POSTER: Cooperative Wavefronts: WFA and GWFA on a 4096-PE MIMD Many-Core

Wenjie Geng
wenjieg@umich.edu
University of Michigan
Ann Arbor, MI, USA

Noah Kaplan
kaplannp@umich.edu
University of Michigan
Ann Arbor, MI, USA

Reetuparna Das
reetudas@umich.edu
University of Michigan
Ann Arbor, MI, USA

Nathaniel Bleier
nbleier@umich.edu
University of Michigan
Ann Arbor, MI, USA

Abstract

Wavefront alignment (WFA) is the standard gap-affine pairwise aligner; graph WFA (GWFA) extends it to pangenome graphs. Prior accelerators target SIMT GPUs, FPGAs, and PIM DPUs, but not the general-purpose MIMD many-core. We present a systematic mapping of WFA and GWFA onto a MIMD many-core (PEZY-SC3, 4096 PEs), with three transferable contributions: **C1**, SIMT thread-scaling inverts below an L1 cache-share threshold (adding independent threads degrades up to 3.7×); **C2**, a power-of-two wavefront length triggers a 6.5× L2 aliasing slowdown invisible to the aggregate L1 hit rate; **C3**, a two-phase host–device pipeline that turns GWFA's heavy tail into balanced PE occupancy. The result is an architecturally explained map: this class wins on GWFA (3.40×/3.37× dataset-level) and stays competitive on gap-affine WFA (21/24 cells over CPU-16t, at parity with the hyperthreaded CPU-32t) — it pays where a query is wide (GWFA) and only ties where it is deep (WFA). We close with a cache-indexing redesign that removes the C2 class.

CCS Concepts

• **Computer systems organization** → **Multicore architectures**;
• **Applied computing** → *Molecular sequence analysis.*

Keywords

wavefront alignment, GWFA, MIMD many-core, PEZY-SC3, cache aliasing, pangenomics, cooperative threading

1 Introduction and Positioning

Wavefront alignment (WFA) [1] is the standard gap-affine pairwise aligner; graph WFA (GWFA) [2] extends it to pangenome graphs. Both have been accelerated on SIMT GPUs, FPGAs, and PIM DPUs [3–5]; the general-purpose MIMD many-core — thousands of independent processing elements — is the one accelerator class that has received the least attention. We present such a mapping of WFA and GWFA onto a MIMD many-core (PEZY-SC3, 4096 PEs), yielding three transferable design contributions. **C1**: the cache-capacity threshold at which SIMT thread-scaling intuition inverts — below it, adding independent threads *degrades* performance (up to 3.7×). **C2**: a class of performance bug structurally hidden by the aggregate L1 hit rate — a power-of-two wavefront length aliases at the shared L2, a 6.5× slowdown the trusted metric cannot see. **C3**: a two-phase host–device pipeline that turns GWFA's heavy-tailed query latency into balanced PE occupancy. Each is stated machine-independently and should apply to any private-small-L1 barrel many-core. We do not claim PEZY-SC3 is universally fastest — the deliverable is an architecturally explained map of where this class wins (GWFA) and ties (WFA), plus the rules that produced it.

Alignment yields many independent, divergent queries whose wavefront diagonals expand and contract with sequence content — task parallelism over divergent control flow, precisely the MIMD model, where SIMT must serialize divergent branches within a warp. A many-core adds scale: thousands of lightweight PEs absorb a heavy-tailed, branch-divergent query stream. The workload is timely: a single reference genome misses up to 10% of base pairs in underrepresented populations [6], so pangenome (graph) alignment [7] is the growing bottleneck. We study PEZY-SC3 [8] because it is shipping, measurable silicon that exposes per-level L1/L2 hit rates and a PE-stall counter — visibility without which C2 could not be diagnosed. It is a barrel many-core (4096 PEs each hiding latency by fine-grained multithreading, a 2 KB private L1 per PE, a shared per-city L2) small enough that *how many threads share the L1* (C1) and *how the wavefront is laid out in it* (C2) dominate. We organize the results by the wavefront's *width* (independent work per query) versus its dependency-chain *depth* (serial latency per query): GWFA is wide, WFA is deep, and §3 shows the same architecture explains both.

2 Three structural rules

C1 — cache-share vs. working-set (the SIMT intuition inverts). On SIMT, more independent threads means more throughput. On a small private L1 it reverses: once per-thread L1 share falls below the wavefront's per-step working set, adding independent threads degrades performance up to 3.7× (worst-case runtime across query sizes); e.g. at 1→4 threads per PE, PE efficiency collapses 79%→29% and L2 read-hit 70%→58%, while the L1 hit barely moves. Stated as cache-share vs. working-set the rule is machine-independent — it sizes thread count on any barrel many-core before tuning.



This work is licensed under a Creative Commons Attribution 4.0 International License.
PACT '26, Chicago, IL, USA
© 2026 Copyright held by the owner/author(s).
ACM ISBN 979-8-4007-2915-7/2026/10
https://doi.org/10.1145/3838684.3846879

Table 1: C2: a power-of-two `wf_length` cliff (stride 4096, 4% err) is invisible in L1 metrics — L1 read-hit stays ~92% across the spikes while L2 dips and runtime cliffs up to 6.5×; a +17 pad restores it. The signal is in L2 and the stall counter, not L1.

`wf_length`	class	time (ms)	L1 rd hit%	L2 rd hit%
768	non-2^n	371.7	91.9	82.9
1024	**2^{10} spike**	**476.0**	91.9	**78.2**
1041	+17 pad	369.5	91.9	83.0
2048	**2^{11} spike**	**886.1**	91.9	**72.4**
2065	+17 pad	372.3	91.9	82.6
4096	**2^{12} spike**	**2462.7**	92.1	**66.6**
4113	+17 pad	380.3	91.8	82.4

Cooperative threading (PEs sharing one wavefront) is then the only viable direction; SRAM compaction plus 2-bit packed sequences add a further ~2×.

C2 — a profiling-blind L2 aliasing bug. A power-of-two wavefront length makes successive diagonals map to the same L2 cache set, a slowdown of up to 6.5×. The trap: the metric programmers trust most — aggregate L1 hit rate — stays flat at 91–92% across the entire cliff (Table 1), because the conflict surfaces one level deeper, at the shared L2, where a power-of-two wavefront stride creates severe L2 set conflicts. The runtime-correlated signal lives in the L2 hit rate and the PE-stall counter, not any L1 metric. A one-line +17 stride pad removes the aliasing. The contribution is the diagnostic recipe (read L2 hit and the stall counter, not L1), which transfers to any private-small-cache hierarchy with power-of-two indexing.

C3 — host–device tail placement. Pangenome graphs produce heavy-tailed per-query latency: >99.95% of queries finish in microseconds, ~0.04% run for hours. On thousands of PEs with no dynamic work-stealing, the contribution is where the split is placed: a Phase-1 device batch clears the easy majority in lockstep; a Phase-2 host-driven per-score-step loop attacks the tail cooperatively; the routing threshold (0.3–0.5%, Table 2) is set wider than the 0.04% tail to catch all of it. This converts the heavy tail into balanced PE occupancy — isolating a tail is old; doing it without hardware scheduling on a barrel many-core is the transferable rule.

3 Results: the map, and why

All comparisons below are same-node and count-exact. CPU baselines: *CPU-16t* (single socket, 16 threads), *CPU-32t* (hyperthreaded, 32 threads); GWFA vs. its pruning-free CPU reference. **Breadth end — GWFA wins.** On the official pangenomicsBench [7] LR (104K queries) and Chrom (291K) graphs the two-phase pipeline delivers 3.40× and 3.37× dataset-level speedup over the CPU reference, and 21.2×/ 20.5× on the two hardest queries — which alone are 69% and 63% of their datasets' CPU time (per-query data omitted for space; totals in Table 2).

Depth end — WFA is competitive. WFA is depth-dominated; PEZY wins 21 of 24 cells over CPU-16t (geomean 1.19×) and beats WFA-GPU on short/low-error, but against CPU-32t it is at parity and loses to the GPU on long/high-error (Table 3) — competitive, not dominant.

Positioning — the split is architectural, not incidental. A post-submission sweep isolates the mechanism: at fixed depth, peak

Table 2: Dataset-level GWFA on pangenomicsBench: the two-phase pipeline (C3) clears >99.5% on-device (Phase 1) and drives the tail from the host (Phase 2), count-exact vs. the CPU reference.

Metric	LR (104,205)	Chrom (291,650)
Phase 1 coverage	99.51%	99.69%
Phase 2 queries	513 (0.49%)	892 (0.31%)
Total PEZY wall-clock	**246 s**	**≈7,658 s**
CPU reference	837 s	25,842 s
Dataset speedup	**3.40×**	**3.37×**
Match (OK / total)	104,204/104,205	291,647/291,650

Table 3: Gap-affine WFA stride×error grid (5 of the 24 cells, ms): PEZY wins 21/24 over CPU-16t (geomean 1.19×), parity with CPU-32t, beats WFA-GPU only on short/low-error — the depth end, competitive not dominant.

stride×err	PEZY (ms)	/CPU-16t	/CPU-32t	/GPU
128×1%	8.3	1.20	1.33	19.2
512×5%	102.8	1.39	0.68	1.68
2048×8%	1367.6	1.59	1.02	0.47
4096×5%	2357	1.57	1.00	0.39
10240×8%	43768	1.06	0.78	0.27
24 cells	—	21/24, gm 1.19	~parity	short win / long loss

frontier width tracks graph fan-out (337→10,019 as branching rises 1→64), so breadth is graph structure. Both datasets fill the machine with queries; the difference is *within* a query. A GWFA tail query is individually wide — one query has enough diagonals to fill thousands of PEs, work a CPU core cannot parallelize intra-query — and since these wide queries are ~two-thirds of dataset CPU time, C3 routing them to the device turns per-query wins (21×) into the dataset win (3.4×). A WFA query is the opposite: narrow (width $\leq 2s+1$) and deep, with little intra-query parallelism, so the CPU clears the short chain quickly and many such queries only match it. (A different axis from C1: threads sharing one L1 there, query width across PEs here.) Where a query is deep, WFA's min-plus inner op suits a fast out-of-order CPU just as well.

4 Conclusion

We gave a systematic mapping of WFA and GWFA to a MIMD many-core, distilled into three transferable rules — C1 (cache-share vs. working-set), C2 (profiling-blind L2 aliasing), C3 (host–device tail placement) — each stated by mechanism and, we argue, should apply to any private-small-L1 barrel many-core (PEZY-SC4, ET-SoC-1, Prodigy [9]). The class wins on wide GWFA queries and ties on deep WFA. We recommend redesigning the small-L1 cache index (skewed associativity [10] or prime-modulus indexing [11]) to remove the C2 class outright.

References

[1] Santiago Marco-Sola, Juan Carlos Moure, Miquel Moretó, and Antonio Espinosa. 2021. Fast gap-affine pairwise alignment using the wavefront algorithm. *Bioinformatics* 37, 4 (2021), 456–463. doi:10.1093/bioinformatics/btaa777

[2] Haowen Zhang, Shiqi Wu, Srinivas Aluru, and Heng Li. 2022. Fast sequence to graph alignment using the graph wavefront algorithm. arXiv:2206.13574 [q-bio.GN]

[3] Quim Aguado-Puig, Max Doblas, Christos Matzoros, Antonio Espinosa, Juan Carlos Moure, Santiago Marco-Sola, and Miquel Moretó. 2023. WFA-GPU: gap-affine pairwise read-alignment using GPUs. *Bioinformatics* 39, 12 (2023), btad701. doi:10.1093/bioinformatics/btad701

[4] Abbas Haghi, Santiago Marco-Sola, Lluc Alvarez, Dionysios Diamantopoulos, Christoph Hagleitner, and Miquel Moretó. 2021. An FPGA accelerator of the wavefront algorithm for genomics pairwise alignment. In *2021 31st International Conference on Field-Programmable Logic and Applications (FPL)*. IEEE, Piscataway, NJ, USA, 151–159. doi:10.1109/FPL53798.2021.00033

[5] Safaa Diab, Amir Nassereldine, Mohammed Alser, Juan Gómez-Luna, Onur Mutlu, and Izzat El Hajj. 2023. A framework for high-throughput sequence alignment using real processing-in-memory systems. *Bioinformatics* 39, 5 (2023), btad155. doi:10.1093/bioinformatics/btad155

[6] Jordan M Eizenga, Adam M Novak, Jonas A Sibbesen, Simon Heumos, Ali Ghaffaari, Glenn Hickey, Xian Chang, Josiah D Seaman, Robin Rounthwaite, Jana Ebler, Mikko Rautiainen, Shilpa Garg, Benedict Paten, Tobias Marschall, Jouni Sirén, and Erik Garrison. 2020. Pangenome Graphs. *Annual Review of Genomics and Human Genetics* 21 (2020), 139–162. doi:10.1146/annurev-genom-120219-080406

[7] Noah Kaplan, Jan-Niklas Schmelzle, Yufeng Gu, Erik Garrison, Christopher Batten, and Reetuparna Das. 2025. PangenomicsBench: a benchmark suite and characterization of pangenomics. In *2025 IEEE International Symposium on Workload Characterization (IISWC)*. IEEE, Piscataway, NJ, USA, 272–285. doi:10.1109/IISWC66894.2025.00031

[8] Naoya Hatta, Shuntaro Tsunoda, Kouhei Uchida, Taichi Ishitani, Ryota Shioya, and Kei Ishii. 2023. PEZY-SC3: A MIMD Many-core Processor for Energy-efficient Computing. arXiv:2301.07510 [cs.AR]

[9] Tachyum, Inc. 2022. Tachyum Prodigy Universal Processor Whitepaper. https://www.tachyum.com/.

[10] André Seznec. 1993. A case for two-way skewed-associative caches. In *Proceedings of the 20th Annual International Symposium on Computer Architecture (ISCA)*. ACM, New York, NY, USA, 169–178. doi:10.1145/165123.165152

[11] Mazen Kharbutli, Keith Irwin, Yan Solihin, and Jaejin Lee. 2004. Using prime numbers for cache indexing to eliminate conflict misses. In *Proceedings of the 10th International Symposium on High-Performance Computer Architecture (HPCA-10)*. IEEE Computer Society, Washington, DC, USA, 288–299. doi:10.1109/HPCA.2004.10015