Deterministic Integer Topologies: Bypassing the Matrix-Multiplication Bottleneck via Cache-Resident Graph Traversal

Matthew Scott Gibson ORCID: 0009-0001-4167-201X Crimson OS 5 September 2026

Reproduce: `python t112_zero_gemm_bench.py` **C hop harness (Linux** `perf`**):** `gcc -O3 -march=native -std=c11 bench_l2_graph.c -o bench_l2_graph` **Telemetry:** `t112_zero_gemm_telemetry_neo.json`, `t112_zero_gemm_telemetry_local.json` **DOI:** https://doi.org/10.5281/zenodo.22406010 **Table (dataset, not a model):** https://huggingface.co/datasets/UltranetCommand/t112-l2-lattice

Abstract

Dense floating-point matrix multiplication (GEMM) is the invoice of modern inference. Continuous vector spaces do not halt: every interpolation is another lookup, another stall, another joule. We replace that loop with a 6-regular integer lattice of order $N = T_{112} = 6328$ whose entire adjacency table occupies **74.16 KiB**. That is 29% of a Kaby Lake **256 KiB** L2 and 3.6% of a Raptor Lake P-core **2 MiB** L2. Same table, two sockets: **9.76 ns/hop** on an i7-7700 (102M hops/s) and **5.75 ns/hop** on an i7-14700F (174M hops/s). Illegal index: **1.5 µs** and **800 ns**. One *1024 x 1024* float32 GEMM on the 7700 is 7.48 ms - **767,000 hops** in that window. This is not a language model. It is a deterministic state router. The comparison is hops versus FLOPs, 256 KiB L2 versus the memory bus, halt versus drift.

1. The float tax

A continuous vector space has no last point. Between any two states you can insert another. In silicon that insertion is a matrix multiply: load tiles, issue FMA, write back, miss a cache line, wait on the bus. Transformers made that the whole product. The memory wall is not a side effect. It is the architecture.

Three facts:

1. **Working set.** A dense *d x d* float32 pair is *8d^2* bytes in, before the product. At *d=1024* that is 8 MiB - off L2, onto DRAM.
2. **No halt.** GEMM has no structural predicate that says this state is illegal. It returns a float. The meter keeps running.
3. **The bill is the math.** Linear interpolation is billed because the math never closes.

The fix is not a smaller GEMM. The fix is a space that cannot interpolate.

2. Architecture: a cache-resident 6-regular lattice

Let $N = T_{112} = 112cdot113/2 = 6328$. Build a 6-regular graph on these N nodes. Store adjacency as `lut : uint16[N][6]`.

Size: *6328 x 6 x 2* B = *75936* B = *74.15625* KiB.

That is the whole engine. A state is a node id in *{0,...,6317}*. A transition is one load: `node <- lut[node][dir]`. If `node >= N`, **stop**.

Circulant 6-regular graph (offsets *{pm1,pm2,pm3} mod N*). Any 6-regular table of this order has the same footprint.

2.1 Memory layout - pad the table, not the node

Each node is six `uint16_t` edges: **12 bytes, packed**. Align the **base** of the array to a 64-byte cache line. Do not pad each node to 64 bytes.

Layout	Bytes	KiB	vs 256 KiB L2
Tight pack (this paper)	75936	**74.16**	resident
Per-node 64 B pad	404992	**395.5**	**spill**

75936 / 64 = 1186.5 lines; *64/12 ~ 5.33* nodes per line.

`aligned_alloc(64, n)` requires `n` multiple of 64. 75936 is not. Allocate **75968** bytes (1187 lines) and use 75936.

3. Experiment

Same script, two sockets. Neo: i7-7700 @ 3.60 GHz, Kaby Lake, **256 KiB L2/core**, Python 3.11.9. Local: i7-14700F, Raptor Lake Refresh, **2 MiB L2/P-core**, Python 3.12.10. Median of seven trials after warmup. *5 x 10^7* hops. GEMM is vendor BLAS float32 *A@B*.

Hops are **not tokens**. GEMM is **not a transformer**.

4. Results

The 14700F alone would leave an architect an out: 2 MiB P-core L2, of course 74 KiB fits. Neo kills that. Kaby Lake L2 is **256 KiB per core**. 74.16 KiB is **28.9%** of that ceiling.

```
Kaby Lake  i7-7700     [256 KiB L2] -> 74.16 KiB (28.9%) -> 9.76 ns/hop
Raptor Lake i7-14700F  [  2 MiB L2] -> 74.16 KiB ( 3.6%) -> 5.75 ns/hop
```

9.76/5.75 ~ 1.70 x (102M to 174M hops/s). That ratio is clock and IPC. A table that is 29% of a 2017 L2 does not need 2024 cache bloat.

4.1 Two-socket table

Benchmark	i7-7700 (Neo)	i7-14700F (local)	Delta
Microarchitecture	Kaby Lake 14 nm, 4c/8t	Raptor Lake Refresh, 20c/28t	7-year gap
L2 per core	**256 KiB**	**2 MiB (P-core)**	proves <= 256 KiB
Lattice	74.16 KiB	74.16 KiB	identical T_112 deg-6
L2 occupancy	**28.9%**	**3.6%**	both under the 256 KiB line
Median hop	**9.76 ns**	**5.75 ns**	1.70x IPC/clock
Throughput	**102.5 M hops/s**	**173.8 M hops/s**	CPU-bound hop
Fail-closed (poison @ 100)	**1.5 µs**	**800 ns**	hard stop, both sockets
256x256 GEMM	235 µs = **24,101 hops**	220 µs = **38,223 hops**	one GEMM >> 20k states
1024x1024 GEMM	7.48 ms = **767,000 hops**	3.56 ms = **619,000 hops**	GEMM leaves L2; the table does not

4.2 GEMM window, both chips

d	Neo	Local
64	10.1 µs / 1,035 hops	5.8 µs / 1,008 hops
256	235 µs / 24,101 hops	220 µs / 38,223 hops
512	1.03 ms / 106k hops	851 µs / 148k hops
1024	**7.48 ms / 767k hops**	3.56 ms / 619k hops

On Neo, one 1024x1024 float32 GEMM is 7.48 ms. The lattice takes **767,000** steps in that window. 256x256 GEMM wall time is almost the same on both chips (235 µs vs 220 µs). The hop is what scaled with the core.

4.3 What this is

	Lattice hop	Dense GEMM
Working set	74.16 KiB	32 KiB - 8 MiB
Datapath	integer load + index	FMA tile
Halt	node >= N	none
Fits 256 KiB L2	**yes (29%)**	only d=64

5. Why this cuts

Hyperscalers are not out of ideas. They are out of bus. A 74 KiB table does not pay that tax. It still does not pay it on a 2017 256 KiB L2.

A SCADA interlock, a protocol validator, a fail-closed router, a bounded actuator map - those are graphs. They need a hop and a halt.

Continuous math will keep billing. Discrete topology stops.

6. Reproduce

```
python verify_t112.py
python t112_zero_gemm_bench.py
```

Linux: `gcc -O3 -march=native -std=c11 bench_l2_graph.c -o bench_l2_graph` then `bash verify_zero_gemm_disasm.sh`.

GEMM numbers are vendor BLAS, not a naive triple loop.

7. Scope

This paper does not claim a language model, a proof of an open PDE, or a replacement for every GEMM in a training run. It claims one measured fact:

A 6-regular integer lattice of 6328 nodes occupies 74.16 KiB (29% of a 256 KiB Kaby Lake L2), hops in 9.76 ns on an i7-7700 and 5.75 ns on an i7-14700F, rejects an illegal state in 1.5 µs / 800 ns, and walks 767,000 hops in the wall time of one 1024x1024 float32 GEMM on the 7700. The GEMM cannot halt.

8. Proofs (automated)

Proof	File	Guarantee
Packed layout	static_assert in bench_l2_graph.c	sizeof(Node)==12, 6328x12=75936<=256 KiB
Topology	verify_t112.py	6-regular, in=out, one component
Golden checksum	verify_t112.py	50M hops, seed 7, node 4972, SHA-256 9d9bc34b…bec68
Fail-closed	verify_t112.py	poisons 6328 / 65535 / 99999 halt at hop 500
Integer-only hop	verify_zero_gemm_disasm.sh	objdump of run_hops has no FMA/SSE scalar FP

Golden matched on **both sockets**. Same bits.

Appendix A. Byte formula

bytes = N * deg * 2 for uint16. N=6328, deg=6: 75936 bytes.

Appendix B. Fail-closed predicate

```
if next >= N: HALT
```

Appendix C. Raw hop samples (seconds, 5e7 hops)

Neo i7-7700: 0.4851, 0.4864, 0.4878, 0.4879, 0.4889, 0.4927, 0.4963 **Local i7-14700F:** 0.2777, 0.2793, 0.2876, 0.2877, 0.2904, 0.2966, 0.2991