# TencentOS Server

# TencentOS AI 应用实践方案

腾讯云

【联系我们】

我们致力于为您提供个性化的售前购买咨询服务，及相应的技术售后服务，任何问题请联系 4009100100或95716。

# 文档目录

# TencentOS AI 应用实践方案
# NVIDIA 部署实践

最近更新时间：2025-12-30 17:42:12

目前，TencentOS Server 4 已实现对 NVIDIA Driver 和 CUDA 的原生支持，为您使用 NVIDIA GPU 提供了完整的 RPM 二进制软件包，包括内核级驱动、系统管理工具、计算库及AI框架适配组件。

本文档将指导如何在 TencentOS Server 4 上快速完成 NVIDIA Driver 和 CUDA 的安装部署，并无缝运行上层 AI 模型与应用。

## 基础环境要求及说明

- **支持 TencentOS 内核版本**：6.6内核（含其间各小版本）。如您使用的是TencentOS 2 或 3系列（5.4内核），建议您升级至 TencentOS 4版本（6.6 内核）。
- **支持的 GPU 驱动和对应 CUDA 版本**：

| 驱动版本 | 对应CUDA版本 |
| --- | --- |
| 525.147.05 | 12.0.1 |
| 530.41.03 | 12.1.1 |
| 535.274.02 | 12.2.2 |
| 545.29.06 | 12.3.2 |
| 550.163.01 | 12.4.1 |
| 555.58.02 | 12.5.1 |
| 560.35.03 | 12.6.3 |
| 570.195.03 | 12.8.1 |
| 575.64.05 | 12.9.1 |
| 580.95.05 | 13.0.1 |

- **支持的 CUDA 版本**：11.8.0、12.0.1、 12.1.1、12.2.2、12.3.2、12.4.1、12.5.1、 12.6.3、12.8.1、12.9.1、13.0.1。

# 安装 NVIDIA Driver

## 安装 TencentOS EPOL源

```
dnf install epol-release
```

## 安装 NVIDIA Driver 驱动包

此处以 Nvidia-Driver-570 为例，若需安装其他版本，请更新以下命令中软件包名的版本号，将570替换为目标版本号即可。

```
# 安装开源模块
dnf install -y kmod-nvidia-570-dkms-open nvidia-570 nvidia-driver-570-
open

# 安装闭源模块
dnf install -y kmod-nvidia-570-dkms nvidia-570 nvidia-driver-570
```

**说明：**

NVIDIA Driver 分为开源和闭源版本，所以部署流程分为开源模块安装和闭源模块安装两部分。

- 闭源版本驱动由 NVIDIA 公司官方提供，内核模块来自 NVIDIA 官方 .run安装包中的闭源代码，通常拥有最新的功能和最优化的性能，包括对 NVIDIA 最新显卡架构的支持，以及对 DirectX、OpenGL 等图形接口的完整支持。缺点是它们不开放源代码，这可能会限制它们与 Linux 内核的兼容性，并且在社区支持方面不如开源驱动。
- 开源版本驱动（如 nouveau）是由社区维护的，内核模块来自 open-gpu-kernel-modules 仓库的开源代码，可以更好地与 Linux 内核集成。尽管它们可能在功能/性能上不如闭源驱动全面，但在某些方面，如内核兼容性，它们表现得更为出色。

## 安装验证

```
# 重启机器
sudo reboot
# 查看已安装的驱动软件包
dnf list installed | grep nvidia
# 查看驱动信息
nvidia-smi
```

> ⚠ **注意：**
> 实例重启会导致正在运行的应用和服务被强制终止，或文件和内存数据会丢失。请先在重启前做好数据保存等操作。

### 卸载驱动安装包（可选）

如需要切换驱动版本，请参见以下步骤卸载已安装的驱动安装包。

```
# 卸载开源模块
dnf remove -y kmod-nvidia-570-dkms-open nvidia-570 nvidia-driver-570-
open

# 卸载闭源模块
dnf remove -y kmod-nvidia-570-dkms nvidia-570 nvidia-driver-570
```

## 安装 CUDA

针对使用场景的不同，TencentOS 团队采用了全新的打包方案。新方案通过将 CUDA 拆分为 runtime 和 devel 两类子包，实现了组件精细化分层管理，既满足了轻量级生产部署的需求，又支持了代码编译和模型训练的完整开发环境。

| 优化方向 | 传统方案 | 新方案 |
| --- | --- | --- |
| 包结构 | 细分子包 | 拆分为 runtime /devel 两类子包 |
| 组件分层 | 静态库/工具与运行时捆绑 | 划分可执行文件、.so 动态库、头文件、.a 静态库 |

其中，runtime / devel，对应组件范围如下：

| 类型 | 组件范围 | 适用场景 |
| --- | --- | --- |
| Runtime | 可执行文件、 .so 动态库 | 轻量级生产部署 |
| Devel | runtime 、头文件、 .a 静态库 | 代码编译/模型训练 |

### 安装 CUDA 组件

此处以 cuda-12-8 为例，如需安装其他版本，请更新以下命令中软件包名的版本号，将12-8替换为目标版本号即可。

- **Runtime 包安装：**

```
dnf install -y cuda-runtime-12-8
```

- **Devel 包安装：**

```
dnf install -y cuda-devel-12-8
```

## 安装验证

```
# 查看已安装的 cuda 软件包
dnf list installed | grep cuda
# 查看 cuda 版本
nvcc --version
```

## 多版本共存支持（可选）

新版打包方案解决了 CUDA 多版本共存的问题，可下载多个版本的 CUDA，并利用 `update-alternatives` 进行版本切换。如需兼容 CUDA 多版本共存，请参见 安装 CUDA，依次安装所需版本。例如，当同时安装 CUDA 12.2 与 12.9 版本，可参见以下方式切换版本：

```
[root@c71022cd9591 /]# nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2025 NVIDIA Corporation
Built on Tue_May_27_02:21:03_PDT_2025
Cuda compilation tools, release 12.9, V12.9.86
Build cuda_12.9.r12.9/compiler.36037853_0
[root@c71022cd9591 /]# nvidia-smi
Wed Oct 29 10:53:38 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.261.03             Driver Version: 535.261.03   CUDA Version: 12.9     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:09.0 Off |                    0 |
| N/A   41C    P8              11W /  70W |      2MiB / 15360MiB |      0%      Default |
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+

+-----------------------------------------------------------------------------------------+
| Processes:                                                                              |
|  GPU   GI   CI        PID   Type   Process name                              GPU Memory |
|        ID   ID                                                               Usage      |
|=========================================================================================|
|  No running processes found                                                             |
+-----------------------------------------------------------------------------------------+
[root@c71022cd9591 /]# update-alternatives --config cuda

There are 2 programs which provide 'cuda'.

  Selection    Command
-----------------------------------------------
*+ 1           /usr/local/cuda-12.9
   2           /usr/local/cuda-12.2

Enter to keep the current selection[+], or type selection number: 2
[root@c71022cd9591 /]# nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2023 NVIDIA Corporation
Built on Tue_Aug_15_22:02:13_PDT_2023
Cuda compilation tools, release 12.2, V12.2.140
Build cuda_12.2.r12.2/compiler.33191640_0
[root@c71022cd9591 /]# nvidia-smi
Wed Oct 29 10:54:34 2025
+-----------------------------------------------------------------------------------------+
| NVIDIA-SMI 535.261.03             Driver Version: 535.261.03   CUDA Version: 12.2     |
|-----------------------------------------+----------------------+----------------------+
| GPU  Name                 Persistence-M | Bus-Id        Disp.A | Volatile Uncorr. ECC |
| Fan  Temp   Perf          Pwr:Usage/Cap |         Memory-Usage | GPU-Util  Compute M. |
|                                         |                      |               MIG M. |
|=========================================+======================+======================|
|   0  Tesla T4                       Off | 00000000:00:09.0 Off |                    0 |
| N/A   38C    P8              12W /  70W |      2MiB / 15360MiB |      0%      Default |
```

```
|                                         |                      |                  N/A |
+-----------------------------------------+----------------------+----------------------+
```

## 卸载 CUDA 组件（可选）

如需切换 CUDA 版本，请参见以下步骤卸载已安装的 CUDA 安装包。

- **Runtime 包：**

```
dnf remove -y cuda-runtime-12-8 cuda-toolkit-12-8-config-common
```

- **Devel 包：**

```
dnf remove -y cuda-runtime-12-8 cuda-toolkit-12-8-config-common cuda-1
2-8 cuda-devel-12-8 cuda-toolkit-12-8
```

# AI 应用安装与验证

## 运行大模型（以 vLLM 框架和 Qwen 示例）

### 安装 uv

```
# 下载 uv
dnf install pip
pip install uv

# 推荐使用 uv 安装，智能处理依赖冲突
uv pip install vllm==0.10.0 --system
```

### 服务启动

```
# 启动 vllm 服务，拉取大模型（此处以 Qwen 为例）
vllm serve "Qwen/Qwen3-0.6B"

# 若机器配置较低，可尝试如下命令启动服务
vllm serve "Qwen/Qwen3-0.6B" --gpu-memory-utilization 0.7 --max-model-
len 1024 --max-num-seqs 4 --dtype float16 --block-size 16
```

### 结果展示：

- **示例：命令行测试**

```
# 另起终端
curl -X POST "http://localhost:8000/v1/chat/completions" -H "Content-
Type: application/json" --data '{"model": "Qwen/Qwen3-0.6B","messages":
[{"role": "user","content": "What is the capital of France?"}]}'
```

- **示例：Python 代码部署和推理**

**创建python程序，内容如下：**

```
from vllm import LLM, SamplingParams
from transformers import AutoTokenizer

# 初始化模型参数
model_path = "Qwen/Qwen3-0.6B"  # 替换为实际模型路径
llm = LLM(
    model=model_path,
    dtype="half",  # 启用半精度推理（FP16）
    trust_remote_code=True,  # 对Hugging Face模型必须开启
    tensor_parallel_size=1,  # 单GPU运行
    gpu_memory_utilization=0.8  # 显存利用率控制
)

# 配置生成参数
sampling_params = SamplingParams(
    temperature=0.7,
    top_p=0.95,
    max_tokens=256,
    repetition_penalty=1.1
)

# 构建对话模板
messages = [
    {"role": "system", "content": "你是一个诗人"},
    {"role": "user", "content": "写一首关于春天的七言绝句"}
]
tokenizer = AutoTokenizer.from_pretrained(model_path)
prompt = tokenizer.apply_chat_template(
    conversation=messages,
```

```
    tokenize=False,
    add_generation_prompt=True
)

# 执行推理
outputs = llm.generate(
    prompts=[prompt],
    sampling_params=sampling_params
)

# 输出结果
for output in outputs:
    generated_text = output.outputs[0].text
    print(f"生成结果：\n{generated_text}")
```

## 运行大模型（以 Ollama + Qwen 示例）

### 安装

```
# 安装依赖
dnf install -y gawk
# 下载安装脚本并安装
curl -fsSL https://ollama.com/install.sh | sh
```

### 服务启动

```
# 服务启动
ollama serve
```

### 运行 Qwen 模型

```
# 拉取大模型（此处以 Qwen 为例）
ollama run qwen3:0.6b
```

# 常见问题

### nvidia-smi 显示的 CUDA Version 与实际 CUDA Toolkit 版本不一致

nvidia-smi 显示的 CUDA Version 与实际 CUDA Toolkit 版本不一致是正常现象，原因在于两者代表不同层面的版本信息。nvidia-smi 命令返回结果中，Driver Version 是当前安装的驱动版本，CUDA Version 代表驱动所支持的最高 CUDA Runtime API 版本，并非当前安装的 CUDA Toolkit 版本。如需查看 CUDA Toolkit 版本，请执行 `nvcc --version` 命令。

## cuda-compat 问题（Driver 与 CUDA 版本兼容）

在 CUDA 生态中，不同 CUDA 版本均有与之最为适配的驱动程序版本。实际上，每个 CUDA 工具包都配套提供了 NVIDIA 显示驱动程序包，此驱动程序能够全面支持该版本 CUDA 工具包所引入的各项功能，是在该 CUDA 版本开发阶段就紧密匹配、功能兼容性最优的版本对应关系。
关于 CUDA 版本与驱动版本的对应关系，请参见 官方版本发布说明 。但是，在实际应用场景里，如果需要突破这种既定对应关系，例如在旧版本的驱动上运行高版本的 CUDA，或者是在较新的驱动上运行旧版本的 CUDA，将不可避免地引发兼容性问题。

为满足高版本 CUDA 在老旧硬件设备上的运行需求，官方专门推出了 cuda-compat 包以实现兼容适配。其实际效果是，当使用该包时，高版本 CUDA 仅能调用当前驱动所支持的最高版本 CUDA 所具备的功能，而无法使用因 CUDA 升级而新增的功能，从功能体验层面来看，类似对 CUDA 版本进行“降级”处理。
具体结论如下：

- 驱动版本高于或等于 CUDA 版本，CUDA 不安装 cuda-compat，运行正常；安装 cuda-compat，可能发生异常。
- 驱动版本低于 CUDA 版本， CUDA 必须安装 cuda-compat。

## 低版本 CUDA 上运行 AI 框架（vLLM）时库文件缺失问题

vLLM 核心架构是自研高性能 C++/CUDA 内核，深度集成 PyTorch，编译时链接特定版本的 CUDA 运行时，与 PyTorch 的 CUDA 版本强相关。而高版本的 vLLM 都是根据较新的 PyTorch 来进行构建，因此在较低版本的系统 CUDA 环境中编译 vLLM 时会出现关键 symbol 缺失问题，具体报错信息可参考如下截图：

```
[root@42ce384e979b /]# nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2024 NVIDIA Corporation
Built on Thu_Mar_28_02:18:24_PDT_2024
Cuda compilation tools, release 12.4, V12.4.131
Build cuda_12.4.r12.4/compiler.34097967_0
[root@42ce384e979b /]# vllm serve "Qwen/Qwen3-0.6B" --gpu-memory-utilization 0.7 --max-model-len 1024 --max-num-seqs 4 --dtype float16 --block-size 16
Traceback (most recent call last):
  File "/usr/local/bin/vllm", line 4, in <module>
    from vllm.entrypoints.cli.main import main
  File "/usr/local/lib64/python3.11/site-packages/vllm/__init__.py", line 14, in <module>
    import vllm.env_override  # noqa: F401
    ^^^^^^^^^^^^^^^^^^^^^^^^
  File "/usr/local/lib64/python3.11/site-packages/vllm/env_override.py", line 5, in <module>
    import torch
  File "/usr/local/lib64/python3.11/site-packages/torch/__init__.py", line 409, in <module>
    from torch._C import *  # noqa: F403
    ^^^^^^^^^^^^^^^^^^^^^^
ImportError: /usr/local/lib64/python3.11/site-packages/torch/lib/libtorch_cpu.so: undefined symbol: cuptiActivityEnableDriverApi, version libcupti.so.12
```

例如，在 CUDA 12.2 中编译 vLLM 时出现 CUPTI 相关 symbol 缺失问题，经分析发现，PyTorch/Triton 自带的 CUPTI 库包含所需符号，但 CUDA 12.2 包含的 CUPTI 库缺少所需符号：

```
[root@89fddb9a7ac6 /]# nm -D /usr/local/lib/python3.11/site-packages/nvidia/cuda_cupti/lib/libcupti.so.12 | grep cuptiActivityEnableDriver
000000000012b7e0 T cuptiActivityEnableDriverApi@@libcupti.so.12
[root@89fddb9a7ac6 /]# nm -D /usr/local/lib64/python3.11/site-packages/triton/backends/nvidia/lib/cupti/libcupti.so.12 | grep cuptiActivityEnableDriver
0000000000107f00 T cuptiActivityEnableDriverApi@@libcupti.so.12
[root@89fddb9a7ac6 /]# nm -D /usr/local/cuda-12.2/extras/CUPTI/lib64/libcupti.so.12 | grep cuptiActivityEnableDriver
```

**切换到高版本的 CUDA 12.9 环境时，测试结果包含目标 symbol 且可以成功运行：**

```
[root@42ce384e979b /]# nm -D /usr/local/lib/python3.11/site-packages/nvidia/cuda_cupti/lib/libcupti.so.12 | grep cuptiActivityEnableDriver
000000000012b7e0 T cuptiActivityEnableDriverApi@@libcupti.so.12
[root@42ce384e979b /]# nm -D /usr/local/cuda-12.2/extras/CUPTI/lib64/libcupti.so.12 | grep cuptiActivityEnableDriver
nm: '/usr/local/cuda-12.2/extras/CUPTI/lib64/libcupti.so.12': No such file
[root@42ce384e979b /]# nm -D /usr/local/cuda-12.9/extras/CUPTI/lib64/libcupti.so.12 | grep cuptiActivityEnableDriver
000000000010b5c0 T cuptiActivityEnableDriverApi@@libcupti.so.12
[root@42ce384e979b /]#
```

```
[root@42ce384e979b /]# nvcc --version
nvcc: NVIDIA (R) Cuda compiler driver
Copyright (c) 2005-2025 NVIDIA Corporation
Built on Fri_Feb_21_20:23:50_PST_2025
Cuda compilation tools, release 12.8, V12.8.93
Build cuda_12.8.r12.8/compiler.35583870_0
[root@42ce384e979b /]# vllm serve "Qwen/Qwen3-0.6B" --gpu-memory-utilization 0.7 --max-model-len 1024 --max-num-seqs 4 --dtype float16 --block-size 16
INFO 11-06 19:46:53 [__init__.py:235] Automatically detected platform cuda.
INFO 11-06 19:46:57 [api_server.py:1755] vLLM API server version 0.10.0
INFO 11-06 19:46:57 [cli_args.py:261] non-default args: {'model_tag': 'Qwen/Qwen3-0.6B', 'dtype': 'float16', 'max_model_len': 1024, 'block_size': 16, 'gpu_memory_uti
lization': 0.7, 'max_num_seqs': 4}
`torch_dtype` is deprecated! Use `dtype` instead!
WARNING 11-06 19:47:08 [config.py:3443] Casting torch.bfloat16 to torch.float16.
INFO 11-06 19:47:08 [config.py:1604] Using max model len 1024
WARNING 11-06 19:47:08 [arg_utils.py:1690] Compute Capability < 8.0 is not supported by the V1 Engine. Falling back to V0.
INFO 11-06 19:47:08 [api_server.py:268] Started engine process with PID 286
INFO 11-06 19:47:14 [__init__.py:235] Automatically detected platform cuda.
INFO 11-06 19:47:18 [llm_engine.py:228] Initializing a V0 LLM engine (v0.10.0) with config: model='Qwen/Qwen3-0.6B', speculative_config=None, tokenizer='Qwen/Qwen3-0
.6B', skip_tokenizer_init=False, tokenizer_mode=auto, revision=None, override_neuron_config={}, tokenizer_revision=None, trust_remote_code=False, dtype=torch.float16
, max_seq_len=1024, download_dir=None, load_format=LoadFormat.AUTO, tensor_parallel_size=1, pipeline_parallel_size=1, disable_custom_all_reduce=False, quantization=N
one, enforce_eager=False, kv_cache_dtype=auto,  device_config=cuda, decoding_config=DecodingConfig(backend='auto', disable_fallback=False, disable_any_whitespace=Fal
```

**以上情况，您可通过设置环境变量，指定动态链接器搜索共享库（.so文件）的路径。当运行需要特定版本 CUDA 库的程序时，系统会优先在这个路径中查找库文件。具体命令如下：**

```
export LD_LIBRARY_PATH=/usr/local/lib/python3.11/site-
packages/nvidia/cuda_cupti/lib:$LD_LIBRARY_PATH
```

# 沐曦 MetaX 部署实践

最近更新时间：2025-12-30 17:42:12

目前，TencentOS Server 已实现对沐曦 MetaX 驱动与 MXMACA SDK 的原生支持，为使用沐曦 GPU 提供了完整的 RPM 二进制软件包，包括内核级驱动、系统管理工具、计算库及 AI 框架适配组件。

本文档将指导如何在 TencentOS Server 4 上快速完成沐曦 MetaX 驱动与 MXMACA SDK 的安装部署，并无缝运行上层 AI 模型与应用。

## 基础环境要求及说明

- **支持 TencentOS 内核版本**：系统及内核要求参见下表，仅支持该表中系统及内核版本。如果您的实例低于支持内核版本，请先升级内核（如 `dnf install kernel-6.6.98-40.2.tl4`）。同时，在使用 MXMACA-sdk 驱动时，建议 GCC 版本与系统发行版保持一致，CMake 版本不低于 3.10。

  > ⚠ **注意：**
  > - 升级操作系统内核风险较高，可能会导致系统不稳定或出现兼容性问题。在操作内核升级前，请充分了解升级可能出现的问题，建议同步备份重要数据并谨慎操作。
  > - 如果您的实例高于支持的内核版本，请联系相关社区，OC 社区与沐曦官方将尽快完善支持。

| CPU架构 | 操作系统 | 支持内核版本 |
| --- | --- | --- |
| x86_64 | TencentOS 3 | 5.4.241-24.0017.23.tl3 |
| x86_64 | TencentOS 4 | 6.6.92-34.1.tl4 |
| x86_64 | TencentOS 4 | 6.6.98-40.2.tl4 |

- **支持的 GPU 设备**：沐曦曦云 C500/C550/C588/C600/N260 系列
- **MetaX 驱动软件版本**：3.1.0.26
- **其他要求**：
  - PCIe 需要支持 Gen5 X16。
  - MMIO 资源满足 GPU 板卡资源需求。
  - 服务器电源满足整机最大工作负载。
  - 单个 PCIe 槽位满足 GPU 单卡的供电需求。

## 环境检查

本文部署实践方案，主要以二进制形式安装驱动包，请通过以下命令确认系统环境符合要求，匹配软硬件系统。检查 CPU 架构、操作系统版本和内核版本满足基础环境要求。若存在任何一项不匹配，请参见 基础环境要求及说明 升级软硬件系统。

```
# 检查 CPU 架构
uname -m

# 检查操作系统版本
lsb_release -a

# 检查内核版本
uname -r
# 如内核版本不满足需求，请先升级至指定内核并设置成默认启动内核
dnf install kernel-6.6.98-40.2.tl4

# 检查是否已安装旧版驱动
yum list installed | grep metax-driver
# 如已安装旧驱动，请执行
yum remove metax-driver

# 检查 GPU 设备是否识别
lspci | grep 9999
```

# 安装 MetaX 驱动及 MXMACA SDK

## 安装 TencentOS EPOL 源

如使用 TencentOS 4 系统，请先安装 EPOL extras 软件源：

> **说明：**
> TencentOS 3系统需要使用编译方式安装驱动。如您存在相关需求，请 联系我们 。

```
dnf install epol-extras-release
```

## 安装 MetaX 驱动包

安装驱动及依赖包：

```
dnf install metax-driver-3.1.0.26
```

## 固件升级（可选）

如当前版本固件存在已知问题（如安全漏洞、稳定性问题）或硬件设备更新等场景下，请升级固件。MetaX 系列 GPU 采用沐曦带内管理工具 mx-smi 对固件进行升级。mx-smi 工具自动安装在驱动安装包的 `/opt/mxdriver/bin` 目录下。

```
# 查看当前固件版本
mx-smi --show-version

# 升级固件（需root）
sudo mx-smi -u /lib/firmware/metax/mxc500/mxvbios-xxx.bin -t 600
```

升级后需重启系统生效。

> ⚠ **注意：**
> 实例重启会导致正在运行的应用和服务被强制终止，或文件和内存数据会丢失。重启前请做好数据保存等操作。

## 虚拟化安装（可选）

如您需要使用 GPU 的 SR-IOV 硬件虚拟化功能，以实现更高效的虚拟化资源分配与管理，请安装 mxgvm 工具包。

> ⚠ **注意：**
> 如您的业务场景不需要使用 SR-IOV 硬件虚拟化功能，建议您不要安装 mxgvm 工具包。因为安装该工具包可能导致系统无法检测到 GPU 硬件设备，引发一系列诸如实例无法获取 GPU 资源、图形处理任务异常中断等问题，影响系统的稳定性和正常使用。

启用 SR-IOV 功能后，物理 GPU 可凭借其硬件虚拟化特性，虚拟出多个 VF（虚拟功能）设备。这些 VF 设备使用灵活，既能在宿主机上，通过 metax-driver 驱动直接调用，实现高效的数据处理与图形渲染，也可绑定至虚拟机，为虚拟环境提供图形处理支持。
安装 mxgvm 时，系统会自动安装 metax-driver，无需手动安装。当使用虚拟机时，libvirt 会自动将 GPU 设备从 metax-driver 解绑，并重新绑定到 vfio（虚拟功能 I/O）驱动，确保虚拟机正确识别设备。
不过，虚拟机内要使 GPU 正常工作、发挥最佳性能，需单独安装 metax-driver 驱动，以满足复杂虚拟化应用场景需求。

```
dnf install mxgvm-3.0.26
```

### 安装验证

查看驱动安装结果。

```
mx-smi
```

## 安装 MXMACA SDK 包

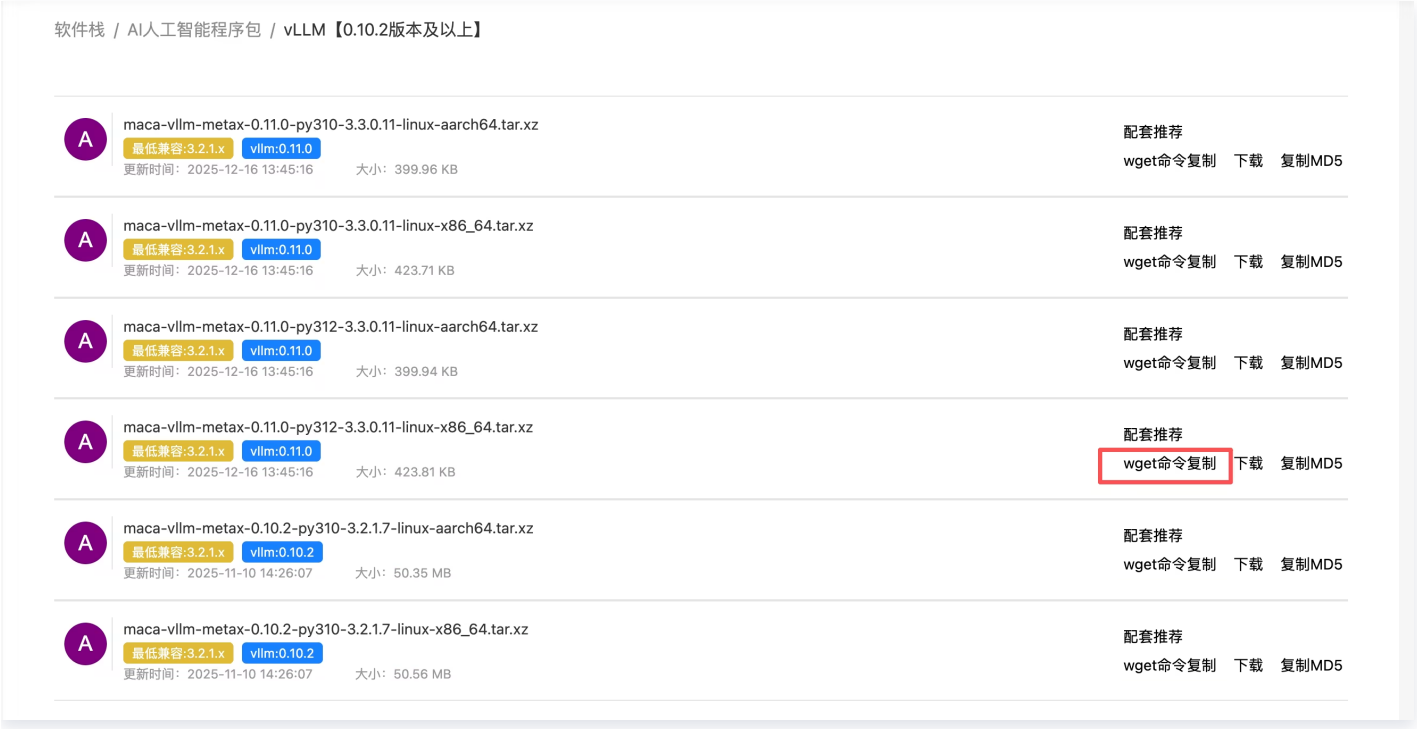
由于相关 RPM 包较多，推荐使用如下命令一键安装。本次适配提供的 MXMACA SDK RPM 包清单，请参见 软件包清单 。

```
dnf install maca_sdk
```

# AI 框架安装与验证

## 拉取 AI 镜像

在沐曦官方开发者社区 > **软件包下载** > **AI 人工智能程序包** 页面选择所需框架，此处以 vLLM 框架 为例，单击 **wget 命令复制**，进行框架的镜像拉取：



> **说明：**
> TencentOS 4 仅支持 Python 3.11 及 3.12 版本，因此在沐曦官网拉取 AI 框架时，请使用 `py311` 或 `py312` 的版本。

## 配置 cu-bridge 环境

请参见以下方式配置 cu-bridge 环境：

```
dnf install -y git cmake
export MACA_PATH=/opt/maca
wget https://gitee.com/metax-maca/cu-bridge/repository/archive/3.1.0.zip
unzip 3.1.0.zip
mv cu-bridge-3.1.0 cu-bridge
sudo chmod 755 cu-bridge -Rf
cd cu-bridge
mkdir build && cd ./build
cmake -DCMAKE_INSTALL_PREFIX=/opt/maca/tools/cu-bridge ../
make && make install

export MACA_PATH=/opt/maca
export CUCC_PATH=/opt/maca/tools/cu-bridge
export PATH=$PATH:${CUCC_PATH}/tools:${CUCC_PATH}/bin
export CUCC_CMAKE_ENTRY=2          # 选择使用 cu-bridge 模拟 CMake 服务
export CUDA_PATH=${CUCC_PATH}      # CUDA_PATH 入口重定向到 cu-bridge 安装位置
```

## 启动容器

在容器中运行 AI 框架及大模型需要使用宿主机 GPU 能力，及直通宿主机 GPU，主要有如下两种方式（可选其一）：

### Docker Run（推荐方式）：

推荐按如下 Docker Run 方式直通宿主机 GPU，执行步骤更便捷，避免基础配置造成的环境差异：

```
docker run -it --restart=always --device=/dev/dri --device=/dev/mxcd --
device=/dev/infiniband --group-add video --name deepspeed_test --
network=host --security-opt seccomp=unconfined --security-opt
apparmor=unconfined --shm-size 100gb --ulimit memlock=-1 --
privileged=true -v /home:/home [image_id] bash
```

## Metax-docker Run：

### 1. 安装 Metax-docker:

访问沐曦开发者社区，选择**云平台工具** > Metax-docker 。在该页面下载离线压缩包，解压完成后请参见以下方式安装。

```
# 安装 metax-docker
mkdir metax-docker
tar -C metax-docker -xvf metax-docker_0.13.1.tar
cd metax-docker
sudo ./metax-docker_0.13.1.<ARCH>.run
```

### 2. 使用 Metax-docker:

您需要安装高于或等于 19.03 版本的 Docker 工具，同时请确保主机上已经正确安装了 MXMACA 软件栈。

```
# 在容器中使用曦云GPU
metax-docker run -it --rm --gpus=all user-application:1.0 /bin/bash
```

Metax-docker 支持官方 Docker 的全部命令及参数，并在 .run 命令下支持的额外参数，详情请参见 Metax-docker 官方指导 。

## 运行大模型（以 vLLM+Qwen 示例）

```
# 安装依赖
dnf install pip curl

# 下载 modelscope
pip install modelscope

# 拉取大模型（以 Qwen 为例）
modelscope download --model 'Qwen/Qwen2-7b'

# 运行服务
vllm serve /root/.cache/modelscope/hub/models/Qwen/Qwen2-7b --port 8000
--served-model-name Qwen2-7b --served-model-name Qwen/Qwen2-7b

# 另起终端利用 curl 对话
```

```
curl http://localhost:8000/v1/chat/completions -H "Content-Type:
application/json"      -d '{"model": "Qwen/Qwen2-7b","messages":
[{"role": "system", "content": "你是一个有帮助的助手"},{"role": "user",
"content": "法国首都在哪？"}],"max_tokens": 100,"temperature": 0.7}'
```

# 附录

## 附录一：软件包清单

| 分类 | 文件名 | 包名 |
| --- | --- | --- |
| 驱动 | metax-driver-3.1.0.26-1.x86_64.rpm | metax-driver |
| | metax-linux-3.1.0.26-1.x86_64.rpm | metax-linux |
| | mxgvm-3.0.26-1.x86_64.rpm | mxgvm |
| | mxfw-3.1.0-1.noarch.rpm | mxfw |
| | mxsmt-3.1.0-1.x86_64.rpm | mxsmt |
| SDK | commonlib_3.1.0-3.1.0.19-1.x86_64.rpm | commonlib |
| | maca_sdk-3.1.0.19-1.x86_64.rpm | maca_sdk |
| | maca_sdk_3.1.0-3.1.0.19-1.x86_64.rpm | maca_sdk |
| | macainfo_3.1.0-3.1.0.19-1.x86_64.rpm | macainfo |
| | mcanalyzer_3.1.0-3.1.0.19-1.x86_64.rpm | mcanalyzer |
| | mcblas_3.1.0-3.1.0.19-1.x86_64.rpm | mcblas |
| | mcblaslt_3.1.0-3.1.0.19-1.x86_64.rpm | mcblaslt |
| | mcccl_3.1.0-3.1.0.19-1.x86_64.rpm | mcccl |
| | mcccltests-3.1.0-3.1.0.19-1.x86_64.rpm | mcccltests |

| | |
|---|---|
| mccompiler_3.1.0-3.1.0.19-1.x86_64.rpm | mccompiler |
| mcdnn_3.1.0-3.1.0.19-1.x86_64.rpm | mcdnn |
| mcfft_3.1.0-3.1.0.19-1.x86_64.rpm | mcfft |
| mcfile_3.1.0-3.1.0.19-1.x86_64.rpm | mcfile |
| mcflashattn_3.1.0-3.1.0.19-1.x86_64.rpm | mcflashattn |
| mcflashinfer_3.1.0-3.1.0.19-1.x86_64.rpm | mcflashinfer |
| mcgpufort_3.1.0-3.1.0.19-1.x86_64.rpm | mcgpufort |
| mchotspot_3.1.0-3.1.0.19-1.x86_64.rpm | mchotspot |
| mcimage_3.1.0-3.1.0.19-1.x86_64.rpm | mcimage |
| mcjpeg_3.1.0-3.1.0.19-1.x86_64.rpm | mcjpeg |
| mckernellib_3.1.0-3.1.0.19-1.x86_64.rpm | mckernellib |
| mcmathlib_3.1.0-3.1.0.19-1.x86_64.rpm | mcmathlib |
| mcpti_3.1.0-3.1.0.19-1.x86_64.rpm | mcpti |
| mcrand_3.1.0-3.1.0.19-1.x86_64.rpm | mcrand |
| mcruntime_3.1.0-3.1.0.19-1.x86_64.rpm | mcruntime |
| mcsolver_3.1.0-3.1.0.19-1.x86_64.rpm | mcsolver |
| mcsolverit_3.1.0-3.1.0.19-1.x86_64.rpm | mcsolverit |
| mcsparse_3.1.0-3.1.0.19-1.x86_64.rpm | mcsparse |

| | |
|---|---|
| mcthrust_3.1.0-3.1.0.19-1.x86_64.rpm | mcthrust |
| mctlass_3.1.0-3.1.0.19-1.x86_64.rpm | mctlass |
| mctoolext_3.1.0-3.1.0.19-1.x86_64.rpm | mctoolext |
| mctracer-3.1.0-3.1.0.19-1.x86_64.rpm | mctracer |
| metax-fabricmanager_3.1.0-3.1.0.19-1.x86_64.rpm | metax-fabricmanager |
| mxccl_plugin_3.1.0-3.1.0.19-1.x86_64.rpm | mxccl_plugin |
| mxcompute_3.1.0-3.1.0.19-1.x86_64.rpm | mxcompute |
| mxdiagease-3.1.0-3.1.0.19-1.x86_64.rpm | mxdiagease |
| mxexporter-3.1.0-3.1.0.19-1.x86_64.rpm | mxexporter |
| mxffmpeg-3.1.0-3.1.0.19-1.x86_64.rpm | mxffmpeg |
| mxffmpeg-dev-3.1.0-3.1.0.19-1.x86_64.rpm | mxffmpeg-dev |
| mxfortran_3.1.0-3.1.0.19-1.x86_64.rpm | mxfortran |
| mxgdrcopy-3.1.0-3.1.0.19-1.x86_64.rpm | mxgdrcopy |
| mxgpu_llvm_3.1.0-3.1.0.19-1.x86_64.rpm | mxgpu_llvm |
| mxkw_3.1.0-3.1.0.19-1.x86_64.rpm | mxkw |
| mxmaca-install-3.1.0-3.1.0.19-1.x86_64.rpm | mxmaca-install |
| mxompi-3.1.0-3.1.0.19-1.x86_64.rpm | mxompi |

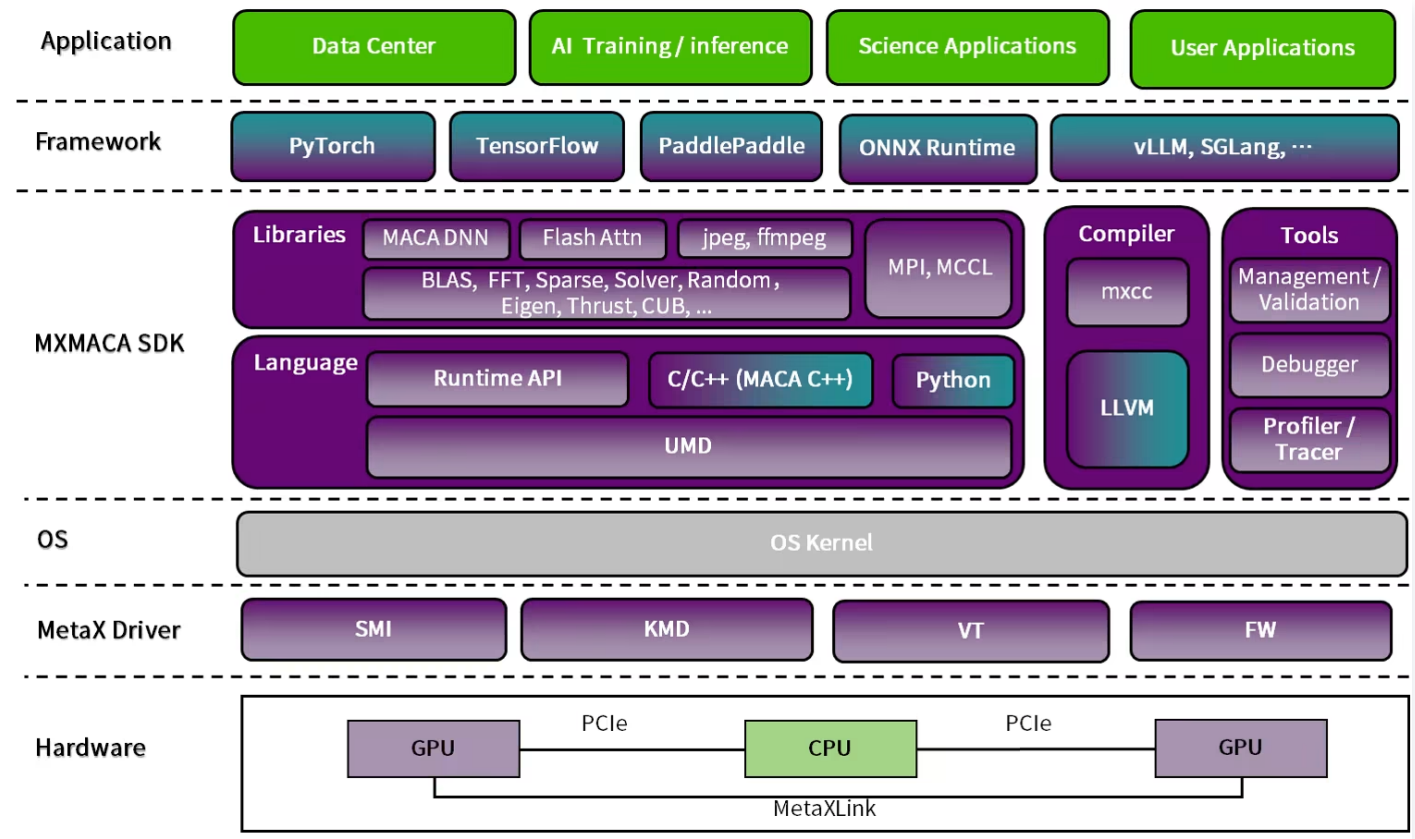
| | | |
|---|---|---|
| | mxreport-3.1.0-3.1.0.19-1.x86_64.rpm | mxreport |
| | mxsm1-devel-3.1.0-3.1.0.19-1.x86_64.rpm | mxsm1-devel |
| | mxucx-3.1.0-3.1.0.19-1.x86_64.rpm | mxucx |
| | mxvpu_3.1.0-3.1.0.19-1.x86_64.rpm | mxvpu |
| | mxvs-3.1.0-3.1.0.19-1.x86_64.rpm | mxvs |
| | sample_3.1.0-3.1.0.19-1.x86_64.rpm | sample |
| | vscode-clangd_3.1.0-3.1.0.19-1.x86_64.rpm | vscode-clangd |

**说明：**

其中：

- metax-driver 是驱动包元信息，安装依赖 metax-linux/mxfw/mxsmt。
- mxgvm 是虚拟化驱动包，安装依赖 metax-linux。

## 附录二：沐曦曦云系列 GPU 应用程序系统架构




## 附录三：沐曦曦云 C500、C550系列硬件适配列表

| 产品 | 适配CPU | 主推拓扑 | 已适配OEM/厂商 | 优势 |
| --- | --- | --- | --- | --- |
| C500 | Intel | common | 浪潮信息、新华三、联想、超聚变、中兴、宁畅等 | • 架构通用：基于经典4U PCIe AI 服务器形态，易于适配、安装、维护，量产机型已覆盖主流 OEM 厂商，在各类整机产品中可适用范围最广。<br>• 拓扑先进：通过 C500 4卡互连拓扑并支持4种 PCIe 服务器经典拓扑（common，balance，cascade，直通），适应各类训练计算场景。<br>• 多元平台：支持 Intel 及海光、飞腾、鲲鹏等国内外主流 CPU 平台。<br>• 成熟稳定：已实现大规模交付并在多个超大规模集群部署并稳定运行。 |
| | 海光4号 | balance | 浪潮计算机、新华三、联想、中兴、中科可控等 | |
| | 飞腾S5000C | balance | 长城等 | |
| | 鲲鹏920 | cascade | 超聚变、华鲲振宇等 | |

| | | | | |
|---|---|---|---|---|
| C550 | Intel | balance | 浪潮信息、新华三、联想、超聚变、中兴等 | • 架构通用：基于经典6U/8U OAM AI 服务器形态，兼容 OAM 1.5/2.0标准，可将 UBB+OAM 作为整体与机头进行适配，量产机型已覆盖主流 OEM 厂商。<br>• 拓扑先进：通过 C550 8卡全互连拓扑实现896GB/s国内领先带宽卡间互连，为各类训练计算场景提供标准服务器单机最强性能。<br>• 多元平台：支持 Intel 及海光、飞腾、鲲鹏等国内外主流 CPU 平台。<br>• 成熟稳定：已实现大规模交付并在多个超大规模集群部署并稳定运行。<br>• 液冷兼容：提供液冷形态模组与液冷 OAM 服务器适配，已实现液冷 OAM 集群大规模部署。 |