NC STATE UNIVERSITY Electrical and Computer Engineering

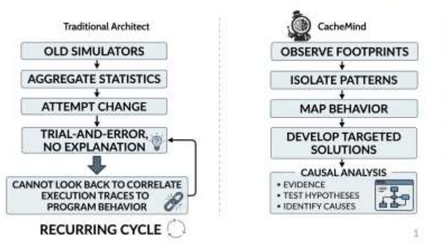
# CacheMind

## From Miss Rates to Why - Natural-language, Trace-grounded Reasoning for Cache Replacement

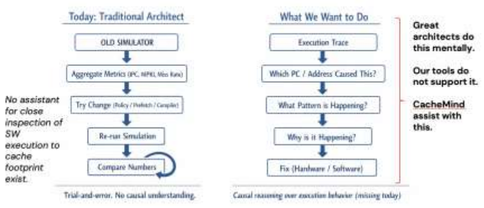
**Kaushal Mhapsekar, Azam Ghanbari, Bita Aslrousta, Samira Mirbagher-Ajorpaz**

## Problem

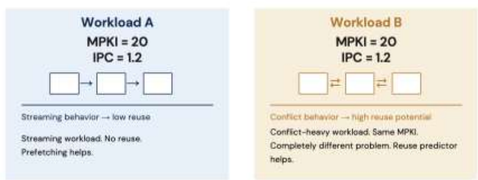
*Great architects do not design cache optimizations from aggregate counters alone; they isolated execution patterns, mapped them back to program behavior, and invented mechanisms that captured those patterns at runtime. CacheMind is the first attempt to bring that human reasoning pipeline into the simulator.*




Caches are still optimized through *trial-and-error over weak signals*, not causal reasoning

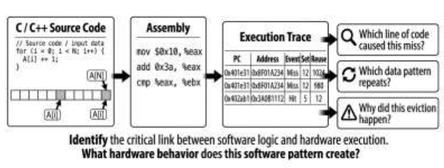



Aggregate metrics hide fundamentally different behaviors.




Same numbers. Completely different root causes and solution needed.

The answer already exists in the execution trace




Simulators already record the evidence. We still **cannot** automatically isolate it, connect it to code and data, and reason over it.

LLMs need retrieval to reason over traces

- **LLMs are not trained on traces**
  LLMs are not trained on microarchitectural simulator traces which are large and highly structured.
- **Traces are massive**
  Single simpoint traces contain ~100,000 accesses (~400 MB). Full suites can span hundreds of gigabytes.
- **Context windows are limited**
  LLM context can fit a few MBs. Precise retrieval is necessary for reasoning.

## Methodology and Novel Benchmark

Experimental Setup

| Category | Subsystem | Configuration |
|---|---|---|
| Simulation Platform | ChampSim | Trace-based |
| | gem5 extension | Used for software intervention studies |
| Workloads | Trace source | SPEC CPU 2006 / CRC-2 derived traces |
| | Trace fields | PC, address, hit/miss, eviction |
| Execution Setup | Run length | 1B instructions |
| | Analysis target | Event-level cache behavior |
| Replacement Policies | Baselines | LRU, Belady, Parrot, Multi-layer perceptron |
| | Learned | 6-wide fetch/decode/execute; 4-wide retire |
| Hardware Configuration | Core | 1 core, 4 GHz |
| | Pipeline | 6-wide fetch/decode/execute; 4-wide retire |
| | OoO structures | 352 ROB, 128 LQ, 72 SQ |
| | Predictor | bimodal |
| Hardware Configuration | L1 I | 32 KB, 64 sets, 8 ways, 4 cyc, 8 MSHR |
| | L1 D | 32 KB, 64 sets, 8 ways, 4 cyc, 16 MSHR |
| | LLC | 2 MB, 2048 sets, 16 ways, 26 cyc, 64MSHR |
| | DRAM | 4 GB DDR4-3200, 1 ch, 1 rank/ch, 8 banks/rank |

**Benchmarking advances LLM reasoning in every field**

*Curated suites with unambiguous ground-truth answers have been the engine of measurable progress across domains TODO*

| GSM8K | MATH | ARC | DROP |
|---|---|---|---|
| Grade-school math | Competition math | Commonsense science | Discrete reasoning |
| Enabled evaluation of multi-step arithmetic reasoning in LLMs | Pushed boundaries of symbolic and algebraic problem solving | Tested scientific reasoning beyond pattern matching | Evaluated numerical reasoning over text passages |

No equivalent benchmark exists for microarchitecture - **CacheMindBench** is the first verified, trace-grounded reasoning suite for cache replacement.

**Existing architectural benchmark does not test *reasoning* over execution behavior**

Trace-Grounded Tier: Testing fine-grained retrieval

| Category | Description | Example Question |
|---|---|---|
| Hit/Miss | Determine if a specific access results in a hit or miss | *Does PC 0x409dc9 and address 0x47ea85d3f1 result in a cache hit in lbm under PARROT?* |
| Miss Rate | Compute miss rate of a specific PC or workload | *What is the miss rate for PC 0x4037ba in mcf with PARROT?* |
| Policy Comparison | Compare miss behavior across replacement policies | *Which policy has the lowest miss rate for PC 0x409270 in astar?* |
| Count | Count frequency of events (accesses, evictions) | *How many times did PC 0x405832 appear in astar under LRU?* |
| Arithmetic | Perform arithmetic over trace statistics | *What is the avg evicted reuse distance of PC 0x40170a for lbm with MLP?* |
| Trick Question | Catch inconsistencies or invalid assumptions | *Does PC 0x4037ba in lbm access address 0x1b73be82e3f? (Answer: TRICK)* |

Binary scoring (1 = correct, 0 = incorrect) using exact-match verification against the trace.

Architectural Reasoning Tier: Testing LLM reasoning

| Category | Description | Example Question |
|---|---|---|
| Microarch. Concepts | Answer general cache questions with architectural logic | *How does increasing cache size affect miss rate? Compare increasing #sets vs. #ways.* |
| Code Generation | Generate code to analyze specific trace conditions | *Write code to compute hits for PC 0x4037ba and address 0xa3e0df33f6l in mcf under LRU.* |
| Policy Analysis | Analyze behavior of policies across contexts | *Why does Belady outperform LRU on PC 0x409270 in astar?* |
| Workload Analysis | Reason about entire workload characteristics | *Which workload has the highest cache miss rate under MLP?* |
| Semantic Analysis | Connect PCs to high-level program behavior | *Why does PC 0x4037ba have a high hit rate? Examine the assembly context and analyze.* |

Qualitative scoring (0–5) based on correctness, use of evidence, and clarity - judged by a microarchitect.

## Contributions

**1. CacheMind: Query-First Microarchitectural Microscope**

- Natural-language → trace-grounded reasoning
- Connects software → instruction → cache behavior
- Answers per-PC, per-address "why" questions

**2. Dual Retrieval Architecture (SIEVE + RANGER)**

- SIEVE: precise symbolic filtering (PC, address, policy)
- RANGER: dynamic code generation for complex queries
- Enables accurate retrieval for unseen query patterns

**3. CacheMindBench:** First Verified Reasoning Benchmark

- 100 trace-grounded questions with ground-truth validation
- Covers factual, arithmetic, and causal reasoning
- Enables evaluation beyond surface-level answers

**4. Actionable Insights:** From Analysis → Optimization

- +7.66% hit rate, +2.04% IPC (bypass optimization)
- +76% speedup (software prefetching fix)
- +0.7% speedup (Mockingjay training improvement)
- Identifies root-cause PCs and execution patterns

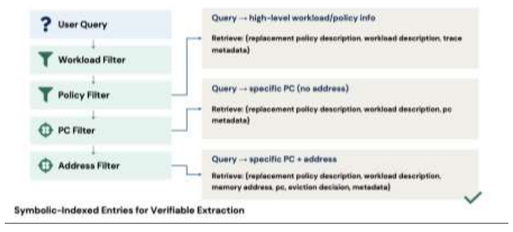
## Our Solution

Why Existing RAG Fails on Trace Reasoning

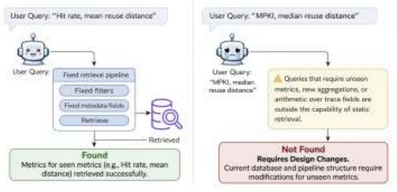
Trace-grounded cache reasoning is a structured retrieval problem disguised as a language problem.

These questions look like language, but they require exact queries, decomposition over structured data and reasoning over structured trace data - not similarity or static queries.

SIEVE filters raw traces to a task-specific slice and returns the most informative evidence for the user's query.

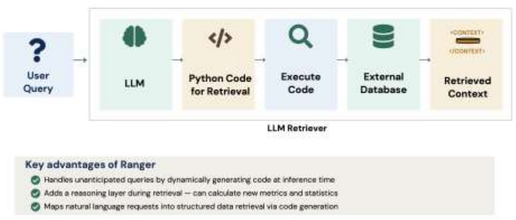



Fixed retrieval fails on unanticipated queries




Queries on new, unaccounted metrics require modifying the infrastructure pipeline.

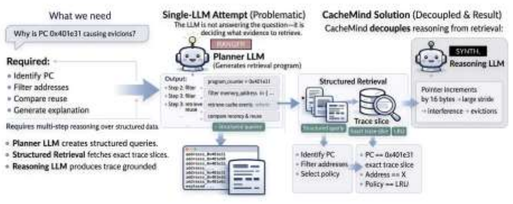
CacheMind leverages LLM's code generation and reasoning capability to handle arbitrary queries




Prompt as modular program specification

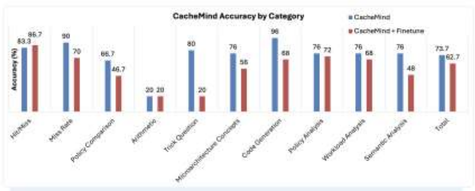
The prompt encodes goal, structure, retrieval logic, and output rules

CacheMind pipeline is a multi-step reasoning over structured data.



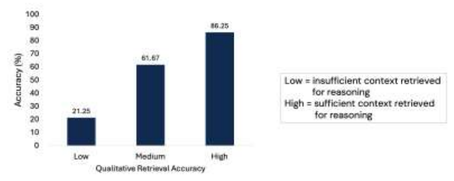

## Results

**Finetuning drops the performance by 11%**



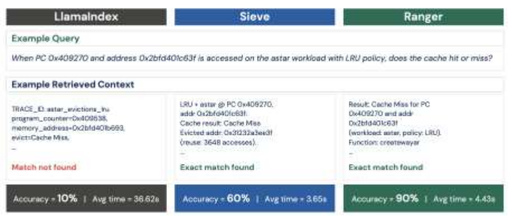

We observed that finetuning narrows generalization and increases hallucination risk

Retrieval quality dominates reasoning performance



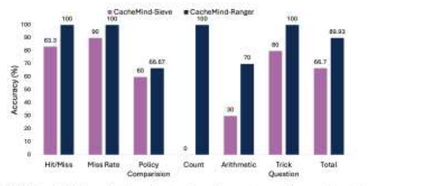

Better retrieval leads to more accurate and consistent outputs.

Sieve fixes RAG limitations; Ranger enhances rigid retrieval



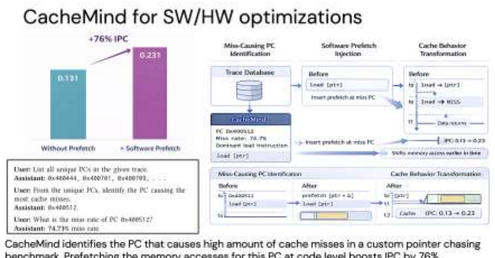

CacheMind Ranger solves retrieval bottleneck

SIEVE limitation - works well for structured queries but doesn't do well for open-ended dynamic queries.




Proposed: RANGER, an LLM-based retriever that writes and executes python code to retrieve relevant data from the database, based on user queries and knowledge of database schema.

CacheMind for SW/HW optimizations




CacheMind identifies the PC that causes high amount of cache misses in a custom pointer chasing benchmark. Prefetching the memory accesses for this PC at code level boosts IPC by 76%

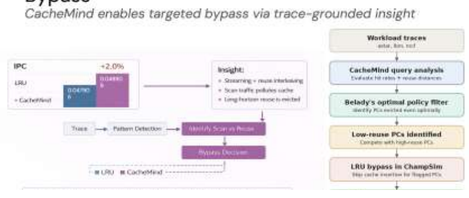
CacheMind can reveal set-level behavior for cache analysis

*Hot set analysis* is beneficial for replacement policies like Mockingjay, Hawkeye, SHiP, SDBP, DRRIP, DIP, which *sample cache sets*

**CacheMind insight**: For astar workload, set hotness depends on workload behavior, not the replacement policy's decisions.

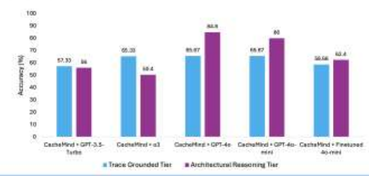
Optimizing Replacement Policy with targeted Bypass

*CacheMind enables targeted bypass via trace-grounded insight*




CacheMind identifies potential optimizations in SOTA replacement policies from just the paper summary and memory access traces!

Stronger LLM backends make CacheMind more reliable




"CacheMind's strength is finding the balance between retrieval quality and LLM reasoning"

### Conclusion

Next-generation simulators must move beyond ***fixed metric*** outputs to engines that can answer ***arbitrary, interactive questions*** about any event in the simulated execution.

| 90% | 9× | 100 | 4 |
|---|---|---|---|
| Retriever accuracy on trace-grounded questions | Better retrieval vs. LlamaIndex | Verified benchmark questions | Actionable insights highlighting utility (76% speedup) |

**CacheMind** demonstrates this concept for cache replacement - a microarchitectural microscope that enables free-form, per-PC, per-data exploration.

ASPLOS