曦云®系列通用计算 GPU

mxcc 编译器用户指南

CSOG-23026-020-F3_V03　　2024-08-05

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
V03	2024-08-05	更新以下章节： 4.3 指定编译器/链接器行为的选项
V02	2024-03-15	新增曦云®系列 GPU 产品信息
V01	2023-10-16	正式版本首次发布

目录

表目录

1. 概述

mxcc 是面向沐曦 GPU 的编译器，可以将 MXMACA®源文件编译为包括主机部分和设备部分的可执行文件。主机部分在 CPU 上执行，设备部分在 GPU 上执行。

2. mxcc 的使用

mxcc [options]文件

代码示例

```
mxcc a.maca -o a.out
```

2.1 支持的输入文件类型

支持的输入文件类型，参见下表。

表 2-1 支持的输入文件后缀

输入文件后缀	描述
.maca	MXMACA 源文件，包括主机代码和设备代码
.c	C 源文件
.C, .cc, .CC, .cp, .cpp, .CPP, .c++, .C++, .cxx, .CXX	C++源文件
.o, .a	目标文件
.so	共享目标文件

3. 使用前准备

请确保已提前安装包括 MXMACA 驱动和 mxcc 工具链的完整 MXMACA 软件包。

4. 支持的选项

大多数选项都有一个长名称和一个短名称。短名称显示在括号中。对于有值的选项，使用空格或等号分隔选项名称和值。因此，`opt value` 和 `opt=value` 均受支持。

4.1 File 和 Path 说明

`--output-file file (-o)`

指定输出文件的名称和位置。

`--pre-include file (-include)`

编译时包含该文件，等效于所编译的源文件首行添加`#include "file"`。

`--library library (-l)`

指定要在链接阶段使用的库，不包含库文件的扩展名。

`--define-macro def (-D)`

定义预处理阶段要使用的宏。`def` 可以为 `name` 或者 `name=definition`。

`--undefine-macro def (-U)`

在预处理或编译阶段，取消定义现有宏。

`--include-path path (-I)`

指定 `include` 的查找路径。

`--system-include path (-isystem)`

指定系统 `include` 的查找路径。

`--library-path path (-L)`

指定库的查找路径。

`--dependency-output file (-MF)`

指定依赖输出文件。此选项指定依赖生成步骤（参见 4.2 指定编译阶段的选项的`-M`）的输出文件。如果设置了依赖输出文件，则必须指定选项`-M` 或`-MM`。

`--generate-dependency-targets (-MP)`

为每个依赖添加一个空目标。此选项将伪目标添加到依赖生成步骤，以避免在删除旧依赖时出现 Makefile 错误。输入文件不会作为伪目标。

`--archiver-binary executable (-arbin)`

指定用-lib 创建静态库的归档工具路径。

```
--maca-path path (-maca-path)
```

指定运行时头文件和库所在目录，位于 MXMACA 安装路径下，如[installation_path]/opt/maca。

```
--maca-host-lib-path path (-maca-host-lib-path)
```

指定 MXMACA 主机库的查找路径，即外部 MXMACA 主机库文件所在路径。MXMACA 提供的主机库位于[installation_path]/opt/maca/lib。

```
--maca-host-lib file (-maca-host-lib)
```

指定外部 MXMACA 主机库文件。MXMACA 发布包中提供了一个主机库：maca_mathlib_host.bc，由 mxcc 自动引入。

```
--maca-device-lib-path path (-maca-device-lib-path)
```

指定 MXMACA 设备库查找路径，即外部 MXMACA 设备库文件所在路径。MXMACA 提供的设备库位于[installation_path]/opt/maca/lib。

```
--maca-device-lib file (-maca-device-lib)
```

指定外部 MXMACA 设备库文件。MXMACA 发布包中提供了两个设备库：maca_kernellib.bc 和 maca_mathlib.bc，由 mxcc 自动引入。

```
--maca-host-input file (-maca-host-input)
```

指示文件 file 为仅含有 host 代码或内容的文件。注意此类文件只能参与 MXMACA 编译流程中的主机编译过程。这个文件不需要再通过-maca-link 指示。

```
--maca-device-input file (-maca-device-input)
```

指示文件 file 为仅含有 device 代码或内容的文件。注意此类文件只能参与 MXMACA 编译流程中的设备编译过程。这个文件不需要再通过-maca-link 指示。

```
--input-is-host (-input-is-host)
```

将所有输入文件视为仅含有 host 代码或内容的文件，其他同-maca-host-input。

```
--input-is-device (-input-is-device)
```

将所有输入文件视为仅含有 device 代码或内容的文件，其他同-maca-host-device。

4.2 指定编译阶段的选项

`--lib (-lib)`

如有必要，将所有输入文件编译为目标文件，并将结果添加到指定的库输出文件中。

默认输出文件名为 `a.a`。

`--dlink-obj (-dlink-obj)`

将输入文件编译为 `dlink` 阶段目标文件。仅和`-fgpu-rdc` 一起使用。

`--dlink-asm (-dlink-asm)`

将输入文件编译为 `dlink` 阶段 `asm` 文件。仅和`-fgpu-rdc` 一起使用。

`--compile (-c)`

将每个`.c`、`.cc`、`.cpp`、`.cxx` 和`.maca` 输入文件编译为目标文件。

源文件扩展名替换为`.o` 以创建默认输出文件名。

`--fatbin (-fatbin)`

将所有`.maca` 输入文件编译为仅限设备的`.mcfb` 文件。

mxcc 使用此选项丢弃每个`.maca` 输入文件的主机代码。

源文件扩展名替换为`.mcfb` 以创建默认输出文件名。

`--device-obj (-device-obj)`

将所有`.maca` 输入文件编译为仅限设备的目标文件。

mxcc 使用此选项丢弃每个`.maca` 输入文件的主机代码。

源文件扩展名替换为`.o` 以创建默认输出文件名。

`--device-bin (-device-bin)`

将输入文件编译为仅限设备的二进制文件。

`--preprocess (-E)`

预处理所有`.c`、`.cc`、`.cpp`、`.cxx` 和`.maca` 输入文件。主机和设备代码均受处理并捆绑在一个输出文件中。

默认在 `stdout` 中生成输出。

`--generate-dependencies (-M)`

生成一个依赖文件，该文件可以包含在`.c`、`.cc`、`.cpp`、`.cxx` 和`.maca` 输入文件的 Makefile 中。

默认在 stdout 中生成输出。

```
--generate-nonsystem-dependencies (-MM)
```

与-M 相同，但忽略在系统目录找到的头文件。

默认在 stdout 中生成输出。

```
--generate-dependencies-with-compile (-MD)
```

生成一个依赖文件，并编译输入文件。依赖文件可以包含在.c、.cc、.cpp、.cxx 和.maca 输入文件的 Makefile 中。

此选项不能与-E 一起指定。依赖文件名的计算方法如下：

- 如果指定-MF，则指定的文件名将用作依赖文件名。
- 如果指定-o，则通过将指定文件名的后缀替换为.d 来计算依赖文件名。
- 否则，通过将输入文件名的后缀替换为.d 来计算依赖文件名。

如果依赖文件名是基于-MF 或-o 计算的，则不支持多个输入文件。

```
--generate-nonsystem-dependencies-with-compile (-MMD)
```

与-MD 相同，但忽略在系统目录找到的头文件。

```
--run (-run)
```

将所有输入文件编译并链接到可执行文件中，然后执行此文件。

4.3 指定编译器/链接器行为的选项

```
--profile (-pg)
```

生成插桩的产物，供 gprof 使用。

```
--debug (-g)
```

生成主机代码的调试信息。

```
--device-debug (-device-debug)
```

生成设备代码的调试信息。

```
--generate-line-info (-lineinfo)
```

为设备代码生成行号信息。

```
--optimization-info kind (-opt-info)
```

为指定的优化类型提供优化报告。

```
--optimize level (-O)
```

指定主机代码的优化级别。默认值为 O3。

```
--ftemplate-backtrace-limit=limit (-ftemplate-backtrace-limit=)
```

为单个警告或错误设置模板实例化注释的最大数量。

```
--ftemplate-depth=limit (-ftemplate-depth=)
```

为模板类设置最大实例化深度。

```
--no-exceptions (-noeh)
```

禁用主机代码的异常处理。

```
--shared (-shared)
```

在链接期间生成共享库。必须和-fPIC 一起使用。

```
--x {c|c++|maca} (-x)
```

显式指定输入文件的语言，而不是使用基于文件名称后缀的默认语言。仅对指定此选项之后的文件有效。

```
--std {c++03|c++11|c++14|c++17} (-std)
```

选择特定的 C++方言。默认值为 c++14。

```
--extended-lambda (-extended-lambda)
```

在 lambda 声明中允许__host__和__device__注释。

```
--expt-extended-lambda (-expt-extended-lambda)
```

-extended-lambda 的别名

```
--fkeep-device-symbols (-fkeep-device-symbols)
```

保留未使用的设备符号。

```
--fno-keep-device-symbols (-fno-keep-device-symbols)
```

删除未使用的设备符号。

```
--default-stream {legacy|null|per-thread} (-default-stream)
```

指定默认情况下将编译程序中的 MXMACA 命令发送到的流。

4.4 传递特定阶段选项的选项

`--compiler-options options,... (-Xcompiler)`

直接向编译器/预处理器指定选项，包括设备侧和主机侧。

`--linker-options option (-Xlinker)`

直接向主机链接器指定选项。每个选项需要一个`-Xlinker`。

`-Wl,options,...`

直接向主机链接器指定选项。可以同时输入多个选项。

`--archive-options option (-Xarchive)`

直接向库管理器指定选项。

`--device-linker-options option (-Xmaca-device-linker)`

直接向设备链接器指定选项。

4.5 指导编译器驱动程序的选项

`--clean-targets (-clean)`

删除同一 mxcc 命令在没有使用此选项时生成的所有非临时文件。

`--run-args arguments,... (-run-args)`

当可执行文件与`-run` 一起使用时，指定可执行文件的命令行参数。

`--dependency-target-name target (-MT)`

指定生成依赖文件（参见 4.2 指定编译阶段的选项的`-M`）时，生成的规则目标名称。

`--maca-device-only (-maca-device-only)`

指示本次编译只执行 MXMACA 编译流程中的设备编译过程。

`--maca-host-only (-maca-host-only)`

指示本次编译只执行 MXMACA 编译流程中的主机编译过程。对非 MXMACA 的编译无影响。

`--maca-link (-maca-link)`

指示中间文件（ir 文件，目标文件等）按照 MXMACA 编译流程进行处理。这些中间文件需要同时包含主机和设备内容。

`--nogpuinc (-nogpuinc)`

不为 MXMACA 添加 `include` 路径，也不包含默认的 MXMACA 适配头文件。

`--nogpulib (-nogpulib)`

不为 MXMACA 设备编译链接设备库。

4.6 调整 GPU 代码生成的选项

`-fgpu-rdc`

启用生成可重定位的设备代码。必须先链接可重定位的设备代码，然后才能执行。

如果禁用此选项，则会生成可执行的设备代码。

`--maxrregcount amount (-maxrregcount)`

指定 GPU 函数可以使用的最大 `mtreg` 数量。

`--maxmregcount amount (-maxmregcount)`

`maxrregcount` 的别名。

`--maxsregcount amount (-maxsregcount)`

指定 GPU 函数可以使用的最大 `streg` 数量。

`--use-fast-math (-use-fast-math)`

使用快速数学库，并设置`-ftz=true -prec-div=false -prec-sqrt=false -fmad=true`。

`--ftz {true|false} (-ftz)`

控制单精度非规格化数的支持。`-ftz=true` 将非规格化数值刷新为零，`-ftz=false` 保留非规格化数值。`-use-fast-math` 会设置`-ftz=true`。默认值为 `false`。

`--prec-div {true|false} (-prec-div)`

控制单精度浮点除法和倒数。`-prec-div=true` 启用 IEEE 最近舍入模式，`-prec-div=false` 启用快速近似模式。`-use-fast-math` 会设置`-precdiv=false`。默认值为 `true`。

`--prec-sqrt {true|false} (-prec-sqrt)`

控制单精度浮点平方根。`-prec-sqrt=true` 启用 IEEE 最近舍入模式，`-prec-sqrt=false` 启用快速近似模式。`-use-fast-math` 会设置`-prec-sqrt=false`。默认值为 `true`。

`--fmad {true|false} (-fmad)`

启用/禁用浮点乘法和加法/减法到浮点乘加运算的收缩。`-use-fast-math` 会设置`-fmad=true`。默认值为 `true`。

4.7 通用工具选项

`--disable-warnings (-w)`

禁止所有警告消息。

`--Werror (-Werror)`

将指定类型的警告改为错误。

`--help (-h)`

显示帮助。

`--version (-V)`

打印此工具的版本信息。

`--options-file file (-optf)`

包含指定文件中的命令行选项。

`-fproc-stat-report`

打印子流程统计信息。

`--fproc-stat-report=file (-fproc-stat-report=)`

将子流程统计信息保存到给定的文件中。

4.8 设备链接器选项

`--disable-warnings (-w)`

禁止所有警告消息。

`--preserve-relocs (-preserve-relocs)`

在链接的可执行文件中保留已解决的重定位。

`--verbose (-verbose)`

启用详细模式，打印代码生成统计信息。

`--warning-as-error (-Werror)`

将所有警告改为错误。

5. 示例

5.1 编译包含设备和主机函数的源文件

假设源文件是 **a.maca**，MXMACA 安装在根目录下。命令如下：

```
$ mxcc a.maca -o a.out
```

5.2 编译 C/C++源文件

假设源文件是 **a.c** 和 **b.cpp**，MXMACA 安装在根目录下。命令如下：

```
$ mxcc a.c -o a.out
$ mxcc b.cpp -o b.out
```

5.3 编译并单独链接

源文件可以先编译成目标文件，然后相互链接。目标文件也可以通过其他链接器进行链接。请注意，如果使用 mxcc 作为链接器，则不应在命令中添加`-x maca`。

假设源文件是 **a.maca** 和 **b.maca**，MXMACA 安装在根目录下。命令如下：

```
$ mxcc -c a.maca -o a.o
$ mxcc -c b.maca -o b.o
$ mxcc a.o b.o -o a.out
```

5.4 编译设备函数定义在其他文件中的源文件

支持使用定义在另一个文件中的设备函数。在这种情况下，需要`-fgpu-rdc`。

假设源文件是 **a.maca** 和 **b.maca**，且 **a.maca** 可能会使用定义在 **b.maca** 中的函数。MXMACA 安装在根目录下。命令如下：

```
$ mxcc -fgpu-rdc a.maca b.maca -o a.out
```

5.5 编译设备函数定义在其他文件中的源文件并单独链接

支持使用定义在另一个文件中的设备函数。在这种情况下，需要`-fgpu-rdc`。链接命令中还需要`-maca-link`。

假设源文件是 **a.maca** 和 **b.maca**，且 **a.maca** 可能会使用定义在 **b.maca** 中的函数。MXMACA 安装在根目录下。命令如下：

```
$ mxcc -fgpu-rdc -c a.maca -o a.o
$ mxcc -fgpu-rdc -c b.maca -o b.o
$ mxcc -fgpu-rdc --maca-link a.o b.o -o a.out
```

5.6 编译设备和主机函数有自定义设备库的源文件

假设源文件是 **a.maca**，自定义设备库是 **self.bc**。MXMACA 安装在根目录下。

1. 创建一个文件夹，并把 **self.bc** 放进去。
2. 将安装在**/opt/maca/lib** 的设备库 **maca_kernellib.bc** 和 **maca_mathlib.bc** 复制到文件夹中。命令如下：

```
$ mkdir selfdir
$ cp /opt/maca/lib/maca_kernellib.bc /opt/maca/lib/maca_mathlib.bc
selfdir
$ mxcc a.maca -o a.out --maca-device-lib-path ./selfdir --maca-device-
lib=maca_kernellib.bc
--maca-device-lib=maca_mathlib.bc --maca-device-lib=self.bc
```

6. 附录

6.1 术语/缩略语

术语/缩略语	全称	描述
mxcc	MXMACA C/C++ Compiler	用于沐曦 GPU 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