We Budgeted 64GB. Super Mode Said Otherwise.

Two generative models on one NVIDIA Jetson 32GB module: how Super Mode and Agentic AI changed the sizing equation

JetPack 7.2 MAXN_SUPER brings Jetson AGX Orin 32GB close to the compute class of the 64GB module for workloads that fit in memory. One engineer supervising an AI agent migrated a complete dual-model video AI stack in days, then benchmarked every step. Here is the full engineering story.

By **Doruk Sonmez, M.Sc.**
AI Solutions Architect, CTai LABS, a department of Connect Tech Inc.
NVIDIA DLI Certified Instructor

Published July 2026 | Benchmark data measured on device by CTai LABS | Technical review: Rob Callaghan, P.Eng., Chief Product Officer, ConnectTech.com

Key results

- NVIDIA Cosmos open physical AI world foundation model throughput up 65 percent at the production request shape, with roughly 6x faster time to first token
- Optimization improves idle free RAM up from 5.3 GB to about 10 GB, enough room to add an entire second generative model to the same module
- Full NVIDIA JetPack 6 to JetPack 7.2 migration, DeepStream 9.0 port, and a 30-row benchmark campaign completed in days
- Three live 720p30 RTSP streams sustained with continuous alerting on a single NVIDIA Jetson AGX Orin 32GB module

In production Edge AI, the model is only one part of the challenge. Integration is where the system succeeds or fails. A new operating environment, updated inference runtimes, hardware video pipelines, and a fixed memory budget all have to work together before a single frame can be analyzed reliably.

When NVIDIA released JetPack 7.2 with new agentic development tools, memory efficiency guidance, and MAXN_SUPER for Jetson AGX Orin 32GB, we put those capabilities to the test on our Scene Analyzer Agent,

a real-time video analysis application built around the NVIDIA Metropolis Blueprint for Video Search and Summarization (VSS). The target: run the entire stack, both generative models included, on one Jetson AGX Orin 32GB module. (A workload we had previously scoped for a 64GB configuration.)

This is the engineering story behind that migration: what moved, how an AI coding agent compressed weeks of platform bring-up into days, and exactly which memory levers made the 32GB module carry the load. Every number below was verified against live device telemetry, and we report the trade-offs as measured, including where the envelope ends.

The workload: dual-model video AI on one Jetson

The Scene Analyzer Agent ingests live RTSP camera streams or recorded video through NVIDIA DeepStream 9.0 SDK hardware pipelines: NVDEC for decode, VIC for scaling and conversion, NVENC for clip encoding. Frame data stays on the hardware-accelerated path and avoids unnecessary CPU memory round trips.

The application keeps two generative models resident and operational on the same module.

The first is a vision-language model, Cosmos-Reason2-2B, served on vLLM. It produces per-chunk scene summaries and powers natural-language semantic alerting, so a rule like “person enters the loading dock after hours” is matched against live video as it happens.

The second is Nemotron-3-Nano-4B in Q4_K_M GGUF format, a hybrid Mamba-2 architecture running on llama.cpp. It rolls dense per-chunk captions into configurable 5-minute and 30-minute digests and powers a knowledge-graph question-answering service over the video history.

Add a SQL database, a vector store, GraphRAG, and a browser-based dashboard with a live low-latency view, and the headline writes itself: two generative models, hardware video ingestion, vector and graph databases, and a full web UI, all sustained on a single Jetson AGX Orin 32GB module.

The agentic bring-up loop

Moving the full application from JetPack 6 to JetPack 7.2 meant a new L4T userspace, DeepStream 9.0, a new CUDA stack, and new model-serving runtimes. This is the kind of platform bring-up that normally runs weeks to months. It took days, because we ran the migration as what we call an agentic bring-up loop: one engineer supervising an AI coding agent equipped with NVIDIA's Jetson and Metropolis developer tooling, working directly against the physical device.

Three NVIDIA Agentic tools did the heavy lifting.

The DeepStream Inference Builder MCP generated the DeepStream 9.0 pipeline scaffold and its plugin configuration straight from the agent conversation, with no hand-written GStreamer boilerplate.

DeepStream handles only the hardware-accelerated video path; the two models are served separately, on vLLM and llama.cpp.

The DeepStream Agentic Skill and Coding Agent gave the agent DeepStream 9.0 and pyservicemaker API knowledge, so it ported our custom NVDEC and VIC ingestion path to DeepStream 9.0 idioms and fixed r39-specific breakages, including GStreamer 1.24 buffer-mapping changes and entrypoint quirks, on the first pass.

Jetson Device Skills gave the agent device-aware diagnostics, memory audits, runtime selection, and structured LLM benchmarking, so every optimization was planned, applied, and verified against live telemetry rather than assumption.

The point is not that an agent wrote code. The point is that it ran the complete engineering loop (plan, apply, benchmark, verify, document, commit) autonomously against real hardware, compressing a normally multi-team, multi-week bring-up into days. The work builds directly on NVIDIA's Maximizing Memory Efficiency to Run Bigger Models on NVIDIA Jetson and Deploy Agentic-Ready AI at the Edge with Memory Efficiency in NVIDIA JetPack 7.2 guidance.

Reclaiming memory for a second model

This is where Super Mode and JetPack 7.2 earn their keep. The starting point was a JetPack 6-era configuration: desktop GUI running, the VLM in BF16 reserving just over half the module's memory up front, software video paths, and little to no room for a second model. From there, the agent applied and telemetry-verified a series of memory levers.

Lever	Memory reclaimed	Notes
Headless operation (multi-user.target)	~1 GB	OS overhead removed
FP8 weights plus FP8 KV cache, replacing BF16	~4 GB	On the VLM
Right-sizing vLLM's upfront reservation, from over half the module to about a third	~3 GB	Zero measured latency cost, verified by benchmark
DeepStream 9.0 hardware video path (NVDEC, VIC, NVENC)	Avoids CPU round trips	Frame data stays on the accelerated path
Serving the Mamba-2 digest model on llama.cpp	~5 GB budget vs ~12 GB as a second vLLM instance	Runtime selection, not just quantization

The result on this stack was substantial. Idle free RAM increased from about 5.3 GB to roughly 10 GB. Comparing the measured baseline against the final configuration, Cosmos-Reason2-2B gained about 65 percent throughput at the production request shape and roughly six times faster time to first token, while

an entire second model was added to the same device. These figures describe the combined effect of JetPack 7.2, MAXN_SUPER, and the agent-applied memory and serving optimizations on this stack.

Why Super Mode changes the sizing equation

JetPack 7.2 introduces MAXN_SUPER for Jetson AGX Orin 32GB, raising rated AI performance from 200 TOPS to 241 TOPS by increasing the available GPU frequency and power envelope. That brings the 32GB module much closer to the compute performance class of Jetson AGX Orin 64GB MAXN, while the 64GB module retains a larger memory capacity and a higher peak rating.

The practical implication matters for system sizing. When a model, and its complete application stack, fit within the 32GB unified memory footprint, the 32GB module can now deliver a level of compute that previously pushed teams toward the 64GB option. To be precise about what this benchmark does and does not show: we did not run a direct 32GB versus 64GB head-to-head, and the modules are not universally equivalent. What the data shows is that, for this workload, memory optimization plus MAXN_SUPER let the 32GB module sustain a dual-model video AI stack we had previously scoped for 64GB.

Our take: for dual-model stacks that fit in memory, the 32GB module is now the default starting point, not the fallback. Huge system cost reduction.

Benchmarking the serving stack

We benchmarked 10 model and runtime variants across three request shapes, producing 30 instrumented rows. Every run started from a clean state, with tegrastats telemetry captured for the exact duration of the measurement. The production request shape was 2048 input tokens, 256 output tokens, and concurrency 4, matching the multi-stream live pipeline summary and alert traffic. One configuration delivered the strongest overall balance of throughput, memory headroom, GPU demand, and power efficiency.

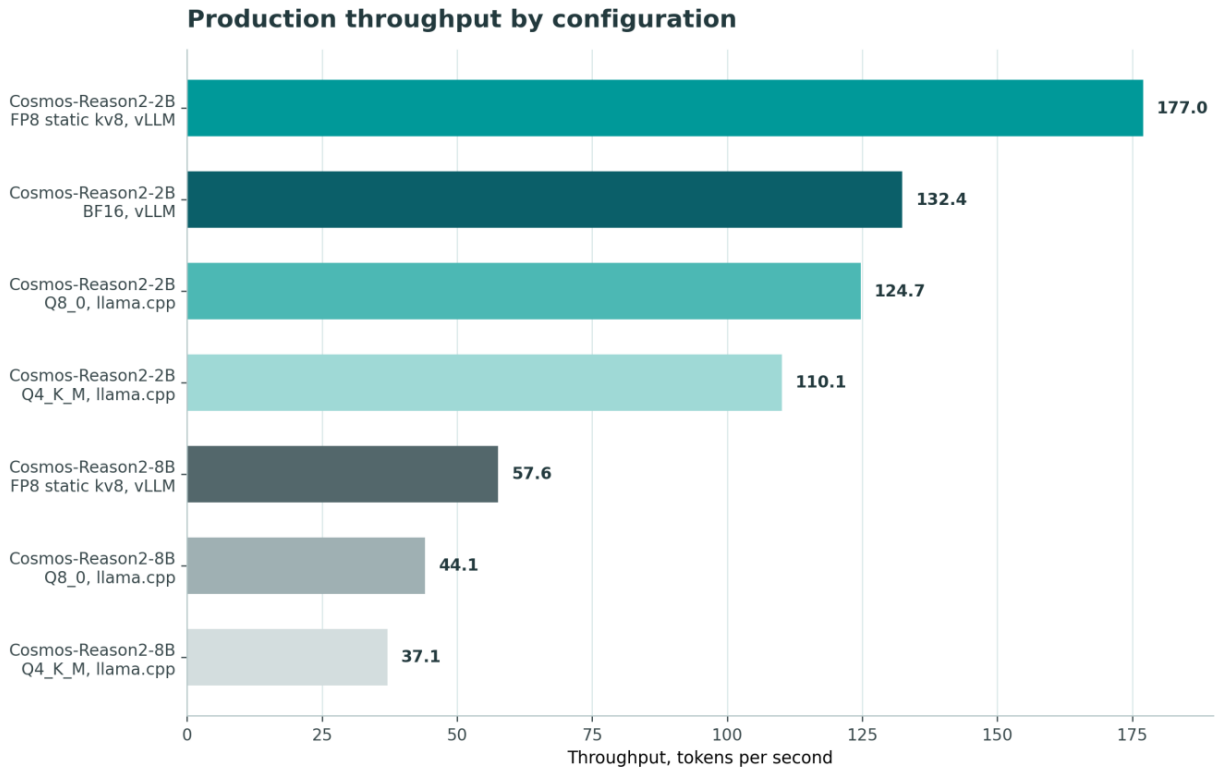

Figure 1. Production throughput at 2048 input tokens, 256 output tokens, concurrency 4, across all seven Cosmos serving configurations. Cosmos-Reason2-2B FP8 static kv8 on vLLM leads at 177.0 tokens per second.

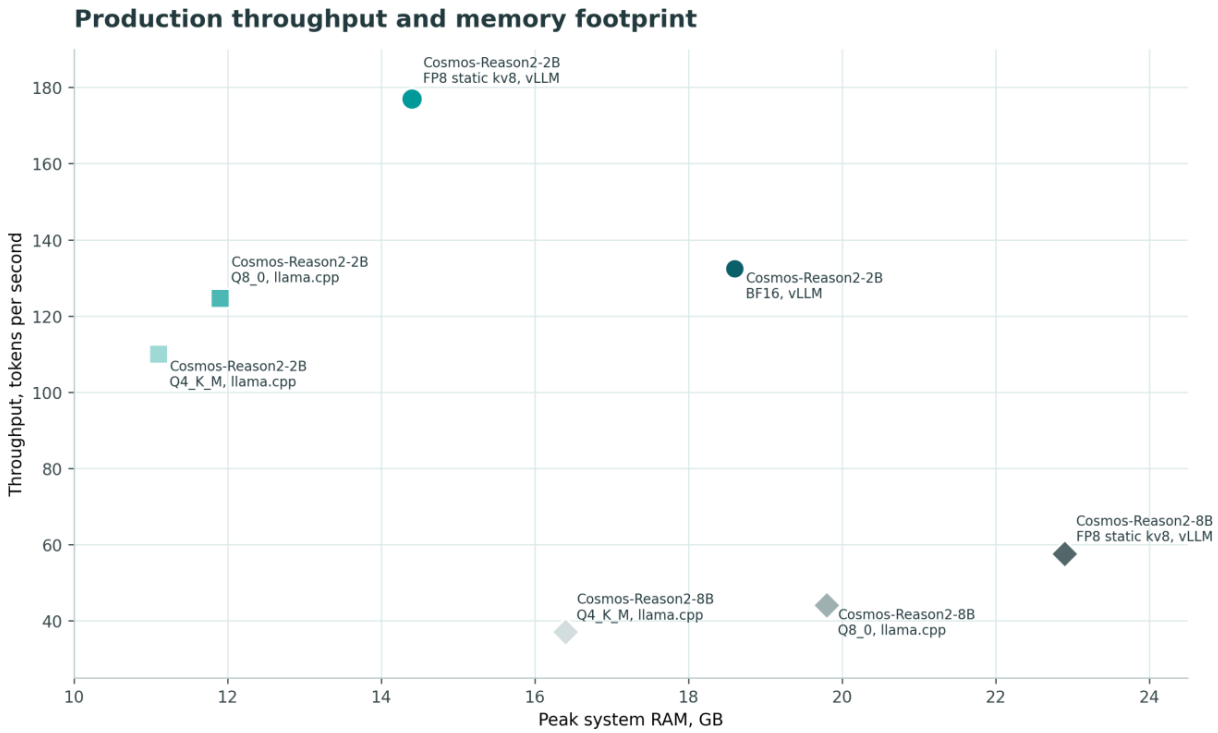

Figure 2. Production throughput against peak system RAM at the production request shape. Cosmos-Reason2-2B FP8 static kv8 on vLLM delivers the highest throughput while using less peak RAM than any other vLLM configuration tested.

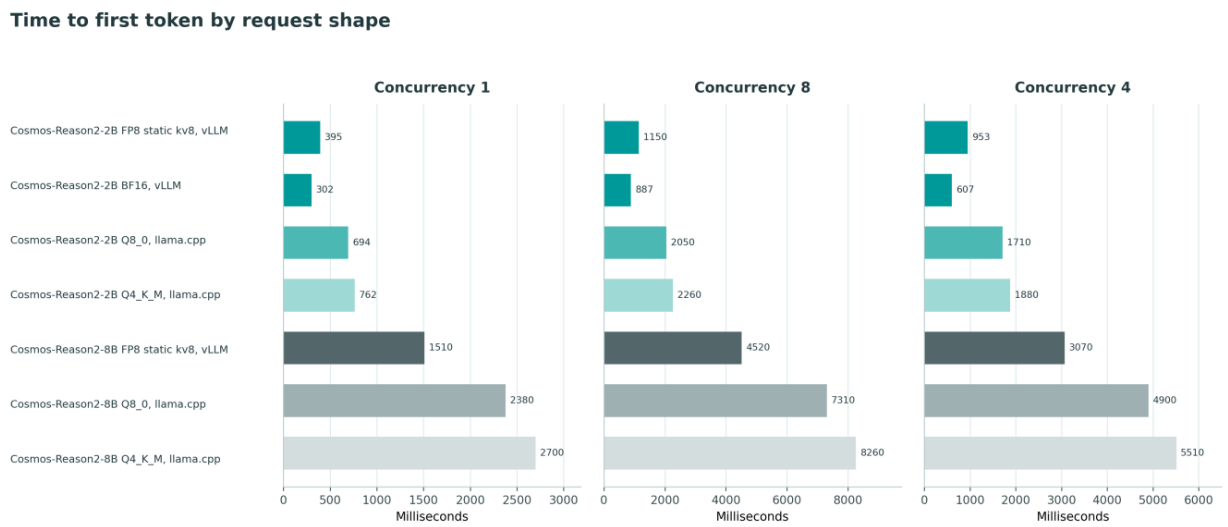

Figure 3. Time to first token at concurrency 1, 8, and 4 for all seven measured configurations. Lower is better. BF16 on vLLM is fastest on this metric; the deployed FP8 configuration trades a little responsiveness for stronger throughput and lower memory use.

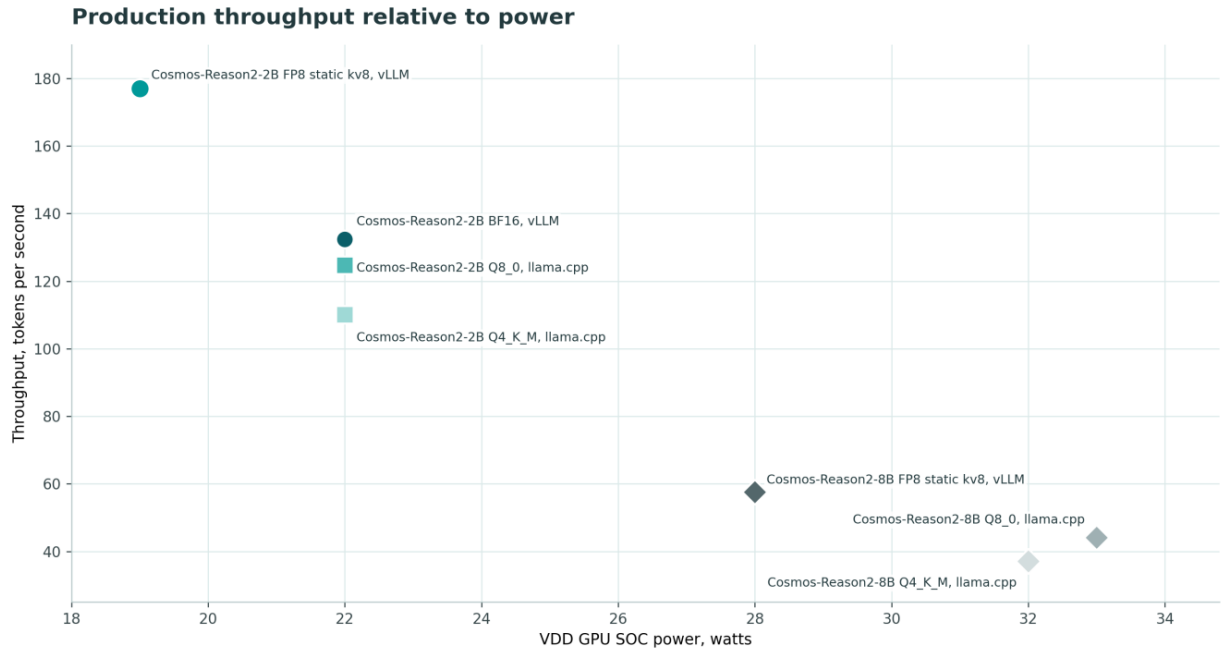

Figure 4. Production throughput against measured VDD_GPU_SOC power at the production request shape. Cosmos-Reason2-2B FP8 static kv8 on vLLM delivers the highest measured throughput at the lowest measured serving power.

The selected FP8 configuration recorded the lowest GPU demand across the measured request shapes, ranging from 47 to 54 percent against roughly 52 to 83 percent for the alternatives. It used the least memory of any vLLM configuration at a 14.4 GB system peak and the lowest measured serving power at 19 to 20 W, while leading throughput at the production request shape.

Its single-stream rate of 58.1 tokens per second sits within one token per second of NVIDIA's published Jetson AI Lab figure of 59 tokens per second for the same model and request shape. That reference was

measured on Jetson AGX Orin 64GB in MAXN mode with jetson_clocks (treat it as directional rather than a direct comparison).

One deliberate trade-off is worth spelling out. At the production request shape, the deployed 0.35 memory reservation reached 177.0 tokens per second and preserved room for the second model, video pipeline, databases, and user interface. Raising the reservation to 0.45 reached 208.8 tokens per second, but we intentionally traded roughly 15 percent throughput for about 3 GB of additional headroom. On a shared module, headroom is a feature.

Field notes: the FP8 trap on Orin. Here is the one we would have wanted someone to tell us. Compressed-tensors FP8 for Cosmos-Reason2-2B runs on Orin SM 8.7, but the tested modelopt FP8 checkpoint for Nemotron-3-Nano-4B requires SM 8.9 or later. Building or quantizing that checkpoint on a newer GPU and copying it to Orin does not make FP8 GEMM execution supported on SM 8.7. Your deployment needs a precision path Orin supports, or that FP8 path will fail to execute.

On the second model itself: Nemotron-3-Nano-4B in Q4 on llama.cpp reached single-request parity with the incumbent Qwen3-4B-Instruct-2507 (24.4 versus 24.5 tokens per second) with faster decode, at a fraction of the memory a second vLLM instance would need. That is the right trade for a serial digest workload.

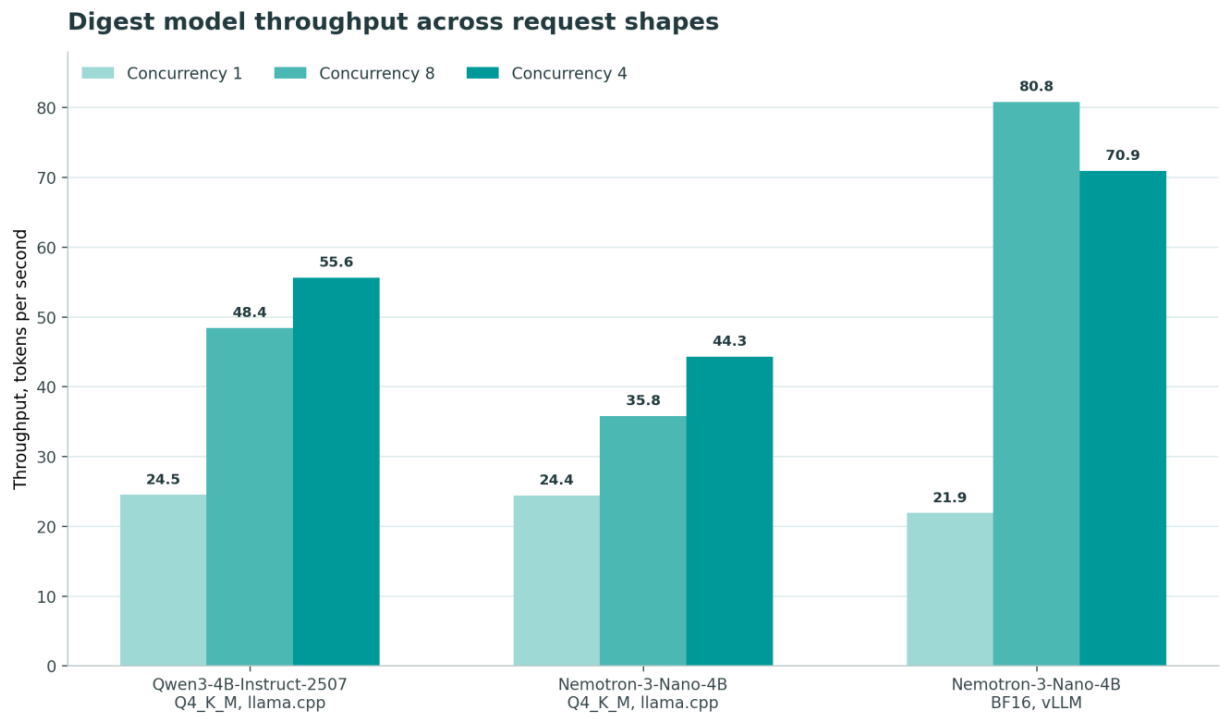

Figure 5. Digest model throughput at concurrency 1, 8, and 4 for Qwen3-4B-Instruct-2507 Q4_K_M, Nemotron-3-Nano-4B Q4_K_M, and Nemotron-3-Nano-4B BF16. Nemotron-3-Nano-4B Q4_K_M remains the selected deployment trade for a serial digest workload.

Sustaining three live video streams

Serving benchmarks are only part of the story; the full application must also stay stable under sustained video load. With three live 720p30 H.264 RTSP streams, continuous semantic alert evaluation, alert clip generation enabled throughout, and periodic digests, the module remained stable across the measured stress run. It saved 32 alert clips and generated 86 captions while averaging 89 percent GPU utilization with a 99 percent peak, 27.7 GB system RAM with a 29.4 GB peak, and 30 W on the tegrastats VDD_GPU_SOC rail. That is not a wall power measurement; total system draw is higher.

All relevant hardware engines were exercised during the run: the GPU served both models, NVDEC decoded the streams, VIC handled scaling and conversion, and NVENC generated alert clips in intermittent bursts.

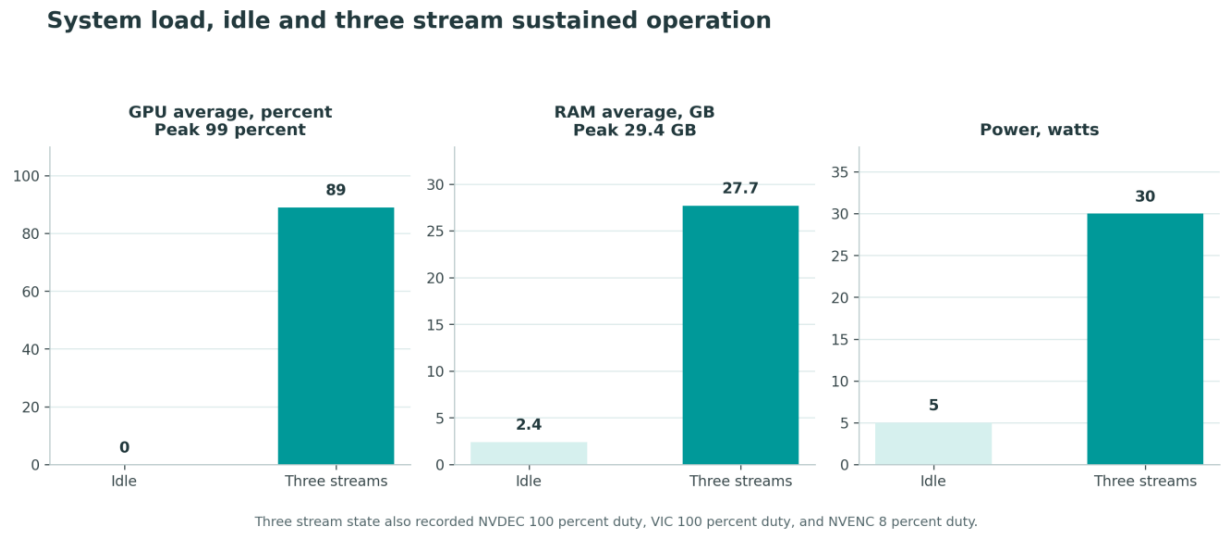

Figure 6. Full application load at idle and during sustained three-stream operation: 89 percent average GPU utilization (99 percent peak), 27.7 GB average system RAM (29.4 GB peak), and 30 W on the VDD_GPU_SOC rail. NVDEC and VIC were active in 100 percent of telemetry samples; NVENC in 8 percent, reflecting intermittent alert clip encoding. The run produced 32 clips over six minutes.

We report the envelope honestly: three concurrent streams with continuous alerting are sustained on this configuration. Push to five streams at a five-second chunk cadence and you exceed the VLM's real-time budget. That is why the full report documents the scaling levers (longer chunks, queue caps, chunk sampling) instead of rounding up.

Deployment implications: time to market and cost

For a team shipping video AI agent solutions at the Edge, this result affects both delivery speed and deployment economics. The JetPack 6 to JetPack 7.2 migration, DeepStream 9.0 port, serving stack retuning, 30-row benchmark campaign, and publication-ready report were completed in days by one engineer supervising an AI agent that benchmarked and verified every step directly on the device.

From a sizing perspective, the optimized Jetson AGX Orin 32GB module with Super Mode now supports a workload previously scoped for a 64GB configuration. The selected serving path averaged 19 to 20 W on the VDD_GPU_SOC rail, and the complete measured application averaged 30 W on that rail. Because

inference stays on the device, video stays local and cloud bandwidth requirements drop. The measured three-stream capacity envelope also gives fleet planners a defensible baseline instead of an estimate.

Looking ahead, we expect this pattern to become the norm rather than the exception. As agentic development tools mature and memory optimization becomes a standard part of platform bring-up, the question for edge deployments shifts from “which module has enough headroom for our worst case” to “how much capability can we verify on the smallest module that fits.” Teams that build that verification muscle now will size fleets more aggressively, and more confidently, than teams still budgeting by rule of thumb.

Turn Physical AI into production edge AI systems with CTai LABS

Every result in this article came from the same discipline: measure, verify, document, and report the envelope as it is. That discipline is not specific to this project. It is how CTai LABS approaches every Physical AI and Agentic AI deployment: bridging foundation models, vision-language models, large language models, agentic workflows, retrieval-augmented generation, and NVIDIA Jetson hardware into complete systems that operate reliably at the edge.

CTai LABS takes end-to-end ownership of the engineering work that slows Physical AI deployment down: migrating workloads from x86 and cloud infrastructure to optimized NVIDIA Jetson platforms, integrating NVIDIA Metropolis, NVIDIA Isaac, and NVIDIA Omniverse technologies, and handling model quantization, NVIDIA TensorRT optimization, NVIDIA DeepStream SDK pipelines, ROS2 orchestration, multimodal sensor fusion, thermal tuning, hardware-in-the-loop validation, and real-time performance engineering. The result is production-ready embodied and agentic intelligence without requiring customers to build a specialized Edge AI team from scratch.

Whether you are bringing a video AI application, autonomous robot, intelligent machine, or multimodal agent to market, CTai LABS moves the solution from concept and simulation to validated edge deployment faster, with less integration risk and reduced dependence on costly cloud or x86 infrastructure.

Bring us the use case. We will make it work at the edge. Book a discovery call at ctailabs.ai/book-a-demo/

About the author

Doruk Sonmez, M.Sc., is an AI Solutions Architect at CTai LABS, the Physical AI and edge AI services department of Connect Tech Inc., an NVIDIA Elite Partner. An NVIDIA DLI Certified Instructor, he specializes in deploying vision-language models, agentic AI workflows, and accelerated video pipelines on NVIDIA Jetson platforms. All benchmark data in this article was measured directly on device by the author.

LinkedIn: https://www.linkedin.com/in/doruk-sonmez/