

References: -

<https://community.frame.work/t/amd-strix-halo-llama-cpp-installation-guide-for-fedora-42/75856> - <https://strix-halo-toolboxes.com/> - <https://github.com/kyuz0/amd-strix-halo-toolboxes>

This report outlines the deployment of the **Ollama LLM runtime** on **Arch Linux** specifically tailored for the **AMD Ryzen AI Max+ 395 APU**. The primary focus is optimizing performance by leveraging the integrated **Radeon 8060S iGPU** through the **Vulkan** backend, and considering the potential of the **XDNA 2 NPU** for heterogeneous acceleration.

## □ Overview of the Target Architecture

The AMD Ryzen AI Max+ 395 APU is a highly-integrated system-on-a-chip (SoC) featuring a heterogeneous architecture critical for efficient AI workloads:

- **Zen 5 CPU Cores:** Provide general-purpose processing and host for the Ollama service.
- **Radeon 8060S iGPU (RDNA 3.5):** The primary accelerator for LLM inference, utilizing shared system memory as **Unified Memory (VRAM)**.
- **XDNA 2 Neural Processing Unit (NPU):** Designed for high TOPS AI acceleration, with potential for hybrid execution (prefill/context ingestion on NPU, decoding on iGPU/CPU).

## □ Arch Linux Prerequisite Setup

Before deploying Ollama, the base Arch Linux installation must have the correct drivers and utilities to fully expose the APU's capabilities, especially for Vulkan and unified memory management.

### 1\ Kernel and Firmware

Set in bios UMA buffer size to 8G or lower.

Ensure the system is running a recent kernel (e.g., \$6.10+\$ or later) for optimal Zen 5 and RDNA 3.5 support.

[snippet.bash](#)

```
# Update system and install a recent kernel if not already running
sudo pacman -Syu linux linux-headers

# disable suspension
sudo systemctl mask sleep.target suspend.target hibernate.target
hybrid-sleep.target
```

add to kernel parameters

```
iommu=pt amdgpu.gttsize=126976 ttm.pages_limit=32505856
```

## 2\. Graphics and Compute Drivers (Vulkan)

The Vulkan backend relies on the open-source **Mesa** stack via the **RADV** driver.

[snippet.bash](#)

```
# Install Mesa with Vulkan support for AMDGPU
sudo pacman -S mesa vulkan-radeon lib32-vulkan-radeon vulkan-headers
```

```
### 3\. ROCm (Optional but Recommended) While the objective is Vulkan,
installing the ROCm stack is often necessary for complete AMD GPU compute
support and may be leveraged by other frameworks or future Ollama features.
The Ryzen AI Max+ 395 is based on the gfx1150 target, which has improved
support in recent ROCm versions (e.g., $6.4+$). ``bash # Install essential
ROCm packages yay -S rocm-core amdgpu_top rocminfo rocm-device-libs yay -S
rocm-hip-sdk rocm-opencl-runtime sudo usermod -a -G render,video $USER # xdna
yay -S amdxdna-driver-bin xrt-npu-git `` IMPORTANT: add /opt/rocm/bin to
PATH ### 4\. Memory Configuration The iGPU uses shared system RAM (Unified
Memory). For optimal LLM performance, a large dedicated memory pool for
graphics (iGPU/VRAM) is essential. * BIOS/UEFI Configuration: Set the
UMA Frame Buffer Size (or equivalent) in the BIOS/UEFI to the maximum
supported value (e.g., 16GB or more, depending on total system RAM). This
is the most crucial step for allocating dedicated VRAM. * Kernel Parameters
(Optional): For fine-tuning, verify kernel parameters if necessary, but the
BIOS setting is typically sufficient. ----- ## Ollama Deployment with
Vulkan Acceleration Ollama relies on an underlying LLM engine (typically a
fork of llama.cpp). Recent Ollama releases (e.g., $0.12.6+$) include
experimental support for Vulkan acceleration, which must be explicitly
enabled via an environment variable. ### 1\. Installation The easiest method
is using the Arch package manager or downloading the official binary. ``bash
# Install Ollama from the Arch Linux repository (or AUR) sudo pacman -S
ollama-vulkan `` ### 2\. Enabling Vulkan Backend To force Ollama to utilize
the more performant Vulkan backend on the AMD iGPU, the OLLAMA_VULKAN
environment variable must be set when running the service. ##### Systemd
Service Configuration (Recommended) Modify the Ollama systemd service unit
to include the required environment variable. 1. Create an override
directory: ``bash sudo mkdir -p /etc/systemd/system/ollama.service.d/ ``
2. Create the override file
(/etc/systemd/system/ollama.service.d/vulkan_override.conf): ``ini [Service]
Environment="OLLAMA_VULKAN=1" # Optional: Set the number of threads for the
CPU fallback/host operations # Environment="OLLAMA_NUM_THREADS=16" `` 3.
Reload the daemon and restart the service: ``bash sudo systemctl daemon-
reload sudo systemctl restart ollama `` ##### Verification Check the
service logs for confirmation that Vulkan initialization was successful:
``bash sudo journalctl -u ollama -f `` Look for messages indicating
Vulkan/GGML initialization on the iGPU. Ollama may also log which accelerator
is being used when a model is run. ### 3\. Testing Model Offload Test a small
model, ensuring the output indicates GPU/Vulkan usage. The number of layers
offloaded (-gpu-layers) is often determined automatically by the available VRAM
```

(shared RAM). ``bash ollama run llama3:8b # After the model downloads, monitor system resource usage (e.g., with htop and radeontop) # The prompt prefill phase will typically show high iGPU usage. `` - - - - - ## ☑

Performance Tuning Notes | Component | Tuning Action | Rationale | | :--- | :--- | :--- | | \*\*System RAM\*\* | Maximize physical RAM (e.g., 64GB/128GB). | The iGPU uses unified memory; more RAM directly equates to more \*\*VRAM\*\* for larger models/context. | | \*\*BIOS/UEFI\*\* | Set UMA Frame Buffer Size to max (e.g., 16GB-32GB). | Crucial for allocating a large, dedicated memory pool for GPU offload. | | \*\*Ollama\*\* | Use \*\*Q4\_K\_M\*\* or \*\*Q5\_K\_M\*\* quantization. | Optimal balance of VRAM usage and inference speed/quality. Larger quantizations (Q6/Q8) may be slower or consume too much VRAM. | | \*\*Vulkan\*\* | Ensure OLLAMA\_VULKAN=1` is set. | Forces the use of the Vulkan backend, which is generally reported as faster than ROCm on APUs for LLM workloads. |

|                        |   |   |
|------------------------|---|---|
| <b>Model Selection</b> | Prioritize <b>MoE (Mixtral, DeepSeek)</b> models. | The architecture excels at MoE models by efficiently leveraging the available memory bandwidth and unified memory for large models with small active experts. |
|------------------------|---|---|

## lemonade-server

[download](#)

```
yay -S lemonade-server
```

oga-hybrid mode: this splits the work, the NPU handles the prefill (prompt processing), and the iGPU handles the generation.

[download](#)

```
lemonade-server run Qwen3-Coder-30B-A3B-Instruct-GGUF --recipe oga-hybrid --llamacpp rocm
```

[download](#)

```
curl http://localhost:8000/api/v1/chat/completions \
  -H "Content-Type: application/json" \
  -d '{
    "model": "Qwen3-Coder-30B-A3B-Instruct-GGUF",
    "messages": [{"role": "user", "content": "Who are you?"}],
    "stream": false
  }'
curl http://localhost:8000/api/v1/stats
```

# Comfyui

pypi packages

download from <https://repo.radeon.com/rocm/manylinux/rocm-rel-7.2/> and install triton, torch, torchvision and torchaudio

[download](#)

```
cd wheels
wget
https://repo.radeon.com/rocm/manylinux/rocm-rel-7.2/torch-2.10.0.dev20260111%2Brocm7.2.0.lw.gitdea53f5b-cp312-cp312-linux_x86_64.whl
wget
https://repo.radeon.com/rocm/manylinux/rocm-rel-7.2/torchaudio-2.10.0%2Brocm7.2.0.git3b0e7a6f-cp312-cp312-linux_x86_64.whl
wget
https://repo.radeon.com/rocm/manylinux/rocm-rel-7.2/torchvision-0.25.0%2Brocm7.2.0.gitaa35ca19-cp312-cp312-linux_x86_64.whl
wget
https://repo.radeon.com/rocm/manylinux/rocm-rel-7.2/triton-3.5.1%2Brocm7.2.0.gita272dfa8-cp312-cp312-linux_x86_64.whl

# install
pip install triton*
pip install torch*
```

install comfyui packages

[download](#)

```
pip install -r requirements.txt
```

native packages **to be checked** after rocm 7.2

```
yay -S python-pytorch-rocm python-torchvision-rocm python-torchaudio-rocm
#uv pip install --pre torchvision torchaudio --index-url
https://download.pytorch.org/whl/nightly/rocm6.3
```

## Benchmark

```
git clone --depth 1 https://github.com/ggerganov/llama.cpp.git
cd llama.cpp
# Use CMAKE to enable Vulkan
```

```
cmake -DGGML_VULKAN=ON -B build
cmake --build build --config Release -j15

# This command downloads the 4.92 GB model file directly.
wget https://huggingface.co/bartowski/Meta-Llama-3-8B-Instruct-
GGUF/resolve/main/Meta-Llama-3-8B-Instruct-Q4_K_M.gguf
https://huggingface.co/bartowski/Llama-3.2-3B-Instruct-
GGUF/resolve/main/Llama-3.2-3B-Instruct-Q4_K_M.gguf

# bench
./build/bin/llama-bench -m Meta-Llama-3-8B-Instruct-Q4_K_M.gguf
./build/bin/llama-bench -m Llama-3.2-3B-Instruct-Q4_K_M.gguf
```

ollama

```
ollama run llama3.2 --verbose 'explain nuclear fusion'
```

```
yay -S python-huggingface-hub
hf download Qwen/Qwen3-VL-8B-Instruct
```

From:  
<https://wiki.csgalileo.org/> - **Galileo Labs**

Permanent link:  
<https://wiki.csgalileo.org/tips/rocm>

Last update: **2026/02/15 08:33**

