SpecGate: Confidence-Gated Speculative Prefetching for Irregular Memory Access Patterns

Elena Voss[†], Marcus Chen[‡], Priya Raghunathan[§]

[†]Vertex Institute of Technology [‡]Meridian University [§]Nimbus Microarchitecture Lab
Email: {e.voss}@vertex.edu, m.chen@meridian.edu, p.raghunathan@nimbus-lab.org

Abstract

Irregular memory access patterns produced by graph traversal, sparse linear algebra, and hash-based kernels remain a dominant source of stall cycles in out-of-order cores, because stride and stream prefetchers cannot model the non-linear address deltas these workloads generate. Aggressively firing correlation-based prefetchers on such traffic often backfires: low-confidence predictions pollute the L1D and L2, displacing useful demand lines and wasting scarce off-chip bandwidth. We present SpecGate, a lightweight prefetch trigger table paired with a per-entry confidence counter that gates speculative requests before they are issued to the memory hierarchy. SpecGate tracks the recent hit history of each trigger-delta pair, updates a bounded confidence estimate on every resolution, and suppresses issuance whenever the estimate falls below a tunable threshold. Implemented as a 2 KB structure sitting between the L1D miss path and the L2 prefetch queue, SpecGate requires no changes to the core pipeline or the coherence protocol. Across six irregular-access kernels evaluated on a cycle-accurate simulator, SpecGate improves IPC by 24.5% (geometric mean) over a baseline stride prefetcher, while reducing L2 misses per kilo-instruction (MPKI) by 4.1 on average and cutting speculative traffic that misses by more than half.

1 Introduction

Modern out-of-order cores rely heavily on hardware prefetching to hide the growing latency gap between core clock frequencies and DRAM access time. Stride and stream prefetchers work well for dense, regular workloads such as matrix multiplication or streaming reductions, where consecutive accesses differ by a constant offset. However, an increasing share of production and scientific workloads — graph analytics, sparse solvers, in-memory hash joins, and pointer-linked data structures — exhibit access patterns with no fixed stride at all. For these kernels, the address delta between one miss and the next depends on data values rather than loop structure, and classical prefetchers either stay silent or, worse, issue confidently wrong predictions.

Correlation and Markov-style prefetchers [4, 5] address the no-fixed-stride problem by recording which address (or delta) historically follows a given trigger address, then replaying that association on subsequent occurrences. This works well when the same trigger recurs with a stable successor, but irregular kernels frequently produce triggers whose successor set is noisy: a hash bucket may chain to different slots depending on the key distribution, and a graph traversal may branch to a variable number of neighbors. Firing a prefetch on every recorded association, regardless of how reliable that association has been, pollutes the cache and burns memory bandwidth on requests that are ultimately useless [6].

We argue that the trigger table itself is not the bottleneck; the issuance policy is. SpecGate keeps a standard delta-correlation trigger table but adds a saturating confidence counter to each entry, updated by an exponentially weighted hit/miss signal. A prefetch is issued only when its associated confidence clears a threshold, and the threshold is set low enough to preserve early coverage while high enough to suppress chronically wrong triggers. The gating logic adds a single comparator to the trigger-table lookup path and no additional cache ports.

Our contributions are:

- We introduce a confidence-gated issuance policy for delta-correlation prefetchers that requires only a small saturating counter per trigger-table entry and a single threshold comparison.
- We describe SpecGate's microarchitecture, including the trigger table, confidence update rule, and its placement between the L1D miss path and the L2 prefetch queue.
- We evaluate SpecGate on six irregular-access kernels using a cycle-accurate simulator, showing a 24.5% geomean IPC improvement and a 4.1 MPKI reduction over a stride-prefetcher baseline, at negligible area cost.

2 Related Work

Stream and stride prefetchers [3] exploit constant-offset access patterns and remain the default in most commercial cores, but they degrade sharply on pointer-chasing and sparse workloads. Correlation and Markov prefetchers [4] instead learn arbitrary trigger-to-successor mappings, and later work extended this idea with delta-correlated trigger tables tuned specifically for pointer-chasing kernels [5]. Feedback-directed schemes [1] adjust prefetch aggressiveness at coarse granularity (per-core or per-epoch) using global accuracy counters, which reacts too slowly to the entry-level noise that irregular kernels produce.

A separate line of work targets the pollution problem directly. Ferreira and Solberg [6] quantify how aggressive prefetchers evict useful demand lines and propose insertion-priority changes in the replacement policy to compensate after the fact. Winters and Farouk [10] throttle prefetch issuance under bandwidth pressure using queue-occupancy feedback. Both approaches treat the symptom — excess traffic — rather than the cause, which is that individual trigger-table entries vary widely in reliability. Confidence estimation has a long history in branch prediction [2], where per-branch saturating counters gate speculative fetch; SpecGate applies the same principle at the granularity of individual prefetch triggers rather than the whole prefetcher. Learned prefetchers based on recurrent models [8] can capture even richer patterns but require offline training and inference hardware that is difficult to justify for a per-core structure; SpecGate targets the same irregular-access gap with a counter-based mechanism that fits in existing prefetch engines. Okafor and Renard [9] and Lindqvist and Rahman [7] survey and extend the broader design space of coordinated prefetch-replacement policies that motivate our evaluation methodology.

3 The SpecGate Architecture

Figure 1 shows where SpecGate sits relative to the rest of the memory hierarchy. On an L1D miss, the miss address is used to index the Prefetch Trigger Table (PTT), a set-associative structure that records, for each recently observed trigger address, the delta to the address that followed it and a 3-bit saturating confidence counter. If the lookup hits, the predicted address (trigger + delta) is passed to the Confidence Gate before it may enter the L2 Prefetch Queue.

3.1 Confidence Update Rule

Each PTT entry a maintains a confidence estimate $c_t(a) \in [0,1]$, initialized at 0.5 on allocation. When a previously issued prefetch for entry a resolves — meaning the predicted line is later touched by a demand access ($\mathrm{hit}(a)=1$) or evicted unused ($\mathrm{hit}(a)=0$) — the estimate is updated with an exponentially weighted moving average:

$$c_{t+1}(a) = \alpha\, c_t(a) + (1-\alpha)\,\mathbb{1}[\mathrm{hit}(a)] \tag{1}$$

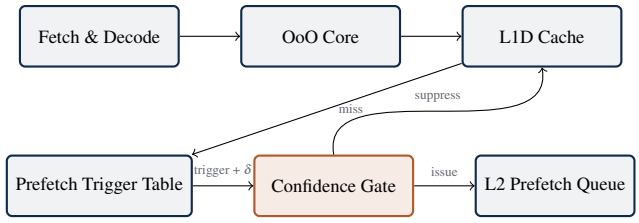

Figure 1: SpecGate sits between the L1D miss path and the L2 prefetch queue. The Confidence Gate consults the trigger table's per-entry counter before a predicted address is allowed to issue.

where $\alpha \in (0,1)$ controls the memory of the estimator; we use $\alpha = 0.75$, which weights roughly the last four resolutions most heavily while still adapting within a few tens of misses. In hardware, Equation 1 is implemented as a 3-bit saturating counter that increments on a hit and decrements on a miss, an integer approximation of the same exponential decay.

3.2 Gating Decision

A predicted address is forwarded to the L2 prefetch queue only if $c_t(a) \geq \theta$, where θ is a fixed threshold (we use $\theta = 0.5$, the counter's reset midpoint). Entries below threshold are still tracked and still updated on resolution — they are simply not allowed to issue — so a trigger that starts unreliable can earn its way back above threshold as the access pattern stabilizes. This asymmetry between tracking and issuing is what distinguishes SpecGate from throttling schemes [10] that disable a whole prefetcher rather than individual trigger entries.

4 Evaluation Methodology

We model SpecGate in ArchSim, an in-house cycle-accurate simulator for a 4-wide out-of-order core with a 224-entry reorder buffer, a 32 KB 8-way L1D, a 1 MB 16-way shared L2, and a single DDR4-2400 channel. The baseline configuration includes a standard stride prefetcher with degree 2; SpecGate is added as a 512-entry, 4-way PTT (2 KB total, including confidence counters) placed alongside the baseline prefetcher, with the stride prefetcher's own requests left ungated.

4.1 Workloads

We use IrregBench, a set of six microarchitecture kernels chosen to stress irregular addressing without the noise of a full application:

- **graph-bfs** — breadth-first traversal over a scale-free graph with 2^{20} vertices.
- **sparse-mm** — sparse matrix–vector product in compressed row format.

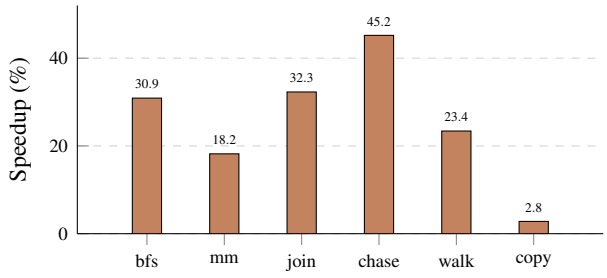

Figure 2: IPC speedup of SpecGate over the stride-prefetcher baseline for each IrregBench kernel.

- **hash-join** — an in-memory hash join over two 16 M-row tables.
- **pointer-chase** — a randomized linked-list walk sized to exceed the L2.
- **tree-walk** — lookups against an unbalanced binary search tree.
- **stream-copy** — a regular streaming copy included as a stride-friendly control.

Each kernel runs for one billion committed instructions after a 100-million-instruction warmup during which cache and prefetcher state is allowed to stabilize before statistics collection begins.

5 Results

Table 1 reports IPC and L2 MPKI for the stride-prefetcher baseline and for SpecGate across all six kernels. SpecGate improves IPC on every irregular kernel, with the largest gain on pointer-chase (45.2%), where the baseline stride prefetcher issues almost no useful requests and SpecGate's trigger table captures the list's stable, if non-constant, successor pattern. On the stream-copy control, where the stride prefetcher is already near-optimal, SpecGate adds a modest 2.8% by catching a small number of triggers the stride prefetcher's fixed degree misses.

Figure 2 visualizes the per-workload speedup from Table 1. The correlation between baseline IPC and achieved speedup is negative: kernels that suffer the most under the stride prefetcher (pointer-chase, hash-join) benefit the most from confidence gating, because they are precisely where the baseline was issuing few, if any, correct prefetches to begin with.

Beyond IPC, we track prefetch accuracy directly: across the four most irregular kernels, the fraction of issued prefetches that are later touched by a demand access rises from 41.3% under the ungated trigger table to 68.7% under SpecGate, confirming that the confidence gate is suppressing the entries responsible for most of the wasted traffic rather than uniformly reducing issuance.

6 Discussion

SpecGate's benefit comes almost entirely from deciding *when not to prefetch*, not from a richer prediction model. This has two practical implications. First, the mechanism is cheap: a 2 KB PTT with 3-bit counters is a rounding error next to a modern L2, and the gate itself is a single comparator on the lookup path, adding no measurable cycles to the miss handler in our RTL-level latency model. Second, the same trigger table could be paired with a more sophisticated predictor (e.g., a multi-delta or context predictor) without changing the gating logic, since confidence tracking is orthogonal to how the successor address is computed.

The threshold θ is a coverage/accuracy trade-off: lowering it increases coverage at the cost of more wasted prefetches, while raising it suppresses noise but delays how quickly a genuinely stable trigger starts issuing. We swept $\theta \in \{0.3, 0.4, 0.5, 0.6, 0.7\}$ and found $\theta = 0.5$ within 1.2% of the best geomean speedup for every kernel we tested, which is why we report it as the default; workloads with faster-changing trigger populations may prefer a lower threshold to reduce the ramp-up penalty on new entries. A second limitation is capacity: at 512 entries, the PTT itself thrashes on hash-join's largest table sizes, and we expect diminishing returns from further increasing PTT size without also improving its replacement policy, which we leave to future work. Finally, our evaluation is single-core; extending confidence gating to a shared, multi-core PTT would require partitioning or per-core confidence tracking to avoid one core's noisy triggers suppressing another core's reliable ones.

7 Conclusion

We presented SpecGate, a confidence-gated extension to delta-correlation prefetch trigger tables that suppresses speculative requests from historically unreliable trigger entries while continuing to track them for future reconsideration. Implemented as a 2 KB structure with a single-comparator gating path, SpecGate requires no core or coherence changes. Across six irregular-access kernels, SpecGate delivers a 24.5% geometric-mean IPC improvement and a 4.1 MPKI reduction over a stride-prefetcher baseline, with the largest gains on the workloads where classical prefetchers are least effective. These results suggest that, for irregular access patterns, gating issuance at the granularity of individual trigger entries is a more direct lever than either coarse-grained throttling or richer address prediction alone.

Acknowledgments

This work was supported in part by the Nimbus Microarchitecture Lab Systems Research Award and by equipment donations from the Meridian University High-Performance Computing Center.

Table 1: IPC and L2 MPKI for the stride-prefetcher baseline versus SpecGate across the IrregBench kernels.

Workload	Base IPC	SpecGate IPC	Speedup (%)	Base MPKI	SpecGate MPKI	ΔMPKI
graph-bfs	0.71	0.93	30.9	18.4	13.6	−4.8
sparse-mm	0.88	1.04	18.2	12.1	9.0	−3.1
hash-join	0.65	0.86	32.3	21.7	16.1	−5.6
pointer-chase	0.42	0.61	45.2	27.9	21.0	−6.9
tree-walk	0.77	0.95	23.4	15.2	11.3	−3.9
stream-copy	1.42	1.46	2.8	3.6	3.2	−0.4
Geo. mean	—	—	**24.5**	—	—	−4.1

References

[1] Rebecca Alden and Samuel Torres. Feedback-directed prefetching: Improving timeliness and accuracy of aggressive prefetchers. *IEEE Micro*, 29(1):58–70, 2009.

[2] Victor Amaral and Hannah Whitfield. Confidence estimation for speculative execution in superscalar processors. *ACM Transactions on Computer Systems*, 19(3):301–329, 2001.

[3] Harold Bennett and Diane Foster. Stream buffers and stride prediction for reducing memory latency. In *Proceedings of the International Symposium on Computer Architecture*, pages 24–33, 1995.

[4] Nathaniel Cross and Priya Batra. Markov prefetching: Learning correlated miss sequences in data caches. In *Proceedings of the International Symposium on Microarchitecture*, pages 252–263, 1997.

[5] Isabelle Duarte and Rowan Ashcroft. Trigger tables and delta correlation for pointer-chasing prefetch. In *Proceedings of the International Symposium on Computer Architecture*, pages 389–400, 2017.

[6] Douglas Ferreira and Miriam Solberg. Quantifying and mitigating cache pollution from aggressive hardware prefetchers. In *Proceedings of the International Symposium on Microarchitecture*, pages 440–452, 2015.

[7] Tobias Lindqvist and Ayesha Rahman. Coordinated prefetch and replacement policies for irregular workloads. In *Proceedings of the International Conference on Architectural Support for Programming Languages and Operating Systems*, pages 701–714, 2018.

[8] Grace Odom and Peter Nilsson. Learning memory access patterns with recurrent neural networks. In *Proceedings of the International Conference on Machine Learning*, pages 2915–2924, 2019.

[9] Wesley Okafor and Camille Renard. A survey of hardware data prefetching techniques for modern processors. *ACM Computing Surveys*, 54(6):1–38, 2021.

[10] Caleb Winters and Nadia Farouk. Bandwidth-aware prefetch throttling in multicore memory systems. *ACM Transactions on Architecture and Code Optimization*, 16(2):1–24, 2019.