Ollama Not Using GPU: Complete Fix Guide

February 18, 2026 · by Mark Bartlett

Download this guide as PDF

> **Quick Answer:** Run `ollama ps` — if it shows 100% CPU, Ollama isn't detecting your GPU. For NVIDIA: check that `nvidia-smi` works (if not, install drivers), then reinstall Ollama so it picks up CUDA. For AMD: check that `rocminfo` shows your GPU (if not, install ROCm and add user to render/video groups). For Mac: Metal works automatically — update Ollama if it doesn't. Common gotcha: model too big for VRAM causes silent CPU fallback — try a smaller quant.

📚 **More on this topic:** Ollama Troubleshooting Guide · Run Your First Local LLM · AMD vs NVIDIA for Local AI · Planning Tool

Ollama is running. The model loads. Tokens generate. But it's painfully slow — because everything is running on your CPU while your GPU sits idle.

Here's how to confirm it and fix it.

Step 1: Confirm the Problem

Before fixing anything, verify that Ollama is actually ignoring your GPU.

Check with ollama ps

```
ollama ps
```

This shows loaded models and which device they're using. Look for the processor column:

```
NAME          ID          SIZE    PROCESSOR    UNTIL
qwen3:8b      abcdef12    5.3 GB  100% GPU     4 minutes from now   ← GPU ✓
qwen3:8b      abcdef12    5.3 GB  100% CPU     4 minutes from now   ← Problem
qwen3:8b      abcdef12    5.3 GB  48% GPU/52% CPU  4 minutes from now   ← Partial offload
```

If it says **100% CPU**, Ollama isn't using your GPU at all. If it shows a GPU/CPU split, the model is partially offloaded — likely because it doesn't fully fit in VRAM.

Check with nvidia-smi

For NVIDIA GPUs, run this while generating:

```
# Start a generation in one terminal
ollama run qwen3:8b "Write a long story about robots"

# In another terminal, watch GPU usage
nvidia-smi -l 1
```

During generation, GPU utilization should spike to 50-100% and memory usage should show the model loaded. If GPU utilization stays at 0%, Ollama is not using the GPU.

NVIDIA Fixes

nvidia-smi Doesn't Work

If `nvidia-smi` returns `command not found` or `NVIDIA-SMI has failed`, the NVIDIA driver isn't installed.

```
# Ubuntu/Debian
sudo apt install nvidia-driver-560
sudo reboot

# Verify after reboot
nvidia-smi
```

Use the latest driver version for your card. The `560` series supports all GPUs from GTX 900 and newer. After installing, reboot — the kernel module must load.

Ollama Installed Before CUDA

Ollama detects available GPU libraries at install time. If you installed Ollama before setting up NVIDIA drivers, it won't know CUDA exists.

Fix: Reinstall Ollama after the driver is working:

```
# Verify driver works
nvidia-smi

# Reinstall Ollama
curl -fsSL https://ollama.com/install.sh | sh
```

The install script detects CUDA and builds the right configuration. This is the most common fix for NVIDIA users.

Docker: Missing GPU Flag

If Ollama runs in Docker, the container doesn't get GPU access by default.

```
# Wrong — no GPU
docker run -d ollama/ollama

# Right — with GPU
docker run -d --gpus all ollama/ollama
```

You also need `nvidia-container-toolkit` installed on the host:

```
# Install the toolkit
sudo apt install nvidia-container-toolkit
sudo systemctl restart docker
```

Without the toolkit, `--gpus all` silently does nothing on some setups.

Multiple GPUs

If you have multiple NVIDIA GPUs, Ollama uses the first one by default. To specify which GPU:

```
# Use only GPU 1 (second GPU)
CUDA_VISIBLE_DEVICES=1 ollama serve

# For systemd service
```

```
sudo systemctl edit ollama
# Add: Environment="CUDA_VISIBLE_DEVICES=0"
```

Check available GPUs with `nvidia-smi -L` to see their indices.

AMD Fixes

AMD GPU support in Ollama requires ROCm. The diagnostics are different.

rocminfo Doesn't Show Your GPU

```
rocminfo
```

Look for an "Agent" entry with your GPU name. If only the CPU agent appears, ROCm isn't detecting your card.

Fix checklist:

```
# 1. Add user to required groups
sudo usermod -a -G render,video $USER
# Log out and back in

# 2. Check that the amdgpu driver is loaded
lsmod | grep amdgpu

# 3. Verify /dev/kfd exists
ls -la /dev/kfd
```

If `/dev/kfd` doesn't exist, the ROCm kernel driver isn't installed. Reinstall ROCm from scratch.

GPU Not in Ollama's Allowlist

Even if `rocminfo` shows your GPU, Ollama has a hardcoded list of supported AMD GPUs. Cards like the RX 6600 (gfx1032) aren't in the list and default to CPU silently.

Fix: Use the HSA_OVERRIDE hack:

```
# For systemd Ollama service
sudo systemctl edit ollama
# Add under [Service]:
# Environment="HSA_OVERRIDE_GFX_VERSION=10.3.0"
sudo systemctl restart ollama
```

Use `10.3.0` for RDNA 2 cards (RX 6600/6700/6800 series) and `11.0.0` for RDNA 3 APUs. See the full ROCm troubleshooting guide for the complete GFX version table.

---

macOS (Apple Silicon)

Metal acceleration should work automatically on M1, M2, M3, and M4 Macs. If it's not:

Update Ollama

Older Ollama versions had Metal bugs on certain chip variants. Update to the latest:

```
brew upgrade ollama
# Or download the latest from ollama.com
```

Verify Metal Is Active

Open **Activity Monitor** → **Window** → **GPU History**. During generation, you should see GPU activity.

You can also check Ollama's logs:

```
cat ~/.ollama/logs/server.log | grep -i metal
# Should show "Metal: enabled" or similar
```

On Apple Silicon, there's no separate driver to install — Metal is part of macOS. If Ollama still uses CPU, it's almost always a version issue. Reinstall the latest version.

Unified Memory Note

On Macs, GPU and CPU share the same memory pool. `ollama ps` may show "GPU" even though you don't have a discrete GPU — that's correct. Apple's Metal uses unified memory for GPU compute. Performance depends on memory bandwidth: M4 Pro and Max are significantly faster than base M1/M2.

Common Gotchas

Model Too Big → Silent CPU Fallback

This is the sneakiest one. If a model doesn't fit in VRAM, Ollama silently falls back to CPU for the layers that don't fit. A 32B Q4 model needs ~20 GB VRAM. If you have 8 GB, most layers run on CPU and it's painfully slow.

Diagnose: `ollama ps` shows the GPU/CPU split. If it says "20% GPU / 80% CPU", the model barely fits.

Fix: Use a model that fits your VRAM:

VRAM	Largest Comfortable Model
6 GB	7B Q4
8 GB	8B Q4 or 14B Q3
12 GB	14B Q4
16 GB	14B Q6 or 32B Q3
24 GB	32B Q4 or 70B Q3 (partial)

See VRAM requirements for the complete table.

VRAM Occupied by Other Processes

Your GPU isn't only for LLMs. Desktop compositors, browsers with hardware acceleration, and video playback all consume VRAM.

```
# Check what's using VRAM
nvidia-smi
```

Look at the process list at the bottom. If Firefox or Chrome is using 1-2 GB, that's VRAM you don't have for inference. Close heavy browser tabs or disable hardware acceleration in your browser settings.

SSH Without GPU Access

If you SSH into a machine running Ollama, the Ollama service on that machine should still use the GPU — SSH doesn't affect GPU access for background services.

But if you're starting `ollama serve` manually via SSH, make sure the user has GPU permissions (render and video groups on AMD, no special groups needed for NVIDIA).

Snap/Flatpak Installs

Snap and Flatpak sandboxing can block GPU access. If you installed Ollama through either:

```
# Remove sandboxed version
snap remove ollama  # or flatpak uninstall ...

# Install directly
curl -fsSL https://ollama.com/install.sh | sh
```

The direct install script handles GPU detection and permissions correctly. Sandboxed installs may miss the GPU driver libraries.

Verification Checklist

After applying fixes, confirm everything is working:

```
# 1. GPU driver works
nvidia-smi          # NVIDIA
rocminfo            # AMD
```

```
# 2. Ollama sees the GPU
ollama run qwen3:8b "Hello"
ollama ps           # Should show GPU in processor column

# 3. Speed matches expectations
# 7B Q4 on RTX 3060:  ~35-45 tok/s
# 7B Q4 on RTX 4090:  ~75-85 tok/s
# 7B Q4 on RX 7900 XTX: ~50-55 tok/s
# 7B Q4 on CPU (good):  ~5-10 tok/s
```

If you're getting CPU-level speeds (~5-10 tok/s) on a GPU that should do 40+, something is still wrong. Walk through the fixes above for your GPU vendor.

Bottom Line

Ollama uses the GPU automatically — when it can find it. NVIDIA needs working drivers installed before Ollama. AMD needs ROCm plus the right group permissions. Mac just works with current Ollama versions. The silent CPU fallback when models are too large catches most people — check `ollama ps` first and make sure the model fits your VRAM.

Source: https://insiderllm.com/guides/ollama-not-using-gpu-fix/