曦云®系列通用计算 GPU

用户指南

CSOG-23018-020-F3_V11　　2024-09-06

沐曦专有和三级保密信息　　本文档受 NDA 管控

更新记录

版本	日期	更新说明
V11	2024-09-06	更新以下章节： 3.1.1 系统支持范围 3.1.2 安装包说明 3.2.3.1 安装环境确认 3.2.3.2 二进制文件直接安装 3.2.4 更新固件 3.2.5 卸载驱动 4.1 mx-smi 工具 4.2.3.3 日志级别操作选项 4.3 mx-exporter 工具 6.1.4 虚拟机配置建议 6.1.5.1 驱动安装与反安装 6.1.7.1 驱动安装与反安装 6.2.2 vGPU FLR 7 容器相关场景支持 新增以下章节： 4.2.3.4 日志模块操作选项
V10	2024-08-05	更新以下章节： 3.1.1 系统支持范围 3.1.2 安装包说明 3.2.3.1 安装环境确认 3.2.3.2 二进制文件直接安装 3.2.5 卸载驱动 6.1.5.1 驱动安装与反安装 6.1.7.1 驱动安装与反安装
V09	2024-06-14	更新以下章节： 3.1.1 系统支持范围 3.2.3.2 二进制文件直接安装 3.2.4 更新固件 3.2.5 卸载驱动 4.2.3.2 日志导出选项 6.1.5.1 驱动安装与反安装

版本	日期	更新说明
V08	2024-05-15	更新以下章节： 2.2 产品外观 3.1.1 系统支持范围 3.2.5 卸载驱动 6.1.5.1 驱动安装与反安装 6.1.5.2 驱动加载与卸载 6.1.7.1 驱动安装与反安装 6.1.7.2 驱动加载与卸载
V07	2024-04-10	更新以下章节： 3.1.1 系统支持范围
V06	2024-03-22	新增曦云®系列 GPU 产品信息 更新以下章节： 2.2 产品外观 3.2.3.1 安装环境确认 3.2.3.2 二进制文件直接安装 3.2.4 更新固件 3.2.5 卸载驱动 6.1.5 Flat 模式 6.1.6 PF 透传 6.1.7 VF 透传 7.1.2 在 Docker 容器中使用板卡
V05	2024-02-29	更新以下章节： 4.2.3.5 附加选项 6.1.5.1 驱动安装与反安装 新增以下章节： 6.1.2 支持虚拟化的固件版本
V04	2024-01-31	更新以下章节： 6.1.5.1 驱动安装与反安装 新增以下章节： 6.1.4 虚拟机配置建议 6.1.9.4 ATS 的限制
V03	2024-01-10	更新以下章节： 4.2.3.3 日志级别操作选项
V02	2023-12-29	更新以下章节： 3.1.1 系统支持范围

版本	日期	更新说明
		3.1.2 安装包说明 3.2.3 安装驱动 4.2.3.5 附加选项 新增以下章节： 6 虚拟化支持
V01	2023-10-16	正式版本首次发布

目录

图目录

表目录

1. 概述

本文档详细描述了曦云®系列 GPU 的运行环境配置，所需软件及工具的安装和使用方法，以及日常管理等内容。

本文档主要适用于以下人员：

- 服务器管理员
- 曦云系列 GPU 用户

2. 产品简介

2.1 概述

曦云®系列 GPU 是针对智算和通用计算的产品，沐曦自主知识产权架构提供强大的多精度混合算力，可广泛应用于智算、通用计算、数据分析等场景。

曦云系列 GPU 提供从数据搬移、ETL、训练到推理部署的全面解决方案，混合精度计算加速，大容量存储和新一代高速 IO 接口配以多卡互联技术使 GPU 算力高效释放，涵盖从智算到通用计算端到端数据处理的全部流程。

2.2 产品外观

曦云 C500/C280 的外观如图 2-1 所示。

图 2-1 曦云 C500/C280 外观图

曦云 C500X 的外观图如图 2-2 所示。

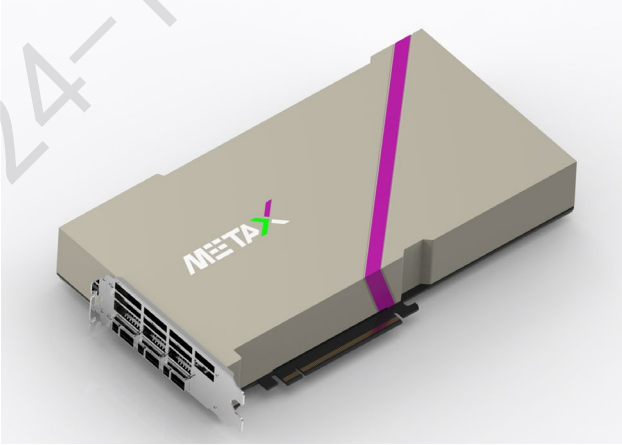

图 2-2 曦云 C500X 外观图

曦云 C290 OAM1.5 的外观如图 2-3 所示。

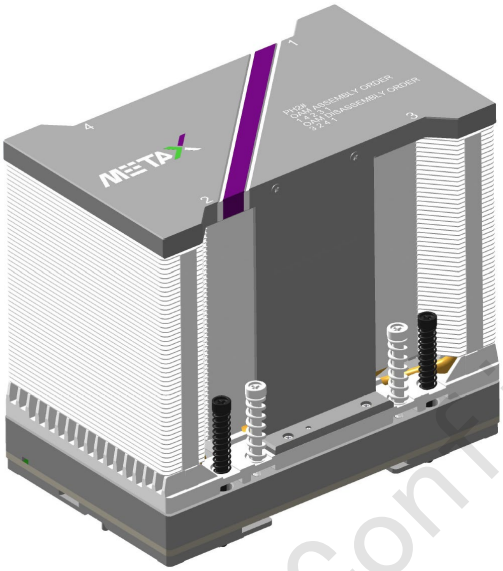

图 2-3 曦云 C290 OAM1.5 外观图

3. 安装与维护

若无特殊说明，本章是以曦云 C500 为例进行撰写。

3.1 用户须知

3.1.1 系统支持范围

目前已支持的 CPU 架构和内核版本，参见表 3-1。

表 3-1 软硬件平台兼容列表

CPU 架构	操作系统	内核版本	状态
x86_64	Ubuntu 18.04	5.4.0-42-generic	支持
		5.4.0-131-generic	
		5.4.0-148-generic	
x86_64	Ubuntu 20.04	5.4.0-26-generic	支持
		5.4.0-42-generic	
		5.15.0-58-generic	
x86_64	Ubuntu 22.04	5.15.0-72-generic	支持
		5.19.0-46-generic	
x86_64	CentOS 8	4.18.0-240.el8.x86_64	支持
x86_64	RedHat 8.3	4.18.0-240.el8.x86_64	支持
x86_64	RedHat 8.6	4.18.0-372.9.1.el8.x86_64	支持
x86_64	RedHat 9	5.14.0-70.22.1.el9_0.x86_64	支持
x86_64	CentOS 7	4.19.0-1.el7.elrepo.x86_64	支持加载 Docker Container 形式
		5.14.0-7.x86_64	
x86_64	BC Linux	4.19.90-2107.6.0.0100.oe1.bclinux	支持
x86_64	BCLinux R8 U2	4.19.0-240.23.11.el8_2.bclinux.x86_64	支持
x86_64	CC Linux	5.15.131-2.cl9.x86_64	支持
x86_64	Kylin V10 SP2	4.19.90-24.4.v2101.ky10.x86_64	支持
x86_64	ALinux3	5.10.134-13.1.al8.x86_64	支持
x86_64	CTYun 23.01	5.10.0-136.12.0.86.ctl3.x86_64	支持

CPU 架构	操作系统	内核版本	状态
x86_64	KeyarchOS 5.8	4.19.91-27.4.19.kos5.x86_64	支持
Arm	Kylin V10 2309a	5.15.0-1.10.6.v2307.ky10h.aarch64	支持
Arm	Kylin V10 SP2	4.19.90-85.0.v2307.ky10.aarch64	支持

3.1.2 安装包说明

曦云系列 GPU 所提供的基础软件包由 Driver 和 MXMACA SDK 两部分组成，其中 Driver 部分通过 run 安装文件发布，MXMACA SDK 部分通过 tar 包发布。以 Ubuntu 系统为例，Driver 和 MXMACA SDK 所包含的内容分别参见表 3-2，表 3-3。

表 3-2 Driver 软件包内容清单

文件名	说明
metax-linux_x.x.x-xxx_amd64.deb	曦云系列 GPU KMD 驱动、工具及相关配置文件
mxgvm_x.x.x-xxx_amd64.deb	曦云系列 GPU Virtualization Manager、工具及相关配置文件
mxfw_x.x.x.x.all.deb	曦云系列 GPU 固件包
mxsmt_x.x.x.x.amd64.deb	mx-smi 系统管理工具，MXSML 系统管理库

表 3-3 MXMACA SDK 软件包内容清单

文件名	说明
commonLib_x.x.x.x_amd64.deb	曦云系列 GPU 通用 lib 库
macainfo_x.x.x.x_amd64.deb	曦云系列 GPU 软件 MXMACA 显示信息
mcanalyzer_x.x.x.x_amd64.deb	曦云系列 GPU 工具库
mcblas_x.x.x.x_amd64.deb	曦云系列 GPU BLAS 库，提供 BLAS API 接口
mccl_x.x.x.x_amd64.deb	曦云系列 GPU 集合通信库，实现对 GPU 的多线程多进程运行控制
mccompiler_x.x.x.x_amd64.deb	曦云系列 GPU 编译器库，提供编译功能
mcthrust_x.x.x.x_amd64.deb	曦云系列 GPU CUB/Thrust 库，提供 CUB/Thrust API 接口
mcdnn_x.x.x.x_amd64.deb	曦云系列 GPU DNN 库，提供 DNN API 接口
mceigen_x.x.x.x.amd64.deb	曦云系列 GPU Eigen 库，提供 Eigen API 接口
mcfft_x.x.x.x_amd64.deb	曦云系列 GPU FFT 库，提供 FFT API 接口
mcfile_x.x.x.x_amd64.deb	曦云系列 GPU Direct Storage 库
mcimage_x.x.x.x.amd64.deb	曦云系列 GPU G2D 库，提供 G2D API 接口
mcjpeg_x.x.x.x.amd64.deb	曦云系列 GPU JPEG 库，提供 JPEG API 接口
mckernellib_x.x.x.x.amd64.deb	曦云系列 GPU 核心库

文件名	说明
mcmathlib_x.x.x.x.amd64.deb	曦云系列 GPU 数学库
mcpti_x.x.x.x_amd64.deb	曦云系列 GPU pti 库，提供 pti API 接口
mcrand_x.x.x.x_amd64.deb	曦云系列 GPU random 库，提供 random API 接口
mcruntime_x.x.x.x.amd64.deb	曦云系列 GPU 运行时库，提供 MXMACA API 接口
mcsolver_x.x.x.x_amd64.deb	曦云系列 GPU SOLVER 库，提供 SOLVER API 接口
mcsolverit_x.x.x.x_amd64.deb	曦云系列 GPU SolverIT 库，提供 SolverIT API 接口
mcsparse_x.x.x.x_amd64.deb	曦云系列 GPU SPARSE 库，提供 SPARSE API 接口
mctlass_x.x.x.x.amd64.deb	曦云系列 GPU Tlass 库，提供 Tlass API 接口
mctoolext_x.x.x.x.amd64.deb	曦云系列 GPU 工具
mctracer_x.x.x.x.amd64.deb	曦云系列 GPU 工具
metax-docker_x.x.x_amd64.deb	曦云系列 GPU docker 工具
mxcompute_x.x.x.x.amd64.deb	曦云系列 GPU UMD 硬件抽象层
mxgpu_llvm_x.x.x.x.amd64.deb	曦云系列 GPU 编译器
mxkw_x.x.x.x.amd64.deb	mxkw 库，为与 KMD 交互的 user-mode API
mxmaca-install_x.x.x.x.amd64.deb	曦云系列 GPU 辅助安装包
mxompi_x.x.x.x_amd64.deb	曦云系列 GPU Open MPI 库，实现 GPU 并行计算
mxreport_x.x.x.x.amd64.deb	mx-report 工具，查询、设置日志级别，收集系统日志
mxucx_x.x.x.x_amd64.deb	曦云系列 GPU UCX 库
sample_x.x.x.x_amd64.deb	曦云系列 GPU sample 库，提供常用库的 sample

3.2 物理机上安装驱动和固件

在安装了 Linux 系统的物理机上，驱动和固件的安装流程如图 3-1 所示。

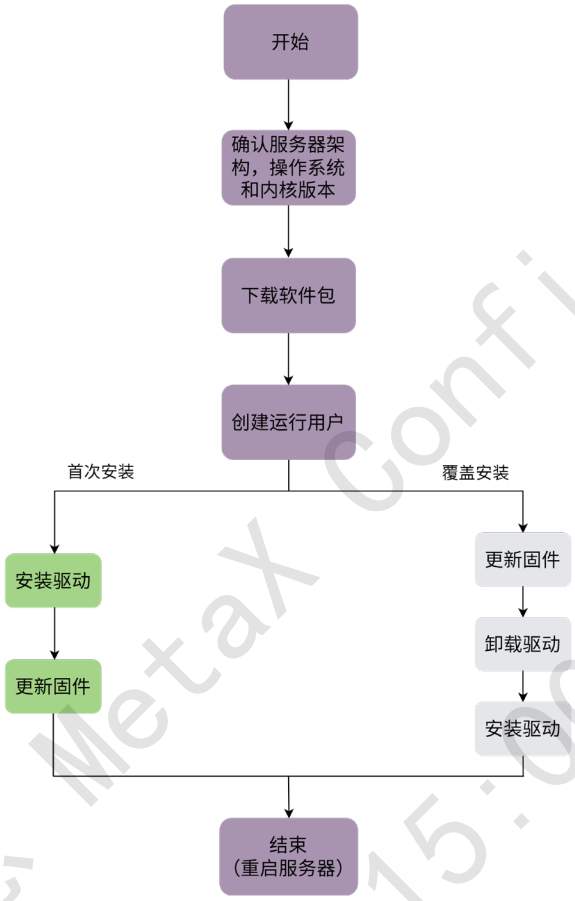

图 3-1 软件包安装流程

说明

- 首次安装场景：服务器上从未安装过驱动，板卡出厂时默认已安装好固件。
- 覆盖安装场景：服务器上安装过驱动且未卸载，当前要再次安装驱动。

3.2.1 确认服务器架构，操作系统和内核版本

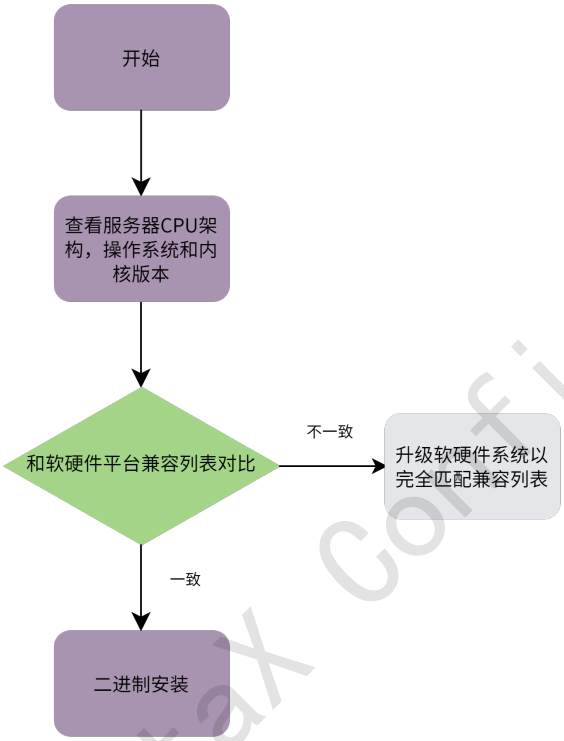

图 3-2 确认服务器架构，操作系统和内核版本

操作步骤

1. 执行以下命令，查询服务器 CPU 架构，操作系统和内核版本。

```
uname -m && uname -r && lsb_release -a
```

2. 对照表 3-1 软硬件平台兼容列表，确认 CPU 架构，操作系统以及内核版本是否在列表中。若有任何一项不匹配，需按照列表升级软硬件系统。

说明

目前驱动安装只支持二进制安装，因此需严格匹配软硬件系统。

3.2.2 创建运行用户

运行用户是软件包安装完成后，使用曦云系列 GPU 的终端用户。安装用户是配置环境，安装软件包的用户，必须有 sudo 权限，一般为服务器管理员。

运行用户可以为 root 用户或者非 root 用户。如果服务器管理员对运行用户有用户权限控制或多用户隔离的需求，可创建非 root 用户作为运行用户。将运行用户加入 video 组即可将其创建为非 root 用户（udev 规则配置文件默认将曦云系列 GPU 使用权限归属于 video 组内）。

操作步骤

例如，创建运行用户 `user` 并将其创建为非 root 用户。

1. 创建运行用户。

```
sudo adduser [user]
```

2. 将运行用户加入 video 组。

```
sudo usermod -a -G video [user]
```

3.2.3 安装驱动

3.2.3.1 安装环境确认

系统兼容性要求

- 若曦云系列 GPU 无法识别为 PCIe 设备（可通过 `lspci | grep 9999` 进行查看），需关闭 BIOS 里兼容性支持模块（CSM）选项。
- 若 PCIe BAR 需要支持 4GB 以上地址空间，需打开 BIOS 里 Large Bar 选项。
- 每张曦云系列 GPU 板卡需要三个 BAR，空间大小分别为 1 MB，8 MB 和 64 GB。

环境检查

表 3-4 环境检查（如无特别说明，以 Ubuntu 18.04 为例）

序号	检查项目	检查命令	说明
1	服务器 CPU 架构	uname -m	对照表 3-1 软硬件平台兼容列表，确认 CPU 架构，操作系统和内核版本是否在列表中。若有任何一项不匹配，则需更新环境。详细信息参见 3.2.1 确认服务器架构，操作系统和内核版本。
2	操作系统	lsb_release -a	
3	内核版本	uname -r	
4	系统是否安装过驱动	lsmod | grep metax	● 若无内容显示，则表示未安装过软件包。可以直接安装软件包。 ● 若有内容显示，则表示安装过软件包。需要先卸载驱动包，再安装新版本软件包。 卸载驱动包，请参见 3.2.5 卸载驱动。
5	板卡是否正常在位 （以曦云 C500 为例）	lspci | grep 9999	如果服务器上有 N（N>0）张曦云 GPU 板卡，回显中含“9999”字段的行数为 N 时，则表示曦云 GPU 板卡正常在位。 例如，若服务器上有 2 张板卡且都正常在位，则回显信息如下所示： `01:00.0 Display controller: Device 9999:4000 (rev 01)`

序号	检查项目	检查命令	说明
			`02:00.0 Display controller: Device 9999:4000 (rev 01)`

6	udev 配置	cat /etc/udev/rules.d/70-metax.rules	● 若有群组管理需要，
			- 只允许 video 组的成员使用曦云 GPU，则配置文件有如下内容：
			`KERNEL=="mxcd", GROUP="video", MODE="0660"`
			`KERNEL=="renderD[0-9]*", GROUP="video", MODE="0660"`
			- 若没有以上内容，则表示只有 root 用户可以使用曦云 GPU。
			● 若无群组管理需要，可删除此配置文件。（deb 包安装会自动创建此配置文件。）

7	虚拟化	dmesg | grep "no space"	支持 SRIOV 功能的曦云系列 GPU 板卡需要分配额外的 PCI BAR 空间给 VF，VF 所需 BAR 空间的大小是 PF 的 8 倍，如果系统无法分配地址空间，对应的设备将无法正常工作。
8	IOMMU 配置	vim /etc/default/grub	● 在 `GRUB_CMDLINE_LINUX=""`里面添加 `iommu.passthrough=1` ● 执行 `sudo update-grub` ● 重启系统，查看/proc/cmdline，确保改动生效
9	gpu_sched 模块	modinfo gpu_sched	● 如果加载 metax 驱动过程中出现 `Unknown symbol drm_sched_xxx` 字样，说明缺少依赖的 gpu_sched 模块 ● 安装内核可选软件包，支持 deb 的系统执行 `apt install linux-modules-extra-$(uname -r)`；支持 rpm 的系统执行 `yum install kernel-modules-extra` ● 重启系统
10	是否允许第三方驱动加载（仅适用于 SLES-15 系统）	modprobe metax	对于 SLES-15 系统，如在加载 metax 驱动时提示 `module metax is unsupported`，则需使用 `modprobe metax --allow-unsupported` 命令加载。 如需开机自动加载 metax 驱动，则需添加 **/etc/modprobe.d/10-unsupported-modules.conf** 文件，并在该文件中填写 `allow_unsupported_modules 1`。

3.2.3.2 二进制文件直接安装

本章节只介绍 Driver 包的安装，MXMACA SDK 的安装参见《曦云®系列通用计算 GPU 快速上手指南》。

操作步骤

1. 将驱动的 run 安装文件下载到目标机器上，进入文件所在目录，执行以下命令安装驱动：

```
sudo ./metax-driver-xxx.run -- -f
```

说明

若 VBIOS 固件和驱动版本不兼容，安装 **metax-linux/mxgvm** 包时会出现如下回显信息：

```
Notice: Please upgrade vbios first, otherwise normal business functions will not be supported
```

此时驱动只提供升级 VBIOS 的功能，不支持正常的业务功能，请根据 3.2.4 更新固件升级 VBIOS。

2. 重启服务器。

```
sudo reboot
```

3. 执行以下命令，查询驱动安装信息。

```
lsmod | grep metax
```

4. 定义环境变量并执行以下命令，若回显信息列出所有曦云 GPU 的信息，则 metax 驱动工作正常。

```
export PATH=$PATH:/opt/mxdriver/bin
export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/opt/mxdriver/lib
mx-smi
```

3.2.4 更新固件

曦云系列 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。关于 mx-smi 工具，参见《曦云®系列通用计算 GPU mx-smi 使用手册》。

操作步骤

1. 确认已经成功加载设备内核驱动。详细内容请参见 3.2.3 安装驱动。
2. 检查更新的 VBIOS 固件文件 **mxvbios-xxx.bin**（例如 **mxvbios-1.1.1.0-17-C500.bin**）已安装到 Linux 的**/lib/firmware/metax/mxc500** 目录下。

说明

若需要使用 SRIOV 功能，应安装带**-VF** 后缀的 VBIOS 固件文件，例如 **mxvbios-1.4.0.0-200-C500-VF.bin**。

3. 确保板卡所有任务已经停止。如果有任务在进行中，需要停止其进程。
4. 使用 mx-smi 工具执行以下命令，升级 VBIOS 固件（需要 Root 权限）。

```
sudo mx-smi -u /lib/firmware/metax/mxc500/mxvbios-xxx.bin -t 600
```

默认对所有板卡进行升级。若屏幕显示以下信息，则表示固件下载成功。

```
vbios-upgrade Done
```

若上述升级VBIOS固件中出现`Bar0Size mismatch`字样，使用以下命令升级（需要Root权限）。

```
sudo mx-smi -U /lib/firmware/metax/mxc500/mxvbios-xxx.bin -t 600 -i ID
```

`ID`是板卡序列号，可以通过`mx-smi -L`查询获取相应板卡的 ID。

5. 重启服务器，以使更新的固件生效。

6. 重启成功并加载驱动后，用 mx-smi 工具执行以下命令查询 VBIOS 固件版本。若与目标版本一致，说明升级安装成功。以曦云 C500 为例，VBIOS 固件版本如图 3-3 所示。

```
mx-smi --show-version
```

```
================== MetaX System Management Interface Log ==================
Timestamp                                         : Wed Jan 24 16:23:21 2024

Attached GPUs                                     : 4
GPU#0  MXC500  0000:4f:00.0
    Version
        MACA                                      : 2.19
        BIOS                                      : 1.1.3.0
        KMD                                       : 2.2.2
        SMP0                                      : 1.1.3.0
        SMP1                                      : 1.1.3.0
        CCX0                                      : 1.1.0.0
        CCX1                                      : 1.1.0.0
        CCX2                                      : 1.1.0.0
```

图 3-3 VBIOS 固件版本

3.2.5 卸载驱动

操作步骤

1. 执行以下命令，卸载驱动。

```
sudo /opt/mxdriver/mxdriver-install.sh -U
```

2. 根据系统提示信息决定是否重启服务器，若需要重启系统，请执行以下命令；否则，请跳过此步骤。

```
reboot
```

4. 工具

4.1 mx-smi 工具

mx-smi 工具的详细介绍，请参见《曦云®系列通用计算 GPU mx-smi 使用手册》。

4.2 mx-report 工具

mx-report 是曦云系列 GPU 的日志管理工具，负责收集 KMD、UMD 和 Firmware 模块的日志，并提供对 KMD、UMD 和 Firmware 模块的日志级别进行查询和设置的功能。

4.2.1 mx-report 工具安装

曦云系列 GPU 软件包安装过程中会默认安装 mx-report 工具。软件包安装完成后，mx-report 工具放置在**/opt/maca/bin/**目录下。

4.2.2 mx-report 工具使用方法

mx-report 是一个 Linux 命令行工具，其调用遵守如下格式。

```
mx-report [选项 1 [参数 1]] [参数 2] …
```

4.2.3 mx-report 命令介绍

mx-report 命令说明，可以通过在工作环境中执行 `man mx-report` 获取。

4.2.3.1 通用选项

```
-h, --help
```

打印使用帮助信息。

```
-v, --version
```

打印 mx-report 的版本信息。

4.2.3.2 日志导出选项

```
./mx-report
```

导出 KMD、UMD、Firmware 的当前日志，存放在当前目录下以当前时间戳命名的文件夹中。

```
./mx-report -c/--continue
```

持续导出 KMD、UMD、Firmware 的日志，存放在当前目录下以当前时间戳命名的文件夹中。

```
--set-fwlog-capacity <log file size, log files number>
```

持续导出日志时，设置 Firmware 单个日志文件的大小和日志文件的数量。若未设置，默认为 20Mb，4 个文件。

```
-p/--pack <target file folder>
```

将导出的日志文件夹打包。

说明

- 非持续导出时 KMD 日志文件是软链接。为保证打包的文件是软链接指向的源文件，推荐使用 `mx-report -p` 进行打包，也可以使用 `tar -cvhf` 或 `zip -r` 进行打包。
- 对于容器场景，一般不建议获取 KMD 日志，需要去对应物理机上获取。若确实需要在容器内获取 KMD 日志，需要在容器启动时将**/var/log** 和**/proc** 挂载到容器内部。

4.2.3.3 日志级别操作选项

```
--show-loglevel
```

显示 KMD、UMD 所有运行中进程（ip: app, mcc, mcr, mxc, mctx, pti, prf, mxkw），Firmware（ip: smp0, smp1, ccx0, ccx1, ccx2）的日志级别。

```
--set-loglevel <moduleName,ipName,loglevel>
```

指定模块名和 ip 设置日志级别，模块名和 ip 可以设置为 `all`，表示全部生效。

4.2.3.4 日志模块操作选项

```
--show-logmodule
```

查询板卡 Firmware（ip: smp0, smp1, ccx0, ccx1, ccx2）日志模块的状态，默认为 `00000001`，表示 ip 第一个模块的日志开关为打开状态。

```
--set-logmodule <ipName,state>
```

设置指定 ip 日志模块状态：`1` 为打开；`0` 为关闭，关闭时不显示该模块日志。若设置 `f`，即为二进制 1111，表示打开 4 个模块日志开关。每个 ip 共 28 个日志模块（0~0fffffff）。本设置仅当 Firmware 该 ip 日志级别为 `5:debug` 时生效。

4.2.3.5 附加选项

```
-i ID, --index ID
```

指定板卡进行日志导出或日志级别查询和设置。如果没有指定，默认对全部板卡生效。ID 是从 0 开始的自然数，可以通过`-L`，`--list` 获取板卡的 ID 信息。

```
--pid pid
```

当设置 UMD 的日志级别时（moduleName 为 umd）需要指定该参数，表示对指定 pid 进行操作，pid 为 all 时对所有进程生效。

当立即导出或持续导出日志时，添加该参数会同时收集与进程 pid 相关的内存使用数据，存放在子目录 umd 下的 memInfo.csv 文件中。

4.2.4 mx-report 工具卸载

曦云系列 GPU 的驱动卸载后，mx-report 工具会自动卸载。卸载驱动请参见 3.2.5 卸载驱动。

在 Root 权限环境中，也可以通过执行以下命令直接删除工作目录下的二进制执行文件来卸载 mx-report 工具。

```
sudo rm /opt/maca/bin/mx-report
```

4.3 mx-exporter 工具

mx-exporter 工具的详细介绍，请参见《曦云®系列通用计算 GPU mx-exporter 使用手册》和《曦云®系列通用计算 GPU mx-exporter Kubernetes 集群监控部署手册》。

5. 维护管理

5.1 带内管理

带内管理通过沐曦 mx-smi 工具提供。mx-smi 工具的使用，参见《曦云®系列通用计算 GPU mx-smi 使用手册》。带内管理的功能有：

提供命令行控制台供系统管理员查询系统软硬件配置和工作。

5.2 带外管理

BMC 提供带外管理功能，包括查询曦云系列 GPU 的配置信息，生产信息，时钟信息，电源功率信息，温度信息，固件版本。通过 MCU，BMC 可以触发曦云 GPU 的重启。

曦云系列 GPU 的带外管理功能，参见《曦云®系列通用计算 GPU 带外管理手册》。

6. 虚拟化支持

曦云系列 GPU 支持基于 SRIOV 的硬件虚拟化，可以将物理 GPU 虚拟为多个 vGPU 使用。开启虚拟化时可以指定 vGPU 数量，以及对哪些 GPU 设备开启虚拟化。

多个 vGPU 的资源是均等的。vGPU 可以在 host 上使用，这种使用方式称为 flat 模式；vGPU 也可以透传到虚拟机中使用，这种使用方式称为透传模式。目前支持的 hypervisor 为 QEMU/KVM。

在虚拟化下使用 GPU 可能需要用到以下两个驱动：

- MetaX GPU Virtualization Manager（mxgvm），即运行于 host 主机的 PF 驱动，负责管理监控 vGPU 的运行（在 PF 透传模式下不需要）。
- MetaX GPU Driver（metax），既是 GPU 的驱动也是 vGPU 的驱动，根据场景可以运行于 host 或虚拟机上。

6.1 配置环境

6.1.1 BIOS 配置说明

开启虚拟化首先需要确认 BIOS 的配置是否满足，常见影响虚拟化的 BIOS 配置如下:

- CPU 的虚拟化配置（以 x86 为例）
  - Intel CPU VT-x Support，选择 `Enabled`
  - AMD CPU AMD-V（或 SVM）Support，选择 `Enabled`
- MMIO 空间相关配置

  MMIO Size，如果有该选项，建议使用最高配置
- PCI 相关配置
  - SRIOV Support，选择 `Enabled`
  - ARI Support，选择 `Enabled`
  - ACS Support，选择 `Enabled`
  - IOMMU（或 SMMU）Support，选择 `Enabled`

说明

由于不同服务器厂商使用的 BIOS 版本不同，有些参数可能不支持或是隐藏的配置，如对 BIOS 的虚拟化支持有疑问，请咨询相关服务器厂商。

6.1.2 支持虚拟化的固件版本

使用 GPU 的 SRIOV 功能需要安装支持虚拟化的固件版本，详细信息参见 3.2.4 更新固件。

6.1.3 Linux 内核虚拟化参数配置

使用透传模式时，Linux 内核参数需要增加 IOMMU 的相关配置。

操作步骤

例如，运行在 Intel CPU 上的 Ubuntu 系统中，使用 root 用户修改/etc/default/grub。

1. 将 `GRUB_CMDLINE_LINUX_DEFAULT=""`修改为 `GRUB_CMDLINE_LINUX_DEFAULT="iommu=pt intel_iommu=on"`。
2. 执行以下命令。

   ```
   sudo update-grub
   ```

3. 重启系统，登入系统查看/proc/cmdline，确保改动生效。

   ```
   cat /proc/cmdline
   ```

x86 平台上 Linux 内核 IOMMU 配置参数，参见表 6-1。Arm 平台上 Linux 内核 IOMMU 配置参数，参见表 6-2。

表 6-1 x86 平台上 Linux 内核 IOMMU 配置参数

参数格式	说明
intel_iommu=on	启用 Intel IOMMU
amd_iommu=on	启用 AMD IOMMU
iommu=pt 或 iommu.passthrough=1	仅在 PCI 设备透传时使用 IOMMU

表 6-2 Arm 平台上 Linux 内核 IOMMU 配置参数

参数格式	说明
iommu.passthrough=1	仅在 PCI 设备透传时使用 SMMU

6.1.4 虚拟机配置建议

关于 QEMU/KVM 的配置建议如下：

- 对于 x86 平台，建议使用 Q35 虚拟硬件平台，Q35 支持 PCIe 相关的特性。
- 配置虚拟机时，需要根据服务器的 NUMA 拓扑合理分配 CPU 与内存资源，否则可能会影响性能。

- 对于虚拟机内支持 PCIe P2P，由于虚拟机依赖 IOMMU，默认需要开启 ACS，开启 ACS 会影响 PCIe 设备的 P2P I/O 路径，引入性能问题。

  建议对支持 ATS 的 PCIe 设备透传后，打开其 P2P 路径上 PCIe bridge 的 `ACS Direct Translated` bit，以降低 ACS 引入的性能损失。

- 使用 OpenStack 的虚拟机管理软件配置 GPU 透传时，根据当前的固件版本选择设备类型：
  - 不支持 SRIOV 的固件，物理 GPU 透传对应的设备类型为 type-PCI
  - 支持 SRIOV 的固件，物理 GPU 透传对应的设备类型为 type-PF
  - 支持 SRIOV 的固件，vGPU 透传对应的设备类型为 type-VF

6.1.5 Flat 模式

6.1.5.1 驱动安装与反安装

操作步骤

1. 将驱动的 run 安装文件下载到目标机器上，进入文件所在目录，执行以下命令安装驱动：

```
sudo ./metax-driver-xxx.run  -- -f -m vt_flat
```

2. 如果安装过程中检测到 VBIOS 版本过低，此时驱动只提供升级 VBIOS 的功能，不支持正常的业务功能，请根据 6.1.5.3 更新固件升级 VBIOS。

3. 重启服务器。

```
sudo reboot
```

4. 执行以下命令，查询 mxgvm 驱动是否已加载。

```
lsmod | grep mxgvm
```

5. 执行以下命令，若回显信息列出所有 GPU 和 vGPU 的信息，则 mxgvm 驱动工作正常。

```
maca-vt
```

说明

安装 mxgvm 包后，会在**/etc/modprobe.d** 路径下生成 **mxgvm.conf** 配置文件，内容如下：

```
options metax vf_only
```

目的是让 metax 驱动只识别 VF 设备，不会自动和 PF 设备绑定。因此若不使用 VF 功能，需要反安装 mxgvm 包，否则 PF 功能无法正常工作。

6. （可选）若要反安装 mxgvm 包，执行以下命令，然后重启系统：

```
sudo /opt/mxdriver/mxdriver-install.sh -U
```

6.1.5.2 驱动加载与卸载

安装驱动包后，在系统启动时会自动加载 mxgvm 和 metax。加载 mxgvm 时会使能 SRIOV 虚拟化，卸载 mxgvm 时会关闭 SRIOV 虚拟化。

若因配置需求，需要手动卸载 mxgvm，请按下列步骤操作。

操作步骤

1. 卸载 vGPU 驱动。

   ```
   sudo modprobe -r metax 或 sudo rmmod metax
   ```

2. 确保 vGPU 没有与任何驱动绑定后，执行以下命令查看 mxgvm 驱动的“Used by”计数。如果计数为 0，说明 mxgvm 没有被 vGPU 使用，可以卸载。

   ```
   lsmod | grep mxgvm
   ```

3. 卸载 mxgvm 驱动。

   ```
   sudo modprobe -r mxgvm 或 sudo rmmod mxgvm
   ```

   说明

   卸载后，GPU 的虚拟化功能关闭，vGPU 设备不再可见。

手动加载 mxgvm 时可以通过 `vf_num` 参数来指定 vGPU 个数。例如，开启 4vGPU：

```
sudo modprobe mxgvm vf_num=4
```

说明

此时 metax 驱动会自动加载，无需再手动加载。

6.1.5.3 更新固件

在 Flat 模式下更新 VBIOS 固件的操作与 PF 模式一致，操作步骤参见 3.2.4 更新固件。

说明

请务必安装带**-VF** 后缀的 VBIOS 固件文件。

更新 VBIOS 并重启 host 服务器后，可使用以下命令检查当前 VBIOS 是否支持 SRIOV：

```
sudo mx-smi --show-vbios | grep SRIOV
```

若显示为 **Support**，则当前 VBIOS 支持虚拟化。

6.1.6 PF 透传

PF 透传不需要开启 SRIOV，因此不需要安装 mxgvm 驱动。

只需要配置好虚拟机将 PF 设备透传进虚拟机后，启动虚拟机；然后在虚拟机里安装 metax 驱动，参见 3.2.3 安装驱动；安装后，虚拟机启动时会自动加载 metax 驱动。

6.1.6.1 更新固件

在 PF 透传模式下，更新固件是在虚拟机中进行，操作步骤参见 3.2.4 更新固件。

说明

- 请使用常规的 VBIOS 固件文件，不要使用带**-VF** 后缀的 VBIOS 固件文件。
- 更新 VBIOS 后，需要重启 host 服务器才能使新 VBIOS 生效，只重启虚拟机无法使其生效。

6.1.7 VF 透传

6.1.7.1 驱动安装与反安装

操作步骤

1. 将驱动的 run 安装文件下载到目标机器上，进入文件所在目录，执行以下命令安装驱：

```
sudo ./metax-driver-xxx.run  -- -f -m vt_pt
```

2. 如果安装过程中检测到 VBIOS 版本过低，此时驱动只提供升级 VBIOS 的功能，不支持正常的业务功能，请根据 6.1.7.3 更新固件升级 VBIOS。
3. 重启服务器。

```
sudo reboot
```

4. 执行以下命令，查询 mxgvm 驱动是否已加载。

```
lsmod | grep mxgvm
```

5. 执行以下命令，若回显信息列出所有 GPU 和 vGPU 的信息，则 mxgvm 驱动工作正常。

```
maca-vt
```

6. 配置虚拟机将 vGPU 透传进虚拟机，启动虚拟机。
7. 在虚拟机里安装 metax 驱动，参见 3.2.3 安装驱动。
8. （可选）若要反安装 mxgvm 包，执行以下命令，然后重启系统：

```
sudo /opt/mxdriver/mxdriver-install.sh -U
```

6.1.7.2 驱动加载与卸载

安装驱动包后，在 host 系统启动时会自动加载 mxgvm，在虚拟机启动时会自动加载 metax。加载 mxgvm 时会使能 SRIOV 虚拟化，卸载 mxgvm 时会关闭 SRIOV 虚拟化。

若因配置需求，需要手动卸载 mxgvm，请按下列步骤操作。

操作步骤

1. 在虚拟机中，执行以下命令卸载 vGPU 驱动。

   ```
   sudo modprobe -r metax 或 sudo rmmod metax
   ```

2. 关闭虚拟机，确保 vGPU 没有与任何驱动绑定后，执行以下命令查看 mxgvm 驱动的“Used by”计数。如果计数为 0，说明 mxgvm 没有被 vGPU 使用，可以卸载。

   ```
   lsmod | grep mxgvm
   ```

3. 卸载 mxgvm 驱动。

   ```
   sudo modprobe -r mxgvm 或 sudo rmmod mxgvm
   ```

   说明

   卸载后，GPU 的虚拟化功能关闭，vGPU 设备不再可见。

手动加载 mxgvm 时可以通过 `vf_num` 参数来指定 vGPU 个数。例如，开启 4vGPU：

```
sudo modprobe mxgvm vf_num=4
```

说明

此时 vGPU 数量变化，可重新配置虚拟机，若虚拟机里已安装 metax 驱动，虚拟机启动时会自动加载 metax。

6.1.7.3 更新固件

在 VF 透传模式下，无法在虚拟机中通过 vGPU 驱动更新 VBIOS 固件，需要在 host 上操作，操作步骤参见 3.2.4 更新固件。

说明

请务必安装带**-VF** 后缀的 VBIOS 固件文件。

更新 VBIOS 并重启 host 服务器后，可使用以下命令检查当前 VBIOS 是否支持 SRIOV：

```
sudo mx-smi --show-vbios | grep SRIOV
```

若显示为 **Support**，则当前 VBIOS 支持虚拟化。

6.1.8 mxgvm 的配置文件和主要参数

除加载 mxgvm 时可以指定参数之外，也可以编辑/etc/mxgvm_config 来配置参数，常用参数参见表 6-3。

表 6-3 mxgvm 的配置参数

参数格式	说明
vf_num=n	n 为 vGPU 的数量，有效值为 1/2/4/8，默认值为 4 开启多 vGPU 时，建议使用 4vGPU，兼顾多实例的同时，相比 8vGPU 可获得更高的利用率
pci-list=<bdf1>,<bdf2>	指定设备列表，配合 list-type 指定目标设备，BDF 格式<domain>:<bus>:<slot>.<func>，例如 0000:01:00.1
list-type=<val>	val=0 时，pci-list 中的设备不开启 SRIOV； val=1 时，pci-list 中的设备开启 SRIOV，默认值为 0

/etc/mxgvm_config 上会自动记录上一次 mxgvm 加载时的配置参数。再次加载时，若不指定任何参数，则默认使用/etc/mxmvm_config 中的配置。

更改 vGPU 的数量无需重启，卸载 mxgvm（参见 6.1.5~6.1.7），再使用新的 `vf_num` 参数重新加载 mxgvm 即可。

卸载 mxgvm 包时，会提示是否需要保留/etc/mxgvm_config 文件，用户根据自己的需求进行选择即可。

6.1.9 限制

6.1.9.1 vGPU 多进程的限制

因受限于物理资源，vGPU 下运行多进程业务时的限制，参见表 6-4。

表 6-4 vGPU 多进程的限制

vGPU 数量	每 vGPU 最大并发进程数
1	11
2	8
4	4
8	2

6.1.9.2 Linux DRM 对显卡数量的限制

Linux 内核 DRM 子系统最大支持 64 个设备。在 8 卡环境下，每张卡开启 8 个 vGPU 时，共有 64 个 vGPU，此时如果服务器自带显卡，有可能会占用一个 DRM 设备，导致最后一个 vGPU 无法正常工作。

6.1.9.3 互联模式的限制

一旦开启 SRIOV，物理 GPU 与 vGPU 将不支持 MetaXLink 互联。

6.1.9.4 ATS 的限制

不支持 ATS。

6.2 mx-smi 的虚拟化支持

6.2.1 显示 vGPU

显示当前系统所有 GPU 和 vGPU 设备。

```
sudo mx-smi -L
```

显示结果中，每个 vGPU 都有特定的 ID，可用于在其它操作中指定目标 vGPU。

6.2.2 vGPU FLR

触发第二个 vGPU 的 FLR。

```
sudo mx-smi -i 1 --vfflr
```

7. 容器相关场景支持

曦云系列 GPU 提供对 Docker 和 Kubernetes 的支持。Kubernetes 的详细介绍，参见《曦云®系列通用计算 GPU 云原生参考手册》。

7.1 官方 Docker 支持

开发人员可以借助于容器引擎，运行预先构建好的容器镜像，快速建立软件开发或运行所需的环境。曦云系列 GPU 提供预构建的 MXMACA 容器镜像。开发人员仅需执行一条 `docker run` 命令就可获得一个干净而完整的板卡开发环境。

在运行 MXMACA 容器镜像前，请确认环境满足以下条件：

- 已正确安装相应曦云 GPU 的内核驱动。详细内容请参见 3 安装与维护。
- 已安装 Docker，Docker 版本≥19.03。

7.1.1 获取 MXMACA 容器镜像

MXMACA 容器镜像是以离线形式发布。用户可在随本文档发布的软件包中找到相关压缩包。本文档中以 **maca-c500-container-2.0.0.tar.gz** 为例，用户应根据实际收到的软件包版本对版本字段进行相应替换。

操作步骤

1. 执行以下命令，完成容器镜像的加载。

```
docker load < ./maca-c500-container-2.0.0.tar.gz
```

7.1.2 在 Docker 容器中使用板卡

7.1.2.1 使用全部曦云 GPU

操作步骤

1. 执行以下命令：

```
docker run -it --device=/dev/mxcd --device=/dev/dri --group-add video \
cr.metax-tech.com/library/maca-c500:2.0.0 /bin/bash
```

7.1.2.2 使用指定曦云 GPU

正确安装内核驱动后，默认在/dev/dri 目录下为每一个曦云 GPU 创建了一个 card 设备文件和一个 renderD 设备文件。通过绑定设备文件来指定期望使用的板卡。如何确认板卡的 ID 与/dev/dri 目录下设备文件之间的对应关系，请参见 7.1.3 GPU 设备文件查询。

操作步骤

例如，指定的板卡所对应的设备文件为/dev/dri/card1 和/dev/dri/renderD129。

1. 执行以下命令：

```
docker run -it --device=/dev/mxcd --device=/dev/dri/card1 \
--device=/dev/dri/renderD129 --group-add video \
cr.metax-tech.com/library/maca-c500:2.0.0 /bin/bash
```

7.1.3 GPU 设备文件查询

操作步骤

1. 执行以下命令查看板卡的设备 ID 和 PCI 总线地址的对应关系。

```
mx-smi -L
```

例如，GPU#0 和 GPU#1 的设备 ID 和 PCI 总线地址的对应关系如下所示。

```
GPU#0 0000:01:00.0
GPU#1 0000:02:00.0
```

2. 在/dev/dri/by-path 路径下，使用 PCI 总线地址筛选得到板卡的设备文件路径。例如，执行以下命令得到 GPU#1 的 dri 设备文件路径。

```
ls -l /dev/dri/by-path | grep 0000:01:00.0
lrwxrwxrwx 1 root root  8 Oct 24 19:03 pci-0000:01:00.0-card -> ../card1
lrwxrwxrwx 1 root root 13 Oct 24 19:03 pci-0000:01:00.0-render
-> ../renderD129
```

8. 附录

8.1 术语/缩略语

术语/缩略语	全称	描述
ACS	Access Control Services	访问控制服务
ARI	Alternative Routing-ID Interpretation	可替换的 Routing ID
BAR	Base Address Register	基地址寄存器
BIOS	Basic Input/Output System	基本输入输出系统
BMC	Baseboard Management Controller	基板管理控制器
CSM	Compatibility Support Module	兼容性支持模块
Docker		一个开源的应用容器引擎
ETL	Extract-Transform-Load	将大量的原始数据经过提取、转换、加载到目标存储数据仓库
IOMMU	Input-Output Memory Management Unit	输入输出内存管理单元
KMD	Kernel-Mode Driver	内核模式驱动程序
KVM	Kernel Virtual Machine	基于内核的虚拟机，是一种内建于 Linux 的开源虚拟化技术
MCU	Microcontroller Unit	微控制单元
MetaXLink		沐曦 GPU D2D 接口总线
MMIO	Memory Mapped I/O	内存映射 I/O，是 PCI 规范的一部分
MXMACA	MetaX Advanced Compute Architecture	沐曦推出的 GPU 软件栈，包含了沐曦 GPU 的底层驱动、编译器、数学库及整套软件工具套件
NUMA	Non-Uniform Memory Access	非一致性内存访问
PCI	Peripheral Component Interconnect	一种连接主板和外部设备的总线标准
PCIe	PCI Express	一种高速串行计算机扩展总线标准
P2P	Peer-to-Peer	PCIe P2P 是 PCIe 的一种特性，使两个 PCIe 设备之间可以直接传输数据
PF	Physical Function	物理功能
QEMU		一个开源的模拟器和虚拟机
SMMU	System Memory Management Unit	系统内存管理单元

术语/缩略语	全称	描述
SRIOV	Single Root I/O Virtualization	将单个物理 PCIe 设备虚拟化为多个 PCIe 设备的技术
UMD	User Mode Driver	用户模式驱动程序
VBIOS	Video BIOS	图形卡的基本输入输出系统
VF	Virtual Function	虚拟功能

声明