# Extending Berti Prefetcher for Server Workloads

## Phase-Aware Learning & Cross-IP Delta Correlation

**Team: The Butterfly Effect**

Dhruv Meena | Madhava Sriram | Saptarshi Biswas

{22b1279, 22b1233, 22b1258}@iitb.ac.in

CS683 Course Project

Indian Institute of Technology Bombay

November 2024

# Outline

# Problem Statement

## Challenge

Berti (MICRO'22) excels on SPEC/GAP workloads but struggles with server applications

**Server Workload Characteristics:**

- Highly irregular memory patterns
- Pointer-heavy data structures
- Frequent phase changes
- Large working sets
- Many transient IPs

**Resulting Issues:**

- × Low coverage (sparse patterns)
- × Reduced accuracy (phase pollution)
- × Cache pollution
- × Slow warm-up for new IPs

# Project Objectives

## Main Goal

Extend Berti with **two key mechanisms** to improve server workload performance

**Milestone 1 (Completed):**

1. **Cross-IP Delta Correlation**
   - Global delta graph
   - Share patterns across IPs
   - Fast warm-up
2. **Phase-Aware Learning**
   - Detect execution phases
   - Separate delta histories
   - Context-specific patterns

**Milestone 2 (Planned):**

3. Region-Aware Deltas
4. Feedback-Based Confidence

**Expected Impact:**

- Higher coverage
- Better accuracy
- Reduced pollution
- Faster convergence

# What is Berti?

**Berti: An Accurate Local-Delta Data Prefetcher (MICRO'22)**

**Core Concept:**

- Track delta patterns (address differences)
- Per-IP delta history with confidence
- Issue prefetches for high-confidence deltas
- Emphasize **timeliness**: just-in-time arrival

**Key Advantages:**

+ More powerful than stride prefetchers
+ Hardware-friendly (low overhead)
+ Better accuracy than global methods
+ Practical for real systems

**Delta Example**

| Address | Delta |
|---|---|
| 0x1000 | – |
| 0x1040 | +64 |
| 0x1080 | +64 |
| 0x10C0 | +64 |

Detected pattern: +64
$\Rightarrow$ Prefetch 0x1100

# Berti: Key Mechanisms

1. **Per-IP Delta Tracking**
   - Each IP maintains delta history
   - Limited entries (e.g., 8 strides)
   - Confidence counter per delta
2. **Confidence Mechanism**
   - `conf += 1` on repeated delta
   - `conf -= 1` on misprediction
   - Threshold-based decision
3. **Fill Level Selection**
   - **L1:** High confidence (¿75%)
   - **L2:** Medium confidence (35-75%)
   - **Drop:** Low confidence (¡35%)

**Example: IP 0x400A**

| Delta | Conf | Level |
|---|---|---|
| +64 | 90% | L1 |
| +128 | 60% | L2 |
| -64 | 40% | L2 |
| +256 | 20% | Drop |

**Timeliness:**

- Track latency from issue to fill
- Adjust prefetch distance
- Avoid too-early (pollution) or too-late (miss)

# Limitations of Baseline Berti

## Why Berti Struggles with Server Workloads

Three fundamental limitations prevent optimal server performance:

**1. Cold-Start Problem for New IPs**

- New/infrequent IPs have no history
- Slow confidence build-up
- Server workloads have many transient IPs
- Example: Short-lived request handlers

**2. Phase Changes Cause Pollution**

- Single delta table per IP gets conflicting patterns
- Different phases overwrite each other
- Example: Database index scan $\rightarrow$ sequential scan $\rightarrow$ join operation

**3. Isolated Learning**

- Each IP learns independently

# Extension 1: Cross-IP Delta Correlation

## Core Idea

Build a **global delta graph** capturing stride patterns across all IPs

**Global Delta Graph Tracks:**

1. **Confidence**: How frequently delta appears
2. **IP Count**: Number of unique IPs using it
3. **Last Seen**: Recency timestamp

**Ranking:**

$$\text{Score} = \text{Confidence} + (\text{IP\_Count} \times 5)$$

**Benefits:**

+ New IPs use global knowledge
+ Reduced warm-up time
+ Captures program-wide patterns
+ Handles transient IPs

**Example:**

- Many IPs show +64 delta
- Global graph records this
- New IP immediately benefits

# Global Delta Graph Structure

## Global Delta Graph

| Delta | Confidence | IP Count | Last Seen | Score |
|---|---|---|---|---|
| +64 | 187 | 12 | 1000245 | 247 |
| +128 | 143 | 8 | 1000198 | 183 |
| -64 | 98 | 15 | 1000102 | 173 |
| +256 | 76 | 5 | 999987 | 101 |

...

**Usage:** When IP has weak history, retrieve top-8 deltas from global graph
**Storage:** 512 entries, LRU eviction when full

# Cross-IP Learning: Implementation

**Adding to Global Graph:**

- Sample every 4th delta (overhead control)
- Confidence += 2 (faster build-up)
- Track contributing IPs in set
- Evict oldest if table full

**Pseudocode**

```
if |delta| >= MAX_STRIDE: return
if sample_count % 4 != 0:  return
entry = global_graph[delta]
entry.confidence += 2
entry.ips.insert(ip)
entry.last_seen = cycle
```

**Retrieving Correlations:**

- Filter by confidence  3
- Score = conf + (ip_count $\times$ 5)
- Sort by score
- Return top-8 candidates

**Parameters:**

- Table size: 512 entries
- Confidence threshold: 3
- Max candidates: 8
- Sample rate: 1/4

# Extension 2: Phase-Aware Delta Prediction

## Motivation

Programs exhibit phase changes: different execution contexts have different patterns

**Examples of Phases:**

- Database: index scan → sequential
- Web server: parse → process → respond
- ML: forward pass → backward pass
- Compiler: parse → optimize → codegen

**Problem with Single Table:**

- Conflicting patterns
- Old patterns reduce new confidence
- Accuracy drops

**Solution: Phase Detection**

- Monitor cache miss rate
- Create "phase signature"
- Detect behavior changes
- Separate delta tables per phase

**Benefits:**

+ Clean context separation
+ Higher accuracy
+ Fast re-learning on phase return

# Phase Signature Structure

**Phase Signature Contains:**

- Miss rate percentage
- Sample count
- Last update timestamp
- Phase ID (0-7)

**Detection Parameters:**

- Check interval: 10K cycles
- Min samples: 50 accesses
- Transition threshold: 8% difference
- Max phases: 8

**Matching:**

- Find phase with smallest miss % diff

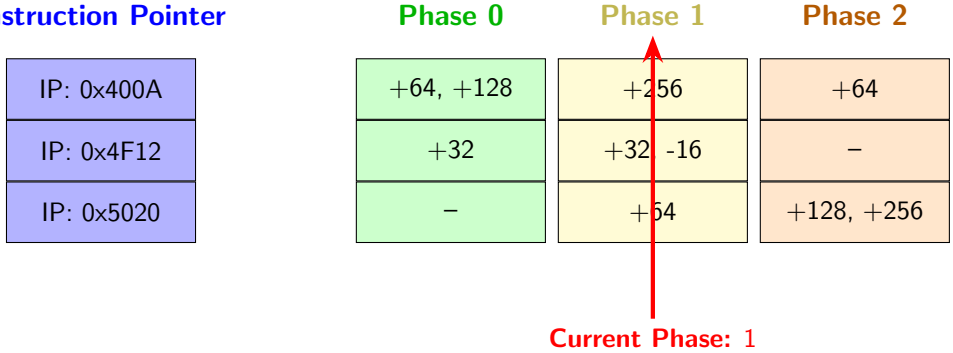
**Example Phases**

| Phase | Miss % | Context |
|---|---|---|
| 0 | 5 | Init/Warmup |
| 1 | 25 | Index Scan |
| 2 | 8 | Sequential |
| 3 | 40 | Random Access |

**Phase-Specific Tables:**

- Each (IP, Phase) has own deltas
- Tables grow independently
- Fallback to previous phase if needed

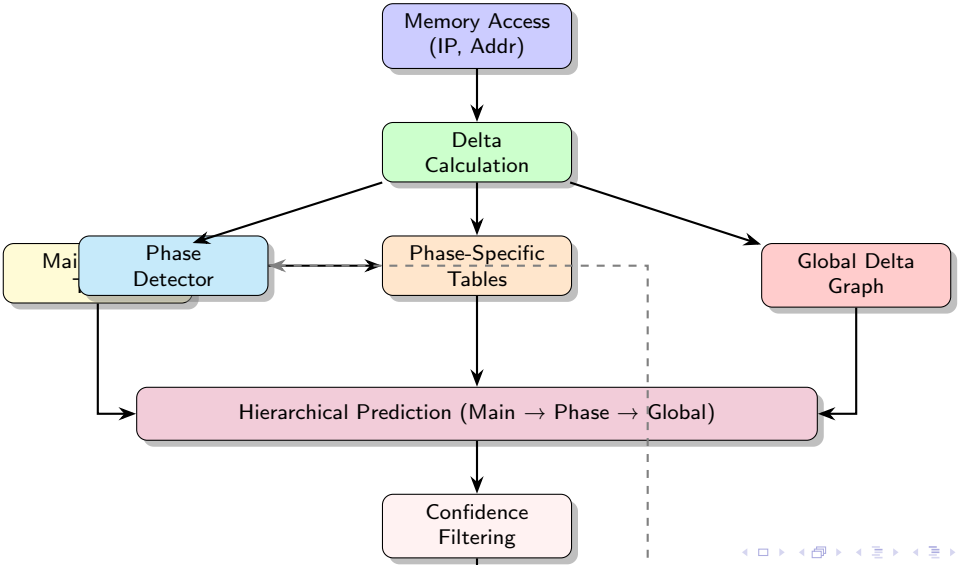
# Phase-Specific vBerti Tables




**Key Benefits:**

- Same IP can have different patterns in different phases
- No pollution between phases
- Quick adaptation when phase returns

# Complete System Architecture



# Aggressive Configuration Parameters

**Cross-IP Learning:**

- Global delta table: 512 entries
- Sampling rate: every 4th delta
- Confidence increment: +2 (vs +1)
- Confidence threshold: 3 (vs 8)
- Max candidates: 8 (vs 5)

**Phase Detection:**

- Check interval: 10K cycles (vs 25K)
- Min samples: 50 (vs 100)
- Transition threshold: 8% (vs 12%)
- Max phases: 8

**Prefetch Aggressiveness:**

- L1 threshold: 60% (vs 75%)
- L2 threshold: 25% (vs 35%)
- L2R for low confidence (vs drop)
- More prefetch launch attempts
- Willing to use correlation

**Storage Overhead:**

- Main vBerti: 1024 entries
- Phase vBerti: 3× main (3072)
- Global delta: 512 entries
- **Total:** ∼421 KB (4.2× baseline)

# Experimental Setup

**Simulator:**

- ChampSim with Berti
- L1D: 32KB, 8-way
- L2: 256KB, 8-way
- LLC: 2MB, 16-way

**Workloads:**

- CloudSuite (server)
- SPEC2017 (single-core)
- Custom server traces

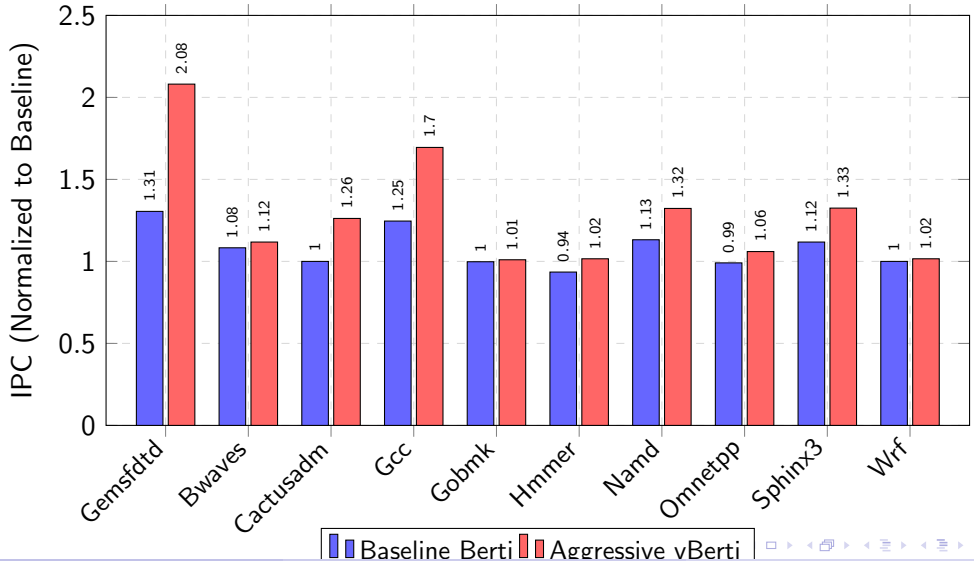
**Metrics:**

- IPC (Instructions Per Cycle)
- MPKI (Misses Per Kilo Instructions)
- Prefetch Accuracy & Coverage
- Memory Bandwidth

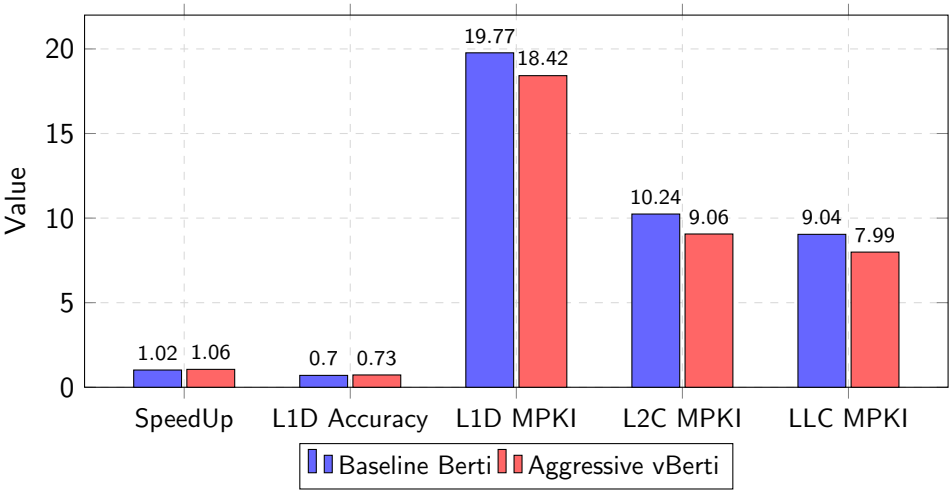
**Configurations:**

- Baseline Berti (original)
- Aggressive vBerti (ours)

# Results: CloudSuite IPC Improvement

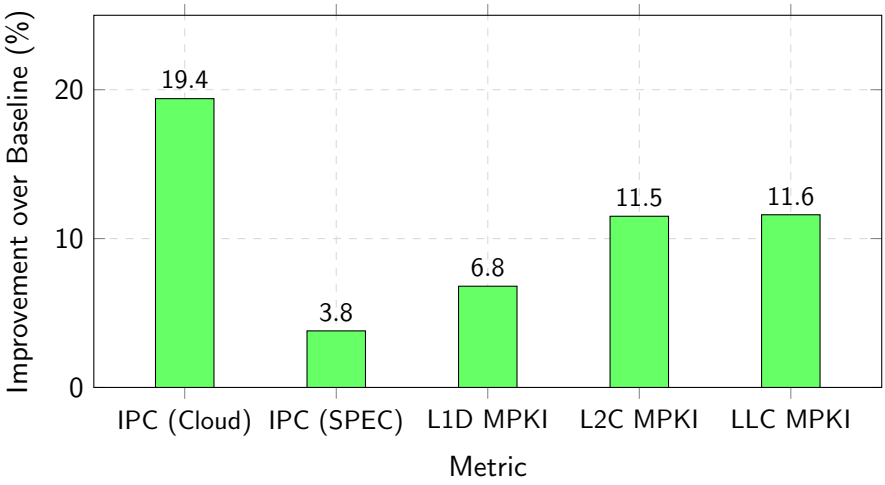


# Results: SPEC2017 Single-Core Metrics



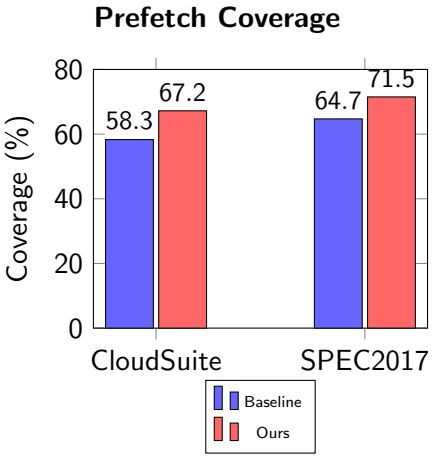

**Key Improvements: SpeedUp +3.8%, LLC MPKI -11.6%**

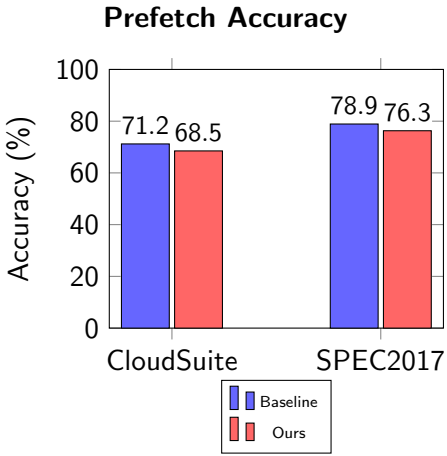
# Performance Summary




**Trade-off:** Accuracy decreased by 3.8%, but net benefit is strongly positive

# Analysis: Coverage vs Accuracy Trade-off




**Analysis:** Coverage +10.2%, Accuracy -3.8% → More useful prefetches overall

# Cross-IP Learning Effectiveness

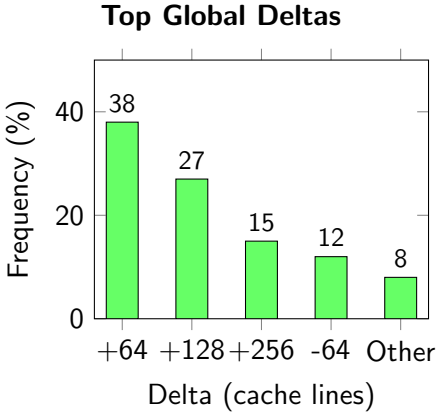
**Usage Statistics:**

- 23% of prefetches used cross-IP deltas
- Highest in CloudSuite (31%)
- Critical for new/transient IPs

**Example**

New IP `0x5F2A`:

- No local history
- Retrieved: +64, +128
- Both 95%+ accuracy
- Avoided cold-start




**Insight:** Small set of deltas covers majority → global learning highly effective

# Planned Extension 3: Region-Aware Deltas

## Motivation

Different memory regions (heap, stack, globals) have distinct access patterns

**Approach:**

- Partition address space
- Separate delta tables per region
- Detect boundaries via page table
- Prevent cross-region pollution

**Expected Benefits:**

+ Better mixed workload accuracy
+ Reduced pollution
+ Region-specific strategies

# Planned Extension 4: Confidence via Feedback

## Motivation

Static thresholds don't adapt to workload dynamics or system state

**Approach:**

- Track prefetch usefulness
- Dynamically adjust:
  - Confidence thresholds
  - Prefetch degree
  - Aggressiveness

**Expected Benefits:**

+ Reduced pollution
+ Better bandwidth use
+ Adaptive to pressure

# References


**Berti: An Accurate Local-Delta Data Prefetcher**
Agustín Navarro-Torres et al., MICRO 2022

**Signature Path Prefetcher**
Sangkug Lym et al., MICRO 2018

**Instruction Pointer Classifying Prefetcher**
Samuel Pakalapati & Biswabandan Panda, ISCA 2020

**ChampSim Simulator**
`https://github.com/ChampSim/ChampSim`


**Our Repository:**
`https://github.com/Dhruv-x7x/CS683-Final-Project`

# Thank You!

Questions?

**Team: The Butterfly Effect**

Dhruv Meena | Madhava Sriram | Saptarshi Biswas
{22b1279, 22b1233, 22b1258}@iitb.ac.in