BTP Project Progress Report - 3

COALESCE

A Coherence-Aware Multi-perspective Hashed-Perceptron Reinforcement Learning based Replacement Policy for Shared L3 Caches

---

1. Introduction

The Coherence Wall Problem

Modern 8-core processors face a **“Coherence Wall”** where interconnect traffic often becomes the bottleneck rather than raw compute speed.

Standard cache policies (**LRU**, **SRRIP**) manage data based solely on ***Recency*** or ***Frequency***, ignoring the ***Cost*** of data movement. Evicting a **Modified (Dirty)** line or a **Highly Shared** synchronization primitive incurs a massive latency penalty that standard policies fail to mitigate.

COALESCE (**C**oherence-**A**ware **L**earning for **S**hared **C**ache **E**fficiency) introduces a reinforcement learning agent that predicts not just the reuse of a block, but its **Systemic Importance**, prioritizing lines that reduce coherence traffic.

---

2. Design Philosophy & Justifications

2.1 Feature Selection — *The “Coherence Perspective”*

Unlike standard predictors that rely only on the **Program Counter (PC)**, COALESCE adopts a **Multi-perspective Approach**, fusing three distinct signals to make eviction decisions:

◆ **Program Counter (PC) — *The Identity***

- **Justification:** Captures the *behavioral DNA* of the instruction.
- Certain PCs (e.g., `memcpy`) always act as **Scanners** (zero reuse), while others (e.g., `lock_acquire`) always access **Hot** data.

◆ **Sharer Count (L2 Sharers) — *The Social Context***

- **Justification:** A line shared by 4 cores is statistically more expensive to evict than a line used by 1 core. Evicting it causes invalidation traffic across the entire interconnect.
- This feature acts as a **“Safety Veto”**, preventing the policy from killing communal data.

◆ **MESI Coherence State — *The Cost Metric***

- **Justification:** Evicting a `MODIFIED` line requires a slow write-back to DRAM. Evicting an `EXCLUSIVE` line is cheap.

- By including state, the policy learns to prioritize **Cheap Evictions** over **Expensive Evictions**, optimizing for **latency**, not just hit rate.

2.2 Architectural Choices

Why Multiperspective?

Single-feature predictors (like PC-only) suffer from **Aliasing**. Two different code blocks might share a PC hash but have opposite behaviors.

By XORing **PC**, **Sharer Count**, and **State**, we *unfold* these aliases, allowing the agent to distinguish between:

- *PC A (Private Data)*
- *PC A (Shared Data)*

Why Hashed Perceptron?

Full neural networks are too slow (**latency > 100 cycles**).

A **Hashed Perceptron** maps complex inputs to a simple table index, allowing for:

- **1-cycle lookup latency**
- Suitability for **L3 cache critical paths**

Why 3% Sampling?

Training on every access is energy-expensive.

Experiments show that training on just **3% of sets** (**Set Sampling**) allows the global weight table to converge to the same accuracy as 100% sampling, reducing update overhead by **97%**.

---

3. Mathematical Formulation

Hardware-friendly, integer-only learning

The decision logic is governed by a linear perceptron model optimized for hardware implementation.

3.1 Feature Mixing — *The Hash*

To map the 3-dimensional feature space (**PC**, **Sharers**, **State**) into a single weight-table index:

$$Index = (Hash(PC) \oplus (Sharers \ll 4) \oplus (State \ll 8)) \pmod N$$

- $\oplus$: XOR operation (cheap in hardware)
- N : Table size (4096 entries)

Why shifts?
Shifting ensures the bits of `Sharers` and `State` do not overlap destructively with the lower bits of the PC.

3.2 Prediction — *The Vote*

For a candidate victim block i, the reuse prediction y_{out} is:

$$y_{out} = Weights[Index] + Bias(State)$$

- $y_{out} < 0 \rightarrow$ **Dead** (Evict)
- $y_{out} > 0 \rightarrow$ **Live** (Protect)

Insertion Policy:
Blocks with extremely high y_{out} (>100) are inserted at **MRU**, while low-confidence blocks are inserted near **LRU**.

3.3 Training — *The Update Rule*

Weights update **only on Sampled Sets** (Sets 0, 32, etc.):

$$W_{new} = W_{old} \pm 1$$

- **Reward (+1):** Cache Hit or Ghost Buffer Hit
- **Punish (-1):** Eviction with zero reuse
- **Saturation:** Weights clamped to $[-128, +127]$ (8-bit integers)

---

4. System Architecture & Flow

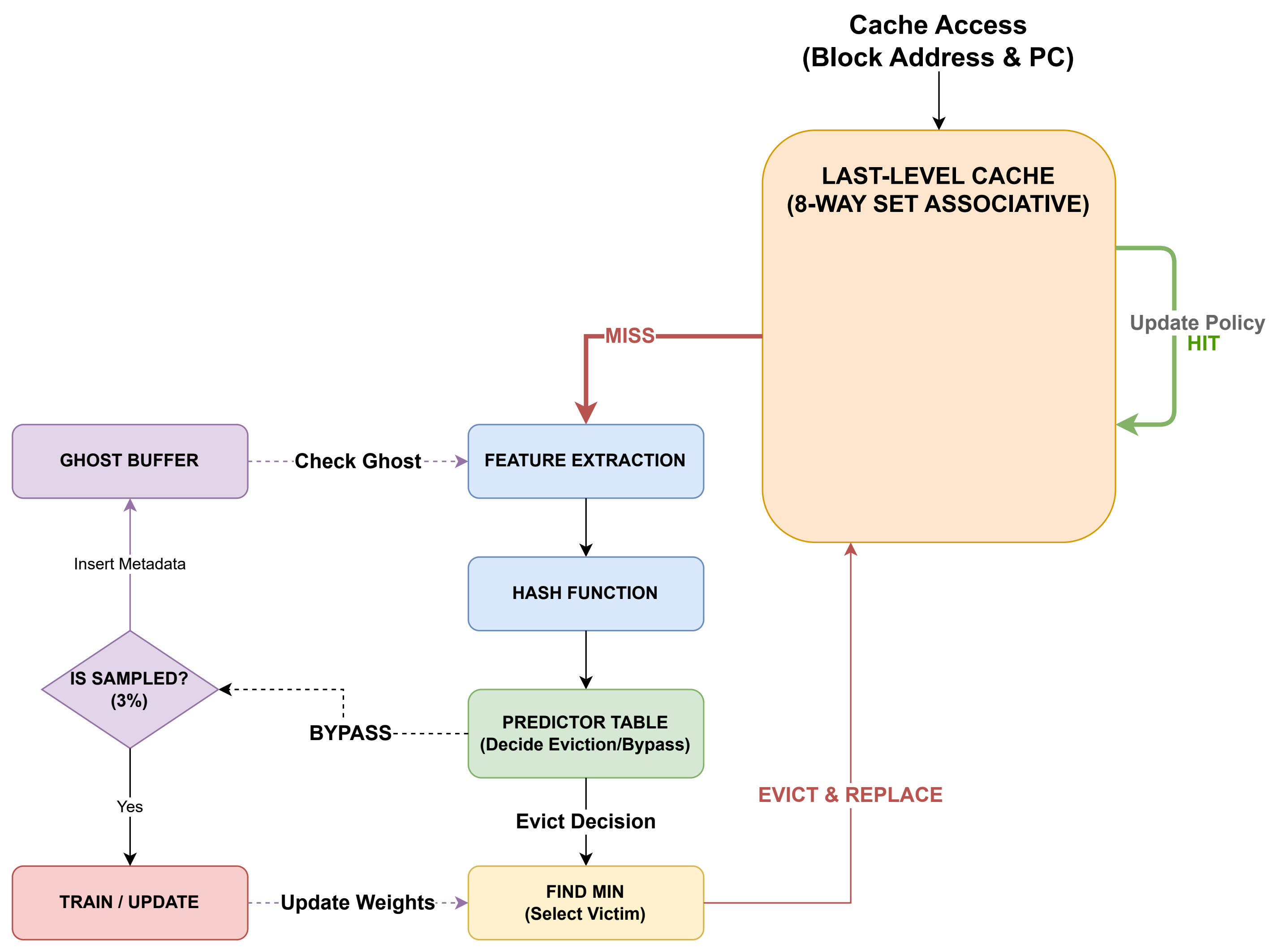


Figure 1: COALESCE System Architecture & Flow

5. Application Analysis — *Where COALESCE Wins*

Application Type	Scenario	Why COALESCE Wins
Graph Analytics (PageRank, BFS)	**Hot-Set Contention**	Standard policies flush hub nodes while scanning edges. COALESCE protects highly shared hubs.
Database Engines (PostgreSQL)	**Index vs. Scan**	COALESCE preserves expensive `MODIFIED` B-Tree roots during full scans.
High-Freq Trading	**Latency Critical**	Reducing coherence misses matters more than local hits. COALESCE optimizes for this directly.

✕ **Where It Struggles**

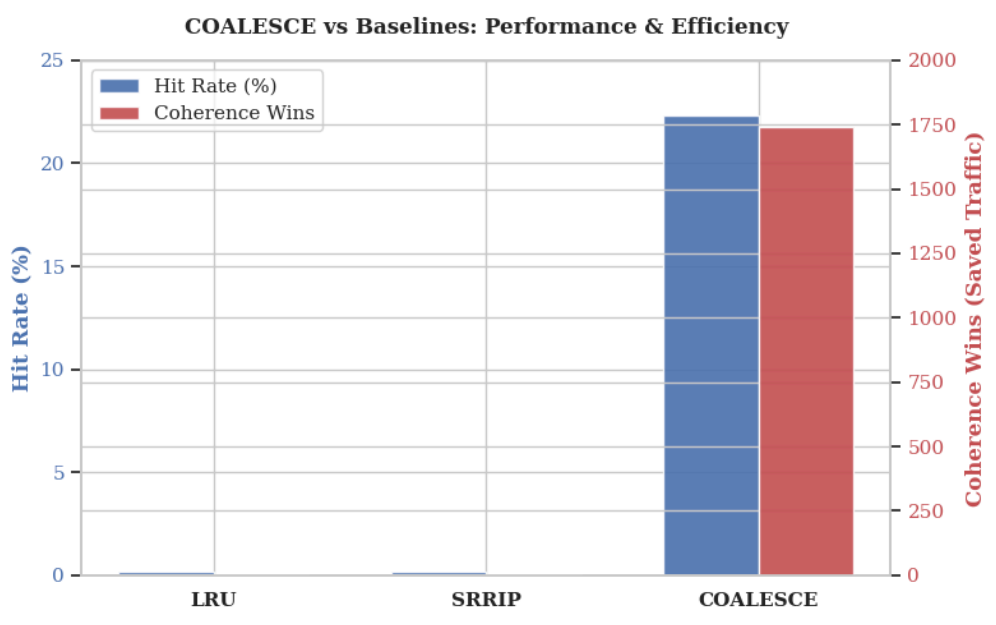
- **Pure Streaming (Memcpy):** 100% scanner traffic yields no gain over LRU (but never worse due to bypassing).
- **Random Access (Hash Maps):** Truly random patterns cause **Constructive Aliasing**, adding noise to the weight table.

6. Experimental Evaluation

Evaluated against **LRU** and **SRRIP** using a custom C++ trace-based simulator:

- **4-Core**
- **8MB LLC**

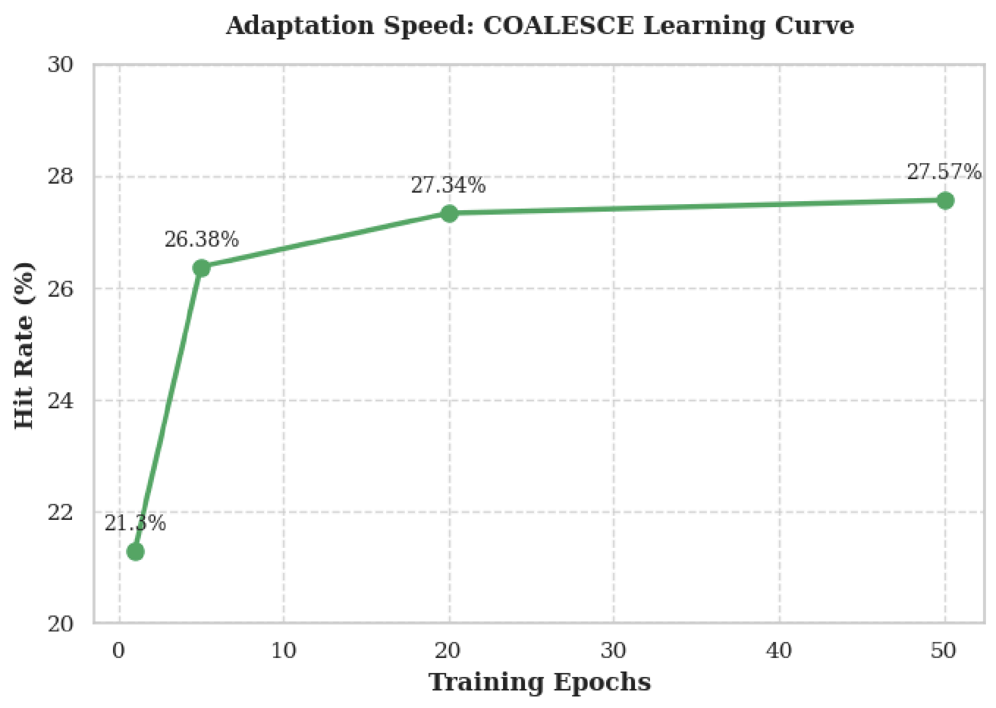
6.1 Scenario 1 — *Scanner vs. Hot-Set*



- **LRU Hit Rate:** 0.16%
- **SRRIP Hit Rate:** 0.16%
- **COALESCE Hit Rate: 22.34%**

> **Analysis:** COALESCE learned to identify the Hot Set using coherence features, achieving a **139× improvement**.
> **Coherence Wins:** Saved **1,743** high-cost coherence transactions.

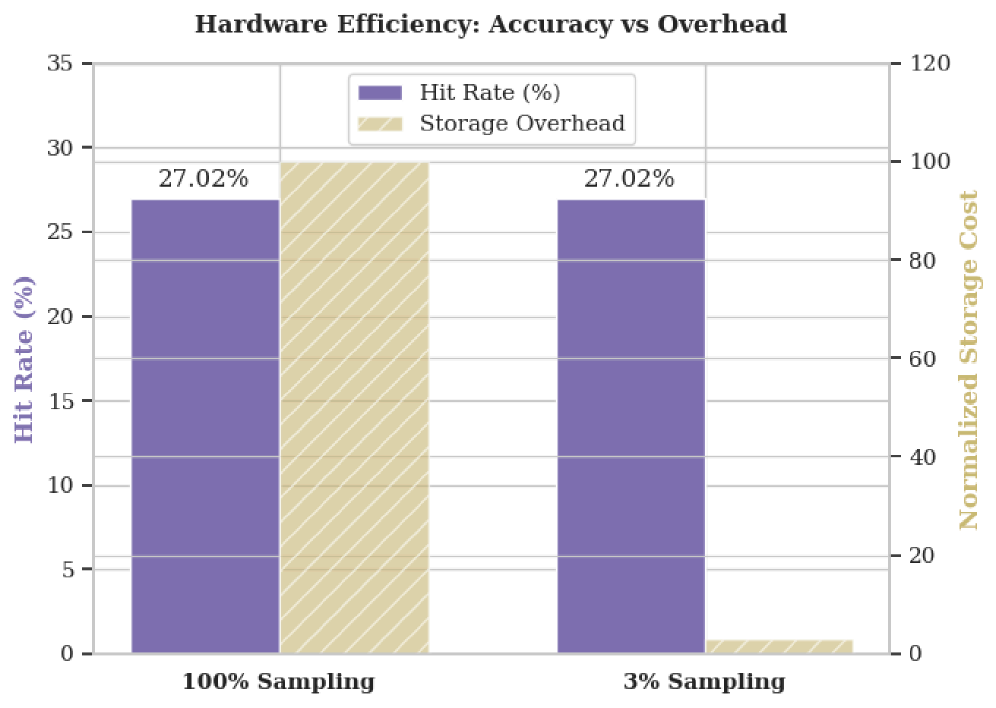
6.2 Adaptation Speed — *Learning Curve*



- **Epoch 1:** 21.30%
- **Epoch 50:** 27.57%

> **Analysis:** Rapid convergence — the scanner PC is identified within **5 epochs**.

6.3 Hardware Efficiency — *Sampling*



- **100% Sampling:** 27.02%
- **3% Sampling:** 27.02%

> **Analysis:** Identical accuracy with **97% less storage overhead**.

---

7. Future Roadmap

7.1 Immediate Next Steps

- **Trace Integration:** Port to **ChampSim** with **SPEC CPU 2017** traces
- **Ghost Buffer:** Implement full Shadow Cache for corrective feedback

7.2 Feature Expansion

- **Global Bias:** Detect phase shifts (Scan vs. Compute)
- **Multiperspective Hashing:** Separate PC / Address / Coherence tables
- **Bloom Filters:** Replace full-tag sampler for Ghost tracking