution and select the one most likely to be useful for prefetching. Hence, it automatically adapts to varying memory access patterns. However, it doesn't consider temporal ordering between delta patterns, and suffer from low accuracy on complex, yet predictable patterns

Variable Length Delta Prefetcher: It builds up delta histories between successive cache line misses within physical pages, then uses these histories to predict the order of cache line misses in new pages. It uses multiple prediction tables, each of which stores predictions based on different length of input history.

Signature Path Prefetcher: It uses an optimized history based scheme that predicts complex address patterns precisely. Unlike many history based algorithms, tracks complex address patterns across page boundaries and continues prefetching immediately as they move to new pages. Finally, it adjusts its prefetching, for each stream based on confidence it has in its predictions, enabling adaptive throttling.

Project Description

We are drawing inspiration from the published paper "**Path Confidence Based Lookahead Prefetching**". While the paper explores path confidence-based techniques for lookahead prefetching, our approach builds upon these ideas but introduces innovations tailored to handling memory access patterns across page boundaries and implementing adaptive throttling in a more efficient manner. **Goal:** Develop and enhance the **Signature Path Prefetching** (SPP) algorithm.

Key Features:

Compressed History Signatures**:** Store memory access patterns in a compressed format
Adaptive Throttling**:** Dynamically adjusts the depth of prefetching based on confidence
Cross-page Boundary Learning**:** Handles memory accesses that span multiple pages

Simulation Tool: The project will use **ChampSim**, a trace-based simulator, to model and evaluate the performance of SPP. ChampSim is ideal for testing prefetching algorithms on microarchitectural blocks and out-of-order processors.

Project Description - SPP Design Spec

The diagram shows the design of the Signature Path Prefetcher which involves 4 components:

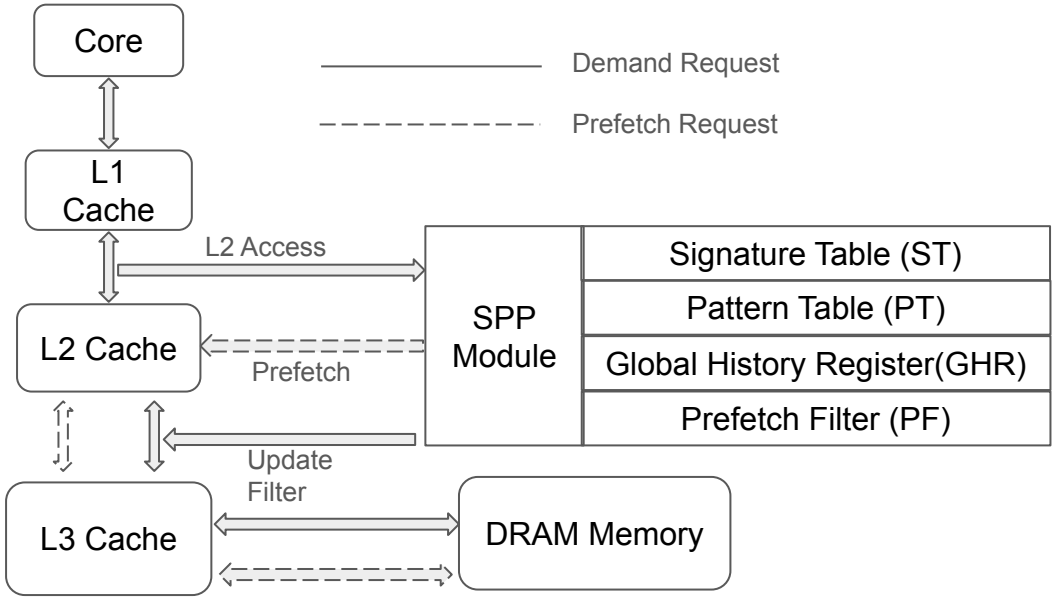
Signature Table: Per Page History

Pattern Table: Per History Confidence

Global History Register: This is an important component in the design that helps the Prefetcher to predict the accesses that cross page boundaries.

Prefetch Filter: Filter to prevent redundant prefetches.

SPP learns from L2 accesses and fills prefetches into L2 or LLC based on prediction confidence.



Project Description
Global History Register

Alternate implementation of Global History Register operation.

What happens if the current **signature** for which the prediction crosses page boundary and is already present as an entry in the **Global History Register** - Update the GHR entry with the latest confidence and delta

Previous implementation only checks for matching offset in a GHR entry

Global History Register (Pre-update)

Signature	P_d	Last Offset	Delta
0x52	0.45	62	+3

New Signature
0x52 with P_d = 0.6

Global History Register (Post-update)

Signature	P_d	Last Offset	Delta
0x52	0.6	45	+5

Project Description - Other Prefetchers

We have implemented the Variable Length Delta Prefetcher (VLDP) as mentioned in the original implementation. Following shows the design specification for this prefetcher

Other Prefetchers used for Comparison:

- Best Offset Prefetcher
- Access Map Pattern Matching Prefetcher

Project Evaluation

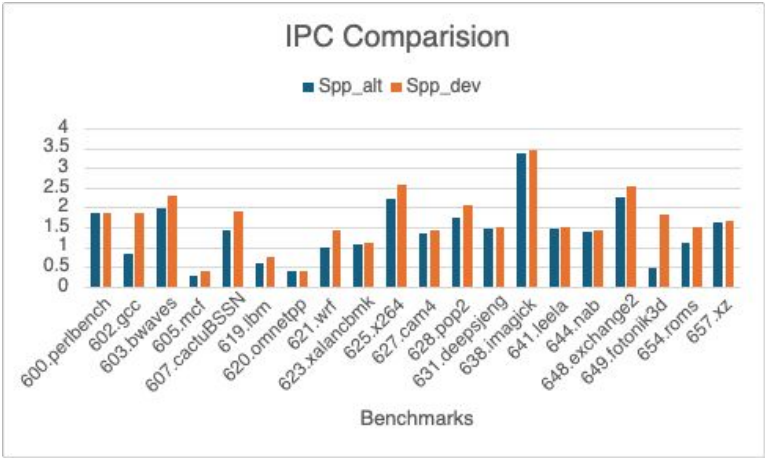
Following comparisons are used to measure the performance of Signature Path Prefetcher:

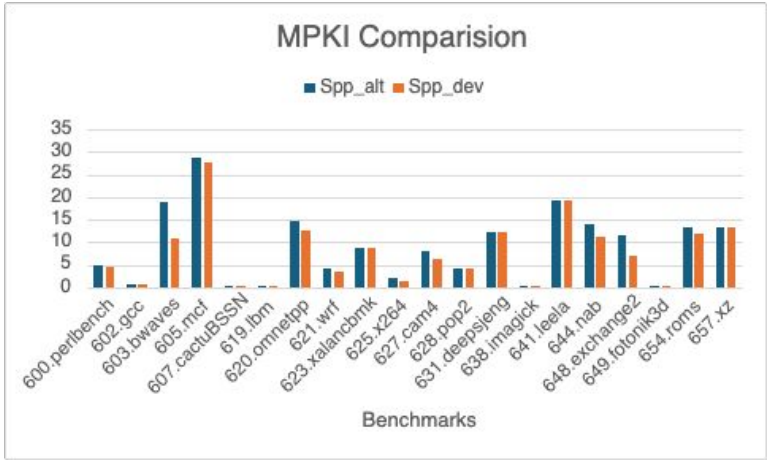
1. SPP_ALT(our implementation) **vs** SPP_DEV(Original implementation), No Prefetcher, Variable Length Delta Prefetcher, Best Offset Prefetcher, DA-AMPM Prefetcher
2. SPP with Global History Register **vs** SPP Without Global History Register

Configuration Used

Core Parameters	1-4 Cores, 3.2 GHz, 256 Entry ROB, 4-wide 64 entry scheduler 64 entry load buffer
Branch Predictor	GShare, Global History Length - 14, History Table Size - 16384, Bits -2
Cache Used	L2 Cache, 256 KB, 8-way, 8 cycles 16 MSHRs, Non - inclusive
Replacement Policy	Least Recently used (LRU)
Benchmarks	SPEC 2017 multi-processor benchmark suite
Warmup Instructions	200000000 (20 Billion)
Simulation Instructions	1000000000 (100 Billion)

Project Evaluation

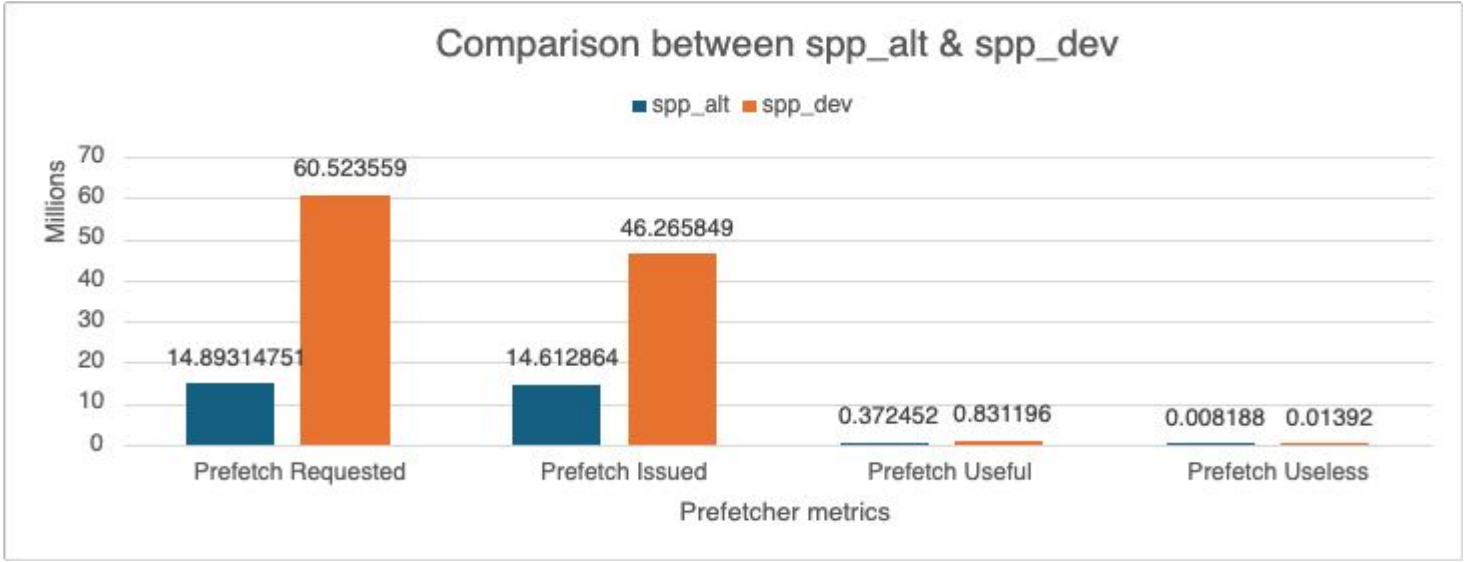




- SPP_ALT achieves IPC values comparable to SPP_DEV across a wide range of benchmarks, showing consistent performance.
- Benchmarks like gcc, bwaves, and mcf show strong IPC values, while cam4, roms, and xalancbmk exhibit minimal difference between the two implementations.

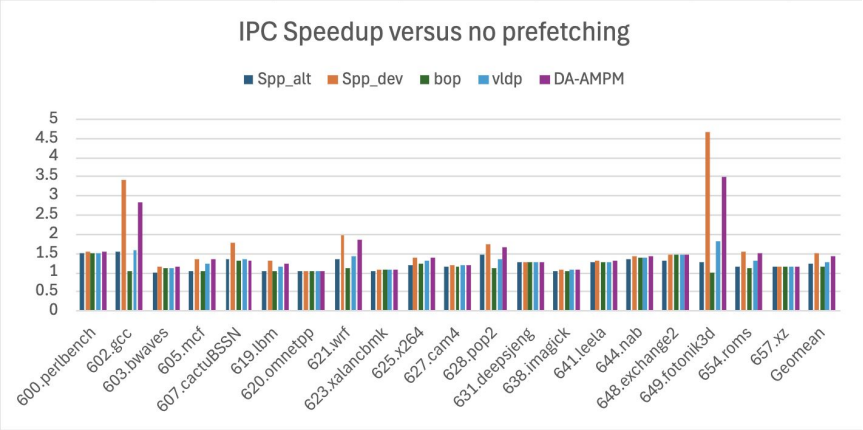
- SPP_ALT achieves MPKI values similar to SPP_DEV across most benchmarks, demonstrating consistent cache miss reduction.
- Workloads like gcc, deepsjeng, and pop2 show identical MPKI, while others such as roms and xalancbmk show close performance trends.

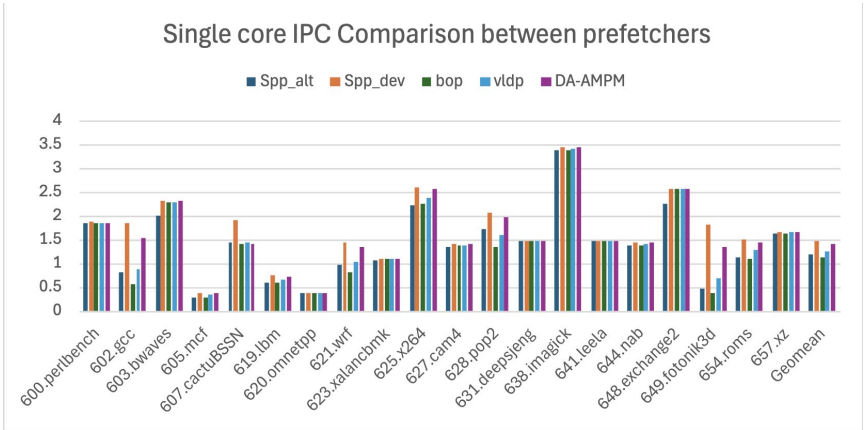
Project Evaluation



- SPP_ALT issues 4x fewer prefetches than SPP_DEV while still maintaining competitive performance, showing efficiency.
- Both useful and useless prefetches are lower in SPP_ALT, indicating a more selective and accurate prefetching approach.

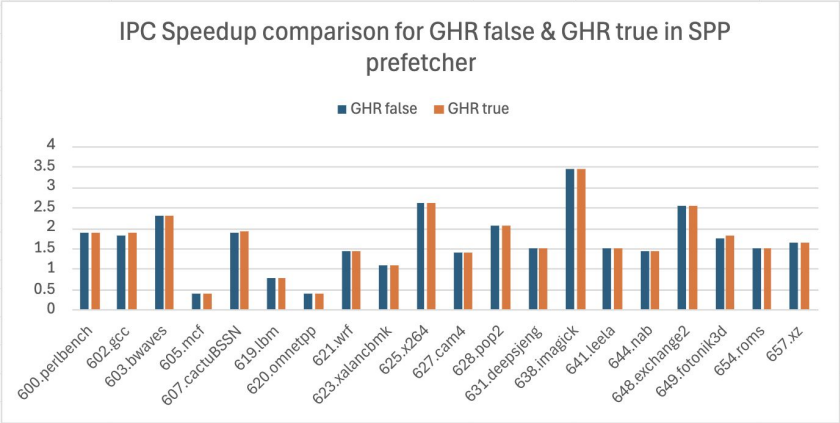
Project Evaluation

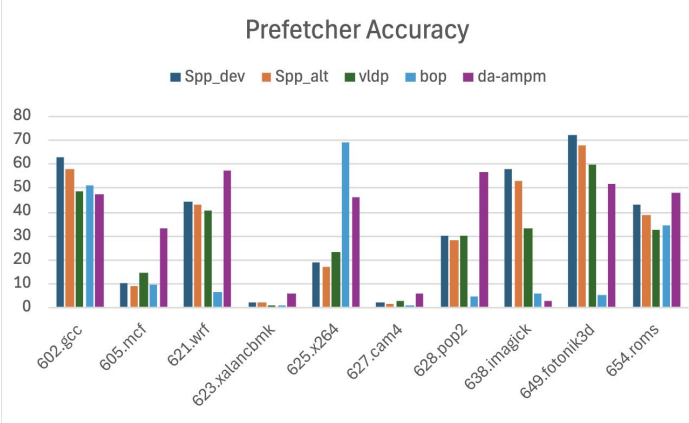




- SPP_ALT achieves a 1.22× IPC speedup over no prefetching and shows competitive performance across all benchmarks.
- Compared to other state-of-the-art prefetchers, SPP_ALT demonstrates balanced and consistent improvement with efficient prefetching.
- SPP_ALT achieves an average IPC of 1.19 across all benchmarks, closely tracking the performance of more aggressive prefetchers.
- It performs consistently well across compute- and memory-intensive workloads, demonstrating versatility in single-core environments.

Project Evaluation



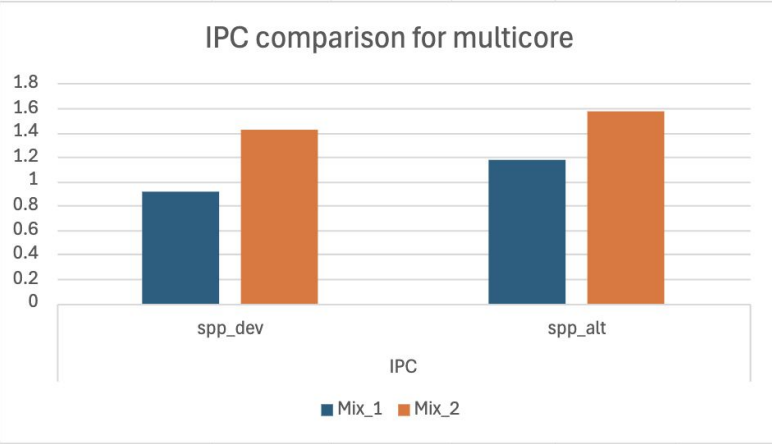


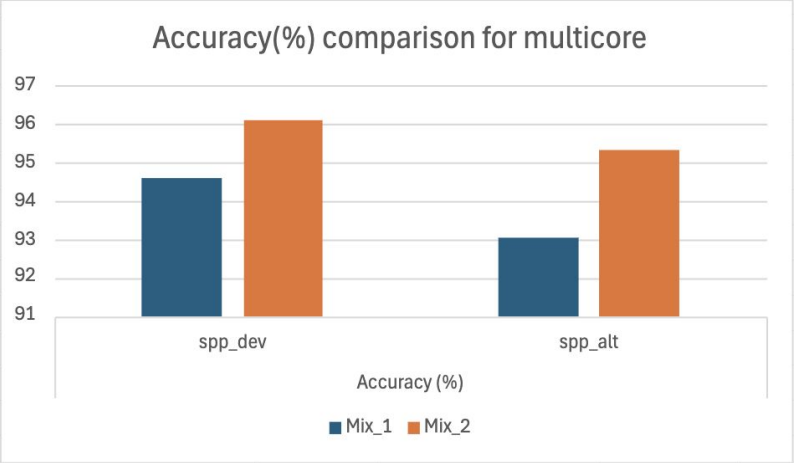
- GHR integration provides better IPC in benchmarks with cross-page memory access patterns, while maintaining stable performance elsewhere.
- Benchmarks like gcc, fotonik 3d, and roms benefit from GHR, confirming its usefulness in enhancing prefetch continuity.

- SPP_ALT maintains high prefetch accuracy across benchmarks, often close to SPP_DEV, and higher than VLDP and BOP in most cases.
- It demonstrates a balanced trade-off between accuracy and volume, ensuring effective prefetching without excessive overhead.

Project Evaluation

We have used 4-core system with 2 DRAM channels.
Mix_1 : perlbench, gcc, bwaves, mcf
Mix_2 : xalancbmk, x264, cam4, pop2





- SPP_ALT delivers higher IPC in both workload mixes, achieving 1.17 (Mix_1) and 1.57 (Mix_2), compared to 0.92 and 1.42 in SPP_DEV.
- This shows better throughput scalability and efficient memory utilization with SPP_ALT in multicore systems.

- Both SPP_ALT and SPP_DEV maintain high accuracy (>93%) in multicore mixes, showing stable prefetch quality under shared resource contention.
- SPP_ALT achieves 93.04% and 95.32% accuracy in Mix_1 and Mix_2 respectively, confirming consistent prediction effectiveness.

Conclusion

- Implemented a confidence-based lookahead prefetcher inspired by the paper *"Path Confidence Based Lookahead Prefetching."*
- Designed our own version, SPP_ALT, incorporating:
  - Compressed signature history
  - Adaptive path confidence throttling
  - Enhanced Global History Register (GHR) for cross-page learning
- Evaluated across 20 SPEC benchmarks in both single-core and multicore (4-core, 2-DRAM channel) setups
- SPP_ALT reduced issued prefetches by up to 4× while maintaining or improving IPC and MPKI compared to SPP_DEV.
- SPP_ALT achieved up to 1.5× IPC improvement over no prefetching and maintained similar or better MPKI compared to SPP_DEV while issuing fewer prefetches.
- Achieved prefetch accuracy up to 67.8% in single-core and above 93% in multicore workloads.
- Outperformed SPP_DEV in multicore IPC, showing better scalability and efficiency under memory contention.
- GHR enhanced cross-page pattern learning, improving continuity and reducing warm-up latency.
- Validated the paper's core idea: confidence-guided lookahead prefetching delivers high performance with reduced overhead.

Link to Git Repo and Video

Video link

Git hub Repository

THANK YOU!!!

Any questions???