Building and Running HunyuanImage-2.1 on AMD GPU (ROCm / Radeon AI PRO R9700, gfx1201)

> This document demonstrates how to build and run **Tencent HunyuanImage-2.1** on **AMD GPU (ROCm)** environment.
> Test environment: **rocm/pytorch:rocm6.4.2_ubuntu24.04_py3.12_pytorch_release_2.6.0** + **AMD Radeon AI PRO R9700 (gfx1201)**.
> Conclusion: **Stage-1 runs successfully**; **Refiner (Stage-2) is prone to being killed by the system when VRAM is insufficient**. It is recommended to generate images with Stage-1 first, or try Refiner with a GPU that has larger VRAM.

Table of Contents

Prerequisites

- AMD/ROCm drivers and hardware are ready (check with `rocminfo` / `hipcc --version`)
- Docker or run directly on host machine
- **VRAM Recommendation**: Official 2K generation requires at least **24 GB VRAM**. When **Refiner** is enabled, it often exceeds 24 GB in practice; if VRAM is insufficient, only run Stage-1 first

---

Launch Container

Pull and Run Docker Container

bash

```
docker pull rocm/pytorch:rocm6.4.2_ubuntu24.04_py3.12_pytorch_release_2.6.0

docker run -it --rm \
  --device=/dev/kfd --device=/dev/dri \
  --group-add video \
  --cap-add=SYS_PTRACE --security-opt seccomp=unconfined \
  --ipc=host --shm-size 16G \
  -v /path/to/workspace:/workspace \
  rocm/pytorch:rocm6.4.2_ubuntu24.04_py3.12_pytorch_release_2.6.0
```

Install Basic Tools in Container

bash

```
apt-get update && apt-get install -y git build-essential ninja wget
pip install --upgrade pip setuptools wheel
```

Install ROCm FlashAttention (Triton Backend)

Important: `triton==3.2.0` must be installed first; `FLASH_ATTENTION_TRITON_AMD_ENABLE=TRUE` is a required environment variable.

bash

```
# 1) Install Triton compiler (required version)
pip install triton==3.2.0

# 2) Install ROCm flash-attention
cd /workspace
git clone https://github.com/ROCm/flash-attention.git
cd flash-attention
git checkout main_perf
FLASH_ATTENTION_TRITON_AMD_ENABLE="TRUE" python setup.py install
```

Verify FlashAttention (AMD/Triton) Import

Set Environment Variable Before Verification

bash

```
export FLASH_ATTENTION_TRITON_AMD_ENABLE=TRUE
```

Run Verification Script

bash

```
python - <<'PY'
import importlib, os

# Check environment variables
env_vars = [
    "FLASH_ATTENTION_TRITON_AMD_ENABLE",
    "HSA_OVERRIDE_GFX_VERSION",
    "TRITON_CODEGEN_AMD_GPU_ARCH"
]
print({k: os.environ.get(k) for k in env_vars})

# Load flash_attn module
m = importlib.import_module("flash_attn")
print("flash_attn loaded from:", getattr(m, "__file__", m))

# Import required symbols
from flash_attn.bert_padding import pad_input, unpad_input
from flash_attn import (
    flash_attn_varlen_func,
    flash_attn_varlen_qkvpacked_func
)
print("✅ AMD/Triton symbols import OK")
PY
```

Expected Output

```python
{
  'FLASH_ATTENTION_TRITON_AMD_ENABLE': 'TRUE',
  'HSA_OVERRIDE_GFX_VERSION': None,
  'TRITON_CODEGEN_AMD_GPU_ARCH': None
}
flash_attn loaded from:
 /opt/conda/envs/py_3.12/lib/python3.12/site-packages/
 flash_attn/__init__.py
✅ AMD/Triton symbols import OK
```

Download HunyuanImage-2.1 and Model Weights

Clone Project and Install Dependencies

```bash
cd /workspace
git clone https://github.com/Tencent-Hunyuan/HunyuanImage-2.1.git
cd HunyuanImage-2.1
pip install -r requirements.txt
```

Download Weights (according to checkpoints-download.md)

```bash
```

```
pip install -U "huggingface_hub[cli]" modelscope
cd /workspace/HunyuanImage-2.1

# Example downloads (adjust according to actual needs;
# use offline copy if network unavailable)
# hf download tencent/HunyuanImage-2.1 --local-dir ./ckpts
# hf download google/byt5-small \
#   --local-dir ./ckpts/text_encoder/byt5-small
# hf download Qwen/Qwen2.5-VL-7B-Instruct \
#   --local-dir ./ckpts/text_encoder/llm
# modelscope download --model AI-ModelScope/Glyph-SDXL-v2 \
#   --local_dir ./ckpts/text_encoder/Glyph-SDXL-v2
```

Run Test Script (hyimage_test.py)

Save as `/workspace/HunyuanImage-2.1/hyimage_test.py`

```
python
```

```python
import os
os.environ['PYTORCH_CUDA_ALLOC_CONF'] = 'expandable_segments:True'
import torch
from hyimage.diffusion.pipelines.hunyuanimage_pipeline import (
    HunyuanImagePipeline
)

# Supported model_name: hunyuanimage-v2.1,
# hunyuanimage-v2.1-distilled
model_name = "hunyuanimage-v2.1"
pipe = HunyuanImagePipeline.from_pretrained(
    model_name=model_name,
    use_fp8=True
)
pipe = pipe.to("cuda")

# The input prompt
prompt = (
    "A cute, cartoon-style anthropomorphic penguin plush toy "
    "with fluffy fur, standing in a painting studio, wearing "
    "a red knitted scarf and a red beret with the word "
    "\"Asrock\" on it, holding a paintbrush with a focused "
    "expression as it paints an oil painting of the Mona Lisa, "
    "rendered in a photorealistic photographic style."
)

# Generate with different aspect ratios
aspect_ratios = {
    "16:9": (2560, 1536),
    "4:3": (2304, 1792),
```

```
    "1:1": (2048, 2048),
    "3:4": (1792, 2304),
    "9:16": (1536, 2560),
}

width, height = aspect_ratios["1:1"]

image = pipe(
    prompt=prompt,
    width=width,
    height=height,
    use_reprompt=False,  # Disable reprompt (save memory)
    use_refiner=False,   # Disable refiner (avoid VRAM shortage)
    num_inference_steps=8 if "distilled" in model_name else 50,
    guidance_scale=3.25 if "distilled" in model_name else 3.5,
    shift=4 if "distilled" in model_name else 5,
    seed=649151,
)

image.save("generated_image.png")
```

Execute Script

```
bash

python hyimage_test.py
```

Performance and Resource Reference (Actual Testing)

Item	Value
Hardware	AMD Radeon AI PRO R9700 (gfx1201)
Output Size	2048×2048 (1:1)
Mode	Refiner disabled (`use_refiner=False`)
Steps	50
Time	✅ Approximately 5 minutes per image (R9700 actual testing)
Peak VRAM	Approximately 20.2 GB before Refiner loading

Additional Notes

- When Refiner is enabled, VRAM requirements surge, R9700 (24GB) is easily killed by the system
- Distilled version (`hunyuanimage-v2.1-distilled`) + 8-step inference takes approximately 1–2 minutes with lower VRAM requirements
- Initial ByT5 loading requires an additional approximately 1.2 GB VRAM

Why Refiner (Stage-2) Cannot Run and Recommendations

Reasons

- Refiner model is large, 2K resolution inference VRAM requirements often exceed 24GB
- Being killed by the system is a normal phenomenon, not a program error

Recommended Solutions

1. **Only run Stage-1** (`use_refiner=False`)
2. **Use GPU with larger VRAM** (≥32GB)
3. **Use distilled model**
4. **Reduce resolution or post-process upscaling**
5. **Advanced**: Release Stage-1 modules before loading refiner

---

Common Notes

Enable ROCm AOTriton SDPA (can reduce VRAM/improve speed)

```bash
export TORCH_ROCM_AOTRITON_ENABLE_EXPERIMENTAL=1
```

AMP Automatic Mixed Precision API Update

```python
# Old: torch.cuda.amp.autocast(...)
# New:
from torch.amp import autocast
with autocast("cuda", dtype=torch.bfloat16):
    ...
```

ByT5 Can Be Disabled

For testing workflow only, `use_byt5=False` can save approximately 1.2GB

Cache Variables

Change `TRANSFORMERS_CACHE` to use `HF_HOME`

Multi-round Generation VRAM Release

```python
import gc, torch
gc.collect()
torch.cuda.empty_cache()
```