GEARFLOW LAB PRODUCTION SRE RUNBOOK 05 | GGUF + VLLM + NVIDIA GB10 UMA

Local LLM VRAM, GQA KV Cache & NVIDIA GB10 128GB Sizing Matrix

Open VRAM Calculator ->

Incident Context: Sizing local LLMs solely by quantized file size triggers mid-generation CUDA OOM or `5 tok/s` PCIe CPU-offload thrashing once long-context KV cache (`32K-128K`) and concurrent agent slots (`OLLAMA_NUM_PARALLEL`) expand. On coherent ARM64 Grace-Blackwell `NVIDIA GB10` (`128GB LPDDR5x UMA`), `nvidia-smi` also prints `Memory-Usage: Not Supported` because there is no discrete PCIe `fb_memory` bar.

1. Exact Weights GiB, GQA KV Cache GiB & Token Decode Bandwidth Formulas

- **Quantized Weights:** `Weights_GiB = (Params_B * 10^9 * bpw / 8) / 2^30` -> `Q4_K_M = 4.85 bpw` | `Q5_K_M = 5.69 bpw` | `Q6_K = 6.56 bpw` | `Q8_0 = 8.50 bpw` | `FP16/BF16 = 16.0 bpw` (+ `0.55 to 1.20 GiB` CUDA/Metal compute graph buffer).
- **GQA KV Cache:** `KV_GiB = (2 * Layers * KV_Heads * Head_Dim * Context_Tokens * Parallel_Slots * Bytes_per_KV) / 2^30` where `Bytes_per_KV` = `2.00 B` (FP16), `1.0625 B` (`Q8_0` / FP8), or `0.5625 B` (`Q4_0`). **Note:** Ollama allocates `Context * OLLAMA_NUM_PARALLEL` upfront!
- **Single-Stream Decode Roofline:** `Decode_tok_s ~= (0.72 * Mem_Bandwidth_GBs) / Active_Weights_KV_GiB` (memory-bandwidth bound at batch=1).

2. Hardware Fit & Single-Stream Decode `tok/s` Reference Matrix (`Q4_K_M` Weights + `Q8_0` KV)

Accelerator / Host	Usable VRAM	Bandwidth	14B at 32K (~11.8G)	32B at 32K (~24.2G)	70B at 64K (~54.8G)
RTX 5070 Ti 16GB GDDR7	16.0 GiB	896 GB/s	Fit (~62 tok/s)	OOM / PCIe Spill	OOM (Needs 4x GPU)
RTX 3090 / 4090 24GB	24.0 GiB	1008 GB/s	Fit (~70 tok/s)	Tight (Use 16K/ Q4 KV)	OOM (Needs 2x 24GB)
Mac mini M4 Pro 48GB / 64GB	36-48 GiB	273 GB/s	Fit (~21 tok/s)	100% Fit (~10 tok/s)	64GB Only (Tight)
Mac Studio M4 Max 128GB	~96 GiB	546 GB/s	Fit (~42 tok/s)	100% Fit (~20 tok/s)	100% Fit (~9 tok/s)
NVIDIA GB10 128GB UMA	~118 GiB	273 GB/s	Fit (~20 tok/s)	100% Fit (~10 tok/s)	100% Fit (~4.5 tok/s)

3. Production Runtime Flags (`ollama` / `vllm` / `llama-server`)

```
# Ollama: Enable Flash Attention + Q8_0 KV quantization (halves KV RAM!)
OLLAMA_FLASH_ATTENTION=1 \
OLLAMA_KV_CACHE_TYPE=q8_0 \
OLLAMA_NUM_PARALLEL=2 \
OLLAMA_CONTEXT_LENGTH=32768 \
ollama serve

# llama.cpp server: Explicit GGUF + Q8_0 K/V cache + 2 parallel slots
llama-server -m Qwen2.5-32B-Instruct-Q4_K_M.gguf \
  -ngl 999 -fa -ctk q8_0 -ctv q8_0 -c 65536 -np 2 --host 127.0.0.1

# vLLM: PagedAttention + FP8 KV cache + explicit VRAM ceiling
vllm serve Qwen/Qwen2.5-32B-Instruct-AWQ \
  --kv-cache-dtype fp8 --max-model-len 32768 \
  --gpu-memory-utilization 0.90 --host 127.0.0.1 --port 8000
```

4. NVIDIA GB10 UMA Telemetry (`nvidia-smi` Fix)

```
# On DGX Spark / ASUS Ascent GX10 / GIGABYTE AI TOP ATOM (GB10):
# nvidia-smi prints "Not Supported" for Memory-Usage by design
# because CPU & GPU share 128GB coherent LPDDR5x UMA!

# 1. Read true allocatable UMA headroom via /proc/meminfo:
awk '/MemTotal|MemAvailable|Cached/ {printf "%-14s %.2f GiB\n", \
  $1, $2/1048576}' /proc/meminfo

# 2. Drop clean Linux page cache before loading 70B+ models
# (prevents premature UMA allocator fallback):
sudo sysctl -w vm.drop_caches=3

# 3. Live GPU compute % + host UMA gauge without NVML error:
nvtop   # or: nvidia-smi dmon -s pucvmet -d 1
```

Quick Prefill Presets & GB10 Guide: GB10 128GB · Qwen 32B | GB10 128GB · Llama 70B 64K | RTX 5070 Ti 16GB · 14B | GB10 UMA Fix (#8280) | All Tools (/tools/)