曦云®系列通用计算 GPU

快速上手指南

CSOG-23017-020-F3_V07　　2024-09-18

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
V07	2024-09-18	更新以下章节： 2.1.1 安装准备
V06	2024-09-06	更新以下章节： 2.1.2 安装 MXMACA 编程环境 2.1.6 卸载 MXMACA 编程环境
V05	2024-08-05	更新以下章节： 2.1.2 安装 MXMACA 编程环境 2.1.6 卸载 MXMACA 编程环境 4.3 MXMACA 人工智能和计算库
V04	2024-06-14	更新以下章节： 2.1 安装 MXMACA 编程环境
V03	2024-05-15	更新以下章节： 2.1.2 安装 MXMACA 编程环境 2.1.4 启用 MXMACA 程序的访问权限 2.1.6 卸载 MXMACA 编程环境
V02	2024-03-22	新增曦云®系列 GPU 产品信息 更新以下章节： 2.1.2 安装 MXMACA 编程环境 2.1.6 卸载 MXMACA 编程环境 2.2.2.1 用参考示例项目的 Makefile 构建 MXMACA 程序
V01	2023-10-16	正式版本首次发布

目录

图目录

表目录

1. 简介

本文档描述了如何快速开始在曦云®系列 GPU 上进行 GPU 编程，包括编程环境的安装验证、MXMACA®的编程模型和 MXMACA 的编程接口等内容介绍。

本文档主要适用于以下人员：

- 在曦云系列 GPU 上进行程序开发工作的客户和用户
- 使用 MXMACA 进行 GPU 编程的程序员

2. 编程环境

2.1 安装 MXMACA 编程环境

本章以 Ubuntu 18.04 系统环境为例进行安装示范。支持的软硬件平台兼容列表，参见《曦云®系列通用计算 GPU 用户指南》中“系统支持范围”章节。

2.1.1 安装准备

操作步骤

1. 执行以下命令，检查是否拥有支持 MXMACA 的 GPU。

```
$ lspci | grep 9999
```

   - 如果机器上有 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，若机器上有 2 张曦云 GPU 且都正常在位，回显信息如下所示。

```
01:00.0 Display controller: Device 9999:4000 (rev 01)
02:00.0 Display controller: Device 9999:4000 (rev 01)
```

   - 如果机器上有曦云系列 GPU 板卡但无内容显示，执行以下命令，检查是否已成功安装对应的 GPU 板卡驱动。

```
lsmod | grep metax
```

     若无内容显示，则表示未成功安装对应的 GPU 板卡驱动。驱动安装参见《曦云®系列通用计算 GPU 用户指南》中“安装驱动”章节。

     说明

     驱动安装成功后，需要重启系统。

2. 验证是否拥有支持 MXMACA 编程环境的 Linux 发行版。

   执行以下命令，确认正在运行的 Linux 发行版及其版本号。

```
$ uname -m && cat /etc/*release
```

   MXMACA 编程环境仅在某些特定的 Linux 发行版上受支持。详情参见《曦云®系列通用计算 GPU 用户指南》中“系统支持范围”章节。

   例如，机器是 64 位系统（x86_64），并且安装了 Ubuntu 18.04 的操作系统，回显信息如图 2-1 所示。

```
x86_64
DISTRIB_ID=Ubuntu
DISTRIB_RELEASE=18.04
DISTRIB_CODENAME=bionic
DISTRIB_DESCRIPTION="Ubuntu 18.04.6 LTS"
NAME="Ubuntu"
VERSION="18.04.6 LTS (Bionic Beaver)"
ID=ubuntu
ID_LIKE=debian
PRETTY_NAME="Ubuntu 18.04.6 LTS"
VERSION_ID="18.04"
HOME_URL="https://www.ubuntu.com/"
SUPPORT_URL="https://help.ubuntu.com/"
BUG_REPORT_URL="https://bugs.launchpad.net/ubuntu/"
PRIVACY_POLICY_URL="https://www.ubuntu.com/legal/terms-and-policies/privacy-policy"
VERSION_CODENAME=bionic
UBUNTU_CODENAME=bionic
```

图 2-1 编程环境验证回显示例

3. 验证系统是否已安装正确版本的 GCC、g++、libstdc++、CMake 和 Make。

   以验证系统上安装的 GCC 版本为例，执行以下命令：

   ```
   $ gcc --version
   ```

   使用 MXMACA 工具包进行开发必须安装 GCC、g++、libstdc++、CMake 和 Make。运行 MXMACA 应用程序不需要。它们通常是作为 Linux 安装的一部分进行安装的，在大多数情况下，与受支持的 Linux 一起安装的 GCC、g++、libstdc++、CMake 和 Make，都会正常工作。

   例如，Ubuntu 18.04 操作系统上安装了 GCC 7.5.0 版本，回显信息如图 2-2 所示。

```
gcc (Ubuntu 7.5.0-3ubuntu1~18.04) 7.5.0
Copyright (C) 2017 Free Software Foundation, Inc.
This is free software; see the source for copying conditions.  There is NO
warranty; not even for MERCHANTABILITY or FITNESS FOR A PARTICULAR PURPOSE.
```

图 2-2 系统验证回显示例

2.1.2 安装 MXMACA 编程环境

说明

如果已安装了旧版本的 MXMACA，请参见 2.1.6 卸载 MXMACA 编程环境的相应方式进行卸载后，再安装新版本的 MXMACA。

操作步骤

1. 安装 MXMACA 依赖。

   MXMACA 的依赖列表，参见表 2-1。

   请确认表格中的依赖库是否都已安装；若没有，请自行获取并安装。

表 2-1 MXMACA 依赖列表

序号	依赖名称	依赖安装命令
1	libmsgpackc2	sudo apt install libmsgpackc2
2	libelf1	sudo apt install libelf1

序号	依赖名称	依赖安装命令
3	libnuma1	`sudo apt install libnuma1`
4	libssl1.1	`sudo apt install libssl1.1`

a. 以 libmsgpackc2 为例，执行以下命令：

```
$ dpkg -l 'libmsgpackc2'
```

b. 如果系统返回 `dpkg-query: no packages found matching libmsgpackc2`，使用 apt 安装 **libmsgpackc2**。

```
$ sudo apt install libmsgpackc2
```

c. 使用以下命令再次进行查询。结果如图 2-3 所示，表示 **libmsgpackc2** 安装成功。

```
$ dpkg -l 'libmsgpackc2'
```

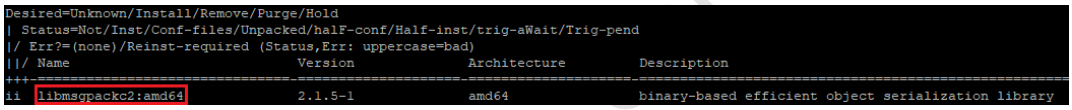

图 2-3 libmsgpackc2 安装成功示例

2. 解压获取安装包。

a. 获取 MXMACA SDK 的 tar 安装包，解压后可以看到 **mxmaca-sdk-install.sh**、**version_compatible_check.sh**、**appimage**、**deb** 等子目录。将安装包放入目标服务器，包内基础文件如图 2-4 所示。

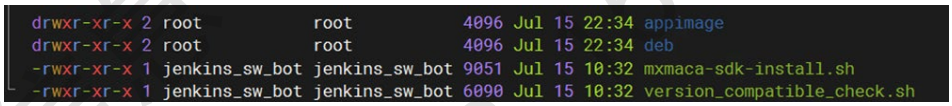

图 2-4 MXMACA SDK 的 deb 安装包内基础文件示例

b. 获取 metax 驱动安装包 **metax-driver-mxc500-x.x.x.x-deb-x86_64.run**。将安装包放入目标服务器。

3. 执行以下命令，安装 MXMACA 编程环境。

```
$ cd xxx ##(MXMACA SDK 解压目录)
$ sudo ./mxmaca-sdk-install.sh -f
$ cd xxx ##(MXMACA 驱动所在目录)
$ sudo chmod +x metax-driver-mxc500-x.x.x.x-deb-x86_64.run
$ sudo ./metax-driver-mxc500-x.x.x.x-deb-x86_64.run -- -f
```

安装成功后的目录如图 2-5 所示。

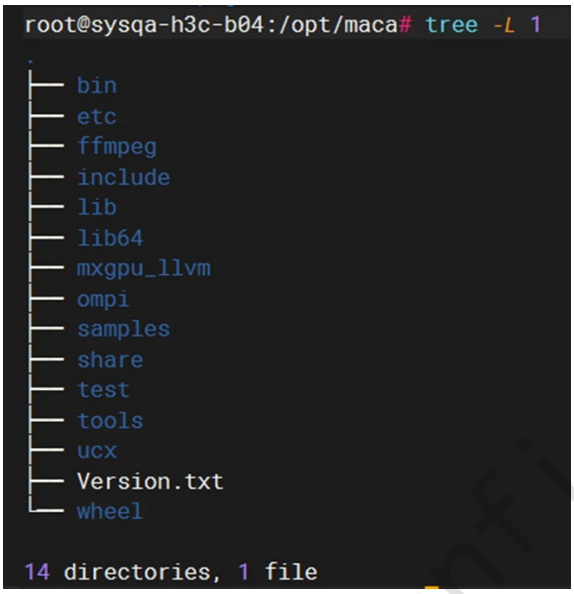

图 2-5 MXMACA 安装成功后的目录示例

2.1.3 配置 MXMACA 编程环境

MXMACA 目前仅支持在默认路径 **/opt/maca** 下进行安装。

操作步骤

1. 将以下 MXMACA 路径加入 `PATH` 环境变量中。

```
$ export PATH=$PATH:/opt/maca/mxgpu_llvm/bin:/opt/maca/bin
```

2. 定义以下 MXMACA 环境变量。

```
$ export MACA_PATH=/opt/maca
$ export LD_LIBRARY_PATH=/opt/maca/lib:/opt/maca/mxgpu_llvm/lib
```

2.1.4 启用 MXMACA 程序的访问权限

MXMACA 程序需要特定文件夹的访问权限，但 MXMACA 程序进程不会修改文件夹中任何文件内容。

操作步骤

1. 执行以下命令，启用访问权限。

```
$ sudo usermod -aG video $USER
```

将用户加入 video 用户组，为其启用访问权限。

2.1.5 验证 MXMACA 编程环境的安装

操作步骤

1. 执行以下命令，检查是否正确安装 MXMACA 编程环境。

```
$ macainfo
```

如图 2-6 所示，如果 macainfo 在有曦云系列 GPU 运行的系统中正常运行，说明 MXMACA 编程环境安装成功。

```
sw@LG-PC-10-2-120-162:/opt/maca$ macainfo
=======================
MXC System Attributes
=======================
Runtime Version:         1.0
System Timestamp Freq:   1000MHz
Signal Max Wait Time:    18446744073709551615(0xffffffffffffffff)
Machine Model:           LARGE
System Endianess:        LITTLE

***********
Agent 1
***********
  Name:                                  AMD Ryzen 7 5800X 8-Core Processor
  Uuid:                                  4350552d-5858-0000-0000-000000000000
  Market Name:                           AMD Ryzen 7 5800X 8-Core Processor
  Vendor Name:                           CPU
  Feature:                               Not Specified
  Profile:                               FULL_PROFILE
  Float Round Mode:                      NEAR
  Max Queue Number:                      0(0x0)
  Queue Min Size:                        0
  Queue Max Size:                        0
  Queue Type:                            MULTI



                                         ......

***********
Agent 2
***********
  Name:                                  xcore1000
  Uuid:                                  ed3398f2-70eb-a673-9f8d-680610255a4d
  Market Name:                           Device 4000
  Vendor Name:                           METAX
  Feature:                               KERNEL_DISPATCH
  Profile:                               BASE_PROFILE
  Float Round Mode:                      NEAR
  Max Queue Number:                      12(0xc)
  Queue Min Size:                        1000
  Queue Max Size:                        20000
  Queue Type:                            MULTI
  Node:                                  1
  Device Type:                           GPU
  Cache Info:
    L1:                                  32 KB
    L2:                                  8192 KB
  Chip ID:                               16384(0x4000)
  Cacheline Size:                        128(0x80)
  Max Clock Freq(MHz):                   1600
  BDFID:                                 2560
  Internal Node ID:                      1
  Accelerator Processors:                104
  PEUs per AP:                           4
  Data Processor Clustes(DPCs):          8
  DPC Arrays per DPC.:                   1
  Watch Pointers on Address Ranges:      4
  Fast Float16 Operation:                FALSE
  Wavefront Size:                        64(0x40)
  Workgroup Max Size:                    1024(0x400)
  Workgroup Max Size per Dimension:
    x                                    1024(0x400)
    y                                    1024(0x400)
    z                                    1024(0x400)
  Max Waves Per AP:                      32(0x20)
  Max Work-items Per AP:                 2048(0x800)
  Grid Max Size:                         4294967295(0xffffffff)
  Grid Max Size per Dimension:
    x                                    4294967295(0xffffffff)
    y                                    4294967295(0xffffffff)
    z                                    4294967295(0xffffffff)
```

图 2-6 MXMACA 编程环境安装验证

2.1.6 卸载 MXMACA 编程环境

操作步骤

1. 执行以下命令，卸载 MXMACA 编程环境。

```
$ sudo /opt/maca/bin/mxmaca-sdk-install.sh -U
$ sudo /opt/mxdriver/mxdriver-install.sh -U
```

说明

使用 MXMACA 安装在系统路径下的 **mxmaca-sdk-install.sh** 和 **mxdriver-install.sh** 脚本进行卸载。

2.2 验证 MXMACA 编程环境

2.2.1 获取 MXMACA 源码示例

操作步骤

1. 参见 2.1 安装 MXMACA 编程环境，成功安装 MXMACA 编程环境后，在 **samples** 目录下获取 MXMACA 源码示例。

图 2-7 samples 目录

2.2.2 验证 MXMACA 编程环境

通过 Makefile 构建 Linux MXMACA 程序。

2.2.2.1 用参考示例项目的 Makefile 构建 MXMACA 程序

示例项目的 Makefile 会使用一些特定选项，可参见表 2-2。

操作步骤

1. 复制一个示例项目（比如 **vectorAdd**）到测试路径下，并将当前路径更改为要构建 MXMACA 程序的示例项目路径。

```
$ mkdir test
$ cp /opt/maca/samples/0_Introduction/vectorAdd/ test/ -R
$ cd test/vectorAdd/
```

2. 选择以下任一方法构建 MXMACA 程序。

   - 仅构建 MXMACA 程序：

     ```
     $ make
     ```

     MXMACA 程序会被构建并生成到示例项目所在路径下，进行运行和启动。

   - 构建并直接启动 MXMACA 程序：

     ```
     $ make run
     [Vector addition of 50000 elements]
     Copy input data from the host memory to the MXMACA device
     MXMACA kernel launch with 196 blocks of 256 threads
     Copy output data from the MACA device to the host memory
     Test PASSED
     MXMACA Sample Done
     ```

3. 执行以下命令，清除构建 MXMACA 程序时生成的项目文件。

```
$ make clean
```

2.2.2.2 用自己的 Makefile 构建 MXMACA 程序

操作步骤

1. 如需从零开始写一个 MXMACA 项目 Makefile，可参见 Makefile 官方指导手册。
2. 新建自己的 Makefile 时，如果需要使用一些特定选项，可参见表 2-2 进行配置。

   表 2-2 列出了 mxcc 编译和构建 MXMACA 程序时支持的典型选项。mxcc 支持的更多选项，参见《曦云®系列通用计算 GPU mxcc 编译器用户指南》。

表 2-2 编译和构建 MXMACA 程序时的可用选项

序号	选项	选项备注
1	# Location of the MXMACA Toolkit MACA_PATH ?= /opt/maca	
2	# start deprecated interface # TARGET_ARCH ?= x86_64	仅支持 x86_64
3	#architecture	仅支持 x86_64

序号	选项	选项备注
	HOST_ARCH := $(shell uname -m) TARGET_ARCH ?= $(HOST_ARCH)	
4	#operating system HOST_OS := $(shell uname - s 2>/dev/null | tr "[:upper:]" "[:lower:]") TARGET_OS ?= $(HOST_OS)	初始支持的操作系统为 Ubuntu 18.04。 其他操作系统将会按需支持。
5	# host compiler HOST_COMPILER ?= mxcc	
6	# Device compiler MXCC := mxcc	
7	# Internal flags MXCCFLAGS := -x maca	
8	#Install directory MACA_INSTALL_TARGET_DIR := /opt/maca/bin	示例项目不会使用此选项

3. 参考 2.2.2.1 用参考示例项目的 Makefile 构建 MXMACA 程序，构建 MXMACA 程序并清除构建 MXMACA 程序时生成的项目文件。

3. 编程模型

本章节描述了 MXMACA 编程模型的几个主要概念。

关于 MXMACA C++的详细描述，参见 4 编程接口。

3.1 概念

3.1.1 GPGPU

GPU 的处理内核远多于 CPU。GPU 最初用于图形处理，后来也用于向量计算和浮点运算，并逐渐用于很多高度并行计算的应用，进而衍生出了 GPGPU，如下图所示。

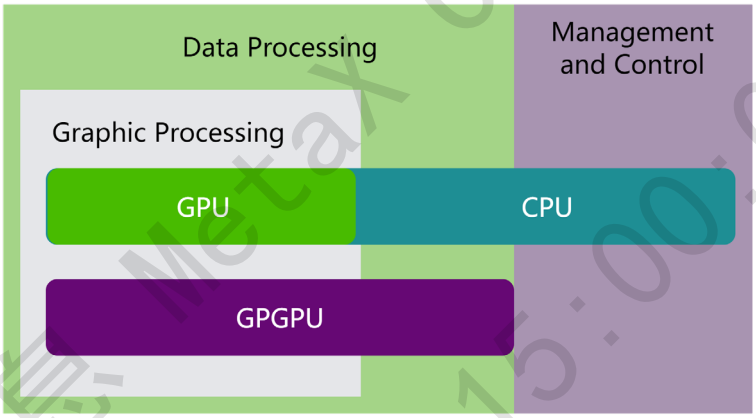

图 3-1 CPU、GPU 和 GPGPU 概念介绍

GPGPU 擅长并行计算，分解是并行处理的核心。分解指的是：

- 将算法划分为单独的任务
- 将数据集划分为可并行操作的离散块

通常，应用程序由并行计算和顺序计算混合组成，因此系统采用 GPU 和 CPU 的混合设计，以最大限度地提高系统整体性能。

3.1.2 MXMACA：通用并行计算平台和编程模型

MXMACA®是一个通用并行计算平台和编程模型，主要用于开发高度并行性的应用程序并部署在沐曦 GPU 上。

MXMACA 的软件环境允许开发人员使用 C++作为高级编程语言。MXMACA 软件栈如图 3-2 所示。

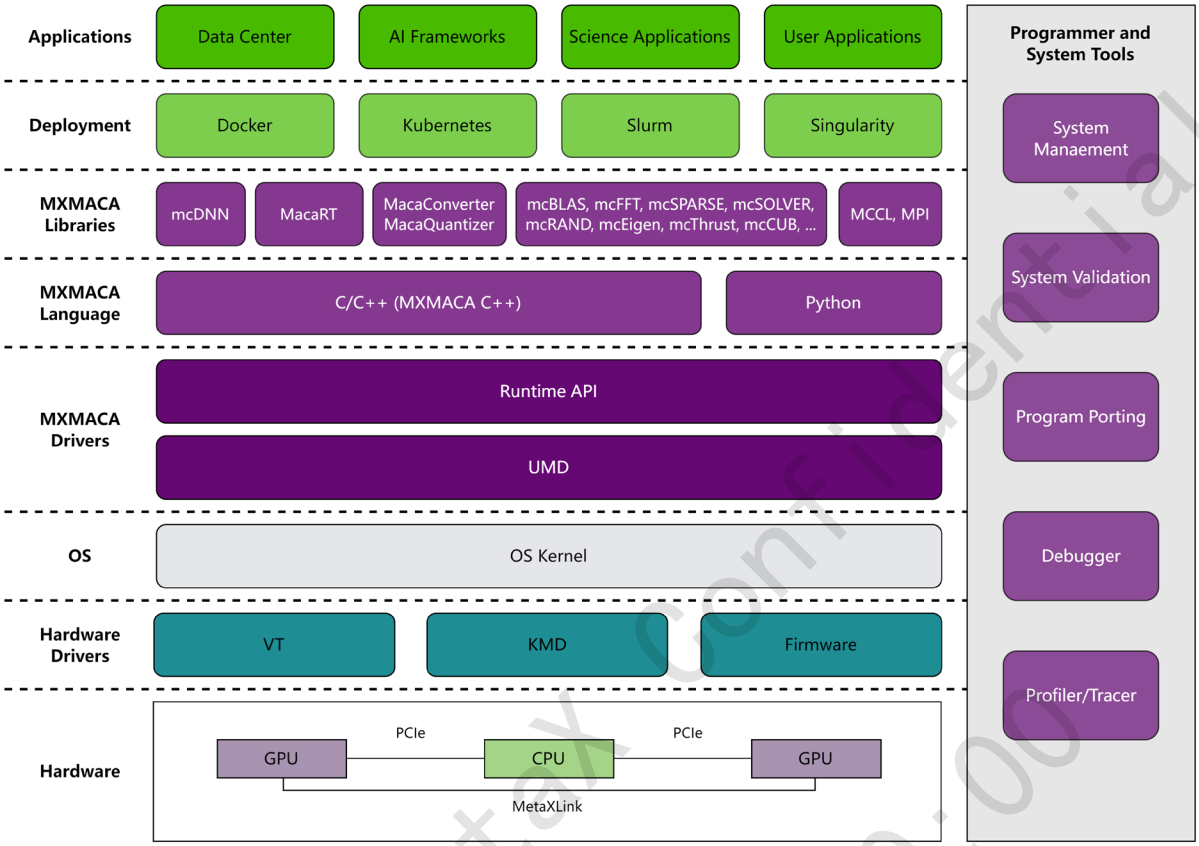

图 3-2 MXMACA 软件栈

MXMACA 支持使用 C/C++编程语言，且包括以下功能：

- Templates
- C++11 Lambdas
- Classes
- namespaces 等

使用 mxcc 编译 MXMACA C++代码，构建在沐曦 GPU 平台上运行的 MXMACA 程序。MXMACA 提供以下 API 接口：

- MXMACA 运行时库：实现 MXMACA 在功能模块上的运行时 API，例如设备管理，流管理，内存管理，执行控制，事件管理等。
- MXMACA 人工智能和计算库（AI & Compute Library，ACL）：为沐曦 GPU 定制的加速库，用于机器学习，数值分析和量子计算等现场应用。

MXMACA 也为开发人员提供了一套专业的记录、跟踪、分析、部署和现场维护工具。

MXMACA 平台支持多个 GPU 的并行计算，可以通过 PCIe 或 MetaXLink 在节点内，也可通过 InfiniBand 或以太网上的 RDMA 跨节点进行。沐曦集合通信库（MetaX Collective Communication Library，MCCL）可用于多 GPU 编程，通过隐藏低级通信的细节来简化编程。

3.2 核函数

MXMACA C++允许程序员定义 MXMACA 内核 C++函数，以此来扩展 C++。N 个不同的 MXMACA 线程（或称为 work-item）会并行执行 N 次内核调用，而不是像常规 C++函数那样只执行一次。

使用`__global__`声明说明符定义内核，并使用新的`<<<...>>>`执行配置句法指定执行内核调用的 MXMACA 线程的数量，详情参见《曦云®系列通用计算 GPU MXMACA C++编程指南》。每个执行内核的线程都分配了一个唯一的线程 ID，可以通过内置变量在内核中访问。

如下图所示，用内置变量`threadIdx`将两个大小为 N 的向量 A 和 B 相加，并将结果存储到向量 C 中。

```
// MACA device kernel definition
__global__ void vectorAdd(const float *A, const float *B, float *C, int numElements)
{
  int i = blockDim.x * blockIdx.x + threadIdx.x;

  if (i < numElements) {
    C[i] = A[i] + B[i] + 0.0f;
  }
}

// Host main routine
int main(void)
{
  ...
  // Launch the Vector Add Kernel with N threads
  vectorAdd<<<1, N>>>(d_A, d_B, d_C, numElements);
  ...
}
```

图 3-3 向量相加代码示例

在此示例中，执行`VecAdd()`的 N 个线程都会执行一次成对加法。

向量加法示例的完整代码，参见 2.2.1 获取 MXMACA 源码示例。

3.3 异构编程

如图 3-4 所示，MXMACA 编程模型假设 GPU 线程在独立的物理设备上执行，该设备作为运行 C++程序的主机协处理器。

例如，可以在沐曦 GPU 设备上执行内核函数，在 CPU 上执行其他 C++程序。

MXMACA 编程模型支持主机和设备都在 DRAM 中维护自己独立的内存空间，分别称为主机内存和设备内存。因此，程序通过调用 MXMACA 运行时库 API（参见 4.2 MXMACA 运行时 API）来管理内核可见的全局和常量内存空间，包括设备内存分配和释放以及主机和设备内存之间的数据传输。

MXMACA 编程模型也支持统一寻址内存（Unified Addressing Memory，UA）技术，该技术创建了一个托管内存池，在该内存池中所有处理器（CPU 或者 GPU）都可以看到具有公共地址空间的单个连贯内存映像，可以用相同的内存地址在 CPU 和 GPU 上访问同一块物理内存。此功能还可实现设备内存的超额分

配，并且无需在主机和设备之间进行显式数据拷贝，从而极大简化了移植应用程序的任务。

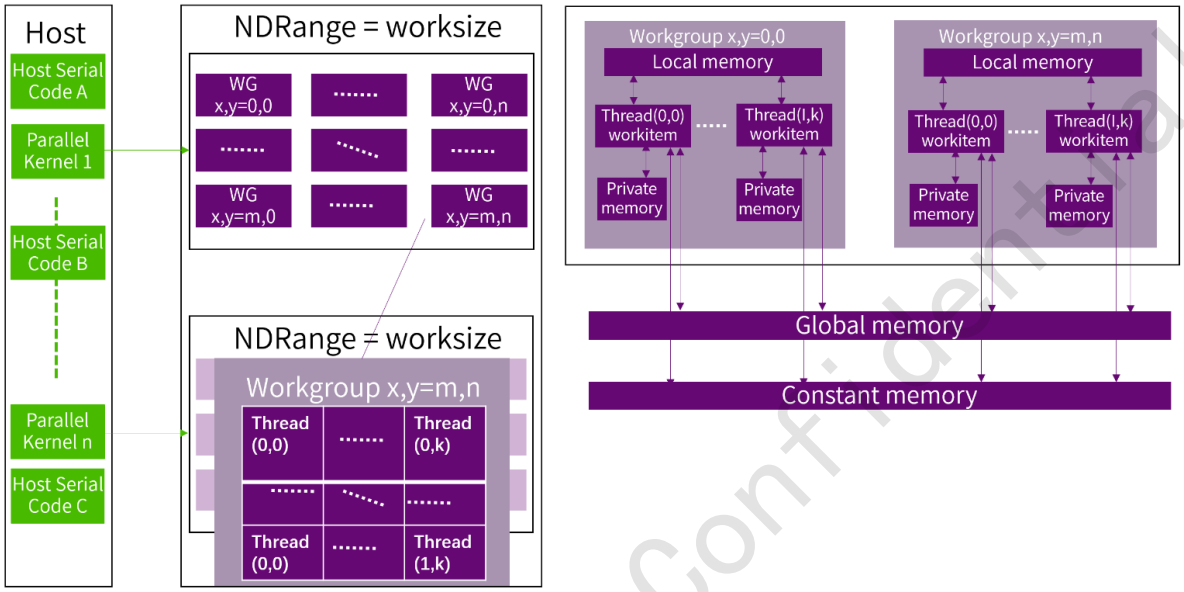

图 3-4 异构编程

说明

串行代码在主机上执行，并行代码在设备上执行。

3.4 存储层次结构

如图 3-5 所示，MXMACA 线程可以从多个内存空间访问数据。每个线程都有私有内存。每个线程块都有对其所有线程可见的本地共享内存，并且内存的生命周期与该线程块相同。所有线程都可以访问相同的全局内存。

所有线程还可以访问一个额外的只读内存空间：常量内存空间。全局和常量内存空间优化是针对不同的内存使用的。全局和常量内存空间在同一应用程序执行的内核启动中是持久的。

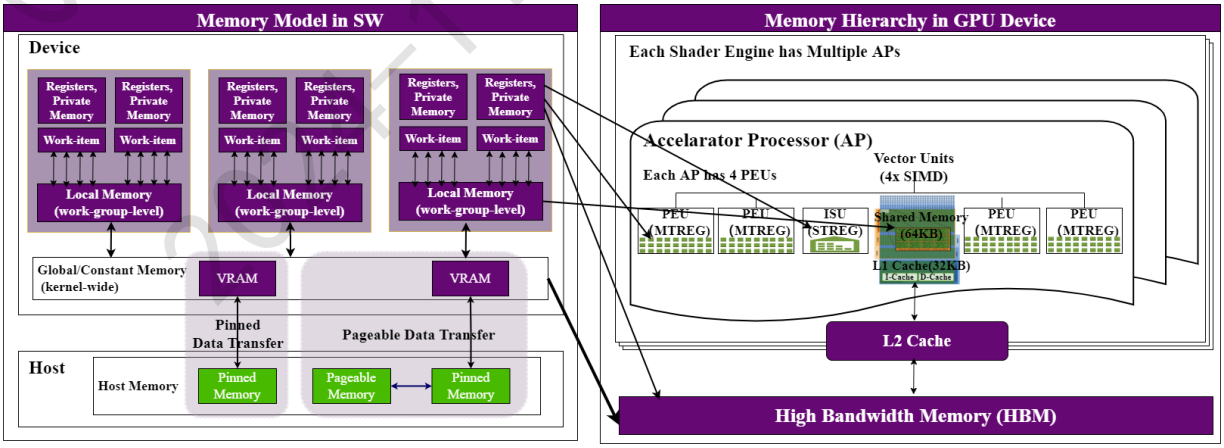

图 3-5 软件内存模型和对硬件的映射

4. 编程接口

MXMACA C++为熟悉 C++编程语言的用户提供了一种简单的途径，可以轻松编写由沐曦 GPU 设备执行的程序。MXMACA C++由C++语言的最小扩展集，MXMACA运行时API和MXMACA人工智能和计算库组成。

4.1 MXMACA C++语言扩展

3.2 核函数中引入了 MXMACA C++语言扩展。它们允许程序员将内核函数定义为 C++ 函数，并在每次调用函数时使用一些新句法来指定 ND-Range 和 Workgroup 维度。任何包含这些扩展的源文件都必须使用 mxcc 进行编译，详情参见《曦云®系列通用计算 GPU mxcc 编译器用户指南》。

4.2 MXMACA 运行时 API

MXMACA 运行时 API 提供了在主机上执行的 C 和 C++函数，用于分配和释放设备内存、在主机内存和设备内存之间传输数据以及管理具有多个设备的系统。运行时 API 的详细介绍，参见《曦云®系列通用计算 GPU 运行时 API 编程指南》。

4.3 MXMACA 人工智能和计算库

支持的 MXMACA 人工智能和计算库，参见下表。

表 4-1 MXMACA 人工智能和计算库

序号	MXMACA 库	描述
1	mcBLAS	基础线性代数程序集
2	mcFFT	快速傅里叶变换库
3	mcDNN	深度神经网络库
4	mcRAND	生成高质量伪随机和准随机数字
5	mcThrust	基于标准模板库的 MXMACA C++模板库
6	mcCUB	C++头库，为 MXMACA 编程模型的每层提供可重用的软件组件
7	mcSPARSE	稀疏矩阵和向量的基本线性代数子程序
8	mcPyTorch	mcPyTorch 基于 PyTorch 2.0/2.1 版本，在 MXMACA 环境中运行，并支持 CPU 和 MXMACA 设备上的张量操作
9	mcSOLVER	基于 mcBLAS 和 mcSPARSE 库的高级包

4.4 开发人工智能相关应用

开发人工智能相关应用，具体文档参见下表。

表 4-2 MXMACA 人工智能相关应用开发文档

序号	开发文档	描述
1	AI 推理用户手册	指导用户如何使用 MacaRT 推理引擎，将训练好的模型部署到曦云系列 GPU 上
2	AI 训练用户手册	指导用户在 AI 训练的场景下，如何使用曦云系列 GPU 进行人工智能算法的训练

4.5 视频编解码相关应用

开发视频编解码相关应用，具体文档参见下表。

表 4-3 MXMACA 视频编解码相关应用开发文档

序号	开发文档	描述
1	视频编解码 VPU 编程指南	指导用户如何使用曦云系列 GPU 视频编解码 VPU，进行硬件加速的软件开发方法

4.6 开发数据中心相关应用

开发数据中心相关应用的文档，具体文档参见下表。

表 4-4 MXMACA 数据中心相关应用开发文档

序号	开发文档	描述
1	mcDF 使用手册	指导用户在曦云系列 GPU 上安装部署 mcDF 和使用 mcDF
2	mcPy 使用手册	指导用户在曦云系列 GPU 上安装部署 mcPy 和使用 mcPy
3	mcFaiss 使用手册	指导用户在曦云系列 GPU 上安装部署 mcFaiss 和使用 mcFaiss
4	mxvs 测试工具套件使用手册	沐曦验收测试套件（MetaX Validation Suite，mxvs）

5. 附录

5.1 术语/缩略语

术语/缩略语	全称	描述
CMake		一个开源的跨平台安装（编译）工具，可以用简单的语句来描述所有平台的安装（编译过程）
GCC	GNU Compiler Collection	一个能够编译多种语言的编译器
GPGPU	General-Purpose GPU	通用 GPU
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
Make		一个智能的批处理工具，用于解释 Makefile 中的指令
Makefile		描述了整个代码工程所有文件的编译顺序、编译规则。Makefile 有自己的书写格式、关键字、函数，可以使用系统 shell 所提供的任何命令来完成想要的工作
mxcc	MXMACA C/C++ Compiler	MXMACA 软件栈中，针对 MetaX GPU 的硬件架构和功能特性设计和发布的编译器
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

声明