Ultra-High Interactivity on NVIDIA GPUs? - TileRT InferenceX

Ultra-High Interactivity on NVIDIA GPUs? - TileRT InferenceX Can TileRT software on NVIDIA GPU compete with Cerebras, Groq LPU, SambaNova? Batch Size 1, Disaggregated engine, high throughput engine Prefill, high interactivity engine decode

AUG 10 https://newsletter.semianalysis.com/p/ultra-high-interactivity-on-nvidia

View in browser

Ultra-High Interactivity on NVIDIA GPUs? - TileRT InferenceX

QQ:1413221435

Can TileRT software on NVIDIA GPU compete with Cerebras, Groq LPU, SambaNova? Batch Size 1, Disaggregated engine, high throughput prefill engine, high interactivity decode engine

BRYAN SHAN, DANIEL NISHBALL, CAM QUILICI, KIMBO CHEN, ALEC IBARRA, AND DYLAN PATEL

AUG 9 • PAID

READ IN APP

Premium-priced “fast modes” are proving that users will pay more for lower latency and faster tokens, potentially yielding higher gross margins. Frontier AI labs such as OpenAI are therefore evaluating purpose-built inference systems, including Cerebras and NVIDIA Groq LPUs that prioritize ultra-high interactivity over maximum batched throughput. Ultra-low latency matters most in interactive workloads, including real-time assistants, and full-duplex voice. OpenAI GPT-Live, for example, can listen and speak simultaneously, making response delay immediately perceptible to the user, described as feeling like Ironman JARVIS.

GPUs perform exceptionally well at high throughput and low-to-medium interactivity, but their architecture is less suited for ultra-low-latency inference. An 8-GPU HGX B200 server provides a theoretical HBM memory bandwidth of 64 TB/s of in aggregate. At batch size 1, GLM-5 at NVFP4 requires only approximately 21

QQ:1413221435

GB of active-parameter traffic per generated token. The B200 HBM bandwidth roofline would therefore **suggest up to 3,047 tokens/s/user without speculative decoding. In practice, GPUs come nowhere close to this limit.**

The gap comes from latency rather than bandwidth. The traditional GPU programming model launches and synchronizes many individual kernels, whose setup and teardown overhead becomes significant at ultra-high levels of interactivity. While these latency costs are less visible at conventional serving speeds, even with CUDA graphs, they dominate as token latency approaches the sub-millisecond Time Per Output Token (TPOT) range. Furthermore, although GPU memory bandwidth increases by roughly 2–3× each generation, memory latency has not improved at all.

While using alternative hardware is popular, there are ways to use GPUs to do this too. This is where TileRT's persistent engine comes in. TileRT statically compiles the entire decode graph into a single persistent kernel on NVIDIA GPUs, maximizing overlap across computation, memory loads and stores, and communication. **On the InferenceX GLM5 FP8 744B benchmark on a single B200 decode server, tileRT has been verified to reach up to 500 tokens/s/user, approximately 3× faster than GB300 NVL72 running traditional inference engines. Iso-cost per output token, TileRT can achieve up to 2x faster interactivity than traditional engines.**

QQ:1413221435

We thank the TileRT maintainers for collaborating on TileRT InferenceX benchmarks and also in general thankful to the vLLM community for their amazing design on the V1 connector. TileRT comes from the same community maintainer organization that built the widely popular TileLang DSL.



Subscribed

With PD disaggregation inference technique, the hyperspecialized TileRT engine handles latency-sensitive decode while throughput-optimized engines such as vLLM and SGLang continuing to serving prefill. The TileRT decode engine is already being deployed in production at Xiaomi for MiMo V2.5 Pro UltraSpeed and ZAI with GLM 5.1 HighSpeed.

In the article, we shall deep dive into the TileRT InferenceX results, what TileRT is, how it composes with the existing inference ecosystem along with the tradeoffs and challenges with TileRT.

QQ:1413221435

We will also elaborate on the tradeoffs of using TileRT on standard GPUs vs. ultra low latency specialized chips like Nvidia Groq LPU, Cerebras and Sambanova, weighing in on if there is a potential for TileRT software running on

GPUs to disrupt these specialist chips' TAM. The SemiAnalysis Accelerator Model provides quarter by quarter estimates of Nvidia LPU30, LPU40, Cerebras WSE-3 & WSE-4 shipments and much more.

InferenceX

InferenceX is our open-source, vendor-neutral, continuously updated AI inference benchmarking and research platform. We measure leading models, inference frameworks, and hardware across the latency-throughput Pareto frontier, tracking how real-world inference performance and economics improve over time.

Thanks for reading SemiAnalysis! This post is public so feel free to share it.

Share

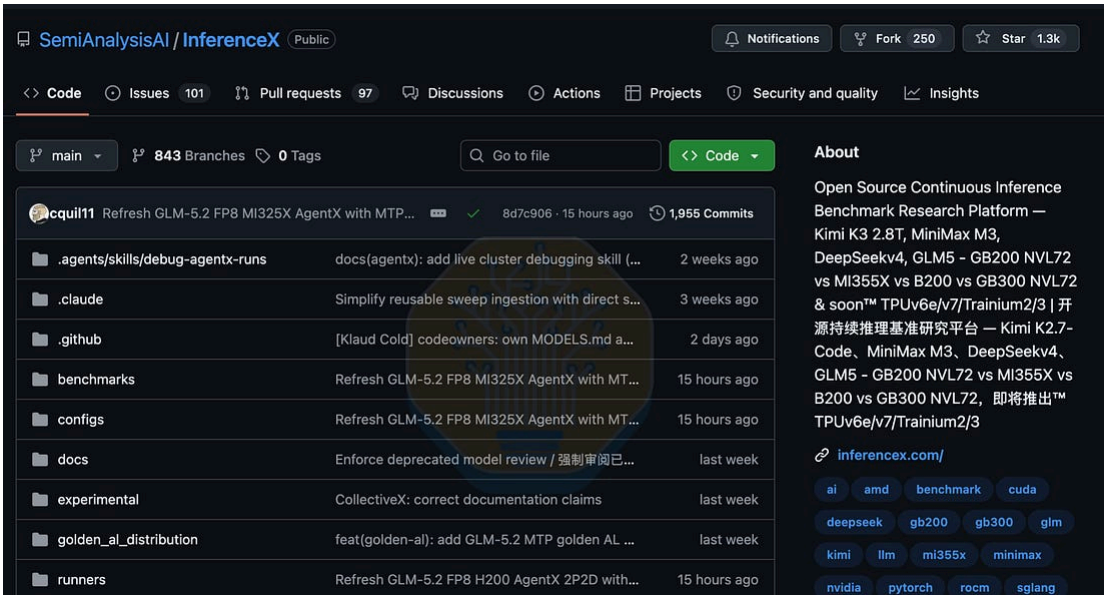
Our benchmark has been widely reproduced, validated and/or supported by almost every major buyer of compute from Google Cloud to Microsoft Azure to Oracle, to Meta and many more. Furthermore, it has the support of the ML community including from vLLM, LMCache, SGLang, PyTorch, Huggingface and the support of major labs like OpenAI, MiniMax, ZAI, Qwen, Moonshot Kimi, etc.



Source: InferenceX

Star the InferenceX GitHub repository if you find the open-source benchmark and data useful!. As previously mentioned, Nvidia has committed to submitting verifiable

Vera Rubin numbers to InferenceX. We will have Google TPUv7 results soon, and AMD has committed to MI455X UALoE72 this year too.



Source: [InferenceX GitHub]

Throughput vs Interactivity Curve

Every inference system must balance two competing goals.

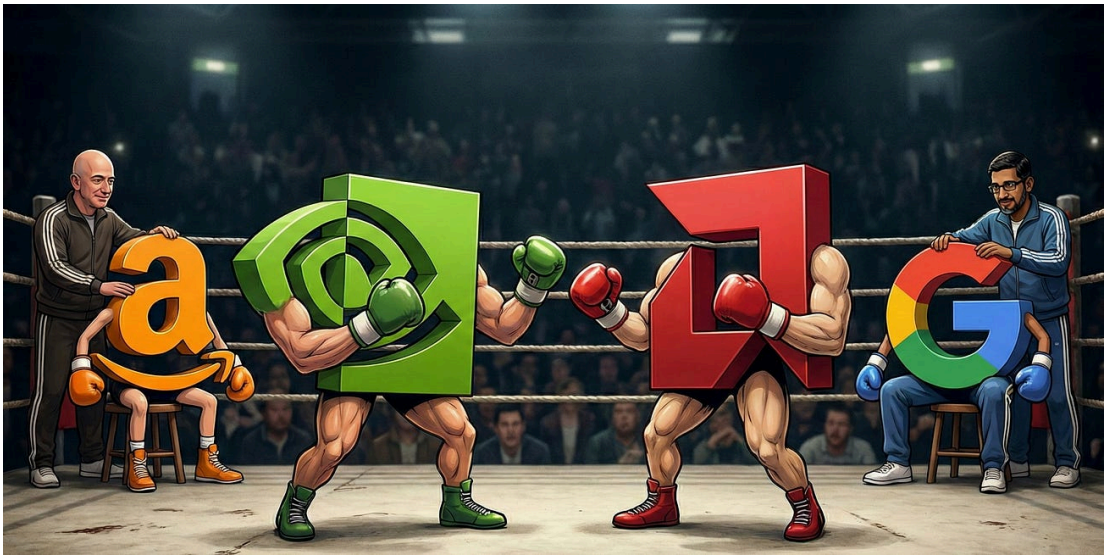
- **Interactivity (tok/s/user)** measures how quickly a single user receives tokens, the inverse of time per output token (TPOT). It determines whether a response feels snappy or sluggish.
- **Throughput (tok/s/GPU)** measures how many tokens the system produces in total across all users. It largely determines the cost per token.

Batching increases aggregate throughput by processing more requests together, but each user typically waits longer for each token. Small batches do the opposite: they improve per-user speed while reducing the amount of useful work each GPU completes in aggregate.

InferenceX v2: NVIDIA Blackwell Vs AMD vs Hopper - Formerly InferenceMAX

DYLAN PATEL, CAM QUILICI, AND 5 OTHERS · FEB 16

QQ:1413221435
QQ:1413221435



Introduction

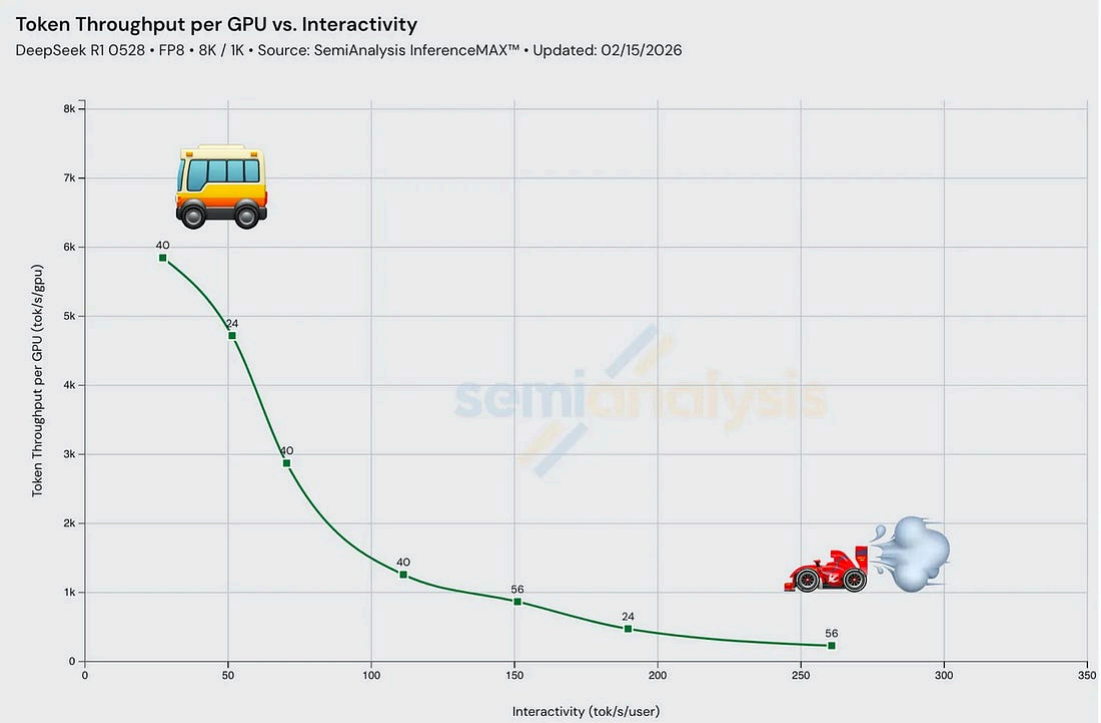
Read full story

A bus amortizes its cost across many passengers but makes each passenger wait for shared stops. A race car carries only one or two people and reaches the destination faster, but at much higher cost per passenger. Inference has the same trade-off: batching improves aggregate throughput and cost per token, while small batches improve per-user responsiveness. There is no one-size-fits-all operating point.

In the configuration shown below, increasing interactivity from roughly 25 to 260 tokens/s/user reduces per-GPU throughput from about 5,900 to 200 tokens/s/GPU. That is roughly a 30× reduction in aggregate throughput for a 10× increase in per-user speed.

QQ:1413221435

QQ:1413221435



Source: SemiAnalysis

TileRT results

As we describe in the next section, GPUs already perform well in high-throughput scenarios but struggle in high-interactivity ones. This weakness has created an entire market segment for dataflow chips. TileRT targets the same weakness and therefore focuses exclusively on high-interactivity operating points.

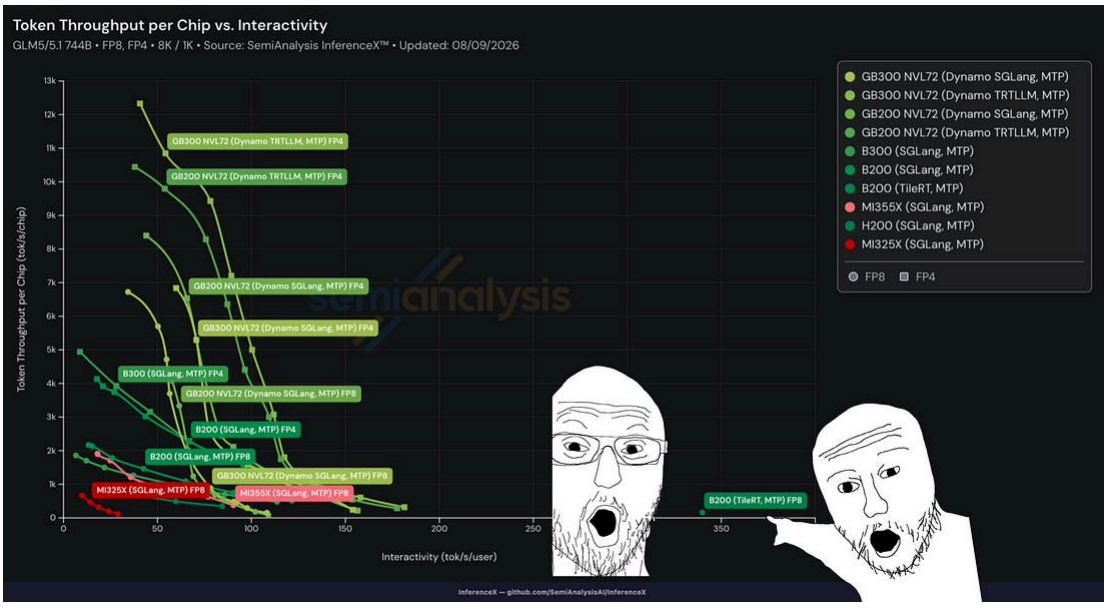
SemiAnalysis is a reader-supported publication.
To receive new posts and support our work,
consider becoming a subscriber.

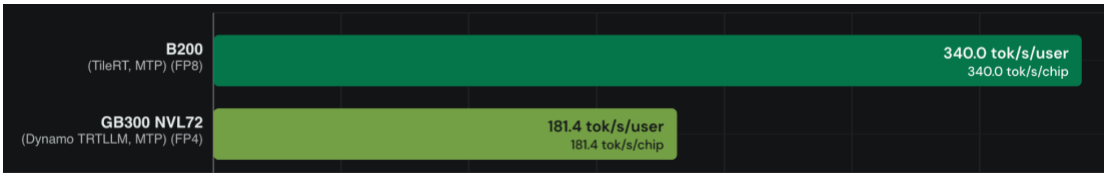
Subscribed

TileRT on B200 is in a class of its own. For the 8k/1k input/output token scenario, TileRT reached 340 tokens/s/user on an eight-GPU B200 node. The fastest result in the current dataset was previously 181.4 tokens/s/user on GB300 NVL72 with NVFP4 and MTP, making TileRT 1.9× faster on this metric. Of course - this is on Batch Size 1, where all that extra trouble to set up the complicated copper backplane in the case of the GB300 NVL72 does not come into play at all in boosting interactivity.

Meanwhile, the fastest FP8 result was 113.6 tokens/s/user on B300 with MTP, making TileRT 3.0× faster at the same precision.

QQ:1413221435





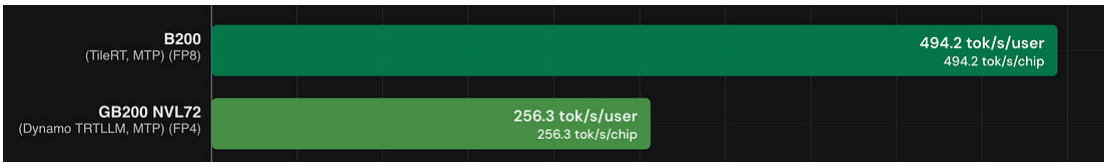
Source: InferenceX

At 1k/1k input/output, TileRT FP8 reached 494.2 tokens/s/user. That was 1.9× the best conventional result, at 256.3 tokens/s/user using FP4, and 3.6× the best conventional FP8 result, at 136.3 tokens/s/user. TileRT doesn't yet have FP4 support, but it is already beating non TileRT FP4 implementations! The result is also notable because it comes from an eight-GPU B200 node rather than the 72-GPU NVLink scale-up domain of GB200 or GB300 NVL72. This comparison concerns per-user interactivity, not aggregate throughput or cost.



Source: InferenceX

QQ:1413221435



Source: InferenceX

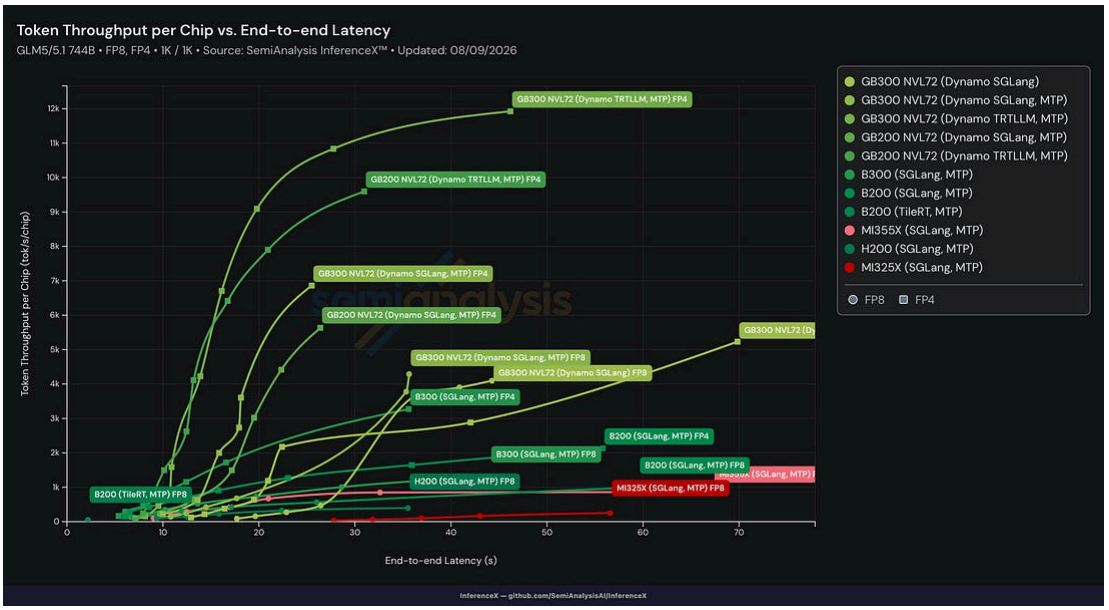
However - there are always tradeoffs when it comes to inference! TileRT's interactivity advantage comes with lower aggregate throughput. Conventional engines can amortize weight loads and fixed kernel costs across more users as concurrency rises. At 8K/1K input/output, the GB300 FP4+MTP point at concurrency 12 delivers approximately 240 total tokens/s/GPU while maintaining 154 tokens/s/user. TileRT delivers 160.4 total tokens/s/GPU while reaching 340 tokens/s/user.

The trade-off is therefore: TileRT provides much higher per-user speed, but the conventional GB300 point completes more aggregate work per GPU. TileRT as of publication also serves only one in-flight request per decode node, making this a deliberately specialized operating point rather than a general throughput configuration. Thus, with support only for a batch size of 1 user, TileRT is not just a race car, but it is more like a private rocket ship with room for just one passenger. Engineering TileRT to support more passengers might be possible, but it is an ambitious goal.

Star InferenceX GitHub

For end-to-end latency, TileRT at FP8 outperforms the best previously recorded GLM-5.1 result by 4.5× at 1k/1k and 3.0× at 8k/1k. As expected, TileRT's time to first token (TTFT) is good but not exceptional. The decisive advantage comes from the decode tail: 3.01 seconds, compared with 6.54 seconds for the best NVFP4 + MTP competitor and 18.18 seconds for MI355X.

QQ:1413221435
QQ:1413221435



Source: InferenceX

But What Exactly is TileRT?

We briefly introduced what TileRT does and showed some benchmark results, but let’s pause and explain more deeply what TileRT is and how it works. Traditional serving engines run as thousands of separate GPU program kernels launched one after another. All that setup and teardown means the GPU spends a surprising amount of time waiting, and while this setup/teardown time might not matter for low to medium interactivity inference but it definitely does for ultra high interactivity inference (aka low latency inference). Worse, each kernel writes its half-finished work out to HBM. At small batch sizes, this is a bigger problem as kernels aren’t large enough to amortize launch latency, synchronization, and scheduling overhead.

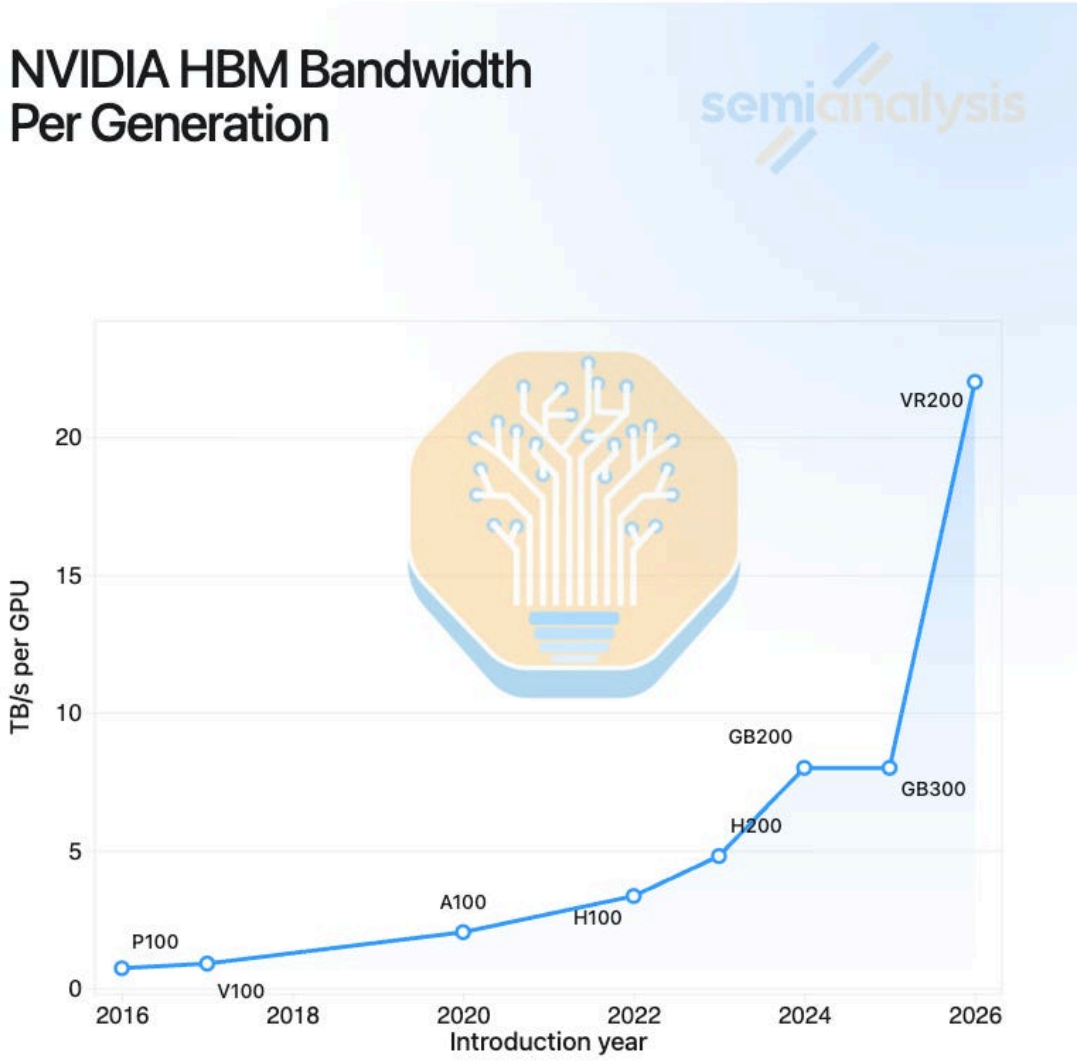
SemiAnalysis is a reader-supported publication.
To receive new posts and support our work,
consider becoming a subscriber.

Subscribed

As mentioned earlier, when running TileRT at batch size 1, for just an single HGX H200 server (38.4TB/s of aggregate HBM memory bandwidth), the active parameter memory bandwidth stands at 42GB per token at MXFP8. In theory, if we were only bound by memory bandwidth, then even without spec decoding, inference should be able to reach up to 1,000 tok/s/user interactivity. This is obviously not the case in the real world! The roadblock is that GPUs’ programming and architecture model is traditionally not built for low latency. Even though memory bandwidth per GPU

QQ:1413221435

increases 2-3x each generation, memory latency has not improved at all, even as HBM prices continually increase!

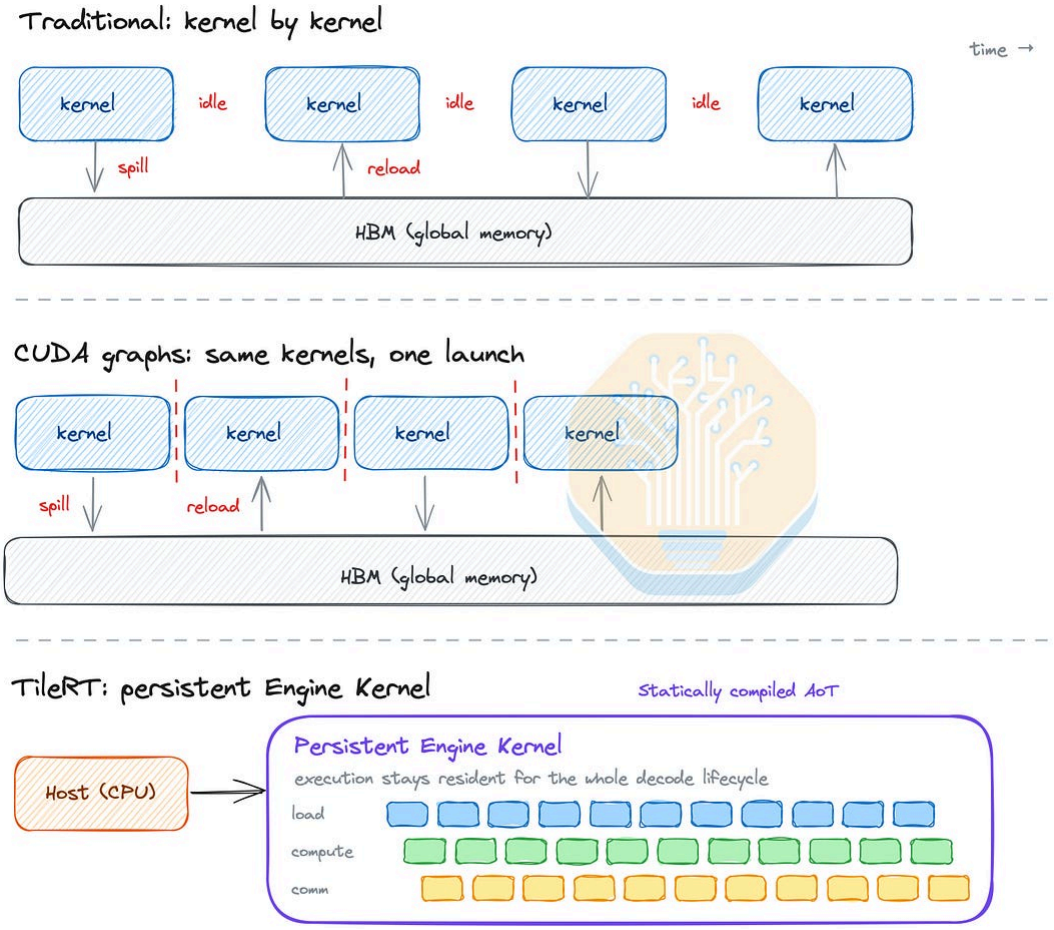


Source: SemiAnalysis, Nvidia

Instead of continuously launching kernels, TileRT has the GPU continuously execute a persistent pipeline, statically compiling the whole model ahead of time into a persistent Engine Kernel: the host launches once, execution stays resident on the GPU for the whole decode lifecycle, and most runtime orchestration moves into compile time.

This is different from CUDA graphs, which captures the DAGs(directed acyclic graphs) of kernel launches and memcpys once, then replays it with a single cudaGraphLaunch. But the kernels themselves are still separate kernels, this boundary between kernels carries device-side costs and the on-chip state is wiped at every boundary. **A CUDA graph optimizes the launching of kernels, while TileRT abolishes the kernel as the unit of execution.**

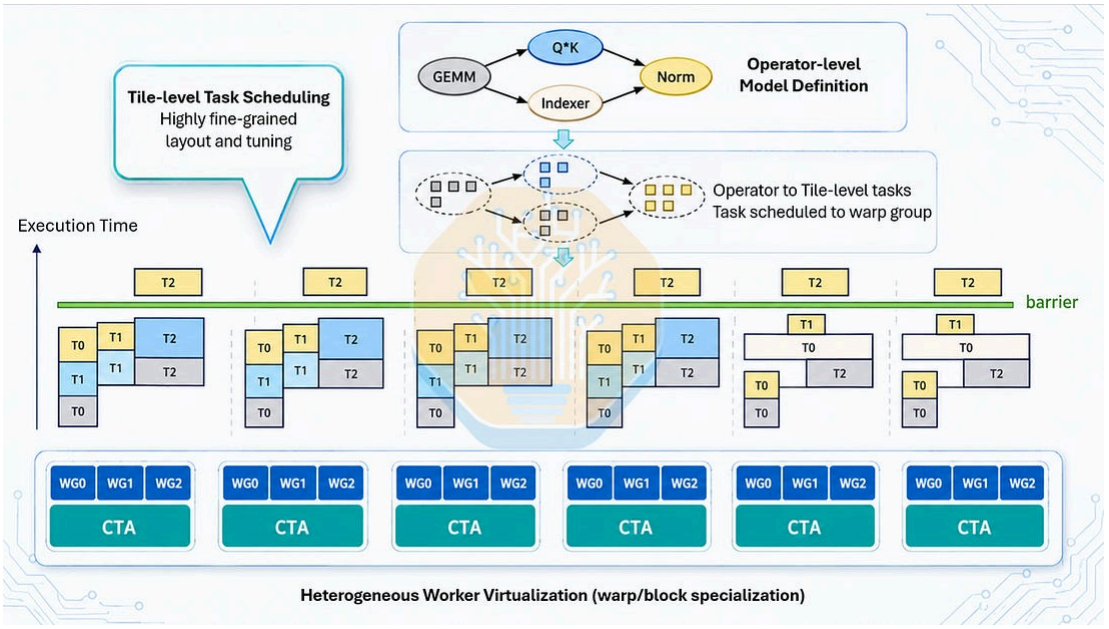
QQ:1413221435



Source: SemiAnalysis

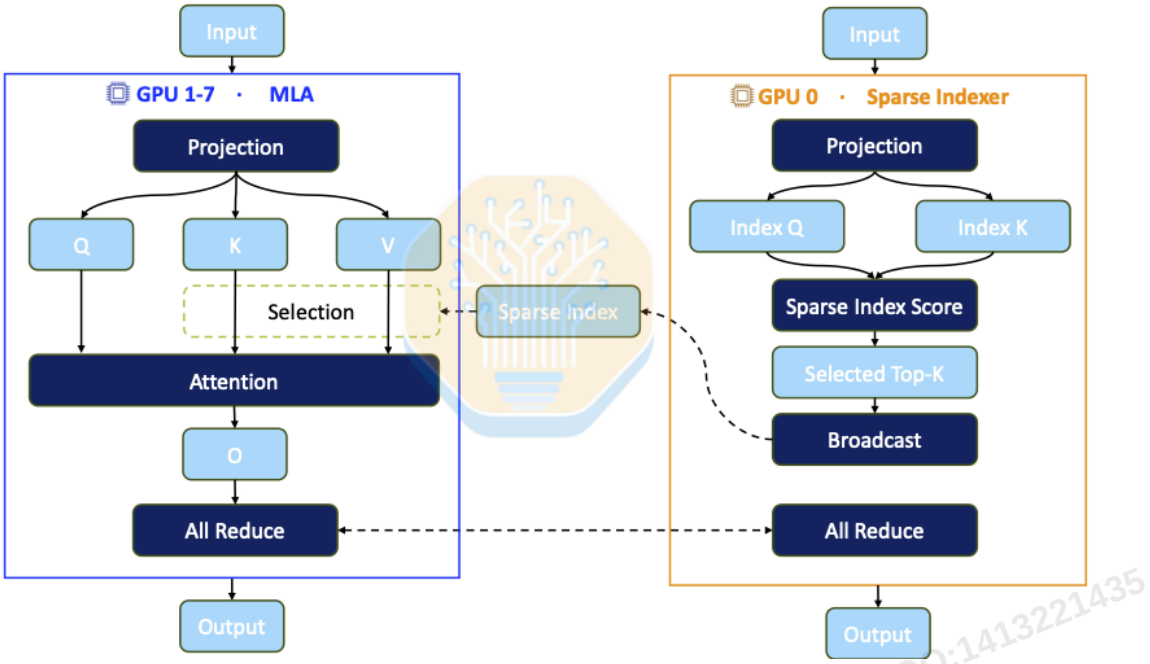
Also, through decomposing work into tile-level tasks with warp and block specialization, the runtime dynamically reschedules computation, I/O, and communication in a highly overlapped way. Inside the Engine Kernel, different warp groups take on different jobs: asynchronous data movement, tensor computation, and communication overlap. Where stages used to run serially as load → barrier → compute → barrier, they now overlap at tile granularity, and intermediate results flow forward through registers, shared memory, and L2 instead of repeatedly spilling to global memory. Effectively, each CTA(Cooperative Thread Array) becomes a small heterogeneous factory rather than a uniform SIMT(Single Instruction Multiple Threads) worker.

QQ:1413221435
QQ:1413221435



Source: [TileRT]

The next optimization TileRT introduces is specialization extended to whole GPUs. Most TP frameworks assume all ranks execute identical logic synchronously, but sparse routing, Top-K selection, dynamic indexing, long-context attention, and MTP don't fit homogeneous scale-out well; they're not compute-heavy but depend on global information, so forcing every rank through them adds redundant work and synchronization amplification. So, if warps can specialize, so can GPUs. In GLM-5.1's attention layer, GPU 0 becomes a Sparse Indexer worker handling Top-K selection, sparse index construction, and routing, while GPUs 1 through 7 run the MLA workers doing RMSNorm, GEMM, flash sparse attention, and AllReduce.



Source: [TileRT]

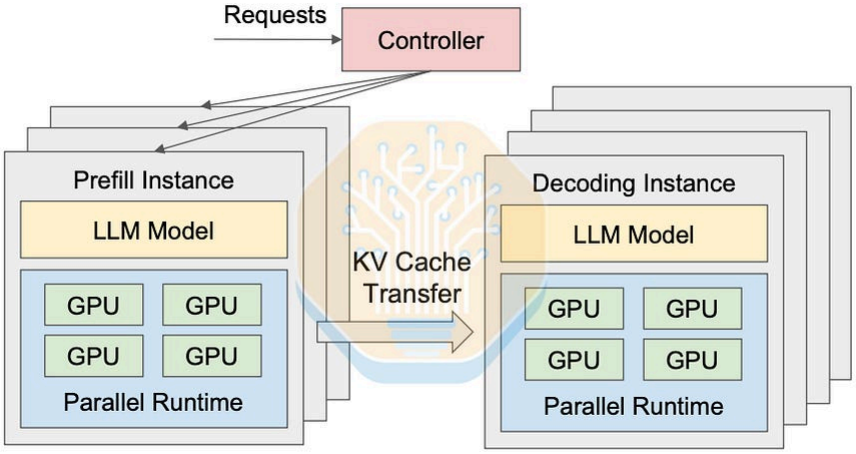
QQ:1413221435

Finally, instead of treating communication as an external stage, broadcasts, reductions, and synchronization execute directly inside the tile-level flow; with TileRT, an entire attention layer corresponds to a single kernel launch at the host, and execution shifts from compute → sync → compute toward a continuously overlapping compute ↔ communication ↔ compute pipeline.

PD Disaggregated Engine with vLLM<>TileRT

LLM inference consists of two distinct phases: prefill and decode. Prefill processes the input prompt in parallel and is primarily compute-intensive, making aggregate throughput the key performance metric. Decode generates tokens sequentially and repeatedly accesses the growing KV cache, making it memory-intensive and highly sensitive to per-token latency.

QQ:1413221435



Source: DistServe

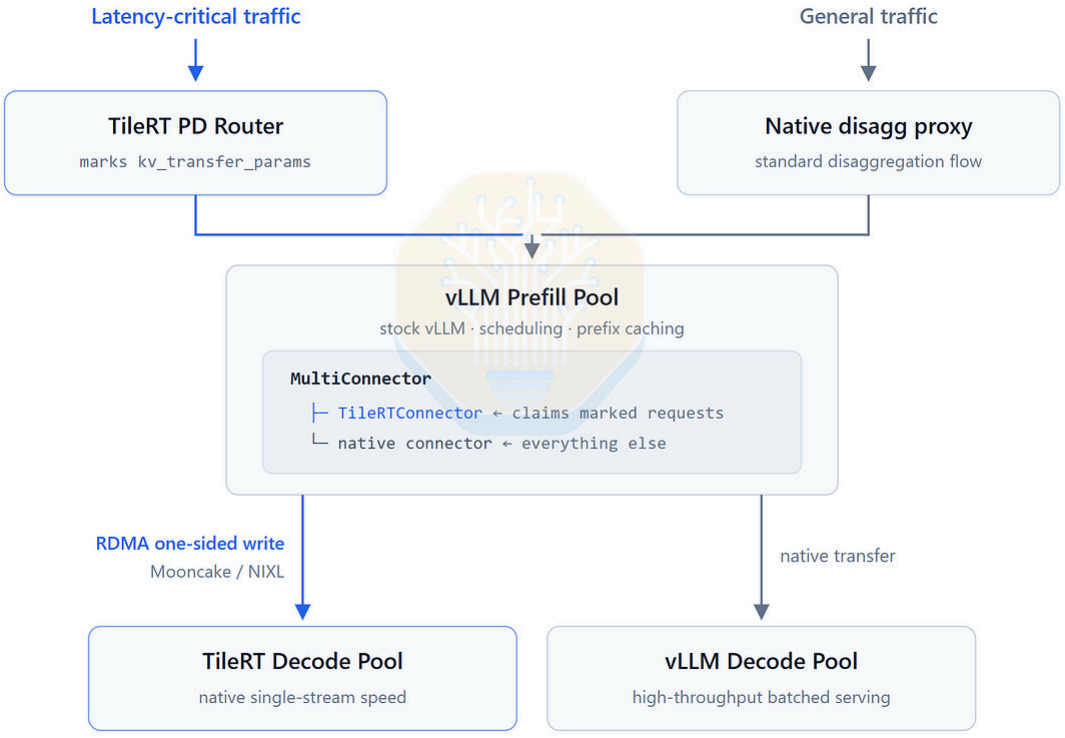
TileRT does not replace vLLM, vLLM remains the high-throughput prefill engine and the surrounding serving layer, including its scheduler, chunked prefill, prefix caching, OpenAI-compatible API, and operational tooling. Only latency-critical decode traffic moves to TileRT. TileRT is engineered to be a single-passenger rocket ship, and vLLM remains the plane, car, bus, and train.

Star InferenceX Github

The prefill and decode phases can be disaggregated into seperate nodes. With disagg, one shared vLLM prefill pool can feed two entirely different decode pools.

QQ:1413221435

- Pool A: Ultra high interactivity decode with TileRT
    - Latency-critical requests pass through the TileRT PD Router, which instructs vLLM to generate the first token and marks the request with the destination TileRT node in kv_transfer_params.
- Pool B: General low to medium interactivity decode with vLLM decode
    - General traffic continues through vLLM's native disaggregation proxy to a conventional vLLM decode pool.



Source: [vLLM & TileRT]

This is done via vLLM's MultiConnector API that composes the TileRTConnector with its native connector. The TileRT connector claims only marked high interactivity traffic class requests and becomes a no-op for everything else, meaning both traffic classes can share the same prefill server. Between the Prefill and Decode, TileRT uses Mooncake Transfer Engine and NIXL Transfer Engine to move KVCache. In TileRT v0.1.5, each decode node serves one in-flight request at a time. The router gates dispatch and applies back-pressure when the node is occupied.

Thanks for reading SemiAnalysis! This post is public so feel free to share it.

Share

QQ:1413221435

How does TileRT compare to Cerebras/Groq/SambaNova?

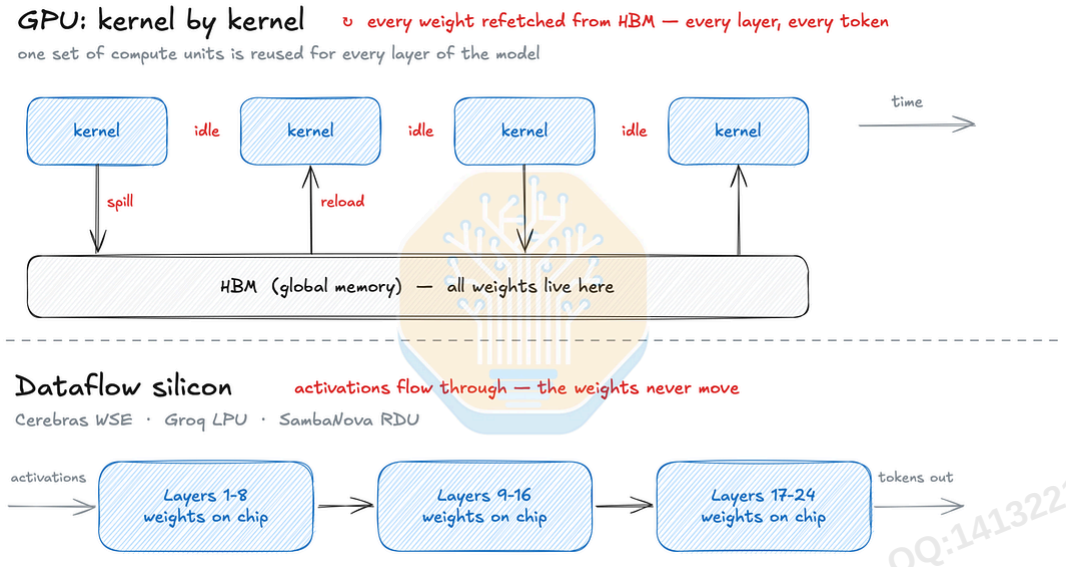
Purpose-built inference vendors identified the same execution bottleneck years ago, but encoded more of the solution in hardware. The SemiAnalysis Accelerator Model has our quarter by quarter estimates of NVIDIA LPU30, LPU40, Cerebras WSE-3 & WSE-4 shipments.

Groq uses deterministic, compiler-orchestrated execution and a large on-chip SRAM hierarchy. Cerebras maps computation spatially across a wafer-scale processor; the CS-3 provides approximately 900,000 cores, 44 GB of on-chip SRAM, and 21 PB/s of memory bandwidth. SambaNova maps model graphs onto reconfigurable dataflow units backed by a tiered SRAM, HBM, and DDR memory system.

SemiAnalysis is a reader-supported publication.
To receive new posts and support our work,
consider becoming subscriber.

Subscribed

The silicon differs, but the systems share the same idea: latency-sensitive inference benefits from reducing runtime scheduling, operator boundaries, synchronization, and unnecessary movement through external memory. At large batch sizes, those costs are easier to amortize. At batch size 1, they occupy a much larger share of each token's latency.



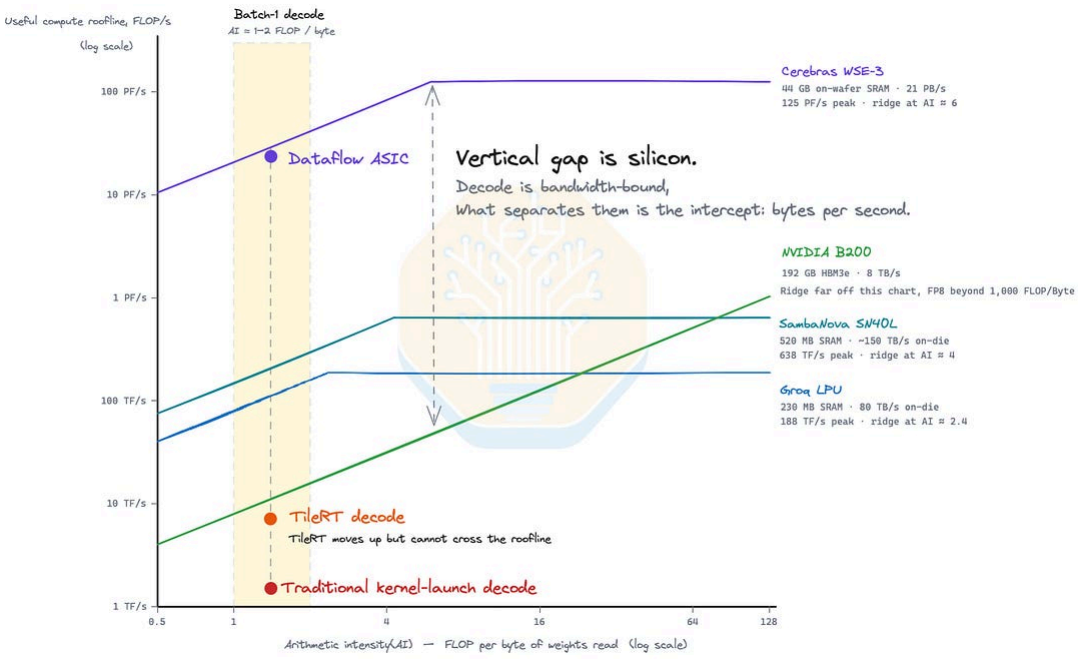
Source: SemiAnalysis

QQ:1413221435

TileRT imports software analogues of several dataflow ideas: AoT scheduling, persistent execution, specialized workers, and tighter overlap between communication and computation. The resemblance is architectural rather than literal. TileRT still runs on a SIMT GPU with dynamic hardware scheduling, HBM, and a model-specific compiled schedule.

Star InferenceX GitHub

However, TileRT is still software only: dataflow is imposed on a machine that was never specialized designed for it. A GPU carries dynamic warp schedulers, a SIMT model, and an HBM hierarchy, and TileRT gets its numbers by spending enormous compiler effort convincing that machinery to impersonate a spatial pipeline through statically expanded persistent kernels, hand-carved warp specialization, and per-model compilation against pinned driver stacks. Native dataflow silicon never fights its own substrate. Purpose-built accelerators encode more of the execution model in hardware and can avoid some of the overhead TileRT must hide in software. Their advantage still depends on the model, precision, memory hierarchy, compiler quality, system scale, and serving configuration. That is why Cerebras serves a dense 70B at speeds no eight-GPU node can reach regardless of scheduling: software can approach the HBM roofline, but it cannot raise it.



Source: SemiAnalysis

The market’s early answer is that purity is negotiable. TileRT’s decode engine is already in production behind Xiaomi’s MiMo V2.5 Pro UltraSpeed and Z.ai’s GLM-5.1 HighSpeed, and the deployment pattern is the tell. Neither company procured a new dataflow chip. They carved a speed tier out of the accelerator cluster they

QQ:1413221435

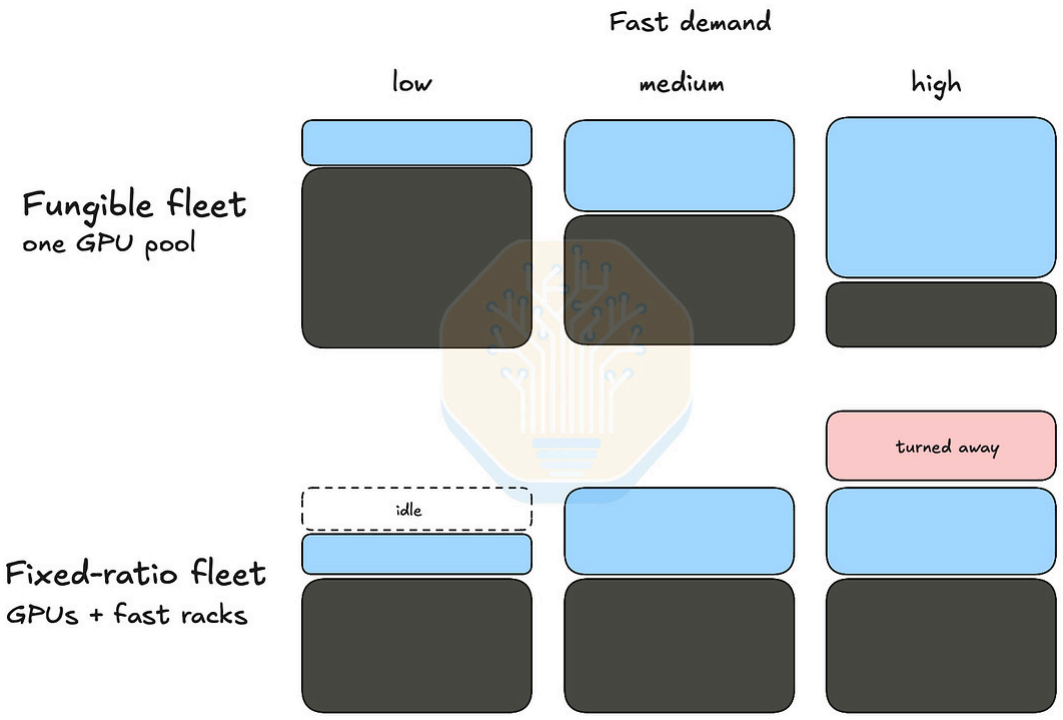
already ran, with vLLM keeping prefill, scheduling, and the API while TileRT takes over decode behind the same endpoint. Good enough on hardware you already own tends to beat architecturally pure on hardware you have to buy.

That points at the deeper structural problem: fungibility and flexibility in prefill-decode (PD) ratio.

A GPU pool is one liquid resource, excellent at prefill, excellent at high to medium-batch decode, and now somewhat credibly strong at ultra-interactive decode, with capacity moving between those roles as software scheduler decision that can follow demand hour by hour. An ASIC fleet is the opposite: the ratio of speed-tier capacity to everything else is fixed in hardware the day the purchase order is signed. Changing the ratio of the physical fleet will take months to physically re-rack and re-cable. That would be fine if the workload mix were stable and known. Unfortunately, the split between users who need ordinary conversational latency and users, increasingly agents, who will pay for extreme-interactivity SLOs has a lot of different variables at play when estimating. Guess wrong with GPUs and you rebalance in software. Guess wrong with dedicated silicon and you either strand capital in idle speed machines or turn away the exact premium traffic you bought them for. On top of that - requirements may shift over time, so a correct guess will only be right for a limited period of time.

Going back to the shared prefill pool mentioned earlier, providers do not need to pay the TileRT premium for all traffic. General requests can stay on throughput-optimized vLLM or SGLang decode pools, while only latency-critical requests are routed to the TileRT decode pool.

QQ:1413221435
QQ:1413221435



Source: SemiAnalysis

None of this kills the top of the speed market. The SRAM roofline is still better, certain sizes of models still favor it, and some workloads will always want maximum tokens per second at any price. But TileRT reframes what most buyers need: not a speed machine, but a speed tier, provisioned dynamically out of the fleet they were going to own anyway. Cerebras, Groq, and SambaNova are no longer competing against a clumsy kernel-launcher. They are competing against their own execution model, running on fungible hardware, reallocated by a config file. TileRT may be a single-passenger rocketship, but it allows providers to strap solid rocket boosters to your Metro Bus instead of having to design an entirely new launch vehicle.

Why Is TileRT Development Slow?

GLM5.1 is a generation behind, and has already been deprecated on mainline InferenceX. TileRT's model catalog is very limited, currently supporting GLM-5/5.1, DeepSeek-V3.2. MiMo-V2.5-Pro-UltraSpeed is the result of a co-design partnership and has yet to be open-sourced.

TileRT inherits ASIC vendors' biggest weakness. Static ahead-of-time compilation means a tiny model catalog (currently GLM-5/5.1 and DeepSeek-V3.2), hard-pinned dependencies, and real engineering effort per new architecture. There is no fully generic path, a persistent engine kernel means the model is statically expanded ahead of time into one resident program, so decisions have to be made on tile shapes, pipeline depth, buffer residency across registers/shared memory/L2, how warp groups split between loading, compute, and communication, where collectives get

QQ:1413221435

fused into the tile flow, and which GPUs take specialized roles like GLM-5.1's dedicated sparse indexer rank. Change the attention mechanism or the routing scheme and much of that schedule is invalidated. Dataflow chips also face this same issue, good compilers can be notoriously difficult to create.

Share SemiAnalysis

Work is being done to simplify this, especially as software development can be accelerated with AI. TileOPs is intended to reduce this burden. Each operator is declared in a machine-readable manifest specifying its signature, workloads, and roofline model. The manifest drives code generation, testing, and benchmarking against hardware bounds rather than only against earlier implementations.

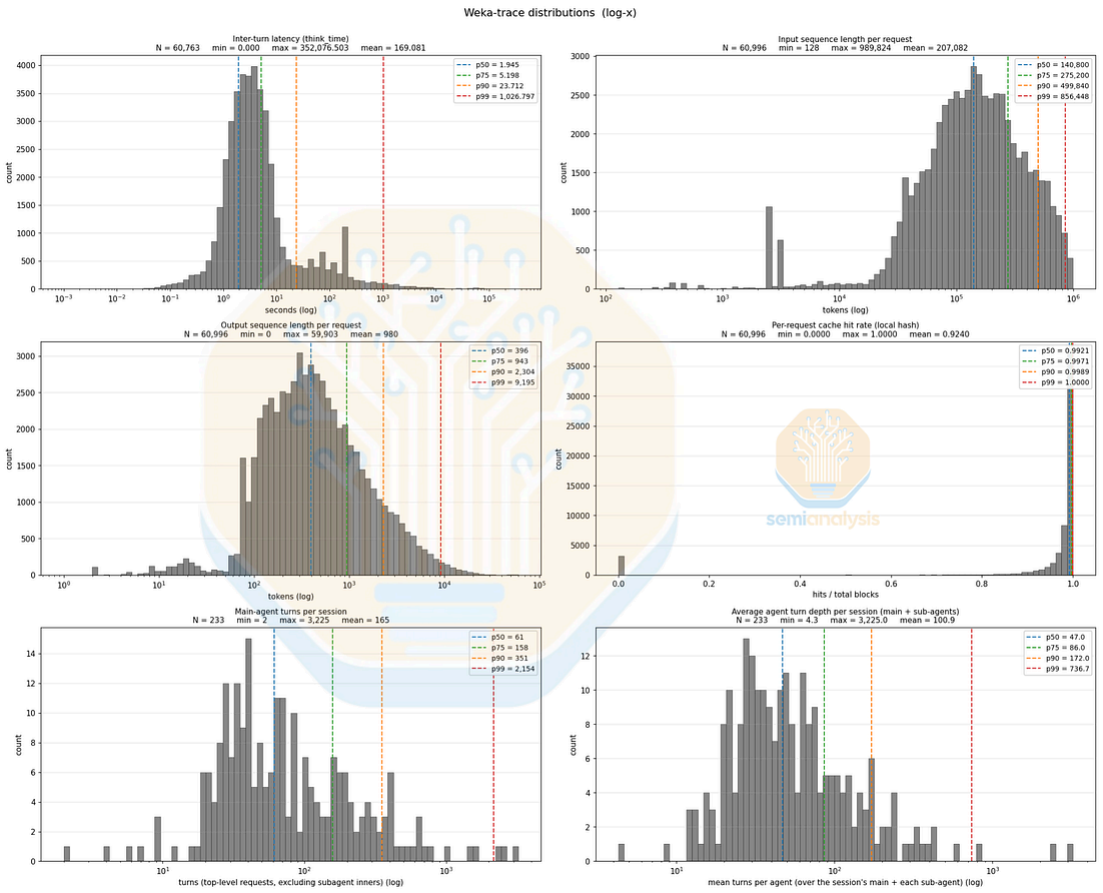
AI coding agents accelerate tuning within known templates, but novel transformations still require expert judgment. A monolithic persistent kernel also reduces the usefulness of conventional per-kernel profiler timelines, making automated feedback loops more difficult.

QQ:1413221435

Next steps with TileRT<>InferenceX

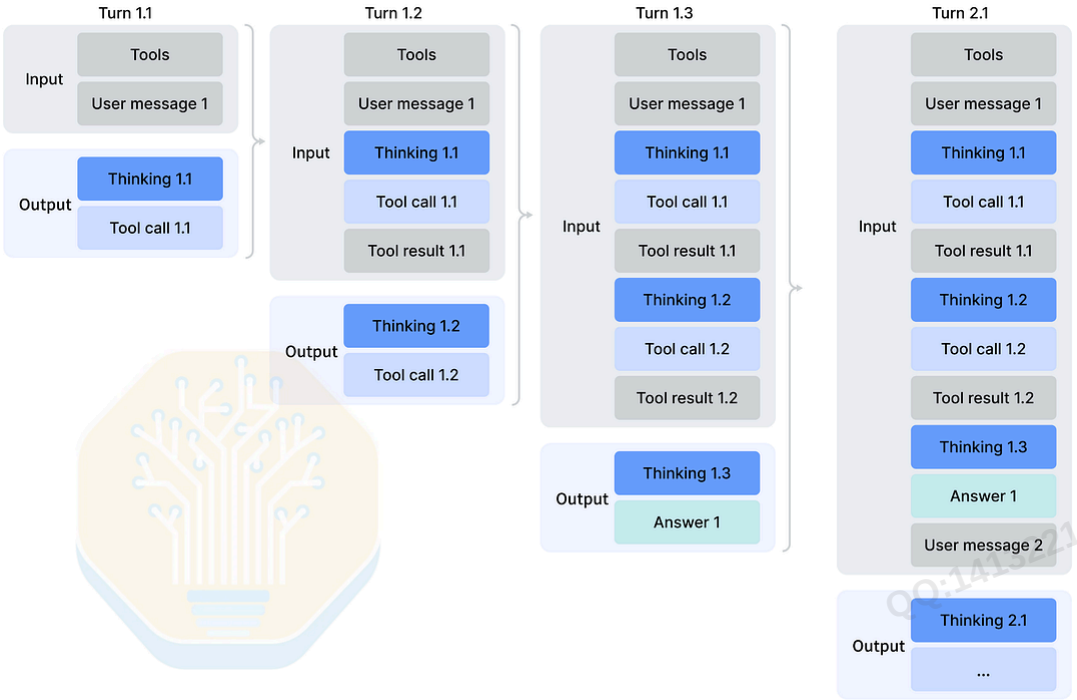
We are actively working on moving TileRT benchmarking from InferenceX's single-turn 8k/1k as well as our new agentic coding benchmark, which we call AgentX. This scenario replays real Claude Code and Codex traces with long-context, multi-turn requests, realistic subagent activity, and dynamic tool-use delays. Its median input length is 140k tokens, while the theoretical median cache-hit rate roofline reaches 99.2%.

QQ:1413221435



Source: [SemiAnalysis]

This workload will test the entire TileRT<> vLLM system, not just decode speed, including incremental KV transfer, prefix-cache reuse, cache retention and offloading, routing, and scheduling. The critical question is whether TileRT can transfer only the newly introduced context between turns while preserving its ultra-high interactivity advantage.



QQ:1413221435

Source: DeepSeek

The second step is to move beyond just batch size one. We will also benchmark TileRT at batch sizes 2, 4, and 8. The goal is to map its throughput–interactivity Pareto frontier and identify the point at which the persistent Engine Kernel's latency advantage begins to flatten.

SemiAnalysis is a reader-supported publication.
To receive new posts and support our work,
consider becoming a subscriber.

Subscribed

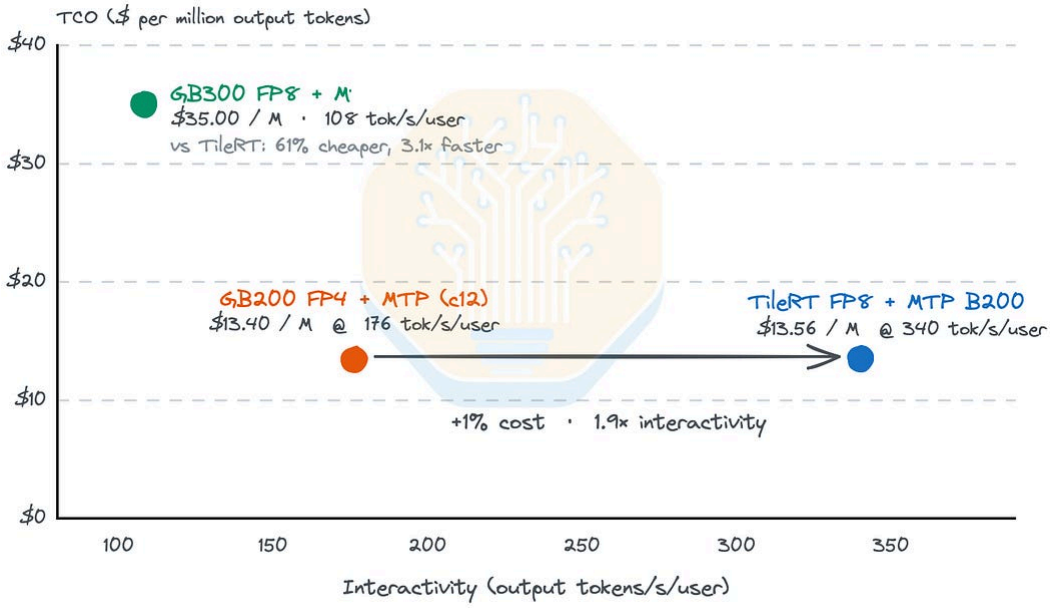
Perf per TCO of TileRT Ultra Fast Speed

Next, we do an deep dive analysis on TileRT's cost per million output tokens at ultra-high interactivity compared to decode at normal lower-interactivity operating points. The results are quite interesting with TileRT boosting up to 1.9x faster interactivity when iso-cost with traditional engines. We use our AI TCO Model as baseline for the capex & opex for each chip SKU.

Cost per token is only meaningful after a system satisfies the latency target. At 339 tokens/s/user, the feasible set in our measured GLM-5.1 GPU results only contains TileRT results. The fastest conventional point reaches 176 tokens/s/user, while the fastest conventional disagg FP8 point, GB300 SGLang with MTP reaches only 108.0 tokens/s/user. At 8k/1k, TileRT reaches 340 tokens/s/user while delivering 35.4 output tokens/s per B200. This comes out to $13.56 per million output tokens.

The FP4 disagg decode setup with the highest interactivity is GB200 FP4 with MTP at concurrency 5. It produces approximately 286 total tokens/s/GPU while maintaining roughly 176 tokens/s/user. At $1.86 per GB200-hour, that works out to $13.4 per million output tokens. TileRT therefore costs only 1% more per token while delivering 1.9× the interactivity. This is much more value than Claude code fast mode, up to 2.5x the interactivity but 2x the price per token. The same-precision comparison is even more striking. The fastest conventional FP8 result is GB300 FP8 with MTP at 108 tokens/s/user. Its endpoint would cost $35 per million output tokens. TileRT reaches 340 tokens/s/user for $13.56 per million, 61% cheaper per output token while delivering 3.1× the interactivity.

QQ:1413221435



Source: SemiAnalysis, TCO Model

Like Comment Restack

© 2026 Dylan Patel
548 Market Street PMB 72296, San Francisco, CA 94104
Unsubscribe

Start writing

QQ:1413221435

QQ:1413221435