HOME: A Hierarchy-Oriented Memory Evaluation Framework for Fast Contention Analysis

Dhruv Gajaria
Pacific Northwest National Laboratory
Richland, WA, USA
dhruv.gajaria@pnnl.gov

Andrés Márquez
Pacific Northwest National Laboratory
Richland, WA, USA
andres.marquez@pnnl.gov

ABSTRACT

Memory bottlenecks caused by contention in modern multicore systems limit scalability for many HPC and scientific workloads. To support rapid, actionable contention analysis, we introduce HOME (Hierarchy-Oriented Memory Evaluation), a lightweight framework that processes sparse PTWRITE traces by temporally interpolating sampled events (implemented as timestamp scaling and count rescaling) to restore the time-varying memory access pressure; HOME then applies cache-and-memory modeling and confidence conditions to flag under-sampled runs. HOME targets the middle ground between slow cycle-accurate simulators and coarse hardware-counter summaries: it preserves time-aware access intensity where it matters while running far faster than full simulation.

We validate HOME on 19 applications drawn from five benchmark suites, comparing against hardware performance counters (`perf`), Dinero, DRAMsim3, and Sniper. HOME achieves average LLC cache contention error rates of 9.92% and memory bandwidth utilization error rates of 7.33% compared to measured hardware, while running approximately 26.8 times faster than Sniper on our GAP workloads. We also present sampling diagnostics and confidence conditions that identify workloads for which temporal interpolation is unreliable, and we provide guidance for increasing sampling density (e.g., merged runs) when needed. HOME is designed to be reproducible and practical for HPC researchers and system engineers who need quick, interpretable contention diagnostics.

1 INTRODUCTION

High-performance computing (HPC) systems underpin scientific advances across various fields, including AI, physics simulations, molecular dynamics, astrophysics, and weather forecasting. The growing demand for these applications has driven rapid advances in both hardware and software capabilities [35]. At the same time, increasing core counts, concurrency, and data volumes place rising pressure on shared memory resources: simultaneous accesses to caches and DRAM often produce contention that degrades throughput and execution efficiency [26]. Typical HPC workloads run with hundreds to thousands of threads, amplifying spatial and temporal interference and exposing complex contention phenomena [10, 13]. Measurements indicate that memory contention can account for a substantial fraction of observed slowdowns (on the order of tens of percent for some workloads) [15, 28], making accurate, practical contention diagnostics a priority for both system designers and application developers.

Cycle-accurate and interval-based simulators (e.g., gem5, Sniper, SimpleScalar) provide detailed emulation of processor and memory behavior and are valuable for examining microarchitectural causes of contention [2, 7, 9]. However, their high computational cost, long runtimes, and complex configuration make them ill-suited for rapid exploration across many workloads or parameter sweeps representative of realistic HPC studies; moreover, simulators can omit platform-specific low-level effects that influence contention, limiting their fidelity in certain cases [32].

As a lightweight alternative, contention models built on hardware performance counters measure runtime metrics such as cache misses, bandwidth utilization, and processor stalls with minimal overhead [3, 13, 20, 36]. These counter-based approaches are attractive for in situ analysis and rapid iteration. Still, because they report aggregated summaries, they often cannot resolve per-address, time-aware interactions across multiple levels of the memory hierarchy or easily support "what-if" architectural explorations. Those trade-offs motivate hybrid approaches that retain the speed of counter-based monitoring while providing finer-grained, address-level insight when needed. To address the limitations of both cycle-accurate simulators and coarse hardware-counter models, we present HOME (Hierarchy-Oriented Memory Evaluation). HOME is a lightweight framework for identifying and analyzing memory contention across cache levels and DRAM, balancing per-address fidelity with practical runtime. Figure 1 illustrates HOME's design point: it provides more address- and time-aware detail than counter-based models while incurring far lower overhead than full simulation.

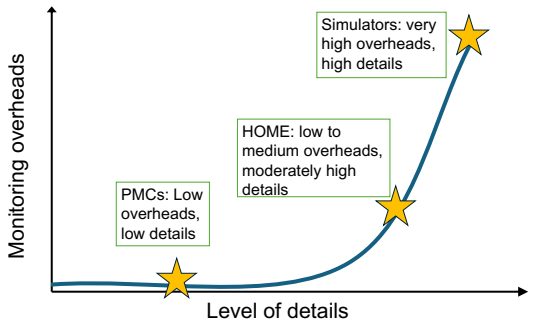

Figure 1: Monitoring overhead vs. level of detail. Cycle-accurate simulators provide the highest microarchitectural fidelity at very high cost; hardware performance counters (PMCs) provide low overhead but coarse summaries; HOME targets the middle ground by consuming sparse PTWRITE traces and producing time- and address-aware pressure profiles with modest overhead.

HOME leverages MemGaze [25] and Intel PTWRITE to collect low-overhead trace events (timestamps, instruction pointers, and access types). Because PTWRITE traces are sparse and can lose samples due to hardware buffering and interrupts, HOME does not recreate the original load/store sequences or unobserved addresses; instead, HOME performs temporal interpolation, which is implemented by timestamp scaling and count rescaling. HOME utilizes these estimated temporal pressure profiles as inputs to configurable cache and DRAM models, applying confidence conditions that flag runs where interpolation-based estimates are likely unreliable. This hybrid approach enables fast, scalable "what-if" exploration of architectural configurations and helps pinpoint code regions that correlate with contention without attempting to synthesize unrecoverable per-access data.

The key contributions of this paper are:

- **HOME framework.** We introduce HOME, a modular pipeline that ingests PTWRITE traces, performs timestamp-scaling interpolation and sampling-ratio scaling to estimate temporal access pressure, and applies cache/DRAM models for contention analysis.
- **Interpolation and confidence conditions.** We formalize temporal interpolation (timestamp scaling) as a practical estimator of access intensity from sparse traces, and we define confidence checks that detect undersampled runs and bound expected error.
- **Configurable hierarchy modeling.** HOME integrates configurable cache and memory hierarchy models, enabling rapid evaluation across different architectural parameters.
- **Empirical validation.** We evaluate HOME on 19 applications from five benchmark suites, comparing against hardware counters (`perf`) and simulators (Dinero, DRAMsim3, Sniper). HOME achieves low average errors for cache and bandwidth metrics while running substantially faster than full simulation (approximately 26.8× on our GAP experiments).
- **Practical diagnostics.** We provide sampling diagnostics and practical recommendations (e.g., merged runs) to improve accuracy for workloads that violate confidence conditions.

Finally, the organization of this paper is as follows: Section 2 provides an in-depth overview of related work on existing tools and models for memory analysis. Section 3 introduces the architecture and mechanisms of HOME, highlighting how it addresses the challenges of sampling drops. Section 4 provides a detailed explanation of the proposed interpolation techniques and boundary conditions. Section 5 presents a comprehensive evaluation of HOME, comparing its performance and accuracy with state-of-the-art simulation tools and real hardware. Section 6 discusses HOME's potential for addressing fine-grained contention in the future, and Section 7 concludes the paper, outlining future directions for extending HOME's capabilities.

2 BACKGROUND AND RELATED WORK

We summarize prior approaches for diagnosing memory contention and position HOME relative to them. Prior work ranges from detailed, cycle-accurate and interval-based simulation to lightweight counter-based modeling, and more recently, low-overhead trace collection. Each approach occupies a different point in the fidelity-overhead design space and exhibits complementary strengths and weaknesses, motivating a hybrid, practical tool such as HOME.

2.1 Contention modeling using simulations

Detailed simulators emulate processor, cache, and memory behavior to expose microarchitectural causes of contention. Cycle-accurate tools (e.g., gem5 [7]) provide fine-grained latency and ordering information useful for debugging coherence, prefetching, and controller interactions. Interval- or sampling-based simulators (e.g., Sniper [9]) trade per-cycle fidelity for much higher scalability and have therefore been used to study larger workloads and parameter sweeps [17, 19, 30]. Formal models (e.g., timed Petri nets [23]) can complement simulators by validating timing and concurrency properties at an abstract level.

Despite their utility, simulators pose two practical challenges for routine contention diagnosis. First, cycle-accurate and detailed interval simulations are often computationally expensive and slow for realistic HPC workloads, which limits interactive exploration and broad parameter studies [32]. Second, simulators can diverge from real hardware in platform-specific behaviors (microcode, memory-controller scheduling, undocumented optimizations), and recent work has documented notable simulator-hardware discrepancies for memory-system experiments [16]. For these reasons, we treat simulation outputs as informative but not absolute ground truth.

HOME adopts a hybrid approach: rather than simulating every cycle and attempting to reproduce every microarchitectural detail from first principles, HOME ingests low-overhead PTWRITE samples (via MemGaze) and feeds timestamp-scaled event profiles into configurable cache and DRAM models. This design preserves useful address- and time-aware fidelity for contention diagnostics while avoiding the cost of full cycle-accurate replay and sidestepping many simulator-specific implementation gaps.

2.2 Performance counters and lightweight monitoring

Hardware performance counters provide low-overhead, in-situ metrics (such as cache misses, bandwidth, and stall cycles) that are convenient for profiling production runs and identifying bottlenecks quickly [3, 22, 37]. Common tooling such as Linux '`perf`', PAPI, and vendor-specific utilities expose counters that are inexpensive to collect and can be aggregated across long or distributed runs, making them well-suited for scalable, production-oriented studies and regression-style performance tracking [4, 38]. For many operational tasks—e.g., identifying whether an application is memory- or compute-bound, or tracking coarse changes after an optimization—counters provide actionable signals with virtually no perturbation to normal execution.

Despite these strengths, performance counters have essential limitations for diagnosing fine-grained contention. Counters are typically aggregated (per-core, per-socket, or per-un-core block) and do not provide per-address or sufficient temporal resolution, so they cannot directly reveal which cache sets, address ranges, or short-lived phases are responsible for observed stalls or bandwidth peaks. In practice, counters are also constrained by a limited number of simultaneous events, vendor-specific event semantics, and

multiplexing artifacts that complicate interpretation. Furthermore, counters may not expose low-level effects, such as set conflicts, row-buffer timing interactions, or undocumented controller policies [11]. These factors reduce the usefulness of counters for detailed root-cause analysis and for architecture "what-if" exploration.

2.3 Traces and emerging low-overhead sampling

Recent tools (e.g., MemGaze [25]) utilize low-overhead mechanisms, such as Intel PTWRITE, to collect sparse address/timestamp samples that are far more informative than counters, while remaining significantly cheaper than full simulation. Such sparse traces require careful processing—HOME employs an explicit temporal interpolation (timestamp scaling and count rescaling) and confidence conditions to estimate time-varying access pressure from sampled events, rather than attempting to recreate unobserved per-access sequences. In practice, we measure sampling ratios r in the range of 1–4%. Under approximate assumptions, timestamp-scaling provide a simple and unbiased estimator of time-varying access intensity (detailed in Section 4.1). This design point allows HOME to support both fast, in-situ diagnostics and configurable hierarchy modeling for exploratory analyses.

Summary. Simulation, counter-based monitoring, and trace sampling are complementary tools: simulators offer detailed microarchitectural insights at high cost, counters provide fast but coarse summaries, and sampled traces offer a practical middle ground. HOME targets this middle ground by combining low-overhead trace collection with lightweight interpolation and configurable hierarchy models, enabling rapid yet informative contention analysis suitable for realistic HPC studies.

3 PROPOSED APPROACH

In this section, we introduce HOME, a novel framework designed to address memory contention by integrating efficient trace collection, hierarchical memory modeling, and architectural optimizations. By leveraging the MemGaze tool alongside HOME's configurable architectural components, we demonstrate how HOME provides precise analysis of contention hotspots while enabling compatibility with detailed memory simulators. The key components of HOME include MemGaze-based trace collection, page map management, coherence space optimizations, and customizable memory hierarchy models.

3.1 Trace collection using MemGaze

HOME begins with trace collection using MemGaze [25], a lightweight tool that leverages Intel Processor Tracing (PTWRITE) [21] to capture detailed information related to program execution and memory access behavior. Processor tracing is a hardware-assisted mechanism integrated into modern CPUs, enabling the recording of runtime execution details, including executed instructions, memory accesses, and timestamps, with minimal overhead. Figure 2 illustrates the flow of MemGaze using processor tracing. As shown in the figure, the collection process involves instrumenting the binary with PTWRITE operations to record load-store addresses

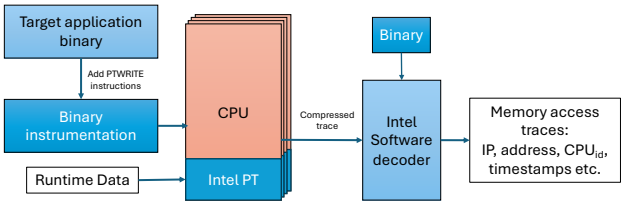

Figure 2: MemGaze + PTWRITE workflow (trace collection pipeline). Instrumentation inserts PTWRITE events, traces are compressed on-chip, decoded offline, and the decoded samples (timestamp, IP, address, CPUid, access type) are fed into HOME for timestamp-scaling and hierarchy modeling.

during program execution. The instrumented binary is then executed on hardware systems that support Intel Processor Tracing (PT). During execution, the processor generates compressed traces containing the memory access information captured by PTWRITE. This compressed trace is subsequently decoded using Intel's decoding utilities, yielding detailed trace data including:

Instruction Pointers (IPs): Track specific instructions being executed, enabling researchers to correlate memory accesses with their corresponding code regions.

Access Types: Identify whether a memory access follows a constant, strided, random, or store pattern. Recognizing these access patterns is essential because they substantially affect memory contention dynamics across hierarchical levels.

Virtual Addresses: Provide the virtual memory addresses of each load and store operation, enabling MemGaze to map memory requests to cache lines, memory banks, and other architectural components of the memory hierarchy. This mapping is critical for identifying contention hotspots.

Cycle-Accurate Timestamps: Deliver precise timing information for every memory operation, offering the high temporal resolution needed to diagnose latency and contention events.

CPU$_{id}$: Indicates the specific CPU or core responsible for initiating each memory request. This information is beneficial for profiling multi-threaded workloads and evaluating contention behaviors in multi-core environments, making MemGaze well-suited for modern high-performance computing (HPC) workloads.

Processor tracing provides fine-grained, low-perturbation signals that are especially useful for memory-contention analysis in HPC workloads. By emitting per-event timestamps, instruction pointers, and access-type annotations, low-overhead mechanisms such as Intel PTWRITE (collected via MemGaze) give far richer, address- and time-aware information than aggregated counters while remaining substantially cheaper than cycle-accurate replay. This makes processor tracing an attractive input for fast, practical contention diagnostics.

A key practical limitation of PTWRITE-style traces is their sparsity and occasional sample drops, as hardware buffering, interrupt handling, and other platform constraints can cause some events to be omitted from the recorded stream. These gaps reduce the raw per-access coverage but do not preclude recovering informative

pressure profiles (time-varying access intensity). HOME, therefore, treats sampled events as noisy, thinned observations of the actual access stream. It applies a temporal interpolation step (implemented as timestamp scaling and count rescaling) to estimate the original access pressure used by our hierarchy models. Importantly, HOME does not attempt to recreate unobserved load/store sequences or synthesize missing addresses; instead, it produces an estimated temporal pressure profile that feeds the cache/DRAM models. Section 4 formalizes the interpolation operator, the underlying assumptions, and the confidence conditions that flag runs where interpolation is likely unreliable.

HOME is modular by design. MemGaze is the default trace source, but the pipeline accepts any data producer that supplies the minimal fields required by our models (timestamp, address, core/CPU identifier, and access type). When inputs are restricted (for example, missing instruction pointers or limited sampling), HOME still operates, but some higher-fidelity diagnostics are disabled; confidence checks inform the user of these limitations. HOME also uses lightweight hardware counters as complementary signals for validation and for triggering additional diagnostics when counters and interpolated pressure profiles disagree.

3.2 HOME architecture

Building upon the trace collection capabilities of MemGaze described in the previous section, this section presents the complete HOME architecture, which is designed to analyze memory contention and bottlenecks across different levels of the memory hierarchy. The input traces collected by MemGaze are passed through configurable cache and memory models, facilitating contention analysis while maintaining adaptability for diverse hardware configurations. The goal of HOME is to capture detailed insights into performance bottlenecks at lower levels of the hierarchy, where timing and latency properties become more critical.

Table 1: Cache configuration inputs for HOME (parameters used to configure the cache hierarchy and coherence behavior).

Parameter	Description
Capacity	Cache size (e.g., 32KB for L1)
Associativity	Number of ways per set
Line Size	Size of the cache line (e.g., 64 bytes)
Access Latency	Cache access latency (in cycles)
Directory-Based Coherence	Coherence tracking enabled at directories

3.2.1 Configurable Cache Hierarchy models. HOME's architecture is designed for computational efficiency and scalability, employing a modular design for modeling the cache and memory hierarchy as seen in Figure 3. At higher levels of the hierarchy, such as private L1 caches and shared L2 caches, contention metrics are analyzed in an event-driven manner, without relying on explicit timing information. Metrics such as miss rates serve as key indicators of contention patterns at these levels. To simplify coherence tracking and improve simulation speed, HOME combines all private caches into a unified model, also unifying coherence directories into a single directory structure at the private cache level. Within the unified private cache, memory requests are checked for hits or coherence misses, leveraging coherence metadata to avoid unnecessary inter-hierarchy communication.

Requests that result in coherence misses propagate directly to the last-level cache (LLC), which acts as the unified memory buffer for resolving unresolved requests. The LLC serves as the final arbiter, ensuring consistency across cores and efficiently managing cache misses. By assuming cache requests are ultimately determined at the LLC, HOME reduces computation complexity while maintaining the hierarchical integrity of the memory system. Using Dinero, a state-of-the-art cache simulator, we empirically evaluated HOME's unified directory and private cache approach across 19 applications from 5 benchmark suites (as detailed in Section 5.1). Our evaluation demonstrates that this approach introduces cache miss rate errors of less than 1%, while improving simulation speed by an average of 38.76% compared to traditional multi-level cache architectures.

HOME's cache hierarchy configuration inputs, summarized in Table 1, include essential attributes such as capacity, associativity, line size, access latency, and whether directory-based coherence is enabled. These configurable parameters allow HOME to adapt to diverse architectural designs, ensuring flexibility and scalability for various applications and workloads. Importantly, all traces and addresses filtered through the cache hierarchy retain and pass down the original metadata collected by MemGaze—such as instruction pointers (IPs), timestamps, and CPU identifiers (CPU_{id}). This detailed metadata is critical for identifying low-level causes of contention across the hierarchy.

HOME's modular design also enables detailed examinations of cache behavior beyond traditional metrics. By combining cache access information with address metadata, HOME can identify frequently accessed cache regions, detect cache set conflicts, and analyze whether cache accesses are distributed across different sets or concentrated on a few overburdened sets. Such insights offer researchers a deeper understanding of cache bottlenecks and opportunities for optimizing memory systems.

3.2.2 Page Map Collection. After memory accesses are filtered using the cache hierarchy models presented in Section 3.2.1, their virtual addresses must be translated into physical addresses using page maps before further analysis at the memory models. To accomplish this, HOME integrates a lightweight page map collection script that operates alongside MemGaze to gather physical address information essential for analyzing hardware-level memory behavior. Although MemGaze primarily collects virtual addresses of load-store operations, physical address resolution is crucial for understanding deeper memory characteristics, such as DRAM row activation patterns, bank-level contention, and resource utilization. The script leverages kernel-level information exposed through */proc/<pid>/pagemap* and */proc/<pid>/maps*, dynamically extracting virtual-to-physical address mappings during program execution. These mappings serve as a critical input for HOME's memory hierarchy analysis.

The page map collection script continuously monitors the target process (PID) for changes in memory usage statistics, including Resident Set Size (RSS) and Proportional Set Size (PSS), using */proc/<pid>/smaps_rollup*. Upon detecting significant changes that

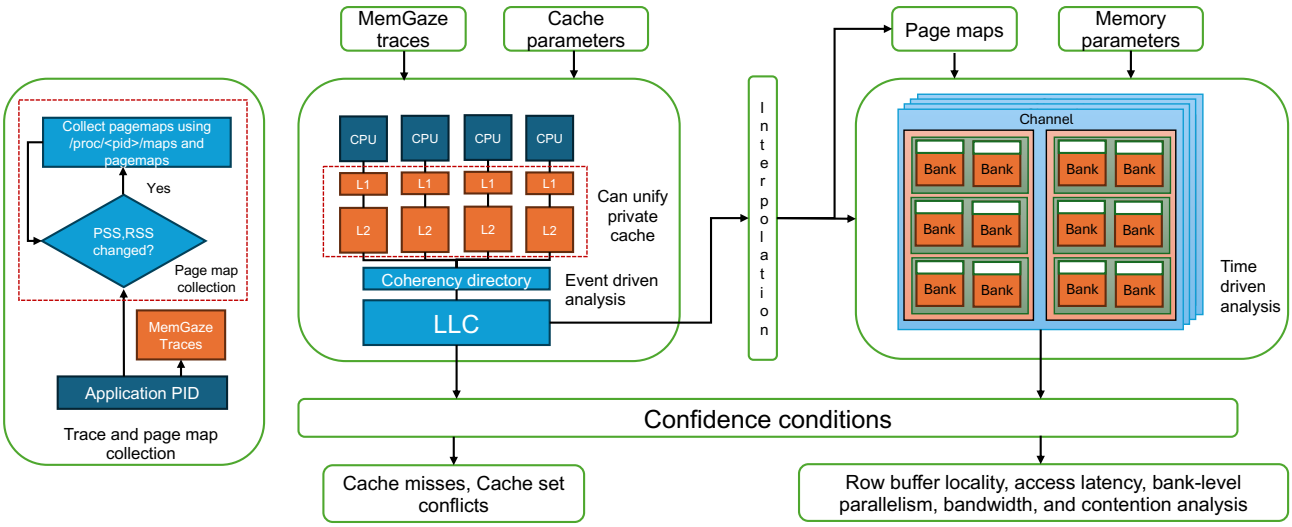

Figure 3: HOME system architecture: trace/page-map collection, configurable cache hierarchy, and time-driven DRAM models. The figure highlights how sampled events and page maps are combined, how cache filtering produces L3→DRAM events, and where interpolation and confidence checks are applied.

exceed predefined thresholds, the script retrieves virtual address ranges from */proc/<pid>/maps* and resolves these addresses to Physical Frame Numbers (PFNs) via */proc/<pid>/pagemap*. The resulting virtual-to-physical address pairs are stored in timestamped output files, ensuring trace completeness and alignment with MemGaze's collected data.

Beyond collecting page map information, HOME also utilizes a secondary script to aggregate all collected physical address mappings and resolve potential address conflicts. This step ensures consistency in the address translation process and addresses inaccuracies arising from dynamic memory remapping or conflicting data. The final compiled page map provides the necessary input for accurate address translation within HOME's analysis framework, enabling reliable identification of hardware-level behaviors such as memory access latency, DRAM row locality, and bank-level parallelism.

The page map collection process is performed concurrently ensuring real-time monitoring without disrupting the execution of the target process. By seamlessly integrating trace collection and physical address resolution, HOME delivers reliable, fine-grained memory analysis across the hierarchy while remaining adaptable for diverse applications and system workloads.

3.2.3 Configurable memory models. At lower levels of the memory hierarchy, such as memory channels, ranks, and banks, timing and contention properties become the primary focus. These levels experience significantly higher access latencies and greater contention, making precise timing information essential for understanding memory access pressure and resource utilization. To address sampling issues in MemGaze traces—where timing gaps may arise—HOME employs interpolation techniques (discussed in Section 4.1) to reconstruct missing timestamps, ensuring accurate representation of memory behavior. This approach allows HOME to integrate both high-fidelity simulation capabilities and efficient real-time modeling workflows.

HOME's memory models feature detailed configurable parameters, similar to those in advanced simulation tools, including channel, rank, bank, and timing settings, as well as address mapping configurations, as shown in Table 2. These parameters allow HOME to adapt to a wide range of hardware architectures and workloads, providing researchers with the ability to model memory contention patterns. Unlike cycle-accurate simulation tools, such as DRAMsim3, which update memory states at every clock cycle, HOME adopts an interval-modeling approach for latency modeling that significantly improves computational efficiency. Instead of continuously updating memory states, HOME evaluates requests only at arrival times. Upon the arrival of a new request, HOME analyzes the completion status of earlier requests, updates the queues of memory components such as banks, bank groups, ranks, and channels, and schedules the new request accordingly. This event-driven process eliminates unnecessary updates during idle periods with high inter-arrival latency, making it considerably faster than granular per-cycle analysis.

This modeling approach is efficient for workloads with bursty or unpredictable patterns, where requests are spaced out by varying inter-arrival times. HOME's selective evaluations ensure accurate modeling of contention and latency metrics while avoiding the computational overhead associated with cycle-by-cycle simulations. Additionally, this design retains key insights into queued request behavior, resource utilization, and access latency without sacrificing accuracy.

HOME's configurable memory model also enables granular analysis of contention dynamics at lower levels of the hierarchy. It provides insights into key metrics, including:

Table 2: Memory configuration parameters for HOME (configurable DRAM channel, timing and queue parameters used by the DRAM model).

Parameter	Description
Channels	Number of memory channels
Ranks	Number of ranks per channel
Bank Groups	Total bank groups
Banks per Group	Banks within each group
Rows	Rows per bank
Columns	Columns per row
Channel Size (MB)	Size of each memory channel
DRAM Size (MB)	Total DRAM size
Transaction Queue Length	Maximum length of the transaction queue
Command Queue Length	Max length of the command queue
Command Queue Per Bank	Whether queue is per bank
tCK	Clock cycle time (ns)
CL	Column access latency
CWL	Column write latency
tRCD	Row-to-column delay
tRP	Row precharge delay
tRAS	Row activation time
tWR	Write recovery time
tRTP	Row-to-precharge delay
tCCD_S	Short col-to-col delay
tCCD_L	Long col-to-col delay
tRRD_S	Short row-to-row delay
tRRD_L	Long row-to-row delay
tRTRS	Rank-to-rank switch latency
Address Mapping	Address mapping scheme used

Row Buffer Locality (RBL): Determines how efficiently consecutive memory accesses use the same row in DRAM, reducing contention and improving performance.

Bank-Level Parallelism (BLP): Measures the ability to overlap multiple memory accesses across different DRAM banks, reducing stalls and increasing throughput.

Bank Access Pressure: Evaluates the intensity of contention at individual memory banks, highlighting potential bottlenecks.

Access Latency: Tracks delays caused by contention, providing an understanding of how resource utilization impacts performance.

Memory Bandwidth Utilization: Assesses how contention reduces overall memory throughput, particularly at shared memory channels.

Transaction Queue Length: Measures the number of requests waiting in memory transaction queues, which directly influences latency under high contention.

Stall Cycles: Tracks cycles where memory operations are stalled, helping quantify periods of heavy contention and resource saturation.

Metrics such as bank access pressure, transaction queue length, and stall cycles are crucial for evaluating the impact of bursts and flurries of memory requests that cause resource contention. On the other hand, metrics such as Row Buffer Locality (RBL) and Bank-Level Parallelism (BLP) offer insights into how aligned data access patterns can be optimized for memory operations and minimized to reduce contention delays. By combining these metrics with metadata from MemGaze, such as instruction pointers (IPs), HOME can trace contention back to specific functions, enabling researchers to identify application-level behaviors that adversely impact memory performance.

4 ERROR MITIGATION

Random sample drops during trace collection produce gaps that can distort timing-driven memory analyses by underestimating the instantaneous access pressure in memory-bound applications. If left unaddressed, these gaps cause contention metrics (e.g., instantaneous pressure or bandwidth utilization) to deviate from real-world execution trends. To mitigate this problem, HOME integrates error-mitigation strategies based on temporal interpolation and confidence conditions; our experimental validation shows these measures keep analysis errors small under a wide range of practical sampling regimes.

4.1 Interpolation algorithm

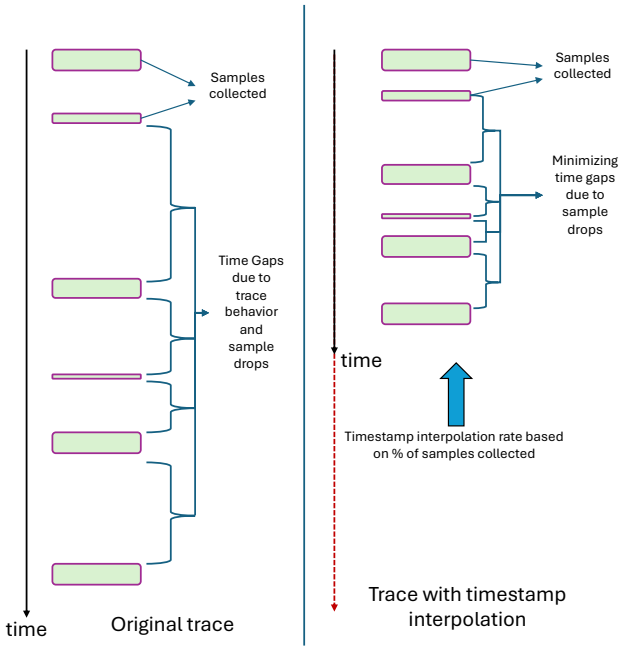

Figure 4: Temporal interpolation (timestamp scaling) example. Left: raw MemGaze sample timeline with gaps caused by sample drops. Right: timestamp-scaled and count-rescaled pressure profile used by HOME's time-driven models. Note: HOME **does not synthesize unobserved addresses; it rescales observed timestamps and counts to estimate time-varying intensity.**

Figure 4 summarizes HOME's temporal interpolation workflow. PTWRITE-style traces are often thinned by platform effects (buffer

overflows, interrupt latency, etc.), so the recorded timestamps form a sparse, noisy sampling of the true event stream rather than a complete log. HOME therefore treats sampled events as thinned observations and produces an *estimated* temporal access-pressure profile rather than attempting to reconstruct unobserved per-access sequences.

Concretely, let $r \in (0, 1]$ denote the measured sampling ratio (fraction of true events observed). For an observed sample with timestamp s_i, HOME computes a scaled timestamp

$$\hat{t}_i = r \cdot s_i,$$

Intuitively, under approximate random thinning of the event stream, these operations restore the time-varying intensity of the original process: scaling compresses the sampled timeline and count-rescaling compensates for omitted events. We refer to this procedure as *temporal interpolation* (implemented via timestamp scaling and count rescaling) and use the resulting pressure profile as input to the cache and DRAM hierarchy models. This temporal interpolation has been widely used in previous studies [12, 13, 37] due to its effectiveness in enabling coarse-grained or macro-level analyses.

HOME determines the sampling ratio r by comparing MemGaze-sampled counts to lightweight hardware counters (e.g., `mem_inst_retired.any`, collected via `perf`). These counters can either be captured concurrently while recording MemGaze traces or by running a separate instance of the program during the trace decoding process. Since the trace decoding stage takes significantly longer than the actual application runtime, capturing these counters does not add overhead to HOME's execution. Since the PTWRITE trace decoding stage takes considerably longer than the actual application runtime, capturing these counters does not add overhead to HOME's execution. Typical r values were on the order of a few percent (commonly 1–4%) for the MemGaze settings used. When r is sufficiently large and the access stream is not dominated by extremely short, high-intensity bursts, temporal interpolation provides reliable pressure estimates for macro- and function-level analyses.

We emphasize the assumptions and failure modes of this method. Timestamp-scaling interpolation assumes sampling is approximately random (thinning) and that the process is locally stationary enough for scaling to be meaningful; it performs poorly when the workload exhibits synchronized, sub-window bursts or highly non-stationary, phase-like behavior that sparse samples cannot capture. To detect such cases, HOME applies explicit *confidence conditions* (described below) that combine sampling diagnostics and counter comparisons to flag estimates that are likely unreliable. When confidence fails, HOME recommends remedy actions (increase sampling density, run merged/replicated traces, or instrument for shorter phases) rather than presenting possibly misleading fine-grained timing results.

HOME's current linear interpolation / temporal-interpolation approach is effective for whole-application and coarse function-level analyses (memory bursts, throughput trends, bandwidth estimates). It is less precise for instruction-level or very short basic-block analysis; as future work we plan to explore IPC-aware and phase-aware interpolation refinements to improve fidelity for highly localized timing analyses.

HOME dynamically adjusts interpolation parameters based on the computed sampling ratio and the confidence checks. The result is a pragmatic balance: lightweight sampling plus temporal interpolation yields time-aware pressure inputs that are adequate for cache/DRAM modeling in many HPC scenarios, while confidence conditions and diagnostics guide the user when higher-fidelity collection is required.

4.2 Confidence conditions

Confidence conditions establish guidelines for achieving acceptable error rates in contention metrics across different levels of the memory hierarchy. These thresholds, derived through empirical observations of real-world workloads, ensure that sampling density is sufficient to produce reliable results, even for sparse sampling scenarios. While synthetic workloads can satisfy certain confidence criteria, they often represent only a subset of possible memory behaviors, failing to capture the diverse and non-uniform address distributions observed in real applications. HOME explicitly establishes and validates its thresholds using practical benchmarks with varied memory access patterns—such as random, strided, and compute-intensive phases—ensuring its heuristics are robust and applicable to real-world conditions.

Additionally, these thresholds have an error tolerance up to 30%. In computer architecture research, simulation models with an error margin up to 30% are often considered sufficient for guiding architectural decisions, given the trade-off between accuracy and simulation speed. Previous work has shown that simulators such as Sniper [9], gem5 [7], and Tejas [33] consistently achieve error rates in the range of 10% to 40% when validated against commercial hardware for various workloads. These error margins are widely recognized as a reasonable trade-off between simulation speed and modeling fidelity, especially when performing architectural exploration or system-level studies [1, 8, 24].

To establish its confidence conditions, HOME estimates sampling density by leveraging the `perf` counter `mem_inst_retired.any`, as described in Section 4.1. By combining sampling density metrics with empirically validated thresholds, HOME ensures reliable modeling of memory hierarchy contention metrics for a diverse range of applications.

4.2.1 L2 Cache. At higher levels of the memory hierarchy, such as the L2 cache, frequent memory accesses naturally result in robust sampling densities, improving reliability for whole-application-level or larger-function-level analysis. Thanks to the law of averages (and pathological cases rendering the L2 ineffective), L2 contention metrics are statistically accurate even if sampling density dips slightly below optimal levels. This makes analysis at the L2 level more resilient to gaps in sampling and extremely effective for evaluating broader workloads with high access rates. The confidence condition for L2 cache, although arbitrary, our heuristic is shown to be:

$$\text{Error Rate (L2)} < 10\% \quad \text{if} \quad \text{Sampling Density (L2)} > 0.0005 \quad (1)$$

This condition requires at least 8 samples per 1000 memory access events, ensuring consistent results when modeling contention dynamics at the L2 level. The frequent memory accesses at this level inherently contribute to more statistically valid sampling results, making L2 cache analysis highly reliable across a variety of applications.

4.2.2 L3 Cache (LLC). In contrast, shared caches such as the L3 cache (LLC) experience fewer memory accesses in workloads with limited spatial locality or low access rates. Sparse sampling at this level can result in scenarios where the L3 cache is not sufficiently warmed up during trace collection. This leads to compulsory misses, where every memory access fails to find relevant data in the cache. Our experimentally validated heuristics demonstrate that for reliable modeling of contention, cache warming conditions require that the number of accesses to L3 exceed twice the number of cache lines available to ensure sufficient data population and minimize inaccuracies. The confidence condition for L3 cache is given by:

$$\text{Error Rate (L3)} < 30\% \quad \text{if} \quad \text{Sampling Density (L3)} > 0.001 \tag{2}$$

For warming conditions, the L3 access threshold is:

$$\text{Accesses (L3)} \geq 2 \times \text{Cache Lines Available} \tag{3}$$

These conditions are specifically tailored for shared cache levels like LLC, where sparse trace sampling or low workload locality may result in unreliable contention metrics without sufficient warming. Ensuring that enough accesses populate the cache reduces inaccuracies caused by compulsory misses during evaluation.

4.2.3 Memory Bandwidth Utilization. Metrics for memory bandwidth utilization rely heavily on the behavior of shared caches, such as the L3 cache (LLC), since bandwidth modeling examines interactions across the memory hierarchy. Sparse workloads may exhibit lower accuracy in bandwidth utilization analysis unless the L3 cache sampling and warming criteria are met. Therefore, memory bandwidth shares the same thresholds as L3 cache evaluation. The confidence condition for memory bandwidth utilization is:

$$\text{Error Rate (BW)} < 30\% \quad \text{if} \quad \text{Sampling Density (BW)} > 0.001 \tag{4}$$

Meeting these thresholds ensures that memory bandwidth contention trends are accurately reflected in HOME's analysis. Workloads with low access rates or bandwidth-intensive applications may require additional sampling techniques to improve reliability.

4.2.4 Enhancing Sampling Density Through Merging. If the sampling density criteria for any level of the hierarchy are not met, future work will focus on running multiple sampling iterations and merging traces to improve coverage. During this merging process, functional granularity (FG) elements—such as instruction pointers (IPs), timestamps, and CPU identifiers—can serve as references to ensure consistency across aggregated traces. This approach will enhance sampling robustness, particularly for workloads that require finer-grained fidelity or exhibit limited locality. These initial confidence criteria establish the reliability of analyses obtained from limited sampling, while advancements in merging workflows will further improve the accuracy and scope of contention modeling. By defining these conditions, HOME establishes a solid foundation for scalable and reliable modeling across diverse workloads while maintaining flexibility for enhancements.

5 EXPERIMENTAL RESULTS

Accurately modeling memory contention across the hierarchy requires evaluating the framework's ability to balance modeling accuracy, computational efficiency, and practical applicability. This section presents the results of HOME's framework through comparisons with state-of-the-art simulation tools and real hardware systems, validating its effectiveness across cache emulation, memory modeling, and computational speedups. By analyzing HOME's outputs across diverse workloads, we assess its applicability in both synthetic scenarios and real-world environments.

The experimental setup used for these evaluations is described in Section 5.1. Trace data was collected using MemGaze, leveraging Intel PTWRITE functionality to generate timestamps, instruction pointers, memory addresses, and CPU identifiers. Benchmarks included multi-threaded HPC workloads with varying memory access patterns (e.g., random, strided, and streaming). HOME's results were compared against simulation tools like Dinero, DRAMsim3, and Sniper, as well as validated against real hardware metrics to test the impact of sampling drops and establish modeling accuracy. The subsections analyze HOME's performance across key components: cache modeling, memory hierarchy analysis, and computational speedup evaluations.

5.1 Experimental Setup

Experiments were conducted on the system specified in Table 4, with the goal of capturing reliable trace data and evaluating HOME across diverse workloads. Traces were collected using MemGaze and processed uniformly across simulation tools and hardware validation workflows. Benchmarks included high-performance computing (HPC) applications drawn from widely used suites, such as GAP [5], NAS Parallel Benchmarks (NPB) [6], miniVite [18], algebraic multigrid (AMG) [27], and Sw4lite [31], covering a range of random, strided, and streaming access patterns. Tools such as Dinero [14], DRAMsim3 [29], and Sniper [9] were configured consistently for fair comparisons.

Table 3 lists the applications used in our evaluation, together with their source benchmark suites and a short description of their dominant memory access pattern. We include all benchmarks actually executed (and mark the few dropped workloads with explanations). These workloads were chosen to cover a diverse mix of memory behaviors (irregular graph traversal, streaming/stencil, strided, compute-bound), and were not selected to favor HOME's results — the set is intended to exercise interpolation and confidence checks across representative HPC patterns.

5.2 Analysis of cache hierachy models

In this section, we analyze HOME's accuracy in single-threaded and multi-threaded environments. To validate the accuracy of HOME's cache model, we first compared its results with those of Dinero, a widely used cache simulator that supports single-core, multi-level cache hierarchy simulations. Since HOME and Dinero can process the same trace inputs, Dinero was used as a baseline for validating HOME's accuracy across single and multi-level cache hierarchies under various cache configurations. Our analysis demonstrates that HOME estimates cache miss rates with an error of less than 0.5% compared to Dinero across different workload types and cache configurations. These experiments underscore HOME's reliability in emulating cache behavior at higher levels of the hierarchy efficiently.

Table 3: Benchmarks used in the evaluation: suite, dominant access/workload type, and usage note. 'Used' denotes run in experiments; 'Dropped' lists excluded benchmarks with reason. This set covers irregular, streaming, strided, and compute-bound patterns.

Application	Suite	Access / Workload Type	Notes (Used / Dropped)
amg	AMG	stencil / streaming (multigrid)	Used
bc	GAP	betweenness-centrality (irregular graph)	Used
bfs	GAP	breadth-first search (irregular graph)	Used
pr	GAP	PageRank (irregular graph / streaming)	Used
cc	GAP	connected components (irregular)	Used
sssp	GAP	single-source shortest path (irregular)	Used
tc	GAP	triangle counting (irregular, heavy)	Dropped – runtime issues
bt	NPB	block tridiagonal solver (strided / compute-bound)	Used
cg	NPB	conjugate gradient (sparse linear algebra)	Used
ep	NPB	embarrassingly parallel (compute-dominated)	Used
ft	NPB	FFT (strided / global communication)	Used
is	NPB	integer sort (random access / memory-traffic heavy)	Used
lu	NPB	LU decomposition (dense / strided)	Used
mg	NPB	multigrid (stencil-like / streaming)	Used
sp	NPB	scalar pentadiagonal solver (stencil/streaming)	Used
miniVite (v1–v4)	miniVite	Louvain community detection (irregular graph accesses, communication-heavy)	Used (variants v1–v4)
sw4lite	SW4lite	seismic wave solver (stencil / streaming)	Used

Table 4: Experimental Setup of real hardware system, validation simulators, as well as our benchmark suites used for validation

Specification	Details
Processor	Intel Core i9-12900KF(24 cores, 48 threads)
RAM	128 GB DDR5 @ 4800 MT/s
Cache Hierarchy	40 KB L1, 1.4MB L2, 30 MB shared L3
Operating System	Ubuntu 22.04.5 LTS (Kernel: 6.8.0-52-generic)
Sampling Tool	MemGaze[25]: Intel PTWRITE with trace output (timestamps, IPs)
Simulation Tools	Dinero, DRAMsim3, Sniper
Benchmark suites used	GAP, NPB, AMG, Sw4lite, miniVite
Trace Content	Timestamps, instruction pointers, virtual memory addresses, CPU IDs

However, Dinero is limited to single-core environments and does not account for multi-core scenarios. To evaluate HOME's performance in multi-core environments, we utilized Sniper, which supports multi-core and multi-threaded workloads. For this experiment, we also compared HOME's outputs with real hardware results, as described in Section 5.1. Both HOME and Sniper were configured with the same cache hierarchy parameters (summarized in Table 4), with the number of threads set to 4. Hardware miss rate data was collected using `perf` during execution on real hardware systems. However, Sniper's evaluation was limited to GAP benchmarks due to compatibility constraints.

Figure 5 showcases the error rates of HOME and Sniper's cache models compared with real hardware results. As seen in the figure, HOME exhibits lower error rates for L2 cache compared to Sniper, with average error rates of 2.64% for HOME and 4.67% for Sniper. This high accuracy at the L2 cache level can be attributed to the higher number of accesses occurring in L2 cache, where the law of averaging smooths out the effects of sampling drops in MemGaze-generated traces. However, for L3 cache, the impacts of sampling drops are more evident in HOME's outputs, leading to error rates of 20.08% and 24.10% for workloads such as *bfs* and *sssp*, respectively. On average, Sniper outperforms HOME at the L3 cache level with an error rate of 4.67%, compared to HOME's average error rate of 13.82%. This is primarily because Sniper collects every single cache access trace accurately, while HOME uses interpolation techniques to handle sampling drops, prioritizing computational efficiency over full trace completeness. However, Sniper's cycle-accurate approach incurs significant computational overhead, as discussed in Section 5.4.

To evaluate the performance of HOME's confidence conditions in low-sample scenarios, we analyzed 19 applications from five benchmark suites (detailed in Table 3) against hardware-derived cache miss data collected using `perf`. Figure 6 shows per-application error rates for the L2 and L3 cache models together with the percentage of trace samples captured by MemGaze. For clarity, we report two averages: AWOCC (average without confidence conditions, i.e., using all traces) and AWCC (average with confidence conditions, i.e., excluding traces that failed the checks).

At the L2 level no application was filtered by the confidence conditions; the maximum single-app error was 8.08% (the *mg* benchmark) and the average error (both AWOCC and AWCC) was 2.10%,

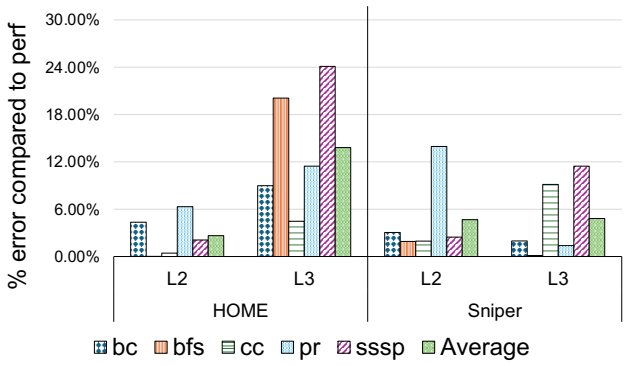

Figure 5: L2 and L3 cache miss-rate error vs. hardware (`perf`) for HOME and Sniper. Bars show mean error per level; error computed as |model `perf`|/`perf`. Sniper captures all accesses (cycle-accurate) and hence tends to be more accurate at L3 but is substantially slower; HOME uses interpolation and confidence checks to trade some L3 fidelity for much higher throughput (see Section 5.4).

indicating robust behavior of HOME's L2 model under our sampling regimes.

At the L3 level, the confidence conditions pre-filtered five applications (*cg*, *ep*, *is*, *sp*, and *sw4lite*) before any L3 miss-rate analysis was performed. These traces failed one or more of the defined criteria (insufficient cache warm-up—measured as total accesses or sampling-density thresholds based on the MemGaze sampling ratio), so they were excluded from AWCC. When these under-sampled traces are nevertheless considered (AWOCC), their L3 errors are high (51.03%, 30.71%, 43.17%, 38.92%, and 12.00% respectively), which explains why the confidence checks rejected them. Note that *sw4lite* is numerically lower than the others but still failed the sampling criterion and was therefore filtered.

Overall, average L3 errors dropped from 16.57% (AWOCC) to 9.92% (AWCC), demonstrating that the confidence conditions successfully remove runs likely to produce unreliable L3 cache miss estimates. The confidence conditions were configured to enforce an operational upper bound on acceptable L3 error (30% in our experiments); all traces that passed the confidence checks produced L3 errors below this bound. When a trace fails confidence, we recommend increasing sampling density (merged or replicated runs) or using targeted instrumentation for the affected code regions rather than relying on the interpolated timing results.

5.3 Analysis of memory model

This section evaluates HOME's memory-modeling accuracy by two complementary comparisons: (1) a trace-driven comparison against DRAMsim3, and (2) validation against real-hardware metrics collected with `perf`. For the DRAMsim3 comparison we begin with the same sampled MemGaze address/timestamp events for both tools. These samples are processed with timestamp-scaling interpolation and count rescaling and are passed through HOME's cache hierarchy; the event stream of L3->DRAM requests that HOME's L3 model produces (addresses observed in the sampled trace, with interpolated timestamps) is then supplied to DRAMsim3. In other words, DRAMsim3 consumes the exact L3->DRAM event stream that HOME would see; therefore, differences in DRAM-level outputs reflect modeling choices (interval-based timeline handling in HOME vs. cycle-accurate DRAM timing in DRAMsim3) rather than input variability.

Table 5 reports percentage errors of HOME relative to DRAMsim3 across three metrics: access latency, inter-arrival latency, and bandwidth. HOME matches DRAMsim3 closely for inter-arrival latency and bandwidth (average errors 0.01% and 0.56%, respectively), demonstrating that timestamp-scaling interpolation preserves the temporal intensity needed for rate- and throughput-oriented metrics. By contrast, HOME exhibits a larger average error for access latency (16.16%). This difference is expected: HOME uses an interval-based modeling approach that advances memory state using completion times and delta timestamps, whereas DRAMsim3 applies cycle-accurate scheduling and updates every clock cycle. Interval modeling is significantly faster but sacrifices the per-cycle granularity required to match cycle-accurate latency in all cases, particularly for workloads with fine-grained timing interactions or where queueing/serialization effects dominate. Future work will explore transaction-level refinements to improve latency fidelity without reverting to full cycle-accurate costs [34].

Table 5: Percentage error of HOME relative to DRAMsim3 on identical sampled inputs. Values show how interval modeling and temporal-interpolation preserve throughput-oriented metrics (inter-arrival, bandwidth) but yield larger errors for per-access latency (see text).

Benchmark	Access Latency (%)	Inter-arrival Latency (%)	Bandwidth (%)
amg	22.35	0.01	0.01
bc	23.34	0.01	0.12
bfs	4.77	0.00	1.07
bt	21.39	0.00	0.83
cc	15.41	0.00	1.33
cg	23.24	0.00	0.01
ep	24.76	0.00	5.87
ft	29.01	0.00	0.17
is	8.12	0.01	0.23
lu	2.00	0.00	0.11
mg	16.90	0.01	0.01
miniVite-v1	12.45	0.00	0.07
miniVite-v2	16.97	0.01	0.12
miniVite-v3	21.45	0.01	0.23
miniVite-v4	1.98	0.01	0.17
pr	21.26	0.02	0.03
sp	3.19	0.01	0.10
sssp	22.02	0.00	0.11
sw4lite	16.37	0.14	0.02
Average	**16.16**	**0.01**	**0.56**

We further validate HOME by comparing bandwidth utilization (observed bandwidth divided by theoretical peak) across 19 applications. For hardware references, we collected total memory reads,

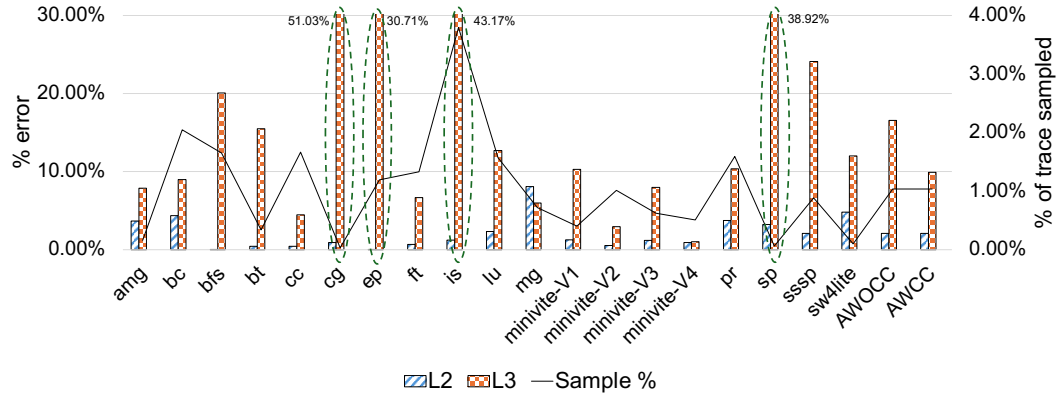

Figure 6: Per-benchmark L2 and L3 cache-miss error (bars) and MemGaze sampling% (trendline). AWOCC shows averages including all traces; AWCC excludes traces failing confidence conditions (marked with dashed ellipsoids). The five benchmarks filtered at L3 are listed in the text; filtering reduces mean L3 error from 16.57% to 9.92%.

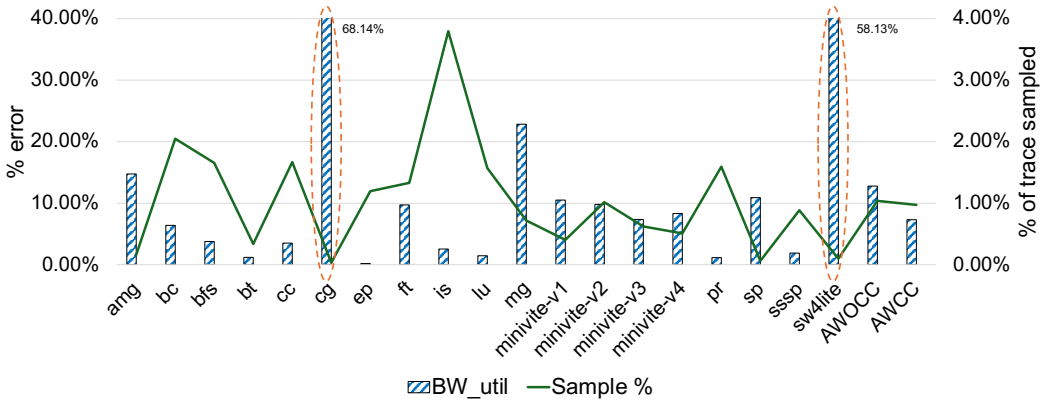

Figure 7: Memory bandwidth utilization (observed / theoretical peak) per benchmark: hardware (`perf`) vs. HOME, shown with and without confidence condition filtering. Bars: per-benchmark utilization (HOME); green trendline: MemGaze sampling% (right axis). Red ellipsoids mark traces rejected by HOME's confidence checks (excluded from AWCC). Mean absolute errors: AWOCC = 12.76%, AWCC = 7.33%.

writes, and execution time using `perf`; theoretical peak bandwidth for each case was computed from the DRAM configuration parameters used in the trace-driven experiments (channels, DIMM width, frequency). Because the physical test machine's DIMM and channel topology differed from the best-available DRAM timing parameters we could find, HOME runs used a best-fit DRAM configuration rather than an exact timing record of the hardware. In the hardware baseline, we therefore report utilization relative to the real machine's peak bandwidth (as derived from its actual configuration) while HOME uses the standardized target DRAM configuration for the trace-driven comparison. These choices ensure that the trace-driven HOME comparison isolates modeling differences, while the hardware comparison (HOME vs `perf`) remains a direct, practical validation of bandwidth utilization estimation. For brevity, we omit per-benchmark DRAMsim3 bandwidth tables from the main text since the bandwidth numbers for HOME and DRAMSIM3 were quite similar.

Figure 7 compares per-benchmark utilization values from HOME and the hardware baseline. Without applying confidence conditions (AWOCC) the mean absolute error is 12.76%; after filtering traces that fail the confidence checks (AWCC) the mean error falls to 7.33%. Confidence checks remove traces with insufficient sampling density or inadequate cache warm-up (e.g., *cg*, *ep*, *is*, *sp*, *sw4lite*), which otherwise present very large interpolation errors (for instance, *cg* shows 68% error when used despite insufficient sampling). Excluding these under-sampled application traces yields a substantially lower average error, illustrating HOME's accuracy when interpolation assumptions hold. We also observed that DRAMsim3's memory-bandwidth-utilization estimates are broadly similar to HOME's for these trace-driven inputs; for brevity, we omit the per-benchmark DRAMsim3 results from Figure 7 and the main text.

5.4 Performance analysis

In this subsection, we analyze the performance speedup achieved by HOME compared to Sniper. In the previous subsection 5.2, we observed that Sniper outperforms HOME in terms of L3 accuracy rates, primarily because Sniper captures detailed and complete traces for analyzing cache miss rates. However, this high accuracy comes at the cost of significantly slower runtime, making Sniper inefficient for larger workloads compared to HOME. Figure 8 illustrates the speedup achieved by HOME over Sniper, showing an average speedup of 26.8x, with a maximum speedup of 50.46x for the *pr* benchmark.

For this analysis, we ran GAP benchmarks to generate and evaluate Kronecker graphs with a degree of 21. Over five applications, Sniper required a cumulative execution time of 19.7 hours, whereas HOME completed the same analysis in 45.46 minutes, demonstrating its computational efficiency. The runtime measured for HOME includes all stages: binary instrumentation, trace collection, decoding the traces, and analyzing them using memory hierarchy models. Among these stages, the most time-consuming task for HOME was address decoding, which is handled by MemGaze and utilizes `perf`. Notably, instruction decoding—the stage that decompresses PT data sampled during runtime—accounts for approximately 79% of the total runtime for HOME across the five GAP benchmarks.

For Sniper, we configured the *clock skew minimization barrier* parameter to 100, enabling it to group 100 events for faster analysis. Reducing this parameter to 1 would have allowed Sniper to perform GEM5-like access-to-access analysis, but the computational time required would have been significantly higher. In contrast, HOME employs a different approach that avoids excessive granularity while still achieving meaningful access-to-access analysis by leveraging Intel's PT. Additionally, HOME mitigates the impact of sample drops through interpolation techniques and confidence conditions integration.

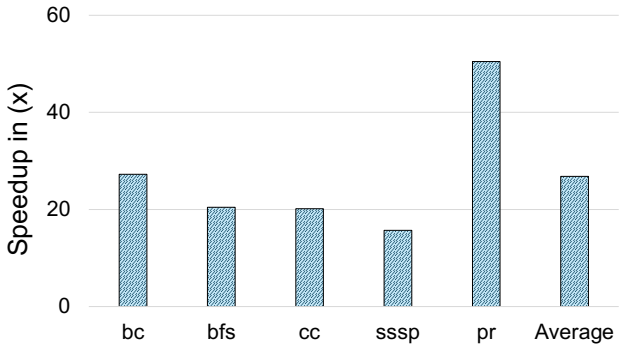

Figure 8: Speedup of HOME vs. Sniper on GAP experiments (lower is better runtime). HOME achieves an average speedup of 26.8× (Sniper timeout/overhead tradeoffs shown in §5.4). Runtime includes instrumentation, trace decoding, and model analysis.

6 DISCUSSION ON FINE-GRAIN CONTENTION ANALYSIS

HOME has proven effective for macro-level analysis, enabling researchers to evaluate contention and bottlenecks across the memory hierarchy with high computational efficiency. However, extending HOME to support micro-level contention analysis remains a key area for future work. Micro analysis often requires a higher sample ratio, which can be achieved by collecting and merging data from multiple runs. This approach would allow HOME to reconstruct memory access pressure with greater fidelity in smaller, critical regions of the workload.

To target specific regions of the application's memory behavior, an IPC-based interpolation algorithm is proposed as a refinement to the existing linear interpolation techniques. IPC (instructions-per-cycle) focuses on capturing the fine-grained fluctuations in memory access patterns across smaller time slices and would mitigate errors introduced by sampling drops in applications with variable execution phases. Furthermore, incorporating confidence conditions based on application access type distributions (e.g., random, strided, constant accesses) as well as instruction mixes (e.g., load/store-heavy phases or compute-intensive operations), would allow HOME to model memory contention more accurately. Instruction mix randomness can exacerbate the impacts of sampling drops, and adapting confidence conditions to include these distributions would reduce inaccuracies in smaller workload regions.

Another avenue for improvement involves the use of call graph information to guide targeted sampling and interpolation during fine-grained analysis. By identifying critical functions or regions contributing to memory bottlenecks, HOME could focus its sampling and trace collection on these high-impact areas, improving data quality with minimal performance overhead. Through these enhancements—higher sampling ratios, IPC-based interpolation, adaptive confidence conditions, and targeted sampling—HOME could extend its applicability to fine-grained analyses, enabling deeper insights into contention dynamics across HPC applications.

7 CONCLUSION

This work introduces HOME (Hierarchy-Oriented Memory Evaluation), a lightweight framework that combines low-overhead PT-based sampling (via MemGaze) with temporal interpolation (timestamp scaling), configurable cache/DRAM models, and runtime confidence checks to provide fast, interpretable contention diagnostics. By estimating time-varying access pressure from sparse samples (rather than attempting to recreate unobserved per-access sequences), HOME occupies a practical point in the fidelity–overhead design space: it preserves time- and address-aware signals where they matter while avoiding the cost of cycle-accurate replay. In our evaluation across 19 applications from five suites, HOME attains average errors of 2.10% (L2), 9.92% (L3 with confidence filtering), and 7.33% (bandwidth with confidence filtering) relative to hardware baselines, while achieving roughly 26.8× speedup over Sniper for the GAP experiments.

We also documented limitations and failure modes: timestamp-scaling interpolation assumes approximately random thinning of the event stream and can fail for extremely short, synchronized bursts or very low sampling ratios; confidence conditions detect

such cases and exclude them from timing-sensitive analyses. HOME's access-latency estimates intentionally trade some per-cycle fidelity for performance (leading to the observed gap vs. cycle-accurate DRAMsim3 in access-latency), and future work will explore transaction-level refinements and IPC-aware interpolation to narrow this gap.

REFERENCES

[1] Ayaz Akram and Lina Sawalha. 2016. × 86 computer architecture simulators: A comparative study. In *2016 IEEE 34th International Conference on Computer Design (ICCD)*. IEEE, 638–645.

[2] Todd Austin, Eric Larson, and Dan Ernst. 2002. SimpleScalar: An infrastructure for computer system modeling. *Computer* 35, 2 (2002), 59–67.

[3] Shouvik Bardhan and Daniel A Menascé. 2014. Predicting the effect of memory contention in multi-core computers using analytic performance models. *IEEE Trans. Comput.* 64, 8 (2014), 2279–2292.

[4] Javier Barrera, Leonidas Kosmidis, Hamid Tabani, Jaume Abella, and Francisco J Cazorla. 2022. Contention tracking in GPU last-level cache. In *2022 IEEE 40th International Conference on Computer Design (ICCD)*. IEEE, 76–79.

[5] Scott Beamer, Krste Asanović, and David Patterson. 2015. The GAP benchmark suite. *arXiv preprint arXiv:1508.03619* (2015).

[6] NAS Parallel Benchmarks. 2006. Nas parallel benchmarks. *CG and IS* (2006).

[7] Nathan Binkert, Bradford Beckmann, Gabriel Black, Steven K Reinhardt, Ali Saidi, Arkaprava Basu, Joel Hestness, Derek R Hower, Tushar Krishna, Somayeh Sardashti, et al. 2011. The gem5 simulator. *ACM SIGARCH computer architecture news* 39, 2 (2011), 1–7.

[8] Anastasiia Butko, Florent Bruguier, Abdoulaye Gamatié, Gilles Sassatelli, David Novo, Lionel Torres, and Michel Robert. 2016. Full-system simulation of big. little multicore architecture for performance and energy exploration. In *2016 IEEE 10th international symposium on embedded multicore/many-core systems-on-chip (MCSOC)*. IEEE, 201–208.

[9] Trevor E Carlson, Wim Heirman, and Lieven Eeckhout. 2011. Sniper: Exploring the level of abstraction for scalable and accurate parallel multi-core simulation. In *Proceedings of 2011 International Conference for High Performance Computing, Networking, Storage and Analysis*. 1–12.

[10] Milind Chabbi and John Mellor-Crummey. 2016. Contention-conscious, locality-preserving locks. In *Proceedings of the 21st ACM SIGPLAN Symposium on Principles and Practice of Parallel Programming*. 1–14.

[11] Sanjeev Das, Jan Werner, Manos Antonakakis, Michalis Polychronakis, and Fabian Monrose. 2019. Sok: The challenges, pitfalls, and perils of using hardware performance counters for security. In *2019 IEEE symposium on security and privacy (SP)*. IEEE, 20–38.

[12] Andreas De Blanche and Thomas Lundqvist. 2015. Addressing characterization methods for memory contention aware co-scheduling. *The Journal of Supercomputing* 71, 4 (2015), 1451–1483.

[13] Alexandre Denis, Emmanuel Jeannot, and Philippe Swartvagher. 2023. Predicting Performance of Communications and Computations under Memory Contention in Distributed HPC Systems. *International Journal of Networking and Computing* 13, 1 (2023), 62–91.

[14] Jan Edler. 1994. Dinero IV: Trace-driven uniprocessor cache simulator. *http://www. cs. wisc. edu/˜ markhill/DineroIV* (1994).

[15] David Eklov, Nikos Nikoleris, David Black-Schaffer, and Erik Hagersten. 2012. Bandwidth bandit: Understanding memory contention. In *2012 IEEE International Symposium on Performance Analysis of Systems & Software*. IEEE, 116–117.

[16] Pouya Esmaili-Dokht, Francesco Sgherzi, Valeria Soldera Girelli, Isaac Boixaderas, Mariana Carmin, Alireza Monemi, Adria Armejach, Estanislao Mercadal, German Llort, Petar Radojković, et al. 2024. A mess of memory system benchmarking, simulation and application profiling. In *2024 57th IEEE/ACM International Symposium on Microarchitecture (MICRO)*. IEEE, 136–152.

[17] Stijn Eyerman, Wim Heirman, Kristof Du Bois, Joshua B Fryman, and Ibrahim Hur. 2018. Many-core graph workload analysis. In *SC18: International Conference for High Performance Computing, Networking, Storage and Analysis*. IEEE, 282–292.

[18] Sayan Ghosh, Mahantesh Halappanavar, Antonino Tumeo, Ananth Kalyanaraman, and Assefaw H Gebremedhin. 2018. miniVite: A graph analytics benchmarking tool for massively parallel systems. In *2018 IEEE/ACM Performance Modeling, Benchmarking and Simulation of High Performance Computer Systems (PMBS)*. IEEE, 51–56.

[19] Valéria S Girelli, Francis B Moreira, Matheus S Serpa, Danilo Carastan-Santos, and Philippe OA Navaux. 2021. Investigating memory prefetcher performance over parallel applications: From real to simulated. *Concurrency and Computation: Practice and Experience* 33, 18 (2021), e6207.

[20] Shingo Igarashi, Takuro Fukunaga, and Takuya Azumi. 2021. Accurate contention-aware scheduling method on clustered many-core platform. *Journal of Information Processing* 29 (2021), 216–226.

[21] Intel Corporation. 2023. *Intel® 64 and IA-32 Architectures Software Developer's Manual: Volume 3 (System Programming Guide)*. https://www.intel.com/content/www/us/en/developer/articles/technical/intel-sdm.html Section 36: Intel® Processor Trace.

[22] Javier Jalle, Mikel Fernandez, Jaume Abella, Jan Andersson, Mathieu Patte, Luca Fossati, Marco Zulianello, and Francisco J Cazorla. 2016. Contention-aware performance monitoring counter support for real-time MPSoCs. In *2016 11th IEEE Symposium on Industrial Embedded Systems (SIES)*. IEEE, 1–10.

[23] Matthias Jung, Kira Kraft, Taha Soliman, Chirag Sudarshan, Christian Weis, and Norbert Wehn. 2019. Fast validation of DRAM protocols with timed petri nets. In *Proceedings of the International Symposium on Memory Systems*. 133–147.

[24] Konstantinos Kanellopoulos, Konstantinos Sgouras, F Nisa Bostanci, Andreas Kosmas Kakolyris, Berkin Kerim Konar, Rahul Bera, Mohammad Sadrosadati, Rakesh Kumar, Nandita Vijaykumar, and Onur Mutlu. 2025. Virtuoso: Enabling fast and accurate virtual memory research via an imitation-based operating system simulation methodology. In *Proceedings of the 30th ACM International Conference on Architectural Support for Programming Languages and Operating Systems, Volume 2*. 1400–1421.

[25] Ozgur O Kilic, Nathan R Tallent, Yasodha Suriyakumar, Chenhao Xie, Andrés Marquez, and Stephane Eranian. 2022. MemGaze: Rapid and Effective Load-Level Memory Trace Analysis. In *2022 IEEE International Conference on Cluster Computing (CLUSTER)*. IEEE, 484–495.

[26] Johannes Langguth, Xing Cai, and Mohammed Sourouri. 2018. Memory bandwidth contention: Communication vs computation tradeoffs in supercomputers with multicore architectures. In *2018 IEEE 24th International Conference on Parallel and Distributed Systems (ICPADS)*. IEEE, 497–506.

[27] Lawrence Livermore National Laboratory. 2017. AMG: Algebraic Multigrid Benchmark (Code-LLNL-738-322). https://github.com/LLNL/AMG. Accessed: 2025-06.

[28] Shang Li, Dhiraj Reddy, and Bruce Jacob. 2018. A performance & power comparison of modern high-speed dram architectures. In *Proceedings of the International Symposium on Memory Systems*. 341–353.

[29] Shang Li, Zhiyuan Yang, Dhiraj Reddy, Ankur Srivastava, and Bruce Jacob. 2020. DRAMsim3: A cycle-accurate, thermal-capable DRAM simulator. *IEEE Computer Architecture Letters* 19, 2 (2020), 106–109.

[30] Biswadip Maity, Bryan Donyanavard, and Nikil Dutt. 2020. Self-aware memory management for emerging energy-efficient architectures. In *2020 11th International Green and Sustainable Computing Workshops (IGSC)*. IEEE, 1–8.

[31] N Anders Petersson and Björn Sjögreen. 2015. Wave propagation in anisotropic elastic materials and curvilinear coordinates using a summation-by-parts finite difference method. *J. Comput. Phys.* 299 (2015), 820–841.

[32] Daniel Sanchez and Christos Kozyrakis. 2013. ZSim: Fast and accurate microarchitectural simulation of thousand-core systems. *ACM SIGARCH Computer architecture news* 41, 3 (2013), 475–486.

[33] Smruti R Sarangi, Rajshekar Kalayappan, Prathmesh Kallurkar, Seep Goel, and Eldhose Peter. 2015. Tejas: A java based versatile micro-architectural simulator. In *2015 25th International Workshop on Power and Timing Modeling, Optimization and Simulation (PATMOS)*. IEEE, 47–54.

[34] Lukas Steiner, Matthias Jung, Felipe S Prado, Kirill Bykov, and Norbert Wehn. 2020. DRAMSys4. 0: a fast and cycle-accurate systemC/TLM-based DRAM simulator. In *Embedded Computer Systems: Architectures, Modeling, and Simulation: 20th International Conference, SAMOS 2020, Samos, Greece, July 5–9, 2020, Proceedings 20*. Springer, 110–126.

[35] Sukeshini, Priyanka Sharma, Mohit Ved, Janaki Chintalapti, and Supriya N Pal. 2021. Big data analytics and machine learning technologies for HPC applications. In *Evolving Technologies for Computing, Communication and Smart World: Proceedings of ETCCS 2020*. Springer, 411–424.

[36] Wei Wang, Tanima Dey, Jack W Davidson, and Mary Lou Soffa. 2014. Dramon: Predicting memory bandwidth usage of multi-threaded programs with high accuracy and low overhead. In *2014 IEEE 20th International Symposium on High Performance Computer Architecture (HPCA)*. IEEE, 380–391.

[37] Felippe Vieira Zacarias, Rajiv Nishtala, and Paul Carpenter. 2020. Contention-aware application performance prediction for disaggregated memory systems. In *Proceedings of the 17th ACM International Conference on Computing Frontiers*. 49–59.

[38] Matteo Zini, Giorgiomaria Cicero, Daniel Casini, and Alessandro Biondi. 2022. Profiling and controlling I/O-related memory contention in COTS heterogeneous platforms. *Software: Practice and Experience* 52, 5 (2022), 1095–1113.