Spark Text Lane Migration: Ollama to TensorRT-LLM on DGX Spark

Mission Control committee report | published May 18, 2026

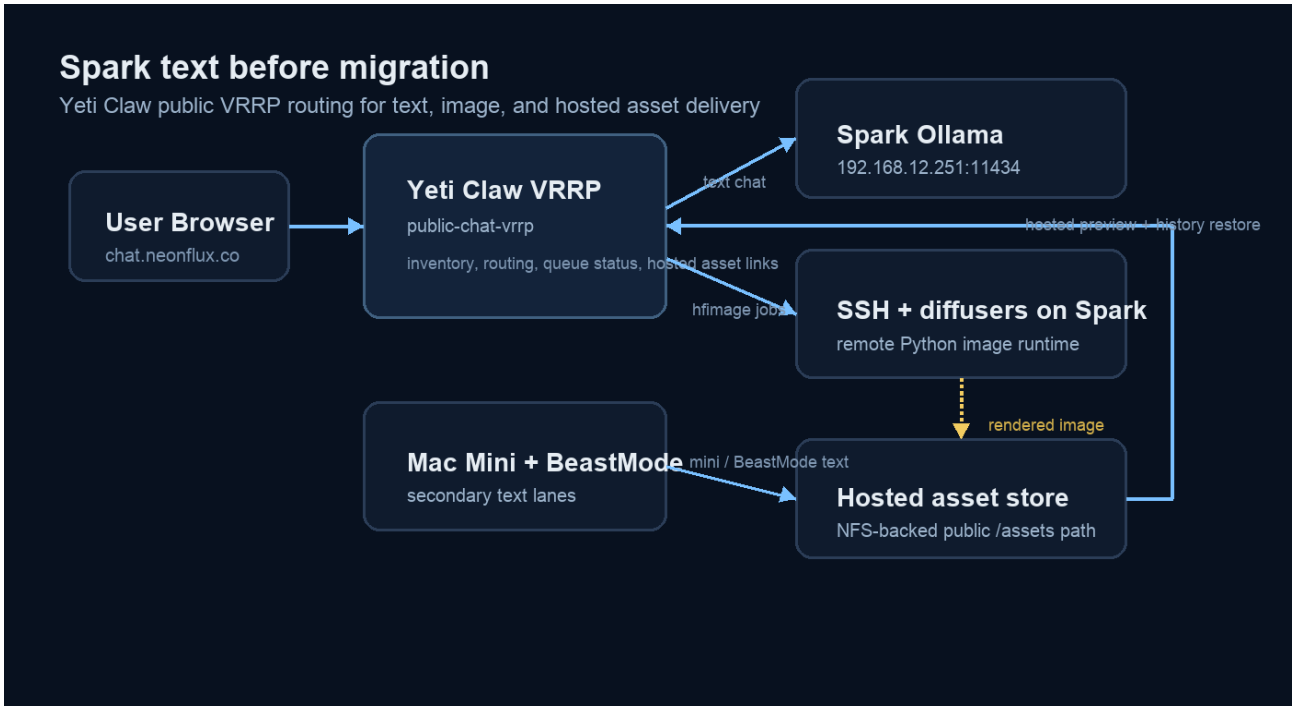
This report captures the before/after transition of the public Spark text lane on Yeti Claw from Ollama-backed **qwen3:8b** to a dedicated TensorRT-LLM service serving **Qwen/Qwen3-8B** through NVIDIA's official **1.3.0rc12.post1** release container on Spark. The goal was to preserve a publishable measurement trail before the public cutover and document the real compatibility work needed on Spark.

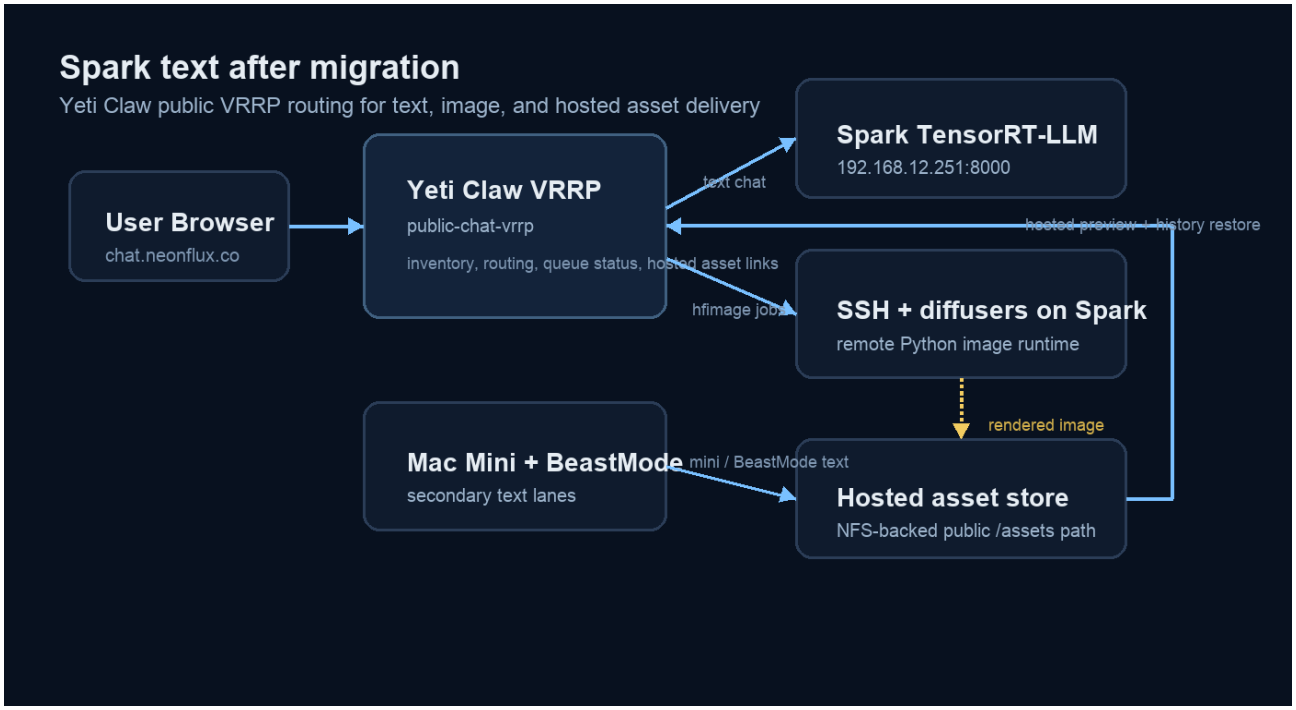
Prompt under test: `In exactly six bullet points, give six practical tips for keeping an inference server responsive under moderate concurrency. Keep each bullet under twelve words.`

Executive Read

- At concurrency 1, average latency moved from 2541.67 ms before to 6997.54 ms after.
- At concurrency 4, aggregate completion throughput moved from 38.47 tok/s to 56.29 tok/s.
- Peak sampled GPU temperature during the concurrency-4 run moved from 65.00 C to 63.00 C.
- Spark image generation remained on the existing SSH + diffusers lane; only the Spark text lane changed.
- The final working stack used NVIDIA's 1.3.0rc12.post1 release container on Spark after two pip-based TensorRT-LLM branches failed in different ways during bring-up.

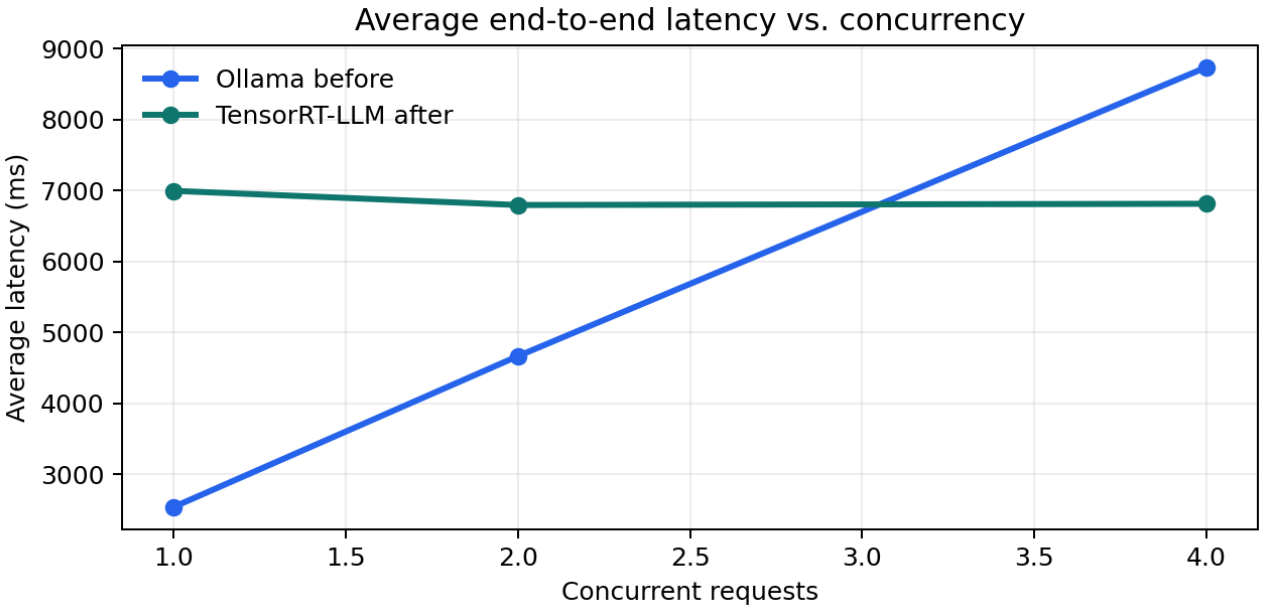
Architecture

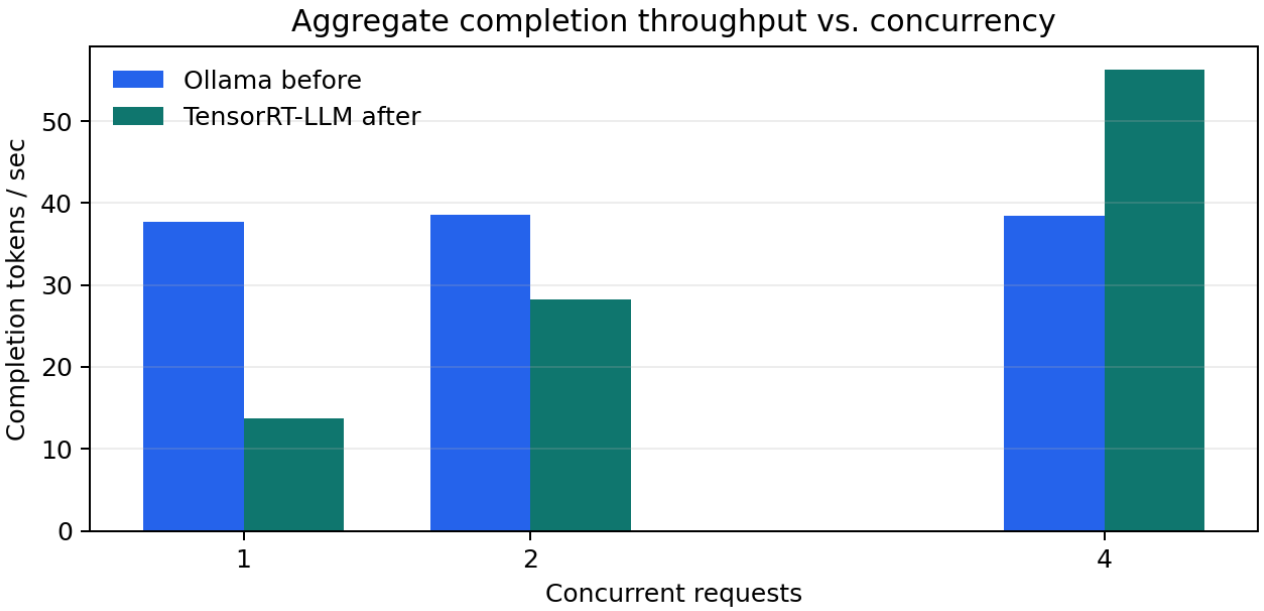



Spark text after migration
Yeti Claw public VRRP routing for text, image, and hosted asset delivery
User Browser
chat.neonflux.co
Yeti Claw VRRP
public-chat-vrrp
inventory, routing, queue status, hosted asset links
text chat
hfimage job
Spark TensorRT-LLM
192.168.12.251:8000
hosted preview + history restore
SSH + diffusers on Spark
remote Python image runtime
rendered image
Mac Mini + BeastMode
secondary text lanes
mini / BeastMode text
Hosted asset store
NFS-backed public /assets path

Before / After Benchmark Table

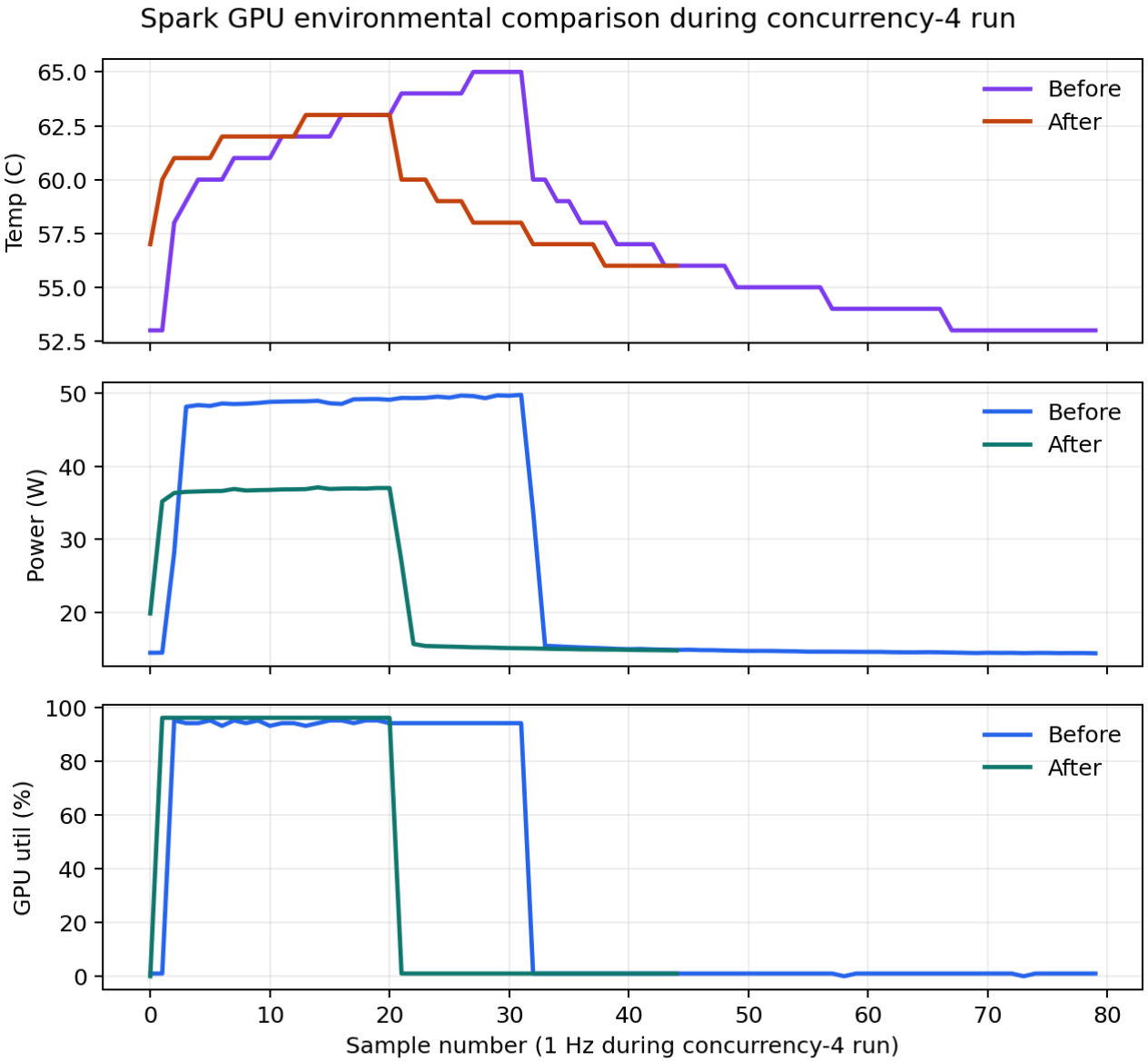
Concurrency	Before avg ms	After avg ms	Latency delta	Before tok/s	After tok/s	TPS delta
1	2541.67	6997.54	+175.3%	37.76	13.72	-63.7%
2	4667.74	6797.99	+45.6%	38.56	28.20	-26.9%
4	8738.93	6815.54	-22.0%	38.47	56.29	+46.3%





Spark Environmentals

Run	Peak temp (C)	Peak power (W)	Peak util (%)	Samples
Before (Ollama c4)	65.00	49.73	95.00	79
After (TRT c4)	63.00	37.10	96.00	44



Pros and Cons Of The Transition

Pros

- Spark text is now served through a dedicated TensorRT-LLM endpoint instead of sharing the same Ollama lane used for legacy text workflows.
- The final cutover uses NVIDIA’s official release container on Spark, which is a cleaner and more supportable path than a hand-built pip environment.
- TensorRT-LLM exposes OpenAI-compatible `/v1/chat/completions`, `/health`, and `/v1/models`, which gives the public stack a cleaner operator surface.
- The Spark image path is unchanged, so the text migration does not disturb the existing hosted-image workflow.
- The working path on Spark stayed inside NVIDIA’s current Blackwell-oriented TensorRT-LLM release container instead of the unsupported pip combinations that failed earlier.
- At concurrency 4, aggregate completion throughput climbed to 56.29 tok/s.

Cons

- The Spark text lane is now a second runtime to operate: separate container, sidecar port, model hydration path, and health checks all add surface area.
- The first-time startup path is heavier than Ollama because it must hydrate a large Hugging Face checkpoint before the API becomes ready.
- The direct pip path was not production-safe on Spark: newer 1.2.x wheels failed to import cleanly, and older fallback builds exposed unsupported-kernel failures on GB10 before the container path succeeded.
- The public Spark text catalog narrowed from 5 models to 1 models during the first cutover.

Operational Notes

- The public Spark text lane now expects an OpenAI-compatible server on port 8000 instead of a native Ollama-only chat path.
- The migration preserves Spark image generation as-is, so hfimage traffic still uses the SSH-driven queue and hosted asset workflow.
- The chosen runtime is TensorRT-LLM 1.3.0rc12.post1 in the NVIDIA release container because that is the path that successfully initialized on Spark and exposed a stable OpenAI-compatible endpoint.
- A small CUDA namespace shim was required so the older TensorRT-LLM code path could import the newer cuda-python package layout present on Spark.