Inference is fragmenting

A whitepaper on agents, specialized silicon, and the largest unfragmented slice of the inference market.

General Compute | May 2026

---

1. The thesis

Every mature compute market eventually fragments. Databases started as one product, Oracle, and split into OLAP, OLTP, time-series, vector, graph, and key-value systems as workloads diversified. Networking silicon split between switching, routing, deep-packet inspection, and SmartNICs. Mobile chips fragmented from a single application processor into dedicated NPUs, ISPs, modems, and security enclaves. In each case, the fragmentation was a response to the same pressure: as the market grew, workload diversity outpaced what any single architecture could serve well, and specialized silicon won the slices where its architectural choices made it the obvious answer.

Inference is at the beginning of this transition. Training will stay on general-purpose GPUs for the foreseeable future, because training is a workload that rewards exactly what GPUs do well. But inference is splitting into workload classes with very different demands. Voice has different requirements than batch. On-device is a different problem than hyperscale serving. Long-context retrieval is a different problem than short-form generation. And agents, which we believe will become one of the largest inference workloads over the next several years, are different from all of them.

General Compute is the inference cloud being built for the agent slice.

Agents reward a specific kind of hardware specialization. Long prompts, short structured outputs, deep sequential trajectories where each step blocks the next, and batch sizes that refuse to fill. The economics of running an agent are bounded by the latency of each step in its trajectory, because every millisecond of decode delay compounds across the entire task. Get the latency wrong and the agent product is uneconomical or unshippable. Get it right and a new class of products becomes viable.

Today, we run agent workloads on SambaNova SN40L, powered by low-cost hydroelectric energy and deployed in existing US colocation. On GPT-OSS-120B, this stack delivers **time-to-first-token 2.6x faster and end-to-end latency 4.6x faster than Together AI**, one of the strongest GPU-based inference providers in the market. That is what one piece of well-chosen silicon already buys you against a well-engineered general-purpose stack.

In Q4 2026, we add two more levers. SambaNova's SN50, which moves total rack throughput on 600B+ parameter models by nearly twenty times over SN40L. And disaggregation across chip types, with AMD

MI300X taking over prefill while SN50 specializes in decode. Each change compounds on the last, and each one is rooted in the same thesis: agent workloads reward hardware that is built for them, and the slice is large enough to be worth building for.

The rest of this paper defines that slice, explains why the hardware specialization works, and describes the stack we are putting against it.

2. Agents are not chatbots

The inference infrastructure the industry built from 2022 to 2025 was built for chatbots. A user types a question, a model generates a response, the user reads it. Throughput matters because you want to serve many users at once. Tail latency matters less because a human is reading the output as it streams, and at reading speed, any decode rate above about fifty tokens per second feels instant.

Agents invert every one of these assumptions.

The input is long. An agent's prompt is not a user question, it is a system prompt, a tool catalog, a memory buffer, retrieved context, and the accumulated history of prior steps. Tens of thousands of tokens is normal. Hundreds of thousands is not unusual. Every step re-sends most of this context, which makes prefill a recurring cost rather than a one-time setup.

The output is short and structured. Agents do not write essays, they emit tool calls, JSON objects, reasoning traces, and intermediate plans. Generations of a few hundred tokens are typical. The model spends most of its time starting and stopping rather than producing sustained output.

The workload is sequential. A chatbot conversation has natural pauses while the user reads and types. An agent trajectory has none. The moment one step finishes, the next begins. There is no human in the loop absorbing latency, so every millisecond of decode delay is a millisecond added to the end-to-end task time, and those milliseconds compound across every step.

The batch is small. Chatbot serving economics rely on batching dozens or hundreds of concurrent user requests together to amortize the cost of reading model weights from memory. Agent workloads are often batch size one: a single long trajectory, running as fast as possible, with no sibling requests to pack alongside. The throughput optimizations that define modern GPU serving stacks simply do not apply.

Put these together and the workload looks like this: long inputs, short outputs, zero tolerance for per-step latency, compounding across dozens of steps, at batches that refuse to fill. This is a different problem from chatbot serving, not a harder version of it.

3. Inference is two workloads, not one

The dominant framing of LLM inference treats it as a single problem. Tokens go in, tokens come out, faster is better. This framing has produced a generation of infrastructure optimized for a weighted average of two fundamentally different computations, and it is starting to break down.

Prefill and decode have almost nothing in common at the hardware level.

Prefill is compute-bound. When a model processes a long prompt, it performs a dense matrix multiplication across the entire input in parallel. The bottleneck is FLOPs. The chip wants wide SIMD, high arithmetic intensity, and plenty of HBM bandwidth to stream weights. This is exactly what modern GPUs are built for. An H100 or MI300X doing prefill is a chip doing what it was designed to do.

Decode is memory-bound. Once generation begins, the model produces one token at a time, autoregressively. Each token requires reading the entire model's weights and the full KV cache from memory, and then doing a trivial amount of math on them. The bottleneck is bandwidth, specifically the latency of getting weights and cache to the compute units. Arithmetic intensity collapses. The chip's FLOPs sit idle waiting for memory. A GPU doing decode at batch size one is a Ferrari in a parking lot, expensive silicon doing almost nothing.

Agent workloads make this split more extreme, not less. Long prompts inflate prefill cost. Small, latency-sensitive generations inflate the relative weight of decode. Because each step in an agent trajectory blocks the next, decode latency compounds across the entire task. A coding agent making twenty tool calls pays the decode tax twenty times. Prefill speed sets the floor on time-to-first-token, and decode speed sets the ceiling on how many steps an agent can take before the user gives up.

The industry's response has been to run both phases on the same chip and optimize for throughput. Batch aggressively, use continuous batching and PagedAttention to pack sequences, amortize weight reads across many requests. This works for chatbots. For agents, where batch size one is closer to the norm than the exception and tail latency per step dominates user experience, it fails on its own terms. You cannot batch your way out of a workload that is inherently sequential.

There are two lines of response to this. The first is to pick better silicon for the whole workload, a chip whose architecture suits both phases better than a GPU. That is what SambaNova SN40L does today, and it is why we already run 4.6x faster than a mature GPU inference provider on the same model with the same prompts. The second, and more powerful, response is to stop using one chip for both phases. Run prefill on a chip designed for prefill, decode on a chip designed for decode, and let each piece of silicon do what it was built for.

The real answer is to disaggregate the hardware. It is what we are moving to in Q4 2026.

4. Why dataflow wins decode

Decode looks like a compute problem but behaves like a memory problem. Every token generated requires moving the full weights of the model from memory to the compute units, along with the accumulated KV cache for the sequence. For a seventy-billion-parameter model, that is hundreds of gigabytes of data movement per token. The matrix multiplications that actually produce the token take a small fraction of the time, and the rest is waiting for memory.

GPUs address this with HBM, the fast, expensive, high-bandwidth memory stacked next to the compute die. HBM is a real engineering achievement and it is the reason modern GPU inference works at all. But HBM bandwidth has scaled more slowly than compute for several generations now, and there is no sign of that curve bending. Each new GPU generation adds more FLOPs than it adds memory bandwidth, which means the bottleneck on decode gets worse, not better, as the hardware improves. The H200 has more HBM than the H100. The B200 has more than the H200. They are all still memory-bound on decode at batch size one, and they always will be, because the memory hierarchy is structurally wrong for the problem.

Dataflow architectures solve this differently. Instead of a small compute die pulling data from a large external memory, a dataflow chip distributes both memory and compute across a reconfigurable grid. Weights live on the chip, close to the compute units that need them. Data moves through a configured pipeline rather than being fetched repeatedly from a bandwidth-constrained pool. The memory hierarchy is not a cliff to be climbed on every token, it is a plane the computation runs across.

SambaNova's RDU is the most mature production instance of this idea. The SN40L and the newer SN50 implement a three-tier memory hierarchy (on-chip SRAM, package-level memory, and off-package DDR) that keeps weights and KV cache resident where they are needed. The practical consequence is that decode at batch size one runs at a fraction of the latency of a GPU, because the chip is not waiting for HBM on every token. The same properties that make dataflow good at batch-size-one decode make it good at radix attention and aggressive prompt caching, which are exactly the patterns agent workloads produce.

These are not theoretical advantages. On GPT-OSS-120B, General Compute running on SN40L delivers a **mean end-to-end latency of 1.76 seconds against Together AI's 8.05 seconds** on the same model, measured with identical prompts sent simultaneously to both providers and with network overhead included for both. Our SN40L stack today is monolithic, prefill and decode on the same silicon, and it is still 4.6x faster than a well-engineered GPU inference provider on a widely-deployed MoE model. The gap widens as generations get longer and as trajectories get deeper, which is the direction agent workloads are going.

There is a long-standing objection to specialized inference chips, which is that general-purpose GPUs benefit from the enormous software ecosystem around CUDA, and specialized silicon does not. This was a serious problem three years ago. It is a smaller problem now. SambaNova's stack supports the model

families agent workloads actually use, including Llama, Qwen, DeepSeek, Mixtral, and other MoE variants, with the serving primitives (KV cache management, prompt caching, structured output) that production agent systems require. The ecosystem gap has closed faster than the hardware gap can be, and the hardware gap is widening as HBM scaling slows.

For prefill, the calculus is different. Prefill is what GPUs are good at. The MI300X in particular offers HBM capacity and FLOPs at a price point that is meaningfully below NVIDIA's comparable parts, and its prefill performance is competitive on the models agent workloads depend on. When we move to a disaggregated stack in Q4 2026, the right answer for the prefill leg is a GPU, and the right GPU, increasingly, is AMD's.

5. The General Compute architecture

General Compute is built in two phases, both grounded in the same thesis: agent workloads reward hardware specialization.

Today, we run monolithic SambaNova SN40L. Prefill and decode both execute on the same dataflow silicon, with our serving layer handling KV cache management, prompt caching, and routing. On GPT-OSS-120B, this stack delivers mean time-to-first-token of **738ms against Together AI's 1,899ms**, and mean end-to-end latency of **1.76s against 8.05s**. That is 2.6x faster to first token and 4.6x faster end-to-end, measured with network overhead included on both sides. These numbers are on the shorter end of our benchmark suite, where GPU providers look their best. The gap widens on longer generations and on the deep trajectories that define real agent workloads. SN40L alone, with no disaggregation and no prefill specialization, is already a different tier of performance from what GPU-based competitors deliver.

In Q4 2026, we move to a disaggregated stack across SambaNova SN50 and AMD MI300X. Prompts will arrive, hit AMD for the parallel compute-bound prefill phase, and hand off the KV cache to SN50 for the sequential memory-bound decode phase. The user still sees a single API. The hardware will see two workloads, each running on the silicon built for it.

This phase unlocks two compounding improvements on top of what SN40L already delivers.

The first is the SN50 step change. On 600B+ parameter models, SN50 moves total rack throughput from roughly **800 tokens per second on SN40L to roughly 15,000 tokens per second**, nearly a twenty-times improvement on the same model class. On GPT-OSS-120B specifically, SN50 reaches interactivity rates of **1,900 tokens per second**, which is in a different tier from what any GPU cloud currently delivers on comparable hardware. This comes before disaggregation is even layered in. It is pure decode specialization on newer silicon.

The second is prefill specialization. Today, SN40L does prefill well, but it is not what dataflow silicon is optimized for. The MI300X is. Moving prefill to AMD lets us use each chip type for what it is best at, and it lets us scale the two phases independently based on actual workload shape. Long-prompt, short-output agent traffic is prefill-heavy. Short-prompt, long-output traffic is decode-heavy. A disaggregated stack provisions each piece to match, rather than buying one chip that compromises on both.

Both phases of the roadmap run on aircooled silicon. SN40L, SN50, and the aircooled MI300X all fit inside existing US colocation facilities at standard rack power densities. The same is not true of the infrastructure most GPU clouds are scaling into. H100 at density, and certainly GB200 NVL72, require purpose-built liquid-cooled data centers with direct-to-chip cooling loops, specialized coolant distribution, and power densities that existing colos largely cannot support. The wait for new liquid-cooled capacity in North America is measured in years, not quarters.

The practical consequence is that our path from signed capacity to racks serving production traffic is structurally shorter than a GPU competitor's path from chip allocation to live inference. We already have options on 15 megawatts of aircooled power, which is enough to support the Q4 2026 architecture and the growth beyond it, at colocation facilities that exist today. A GPU cloud targeting similar density is waiting for buildings that do not yet exist, on timelines determined by construction permits and cooling-infrastructure supply chains.

Underneath both phases of the roadmap is power. Our capacity runs on low-cost hydroelectric energy, which at scale is the dominant operating cost of an inference cloud. The combination of efficient silicon, inexpensive clean power, and aircooled deployment into existing colocation is what makes our pricing defensible over the long run rather than a short-term discount. It is also what makes the roadmap credible. We are not promising capacity that depends on infrastructure that has to be built first.

On the commercial side, we have an executed LOI with SambaNova covering dedicated SN40L capacity today and a path to SN50 as it comes online, an active partnership track with AMD on the prefill side of the Q4 2026 architecture, and distribution going live through OpenRouter at our May 1 launch. The first cohort of production customers is onboarding now on SN40L. The next phase of the stack lands in Q4 2026, against the same customers, with significantly more headroom.

We are not trying to be a general-purpose cloud. We are not trying to serve every model for every use case. We are building the best possible inference substrate for a specific, large, and rapidly growing workload class, which is agents, and we believe being the best at that will matter more over the next five years than being adequate at everything.

6. What this unlocks

If the economics of an agent are bounded by latency, raising the ceiling on latency changes what gets built.

The agent products in market today are shaped by the latency budget available to them. Coding agents make a handful of tool calls before returning control to the user because each call costs seconds. Research agents retrieve a small number of documents because longer trajectories become unusable. Voice agents constrain themselves to shallow reasoning because the turn-taking budget is brutal. These look like product choices, but most of them are latency constraints wearing product clothing.

When the decode ceiling rises, the shape of what is possible changes. A coding agent can take thirty tool calls instead of five, and the quality difference is not linear. It is the difference between a system that drafts code and a system that builds and tests it. A research agent can read fifty documents instead of five, and produce synthesis rather than summary. A voice agent can reason about what to say before it says it, instead of generating the first plausible response. Long-horizon agents, which the field has talked about for two years and largely failed to ship, become practical when the per-step cost drops below the threshold where a multi-hour trajectory is economically viable.

The companies that will build these products need infrastructure that treats agent inference as a first-class workload, not a side case of chatbot serving. That infrastructure does not exist on GPU-only clouds, and it cannot be retrofitted onto them, because the underlying hardware is optimized for the wrong half of the problem.

There is a second kind of speed that matters here too. If the goal is to raise the amount of useful agent intelligence in the world, per-token latency is only half of the equation. The other half is how quickly new capacity comes online. Every quarter of deployment delay is a quarter of agent products that do not get built, and the infrastructure the industry is betting on, liquid-cooled hyperscale GPU clusters, is the kind of infrastructure that ships on construction timelines. Aircooled silicon in existing colos ships on installation timelines. That is the difference between raising the ceiling next year and raising it in three.

General Compute is being built for the companies that understand this and want to ship against it.

If that is the future you see too, we would like to talk.