Beyond Core Isolation: A Characterization of Non-Linear Memory Bandwidth Interference in Multi-Tenant Workloads

Wei-An Chen

Abstract

Although contention on shared LLC and DRAM bandwidth is a known limitation of multi-tenant systems, the practical behavior of such interference under strict CPU core pinning has not been fully characterized. Rather than questioning the existence of interference, this work focuses on when it emerges, how abruptly performance degrades, and whether the degradation follows a linear or regime-based pattern. This study presents a black-box characterization of memory interference in a virtualized WSL2 environment. We hypothesize that memory interference exhibits non-linear behavior and is dependent on memory access granularity. By systematically sweeping block sizes from 1KB to 64MB, we identify a distinct performance cliff: workloads whose working sets fit within the LLC remain largely protected (latency-bound), while workloads exceeding the LLC capacity experience severe performance degradation due to bandwidth contention (bandwidth-bound). Our results show a peak performance drop of 49.9% at a block size of 32MB, demonstrating that physical CPU core isolation alone is insufficient to guarantee QoS for memory-intensive applications.

1 Introduction

While it is well understood that shared LLC and DRAM bandwidth can induce cross-tenant interference, the shape of this interference remains under-characterized in practical multi-tenant settings. Rather than asking whether interference exists, we ask when it emerges, how abruptly it degrades performance, and whether the degradation is monotonic with respect to access granularity. Under strict core pinning, we systematically vary the victim's memory access block size and observe a clear regime shift from cache-resident behavior to bandwidth-bound collapse, including non-monotonic phenomena at intermediate sizes. Hypothesis: We hypothesize that memory interference exhibits non-linear behavior. We predict that workloads will remain protected as long as their working set fits within the LLC (Shielding Effect). However, once the access granularity exceeds the effective LLC capacity, the system will hit a "Bandwidth Wall," triggering a sharp performance cliff and significant Quality of Service (QoS) degradation.

2 Background and Motivation

2.1 The Multi-tenant Interference Problem

Modern processors share the Last-Level Cache (LLC) and the memory controller across all cores.1 When a high-bandwidth "aggressor" (e.g., a streaming application) runs alongside a victim application, it can evict the victim's data from the LLC ("Cache Thrashing") or saturate the memory bus ("Bandwidth Saturation"). This creates a scenario where a tenant's performance depends heavily on the behavior of its neighbors, violating the principle of isolation.

2.2 Challenges in Virtualized Environments

Conducting architectural profiling in consumer-level virtualized environments such as Windows Subsystem for Linux 2 (WSL2) introduces a fundamental black-box constraint. Unlike bare-metal Linux systems, where hardware Performance Monitoring Units (PMUs) expose fine-grained events (e.g., LLC misses or memory bandwidth counters), WSL2 abstracts these low-level hardware interfaces, preventing direct observation of uncore resource usage. Rather than treating this limitation as a drawback, we adopt it as a deliberate methodological choice. In practical cloud environments, tenants similarly lack privileged access to host-level hardware counters and must reason about performance interference using only application-visible metrics. Therefore, a black-box characterization approach more accurately reflects the perspective of real-world cloud users. By relying exclusively on application-level throughput measurements and controlled sensitivity sweeps, we infer underlying architectural bottlenecks without assuming access to internal hardware signals. In particular, sharp and repeatable transitions in performance behavior can be used to identify regime shifts between cache-resident and bandwidth-bound execution. This methodology enables robust architectural characterization under constrained visibility and ensures that our findings remain applicable to public cloud settings where low-level counters are unavailable.

3 Experimental Setup

3.1 Host Hardware Specifications

All experiments were conducted on a host machine featuring the Intel Core Ultra 5 125H processor (Meteor Lake architecture). The hardware specifications are critical for interpreting the interference boundaries:

- `Processor`: Intel Core Ultra 5 125H (14 Cores: 4P + 8E + 2LPE).
- `Cache Hierarchy`:
  - L1 Cache: 112 KB per P-core.
  - L2 Cache: 2 MB per P-core.
  - Last-Level Cache (L3): 18 MB (shared).
    Note: This capacity is the key architectural parameter tested in our block size sweep.
- `Memory`: 32GB DDR5 (Host), utilizing high-bandwidth memory to support the integrated NPU/GPU architecture.

3.2 Virtualization Environment (WSL2 Configuration)

The test environment runs on Ubuntu 22.04 LTS within WSL2. To mitigate host-side noise and ensure consistent resource availability, we explicitly configured the WSL2 VM parameters via .wslconfig:

- Memory Limit: Restricted to 8GB. This constraint is critical as it forces memory contention to occur earlier and prevents

Table 1: Host Hardware and Virtualization Specifications

Component	Specification	Notes
Processor Model	Intel Core Ultra 5 125H	Meteor Lake Architecture
Core Count	14 Cores (4P + 8E + 2LPE)	Victim/Aggressor pinned to P-Cores
L1 Cache	112 KB (per P-core)	Private
L2 Cache	2 MB (per P-core)	Private
L3 Cache (LLC)	**18 MB**	Shared (Key Bottleneck)
Host Memory	32 GB DDR5	High-bandwidth
WSL2 VM Limit	**8 GB**	Configured via .wslconfig
Logical Processors	4	Restricted for isolation

the Linux kernel from utilizing excessive page cache to mask memory latency during large block transfers.

- `Processor Limit`: Restricted to 4 logical processors. This strictly controls the placement of the Victim (Core 0) and Aggressor (Core 1) threads while leaving ample headroom for OS background tasks.

3.3 Workload Configuration

- `Victim`: sysbench memory
  - Access Pattern: Random Read/Write.
  - Block Sizes: 1KB, 128KB, 4MB, 16MB, 32MB, 64MB.
- `Aggressor`: stress-ng –stream 4
  - Mechanism: Spawns 4 workers performing continuous streaming memory operations (non-temporal stores) to flood the memory bus.
- `Isolation Strategy`:
  - Victim pinned to Core 0 (using taskset -c 0).
  - Aggressor pinned to Core 1 (using taskset -c 1).

3.4 Preliminary Control Experiment

Specificity Validation To validate that the observed performance degradation stems specifically from memory subsystem contention rather than CPU scheduler latency, we conducted a preliminary control experiment. We compared a compute-bound victim against our memory-bound victim under the same interference conditions.

- Compute-bound Victim: We executed sysbench cpu (prime number calculation), which operates almost entirely within CPU registers and the private L1 cache.
- Memory-bound Victim: We executed sysbench memory with a working set larger than the LLC.

Observation: Under identical aggression from stress-ng, the compute-bound victim maintained 99.8% of its baseline performance, whereas the memory-bound victim suffered significant degradation. This control experiment confirms that core isolation (pinning) successfully mitigates pipeline contention but fails to isolate the shared memory path. Consequently, the remainder of this study focuses exclusively on the memory-bound victim to characterize this specific vulnerability.

3.5 Methodology: The Block Size Sweep

We conducted a systematic parameter sweep of memory block sizes. For each block size, we:

Table 2: Control Experiment: Sensitivity Analysis

Victim Type	Workload	Isolation	Drop (%)
Compute Bound	sysbench cpu	Pinned	**< 0.2%**
Memory Bound	sysbench memory	Pinned	**> 40.0%**

Note: The compute-bound workload is immune to interference under core pinning, whereas the memory-bound workload suffers significant degradation.

Table 3: Performance Degradation by Block Size

Block Size	Average Drop (%)	State Classification
1KB	-0.7%	Latency Bound (Shielded)
128KB	-7.8%(gain)	System Anomaly
4MB	3.5%	LLC Resident
16MB	22.8%	The Architectural Cliff
32MB	49.9%	Bandwidth Saturated
64MB	38.6%	Bandwidth Saturated

(1) Measured the baseline throughput with the victim workload running alone.
(2) Measured the interference throughput with the victim and aggressor running concurrently.
(3) Calculated the performance drop as a percentage relative to the baseline.
(4) Repeated each experiment three times and reported the average to reduce measurement noise.

4 Results and Characterization

Our experiments revealed a clear, non-linear relationship between access granularity and interference sensitivity. The results are summarized below: The overall results are summarized in Table 3 and Figure 1.

4.1 The Shielding Effect (1KB - 4MB)

For small block sizes (1KB) and medium sizes (4MB), the performance degradation was negligible (< 4%). This validates the "LLC Shielding Effect." Since these block sizes fit within the processor's 18MB L3 cache, the victim workload rarely needed to access the main memory. As a result, it remained immune to the bandwidth contention caused by the aggressor on the memory bus.

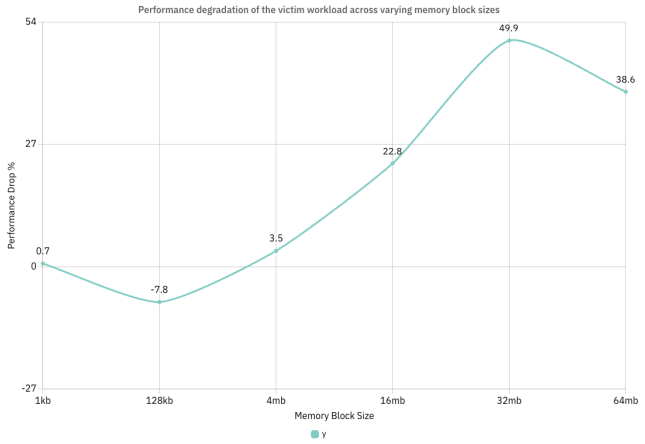

Figure 1: Line Chart showing Drop % vs. Block Size
Performance degradation of the victim workload across varying memory block sizes. The data reveals a non-linear "Performance Cliff" starting at 16MB, marking the transition from LLC-shielded to bandwidth-bound states.

4.2 The System Anomaly (128KB)

Contradictory to intuition, the 128KB workload experienced a performance improvement of 7.8% under interference. This "negative interference" is attributed to System-level Power Management (DVFS/Turbo Boost). The introduction of the aggressor workload increased the overall CPU package utilization, prompting the hardware to boost the clock frequency. Since the 128KB workload is still largely cache-resident, it benefited from the higher clock speed without suffering from bandwidth contention.

4.3 The Performance Cliff (16MB - 64MB)

A distinct "Performance Cliff" was identified at the 16MB block size, where degradation jumped to 22.8%. This marks the boundary of the L3 Cache (18MB). Once the working set exceeded the cache capacity (at 32MB and 64MB), the system transitioned into a Bandwidth Bound state. In this region, every memory request forced a DRAM access, colliding directly with the aggressor's stream. This resulted in a catastrophic performance collapse, peaking at a 49.9

5 Discussion

This study reveals three critical architectural phenomena that dictate interference sensitivity:

5.1 The 128KB Anomaly: Power Management vs. Resource Contention

Our observation of a 7.8% performance gain at 128KB highlights the multi-dimensional nature of system performance. In isolation, the victim workload places a light load on the CPU, causing it to run at base frequencies to conserve power. However, the introduction of the aggressor increases the total package power draw, triggering hardware mechanisms such as Intel Turbo Boost or Speed Shift, which elevate the operating frequency of all active cores. Since the 128KB workload fits entirely within the private L2 cache, it benefits from the higher clock cycles without being penalized by the bandwidth contention on the memory bus. This demonstrates that for small-granularity workloads, power management policies can outweigh resource contention effects. The Role of Uncore Frequency Scaling: While core frequency boosting (Turbo Boost) explains part of the 128KB anomaly, the impact of Uncore Frequency Scaling (UFS) is likely a dominant factor. The specific processor (Intel Meteor Lake) utilizes a complex ring interconnect to link P-cores, E-cores, and the LLC. The activation of the aggressor workload signals the Power Control Unit (PCU) to elevate the voltage and frequency of the entire compute tile, including the Ring Interconnect. Since the 128KB victim is latency-sensitive, the reduced latency of the interconnect (facilitating faster L2-L1 transfers and coherency checks) outweighs the negligible bandwidth contention. This observation highlights that in low-contention regimes, "noisy neighbors" can inadvertently act as "helpful neighbors" by keeping the shared infrastructure in a high-performance P-state.

5.2 The 16MB Cliff: Defining the "Effective" LLC Boundary

Although the host processor features an 18MB L3 Cache, we observed a significant 22.8% performance drop at a block size of 16MB. Theoretically, 16MB should fit within 18MB. However, this discrepancy illustrates the concept of "Effective Capacity" in multi-tenant systems. The L3 cache is a shared resource utilized not only by the victim but also by the OS kernel, background processes, and metadata associated with the aggressor. Consequently, a 16MB working set occupies nearly 88% of the physical cache, leading to severe cache thrashing and conflict misses. This finding underscores that tenants cannot rely on the theoretical maximum cache size; the usable boundary is significantly lower in shared environments.

5.3 The Collapse at 32MB: Worst-Case Resonance

Notably, the performance degradation peaked at 49.9% for the 32MB block size, which was more severe than the drop observed at 64MB (38.6%). This non-monotonic behavior suggests a worst-case resonance or alignment conflict in the memory subsystem. At 32MB, the victim's access pattern likely caused maximum contention for DRAM Row Buffers or Translation Lookaside Buffers (TLBs) when interleaved with the aggressor's stream. In contrast, at 64MB, the workload becomes purely bandwidth-saturated, and hardware prefetchers may adapt more effectively to the sustained stream, slightly mitigating the penalty. This indicates that interference severity is not determined solely by data volume, but also by how specific access strides align with the memory controller's architecture.

Refining the "Worst-Case" Resonance (Expanded Analysis): The counter-intuitive recovery of performance at 64MB (38.6% drop) compared to 32MB (49.9% drop) suggests a shift in the hardware's management of memory requests. We posit that at 32MB, the workload resides in a "Thrashing Valley." The working set is large enough to evict useful lines constantly but lacks the distinct streaming signature required to fully engage the memory controller's optimization logic. Conversely, at 64MB, the access pattern becomes definitively

indistinguishable from a linear stream. This likely allows the Hardware Prefetcher to maximize Bank-Level Parallelism (BLP) and potentially triggers LLC Bypassing policies (where data is streamed directly to L2/L1, bypassing the polluted L3). This "streaming optimization" reduces the coherency traffic overhead, mitigating the penalty slightly despite the higher data volume.

5.4 Implications for Cloud Schedulers and Developers

Our findings have significant implications for both cloud providers and application developers:

- Scheduler-Level: Cloud orchestrators (e.g., Kubernetes) currently schedule pods primarily based on CPU and RAM quantity. Our data suggests they must also account for Memory Bandwidth Intensity. Placing two high-bandwidth tenants (e.g., >16MB working sets) on the same physical socket ensures performance degradation. Schedulers should employ anti-affinity rules for bandwidth-heavy workloads.
- Application-Level: Developers can optimize for multi-tenancy by strictly applying Cache Blocking (Tiling) techniques. Ensuring that inner-loop working sets remain within the "Safe Zone" (e.g., <4MB in our testbed) can effectively shield applications from noisy neighbors, utilizing the LLC as a hardware isolation buffer.

6 Threats to Validity

Several factors may influence the interpretation of our results. First, the use of WSL2 introduces additional layers of address translation (SLAT/EPT), which may amplify memory latency compared to bare-metal environments. While this may affect absolute performance values, it does not invalidate the observed regime transitions. Second, our experiments utilize standard 4KB pages. The use of Transparent Huge Pages (2MB) could reduce TLB pressure and potentially shift interference boundaries. Finally, background activity from the host operating system may introduce noise, although repeated measurements and averaging mitigate this effect. Despite these limitations, the qualitative behaviors observed in this study remain robust and reproducible.

7 Conclusion

This characterization study highlights the limitations of CPU core isolation in virtualized environments. By systematically sweeping memory block sizes, we mapped the architectural boundaries of interference. We conclude that:

(1) Interference is Non-Linear: It follows a step-function behavior based on cache residency.
(2) Core Pinning is Insufficient: It fails to protect workloads larger than the LLC, leading to performance drops of up to 50%.

Future cloud schedulers must account for Memory Bandwidth Intensity—not just CPU usage—to ensure strict Quality of Service for large-granularity workloads.