目录

1 UPTK 基础安装

UPTK 是一个通用异构程序编程框架，能够支持 cuda 程序快速迁移运行至国产 GPU 平台。

UPTK 安装前需加载 dtk 环境，dtk 环境加载步骤如下：

```
source dtk/env.sh
```

UPTK 构建安装步骤如下：

```
cd UPTK
mkdir output    //创建 output 目录，后续编译生成的库安装到该目录下
mkdir build
cd build
cmake .. -DCMAKE_INSTALL_PREFIX=../output     //CMAKE_INSTALL_PREFIX 指定为项目根目录下 output 目录
make
make install
```

UPTK 具体使用步骤可参考以下示例，步骤如下：

```
cd UPTK/example
ls
cd demo
make run
```

2 UPTK API 说明

2.1 流功能接口

2.1.1 UPTKStreamCreate

__host__UPTKError_t UPTKStreamCreate (UPTKStream_t pStream)*

创建一个异步流。

参数

pStream - 指向新流标识符的指针

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

在调用主机线程的当前上下文中创建一个新的异步流。如果没有当前上下文与调用主机线程关联，则会选择一个设备的主上下文，使其与调用线程关联，并在其上初始化，然后在其上创建流。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。

正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。

2.1.2 UPTKStreamCreateWithFlags

__host__UPTKError_t UPTKStreamCreateWithFlags (UPTKStream_t pStream, unsigned int flags)*

创建一个异步流。

参数

`pStream` - 指向新流标识符的指针

`flags` - 流创建

参数

返回 UPTKSuccess, UPTKErrorInvalidValue

描述

在调用主机线程的当前上下文中创建一个新的异步流。如果没有当前上下文与调用主机线程关联，则会选择一个设备的主上下文，使其与调用线程关联，并在其上初始化，然后在其上创建流。`flags` 参数决定了流的行为。`flags` 的有效值为：

- `UPTKStreamDefault`: 默认流创建标志。
- `UPTKStreamNonBlocking`: 指定在创建的流中运行的工作可以与流 0（NULL 流）中的工作并发运行，并且创建的流不应与流 0 执行任何隐式同步。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。

正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。

2.1.3 UPTKStreamDestroy

__host__UPTKError_t UPTKStreamDestroy (UPTKStream_t stream)

销毁并清理一个异步流。

参数

`stream` - 流标识符

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidResourceHandle

描述

销毁并清理由 `stream` 指定的异步流。

如果在调用 `UPTKStreamDestroy()` 时设备仍在流 `stream` 中执行工作，该函数将立即返回，并且与 `stream` 关联的资源将在设备完成流中的所有工作后自动释放。

注意

此函数使用标准默认流语义。

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。

正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。 此调用后使用该句柄是未定义行为。

2.1.4 UPTKStreamSynchronize

__host__UPTKError_t UPTKStreamSynchronize (UPTKStream_t stream)

等待流任务完成。

参数

`stream` - 流标识符

返回

UPTKSuccess, UPTKErrorInvalidResourceHandle

描述

阻塞直到 `stream` 完成所有操作。如果为此设备设置了 `UPTKDeviceScheduleBlockingSync` 标志，主机线程将阻塞，直到流完成其所有任务。

注意

此函数使用标准默认流语义。

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。

正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。

2.1.5 UPTKStreamWaitEvent

__host__UPTKError_t UPTKStreamWaitEvent (UPTKStream_t stream, UPTKEvent_t event, unsigned int flags = 0)

使计算流等待一个事件。

参数

`stream` - 要等待的流

`event` - 要等待的事件

`flags` - 操作的参数（见上文）

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidResourceHandle

描述

使提交到 `stream` 的所有未来工作等待 `event` 中捕获的所有工作。有关事件捕获内容的详细信息，请参见 `UPTKEventRecord()`。同步将在适用时在设备上高效执行。`event` 可以来自与 `stream` 不同的设备。

`flags` 包括： * `UPTKEventWaitDefault`: 默认事件创建标志。 * `UPTKEventWaitExternal`: 在执行流捕获时，事件在图中被捕获为外部事件节点。

注意

此函数使用标准默认流语义。

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。

正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。

2.1.6 UPTKStreamGetDevice

__host__UPTKError_t UPTKStreamGetDevice(UPTKStream_t stream, UPTKStreamCaptureMode mode)

在流上开始图捕获。

参数

`stream` - 要在其中启动捕获的流

`mode` - 控制此捕获序列与其他可能不安全的 API 调用的交互。

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

在 `stream` 上开始图捕获。当流处于捕获模式时，推入流的所有操作将不会执行，而是会被捕获到一个图中，该图将通过 `UPTKStreamEndCapture` 返回。如果 `stream` 是 `UPTKStreamLegacy`，则无法启动捕获。捕获必须在启动它的同一流上结束，并且只有在流尚未处于捕获模式时才能启动。可以通过 `UPTKStreamIsCapturing` 查询捕获模式。可以通过 `UPTKStreamGetCaptureInfo` 查询表示捕获序列的唯一 id。

如果 `mode` 不是 `UPTKStreamCaptureModeRelaxed`，则必须从同一线程在此流上调用 `UPTKStreamEndCapture`。

注意

使用此 API 捕获的内核不得使用纹理和表面引用。通过任何纹理或表面引用进行读写是未定义行为。此限制不适用于纹理和表面对象。 此函数可能还会返回来自先前异步启动的错误代码。

2.1.7 UPTKStreamEndCapture

__host__UPTKError_t UPTKStreamEndCapture (UPTKStream_t stream, UPTKGraph_t pGraph)*

结束流上的捕获，返回捕获的图。

参数

`stream` - 要查询的流

`pGraph` - 捕获的图

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorStreamCaptureWrongThread

描述

结束 stream 上的捕获，通过 pGraph 返回捕获的图。必须已通过调用 UPTKStreamBeginCapture 在 stream 上启动了捕获。如果捕获因违反流捕获规则而无效，则将返回 NULL 图。

如果 UPTKStreamBeginCapture 的 mode 参数不是 UPTKStreamCaptureModeRelaxed，则此调用必须与 UPTKStreamBeginCapture 来自同一线程。

此函数可能还会返回来自先前异步启动的错误代码。

2.1.8 UPTKStreamIsCapturing

__host__UPTKError_t UPTKStreamIsCapturing (UPTKStream_t stream, UPTKStreamCaptureStatus pCaptureStatus)*

返回流的捕获状态。

参数

stream - 要查询的流

pCaptureStatus - 返回流的捕获状态

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorStreamCaptureImplicit

描述

通过 pCaptureStatus 返回 stream 的捕获状态。成功调用后，pCaptureStatus 将包含以下之一：

- UPTKStreamCaptureStatusNone: 流未在捕获。
- UPTKStreamCaptureStatusActive: 流正在捕获。
- UPTKStreamCaptureStatusInvalidated: 流正在捕获，但错误已使捕获序列无效。必须在其启动的流上使用 UPTKStreamEndCapture 终止捕获序列，才能继续使用流。

如果在同一设备上的阻塞流正在捕获时在 UPTKStreamLegacy（“空流”）上调用此函数，它将返回 UPTKErrorStreamCaptureImplicit，并且调用后 pCaptureStatus 是未指定的。阻塞流捕获不会失效。

当阻塞流正在捕获时，传统流处于不可用状态，直到阻塞流捕获终止。传统流不支持流捕获，但尝试使用会隐式依赖于正在捕获的流。

注意

此函数可能还会返回来自先前异步启动的错误代码。

2.1.9 UPTKStreamBeginCapture

__host__ UPTKError_t UPTKStreamBeginCapture (UPTKStream_t stream, UPTKStreamCaptureMode mode)

在流上开始图捕获。

参数

- `stream` – 要在其中启动捕获的流
- `mode` – 控制此捕获序列与其他可能不安全的 `API` 调用的交互

返回

`UPTKSuccess, UPTKErrorInvalidValue`

描述
在 stream 上开始图捕获。当流处于捕获模式时，所有推入流的操作都不会执行，而是被捕获到图中。

2.1.10 UPTKStreamCreateWithPriority

__host__ UPTKError_t UPTKStreamCreateWithPriority (UPTKStream_t pStream, unsigned int flags, int priority)*
创建具有指定优先级的异步流。

参数

- `` `pStream` `` – 指向新流标识符的指针
- `` `flags` `` – 流创建标志
- `` `priority` `` – 流的优先级，较低的数字表示较高的优先级

返回
UPTKSuccess, UPTKErrorInvalidValue

描述
创建具有指定优先级的流，并在 pStream 中返回句柄。

2.2 事件功能接口

2.2.1 UPTKEventCreate

__host__UPTKError_t UPTKEventCreate (UPTKEvent_t event)*

创建一个事件对象。

参数

`event` - 新创建的事件

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorLaunchFailure, UPTKErrorMemoryAllocation

描述 使用 `UPTKEventDefault` 为当前设备创建一个事件对象。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。

正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。

2.2.2 UPTKEventCreateWithFlags

__host__UPTKError_t UPTKEventCreateWithFlags (UPTKEvent_t event, unsigned int flags)*

使用指定标志创建事件对象。

参数

`event` - 新创建的事件

`flags` - 新事件的标志

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorLaunchFailure, UPTKErrorMemoryAllocation

描述

使用指定的标志为当前设备创建事件对象。有效标志包括：

- `UPTKEventDefault`: 默认事件创建标志。
- `UPTKEventBlockingSync`: 指定事件应使用阻塞同步。使用 `UPTKEventSynchronize()` 等待以此标志创建的事件的宿主线程将阻塞，直到事件实际完成。
- `UPTKEventDisableTiming`: 指定创建的事件不需要记录计时数据。创建时指定了此标志且未指定 `UPTKEventBlockingSync` 标志的事件，当与 `UPTKStreamWaitEvent()` 和 `UPTKEventQuery()` 一起使用时，将提供最佳性能。

- UPTKEventInterprocess: 指定创建的事件可以通过 UPTKIpcGetEventHandle() 用作进程间事件。UPTKEventInterprocess 必须与 UPTKEventDisableTiming 一起指定。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。

正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.2.3 UPTKEventDestroy

__host__UPTKError_t UPTKEventDestroy (UPTKEvent_t event)

销毁事件对象。

参数

event - 要销毁的事件

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidResourceHandle, UPTKErrorLaunchFailure

描述

销毁由 event 指定的事件。

事件可以在完成之前销毁（即，当 UPTKEventQuery() 将返回 UPTKErrorNotReady 时）。在这种情况下，调用不会阻塞事件的完成，并且任何关联的资源将在完成时自动异步释放。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。

正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。 此调用后使用该句柄是未定义行为。 如果传入 NULL 作为输入事件，则返回 UPTKErrorInvalidResourceHandle。

2.2.4 UPTKEventSynchronize

__host__ UPTKError_t UPTKEventSynchronize (UPTKEvent_t event)

等待事件完成。

参数

event - 要等待的事件

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidResourceHandle, UPTKErrorLaunchFailure

描述

等待直到当前捕获在 event 中的所有工作完成。有关事件捕获内容的详细信息，请参见 UPTKEventRecord()。

等待使用 UPTKEventBlockingSync 标志创建的事件将导致调用 CPU 线程阻塞，直到设备完成该事件。如果未设置 UPTKEventBlockingSync 标志，则 CPU 线程将忙等待，直到设备完成该事件。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。

正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。 如果传入 NULL 作为输入事件，则返回 UPTKErrorInvalidResourceHandle。

2.2.5 UPTKEventQuery

__host__ UPTKError_t UPTKEventQuery (UPTKEvent_t event)

查询事件的状态。

参数

event - 要查询的事件

返回

UPTKSuccess, UPTKErrorNotReady, UPTKErrorInvalidValue, UPTKErrorInvalidResourceHandle, UPTKErrorLaunchFailure

描述

查询当前由 event 捕获的所有工作的状态。有关事件捕获内容的详细信息，请参见 UPTKEventRecord()。

如果所有捕获的工作已完成，则返回 UPTKSuccess；如果任何捕获的工作未完成，则返回 UPTKErrorNotReady。

就统一内存而言，返回值 UPTKSuccess 等同于调用 UPTKEventSynchronize()。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。

正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。 如果传入 NULL 作为输入事件，则返回 UPTKErrorInvalidResourceHandle。

2.2.6 UPTKEventRecord

__host__UPTKError_t UPTKEventRecord (UPTKEvent_t event, UPTKStream_t stream = 0)

记录一个事件。

参数

event - 要记录的事件

stream - 要在其中记录事件的流

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidResourceHandle, UPTKErrorLaunchFailure

描述

在此调用时捕获 stream 的内容到 event 中。event 和 stream 必须在同一个 UPTK 上下文中。然后，诸如 UPTKEventQuery() 或 UPTKStreamWaitEvent() 之类的调用将检查或等待所捕获工作的完成。此调用之后对 stream 的使用不会修改 event。有关默认情况下捕获内容的说明，请参见默认流行为注释。

可以在同一事件上多次调用 UPTKEventRecord()，并且将覆盖先前捕获的状态。其他 API（如 UPTKStreamWaitEvent()）使用 API 调用时最近捕获的状态，并且不受后来调用 UPTKEventRecord() 的影响。在第一次调用 UPTKEventRecord() 之前，事件表示一组空的工作，因此例如 UPTKEventQuery() 将返回 UPTKSuccess。

注意

此函数使用标准默认流语义。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为

诊断信息。 如果传入 NULL 作为输入事件，则返回 `UPTKErrorInvalidResourceHandle`。

2.2.7 UPTKEventElapsedTime

`__host__ UPTKError_t UPTKEventElapsedTime ( float* ms, UPTKEvent_t start, UPTKEvent_t end )`计算事件之间的时间间隔。

参数

`ms` - 开始和结束之间的时间（毫秒）

`start` - 开始活动

`end` - 结束事件

返回

`UPTKSuccess`, `UPTKErrorNotReady`, `UPTKErrorInvalidValue`, `UPTKErrorInvalidResourceHandle`, `UPTKErrorLaunchFailure`, `UPTKErrorUnknown`

描述

计算两个事件之间的时间间隔（以毫秒为单位，精度约为 `0.5` 微秒）。

2.3 内存功能接口

2.3.1 UPTKMalloc

*__host__UPTKError_t UPTKMalloc (void** devPtr, size_t size)*

在设备上分配内存。

参数

`devPtr` - 指向已分配设备内存的指针

`size` - 请求的分配大小（字节）

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorMemoryAllocation

描述

在设备上分配 `size` 字节的线性内存，并在 `*devPtr` 中返回指向已分配内存的指针。分配的内存针对任何类型的变量进行了适当对齐。内存不会被清除。如果分配失败，`UPTKMalloc()` 返回 `UPTKErrorMemoryAllocation`。

设备的 `UPTKFree` 版本不能与使用主机 API 分配的 `*devPtr` 一起使用，反之亦然。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。

正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.3.2 UPTKFree

__host__UPTKError_t UPTKFree (void devPtr)*

释放设备上的内存。

参数

devPtr - 要释放内存的设备指针

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

释放由 devPtr 指向的内存空间，该指针必须是由先前调用以下内存分配 API 之一返回的 - UPTKMalloc(), UPTKMallocPitch(), UPTKMallocManaged(), UPTKMallocAsync(), UPTKMallocFromPoolAsync()。

注意

如果指针是使用 UPTKMallocAsync 或 UPTKMallocFromPoolAsync 分配的，此 API 不会执行任何隐式同步。调用者必须确保在调用 UPTKFree 之前，对这些指针的所有访问都已完成。为了获得最佳性能和内存重用，用户应使用 UPTKFreeAsync 来释放通过流顺序内存分配器分配的内存。对于所有其他指针，此 API 可能会执行隐式同步。

如果之前已经调用过 UPTKFree(devPtr)，则返回错误。如果 devPtr 为 0，则不执行任何操作。如果失败，UPTKFree() 返回 UPTKErrorValue。

设备的 UPTKFree 版本不能与使用主机 API 分配的 *devPtr 一起使用，反之亦然。

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。

正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.3.3 UPTKMemcpy

__host__UPTKError_t UPTKMemcpy (void dst, const void* src, size_t count, UPTKMemcpyKind kind)*

在主机和设备之间复制数据。

参数

`dst` - 目标内存地址

`src` - 源内存地址

`count` - 要复制的字节大小

`kind` - 传输类型

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidMemcpyDirection

描述

将 `count` 字节从 `src` 指向的内存区域复制到 `dst` 指向的内存区域，其中 `kind` 指定了复制方向，并且必须是 `UPTKMemcpyHostToHost`、`UPTKMemcpyHostToDevice`、`UPTKMemcpyDeviceToHost`、`UPTKMemcpyDeviceToDevice` 或 `UPTKMemcpyDefault` 之一。建议传递 `UPTKMemcpyDefault`，在这种情况下，传输类型是从指针值推断出来的。但是，`UPTKMemcpyDefault` 仅在支持统一虚拟寻址的系统上允许。使用与复制方向不匹配的 `dst` 和 `src` 指针调用 `UPTKMemcpy()` 会导致未定义行为。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。

正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。 此函数在大多数用例中表现出同步行为。 请求的内存区域必须要么完全在 UPTK 中注册，要么在主机可分页传输的情况下，完全未注册。不支持跨越既在 UPTK 中注册又未注册的分配的内存区域，并将返回 `UPTK_ERROR_INVALID_VALUE`。

2.3.4 UPTKIpcGetMemHandle

UPTKresult UPTKIpcGetMemHandle (UPTKipcMemHandle pHandle, UPTKdeviceptr dptr)*

获取现有设备内存分配的进程间内存句柄。

参数

`pHandle` - 指向用户分配的

UPTKipcMemHandle 的指针，用于返回句柄。

dptr - 指向先前分配的设备内存的基本指针

返回

UPTK_SUCCESS, UPTK_ERROR_INVALID_HANDLE, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_OUT_OF_MEMORY, UPTK_ERROR_MAP_FAILED, UPTK_ERROR_INVALID_VALUE

描述

获取通过 UPTKMemAlloc 创建的现有设备内存分配的指针，并导出它以供另一个进程使用。这是一个轻量级操作，可以在一个分配上多次调用而不会产生不利影响。

如果使用 UPTKMemFree 释放了某个内存区域，并且后续的 UPTKMemAlloc 调用返回具有相同设备地址的内存，则 UPTKIpcGetMemHandle 将为新内存返回一个唯一句柄。

IPC 功能仅限于在 Linux 和 Windows 操作系统上支持统一寻址的设备。Windows 上的 IPC 功能为了兼容性目的而受支持，但不推荐使用，因为它会带来性能成本。用户可以通过使用 UPTK_DEVICE_ATTRIBUTE_IPC_EVENT_SUPPORTED 调用 UPTKapiDeviceGetAttribute 来测试其设备的 IPC 功能。

2.3.5 UPTKIpcOpenMemHandle

UPTKresult UPTKIpcOpenMemHandle (UPTKdeviceptr pdptr, UPTKipcMemHandle handle, unsigned int Flags)*

打开从另一个进程导出的进程间内存句柄，并返回可在本地进程中使用的设备指针。

参数

pdptr - 返回的设备指针

handle - 要打开的 UPTKipcMemHandle

Flags - 此操作的标志。必须指定为 UPTK_IPC_MEM_LAZY_ENABLE_PEER_ACCESS

返回

UPTK_SUCCESS, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_MAP_FAILED, UPTK_ERROR_INVALID_HANDLE, UPTK_ERROR_TOO_MANY_PEERS, UPTK_ERROR_INVALID_VALUE

描述

将使用 UPTKIpcGetMemHandle 从另一个进程导出的内存映射到当前设备地址空间。对于不同设备上的上下文，UPTKIpcOpenMemHandle 可以尝试启用设备之间的点对点访问，就像用户调用了 UPTKCtxEnablePeerAccess 一样。此行为由 UPTK_IPC_MEM_LAZY_ENABLE_PEER_ACCESS 标志控制。UPTKDeviceCanAccessPeer 可以确定映射是否可能。

可以打开 UPTKipcMemHandle 的上下文受到以下限制：给定进程中每个 UPTKdevice 的 UPTKipcMemHandle 只能由每个其他进程的每个 UPTKdevice 的一个 UPTKcontext 打开。

如果当前上下文已经打开了内存句柄，则该句柄的引用计数增加 1，并返回现有的设备指针。

从 UPTKIpcOpenMemHandle 返回的内存必须使用 UPTKIpcCloseMemHandle 释放。

在导入上下文中调用 UPTKIpcCloseMemHandle 之前，对导出内存区域调用 UPTKMemFree 将导致未定义行为。

IPC 功能仅限于在 Linux 和 Windows 操作系统上支持统一寻址的设备。Windows 上的 IPC 功能为了兼容性目的而受支持，但不推荐使用，因为它会带来性能成本。用户可以通过使用 UPTK_DEVICE_ATTRIBUTE_IPC_EVENT_SUPPORTED 调用 UPTKapiDeviceGetAttribute 来测试其设备的 IPC 功能。

注意

不保证返回在 *pdptr 中的地址。特别是，多个进程可能不会为同一句柄接收相同的地址。

2.3.6 UPTKIpcCloseMemHandle

UPTKresult UPTKIpcCloseMemHandle (UPTKdeviceptr dptr)

尝试关闭使用 UPTKIpcOpenMemHandle 映射的内存。

参数

dptr - 由 UPTKIpcOpenMemHandle 返回的设备指针

返回

UPTK_SUCCESS, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_MAP_FAILED, UPTK_ERROR_INVALID_HANDLE, UPTK_ERROR_INVALID_VALUE

描述

将由 UPTKIpcOpenMemHandle 返回的内存的引用计数减 1。当引用计数达到 0 时，此 API 取消映射内存。导出进程中的原始分配以及其他进程中的导入映射将不受影响。

用于启用点对点访问的任何资源将在这是使用它们的最后一个映射时被释放。

IPC 功能仅限于在 Linux 和 Windows 操作系统上支持统一寻址的设备。Windows 上的 IPC 功能为了兼容性目的而受支持，但不推荐使用，因为它会带来性能成本。用户可以通过使用 UPTK_DEVICE_ATTRIBUTE_IPC_EVENT_SUPPORTED 调用 UPTKapiDeviceGetAttribute 来测试其设备的 IPC 功能。

2.3.7 UPTKMemcpyPeerAsync

__host__ UPTKError_t UPTKMemcpyPeerAsync (void dst, int dstDevice, const void* src, int srcDevice, size_t count, UPTKStream_t stream = 0)在两个设备之间异步复制内存。*

参数

`dst` - 目标设备指针

`dstDevice` - 目标设备

`src` - 源设备指针

`srcDevice` - 源设备

`count` - 内存复制的字节大小

`stream` - 流标识符

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidDevice

描述

将内存从一个设备异步复制到另一个设备上的内存。

2.3.8 UPTKMemcpyPeer

__host__ UPTKError_t UPTKMemcpyPeer (void dst, int dstDevice, const void* src, int srcDevice, size_t count)*

在两个设备之间复制内存。

参数

`dst` - 目标设备指针

`dstDevice` - 目标设备

`src` - 源设备指针

`srcDevice` - 源设备

`count` - 内存复制的字节大小

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidDevice

描述

将内存从一个设备复制到另一个设备上的内存。

2.3.9 UPTKMemcpyAsync

__host__ __device__ UPTKError_t UPTKMemcpyAsync (void dst, const void* src, size_t count, UPTKMemcpyKind kind, UPTKStream_t stream = 0)*
在主机和设备之间复制数据。

参数

`dst` – 目标内存地址

`src` – 源内存地址

`count` – 要复制的字节大小

`kind` – 传输类型

`stream` – 流标识符

返回

`UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidMemcpyDirection`

描述

将 `count` 字节从 `src` 指向的内存区域复制到 `dst` 指向的内存区域。

2.3.10 UPTKMallocManaged

*__host__ UPTKError_t UPTKMallocManaged (void** devPtr, size_t size, unsigned int flags = UPTKMemAttachGlobal)*

分配由统一内存系统自动管理的内存。

参数

`devPtr` – 指向分配的设备内存的指针

`size` – 请求的分配大小（字节）

`flags` – 必须是 `UPTKMemAttachGlobal` 或 `UPTKMemAttachHost`（默认为 `UPTKMemAttachGlobal`）

返回

`UPTKSuccess, UPTKErrorMemoryAllocation, UPTKErrorNotSupported, UPTKErrorInvalidValue`

描述

在设备上分配 `size` 字节的托管内存，并在 `*devPtr` 中返回指向分配内存的指针。

2.3.11 UPTKMallocAsync

*__host__ UPTKError_t UPTKMallocAsync (void** devPtr, size_t size, UPTKStream_t hStream)*

以流式语义分配内存。

参数

`devPtr` - 返回的设备指针

`size` - 要分配的字节数

`hStream` - 建立流排序约定和要从中分配的内存池的流

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorNotSupported, UPTKErrorOutOfMemory

描述

向 hStream 中插入一个分配操作。分配操作完成后才能访问分配的内存。

2.3.12 UPTKFreeAsync

__host__ UPTKError_t UPTKFreeAsync (void devPtr, UPTKStream_t hStream)*

以流式语义释放内存。

参数

`devPtr` - 要释放的设备指针

`hStream` - 建立流排序承诺的流

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorNotSupported

描述

向 hStream 中插入一个释放操作。流执行到释放操作后，不得再访问已分配的内存。

2.4 设备功能接口

2.4.1 UPTKChooseDevice

__host__UPTKError_t UPTKChooseDevice (int device, const UPTKDeviceProp* prop)*

选择最符合标准的计算设备。

参数

`device` - 最佳匹配的设备

`prop` - 所需的设备属性

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

在 `device` 中返回其属性与 `prop` 最匹配的设备。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。

正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。

2.4.2 UPTKSetDevice

__host__ UPTKError_t UPTKSetDevice (int device)

设置用于 GPU 执行的设备。

参数

`device` - 活动主机线程应在其上执行设备代码的设备。

返回

UPTKSuccess, UPTKErrorInvalidDevice, UPTKErrorDeviceUnavailable,

描述

将 `device` 设置为调用主机线程的当前设备。有效的设备 ID 是 0 到 (`UPTKGetDeviceCount()` - 1)。

此后从此主机线程使用 `UPTKMalloc()`、`UPTKMallocPitch()` 或 `UPTKMallocArray()` 分配的任何设备内存将物理驻留在 `device` 上。从此主机线程使用 `UPTKMallocHost()` 或 `UPTKHostAlloc()` 或 `UPTKHostRegister()` 分配的任何主机内存将具有与 `device` 关联的生存期。从此主机线程创建的任何流或事件将与 `device` 关联。从此主机线程使用 `<<<>>>` 操作符或 `UPTKLaunchKernel()` 启动的任何内核将在 `device` 上执行。

此调用可以从任何主机线程、对任何设备、在任何时间进行。此函数不会与先前或新设备进行任何同步，并且只应在初始化运行时的上下文状态时花费显著时间。此调用将指定设备的主上下文绑定到调用线程，所有后续的内存分配、流和事件创建以及内核启动都将与主上下文关联。此函数还将立即在主上下文上初始化运行时状态，并且上下文将立即在设备上变为

当前上下文。如果设备处于 UPTKComputeModeExclusiveProcess 模式并被另一个进程占用，或者设备处于 UPTKComputeModeProhibited 模式，此函数将返回错误。

在使用此函数之前不需要调用 UPTKInitDevice。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。

正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.4.3 UPTKGetDevice

__host__ UPTKError_t UPTKGetDevice (int device)*

返回当前正在使用的设备。

参数

device - 返回活动主机线程执行设备代码所在的设备。

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorDeviceUnavailable,

描述

在 device 中返回调用主机线程的当前设备。

注意

此函数可能还会返回来自先前异步启动的错误代码。

如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。

正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.4.4 UPTKDeviceSynchronize

__host__ UPTKError_t UPTKDeviceSynchronize (void)

等待计算设备完成。

返回

UPTKSuccess

描述

阻塞直到设备完成所有先前请求的任务。如果先前的任务之一失败，`UPTKDeviceSynchronize()` 将返回错误。如果为此设备设置了 `UPTKDeviceScheduleBlockingSync` 标志，主机线程将阻塞，直到设备完成其工作。

注意

在设备代码中使用 `UPTKDeviceSynchronize` 在 UPTK 11.6 中已弃用，并在 compute_90+ 编译中移除。对于计算能力 < 9.0，目前需要在编译时通过指定 `-D UPTK_FORCE_CDP1_IF_SUPPORTED` 来选择加入，以继续在设备代码中使用 `UPTKDeviceSynchronize()`。这与主机端的 `UPTKDeviceSynchronize` 不同，后者仍然受支持。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。 正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。

2.4.5 UPTKDeviceReset

__host__ UPTKError_t UPTKDeviceReset (void)

销毁当前进程中当前设备上的所有分配并重置所有状态。

返回

UPTKSuccess

描述

显式销毁并清理与当前进程中当前设备关联的所有资源。调用者有责任确保这些资源不会被后续的 API 调用访问或传递，否则将导致未定义行为。这些资源包括 UPTK 类型 `UPTKStream_t`、`UPTKEvent_t`、`UPTKArray_t`、`UPTKMipmappedArray_t`、`UPTKPitchedPtr`、`UPTKTextureObject_t`、`UPTKSurfaceObject_t`、`textureReference`、`surfaceReference`、`UPTKExternalMemory_t`、`UPTKExternalSemaphore_t` 和 `UPTKGraphicsResource_t`。这些资源还包括通过 `UPTKMalloc`、`UPTKMallocHost`、`UPTKMallocManaged` 和 `UPTKMallocPitch` 进行的内存分配。任何后续对此设备的 API 调用都将重新初始化该设备。

此函数将立即重置设备。调用者有责任确保在调用此函数时，设备未被进程中的任何其他主机线程访问。

注意

`UPTKDeviceReset()` 不会销毁由 `UPTKMallocAsync()` 和 `UPTKMallocFromPoolAsync()` 进行的内存分配。这些内存分配需要显式销毁。 如果非主 CUcontext 是线程的当前上下文，`UPTKDeviceReset()` 将仅销毁该 CUcontext 的内部 UPTK RT 状态。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。 正如

UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.4.6 UPTKDeviceGetName

UPTKresult UPTKDeviceGetName (char name, int len, UPTKdevice dev)*
返回设备的标识符字符串。

参数

`name` – 返回设备的标识符字符串

`len` – 名称中要存储的字符串的最大长度

`dev` – 获取设备标识符字符串

返回
UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_INVALID_DEVICE

描述
返回一个 ASCII 字符串，用于标识 name 指向的以 NULL 结尾的字符串中的设备 dev。

2.4.7 UPTKDeviceGetAttribute

__host__ __device__ UPTKError_t UPTKDeviceGetAttribute (int value, UPTKDeviceAttr attr, int device)*

返回设备相关信息。

参数

`value` – 返回的设备属性值

`attr` – 要查询的设备属性

`device` – 要查询的设备编号

返回

UPTKSuccess, UPTKErrorInvalidDevice, UPTKErrorInvalidValue

描述

返回 *value 中设备 device 上属性 attr 的整数值。

2.4.8 UPTKDeviceEnablePeerAccess

__host__ UPTKError_t UPTKDeviceEnablePeerAccess (int peerDevice, unsigned int flags)

允许直接访问对等设备上的内存分配。

参数

`peerDevice` - 对等设备，以便从当前设备直接访问

`flags` - 保留供将来使用，必须设置为 0

返回

UPTKSuccess, UPTKErrorInvalidDevice,
UPTKErrorPeerAccessAlreadyEnabled, UPTKErrorInvalidValue

描述

成功后，当前设备将立即可以访问来自 peerDevice 的所有分配。

2.4.9 UPTKGetDeviceCount

__host__ __device__ UPTKError_t UPTKGetDeviceCount (int count)*

返回计算能力设备的数量。

参数

`count` - 返回计算能力大于或等于 2.0 的设备数量

返回

UPTKSuccess

描述

返回在 *count 中可用的计算能力大于或等于 2.0 的设备数量。

2.4.10 UPTKInit

UPTKresult UPTKInit (unsigned int Flags)

初始化 UPTK 驱动程序 API。

参数

`Flags` - UPTK 的初始化标志，目前必须为 0

返回

UPTK_SUCCESS, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_INVALID_DEVICE,
UPTK_ERROR_SYSTEM_DRIVER_MISMATCH,
UPTK_ERROR_COMPAT_NOT_SUPPORTED_ON_DEVICE

描述

初始化驱动程序 `API`，必须在当前进程中调用任何其他驱动程序 `API` 函数之前调用。

2.5 模块功能接口

2.5.1 UPTKModuleLoad

UPTKresult UPTKModuleLoad (UPTKmodule module, const char* fname)*

加载计算模块。

参数

`module` - 返回的模块

`fname` - 要加载的模块的文件名

返回

UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED,
UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_VALUE,
UPTK_ERROR_INVALID_PTX, UPTK_ERROR_UNSUPPORTED_PTX_VERSION,
UPTK_ERROR_NOT_FOUND, UPTK_ERROR_OUT_OF_MEMORY,
UPTK_ERROR_FILE_NOT_FOUND, UPTK_ERROR_NO_BINARY_FOR_GPU,
UPTK_ERROR_SHARED_OBJECT_SYMBOL_NOT_FOUND,
UPTK_ERROR_SHARED_OBJECT_INIT_FAILED, UPTK_ERROR_JIT_COMPILER_NOT_FOUND

描述

获取文件名 `fname` 并将相应的模块 `module` 加载到当前上下文中。UPTK 驱动程序 API 不会尝试延迟分配模块所需的资源；如果无法分配模块所需的函数和数据（常量和全局）的内存，`UPTKModuleLoad()` 将失败。该文件应该是 nvcc 输出的 UPTKbin 文件，或者是 nvcc 输出或手写的 PTX 文件，或者是从工具链 4.0 或更高版本的 nvcc 输出的 fatbin 文件。

注意

此函数可能还会返回来自先前异步启动的错误代码。

2.5.2 UPTKModuleLoadData

UPTKresult UPTKModuleLoadData (UPTKmodule module, const void* image)*

加载模块的数据。

参数

`module` - 返回的模块

`image` - 要加载的模块数据

返回

UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_INVALID_PTX, UPTK_ERROR_UNSUPPORTED_PTX_VERSION, UPTK_ERROR_OUT_OF_MEMORY, UPTK_ERROR_NO_BINARY_FOR_GPU, UPTK_ERROR_SHARED_OBJECT_SYMBOL_NOT_FOUND, UPTK_ERROR_SHARED_OBJECT_INIT_FAILED, UPTK_ERROR_JIT_COMPILER_NOT_FOUND

描述

获取指针 `image` 并将相应的模块 `module` 加载到当前上下文中。

注意

此函数可能还会返回来自先前异步启动的错误代码。

2.5.3 UPTKModuleUnload

UPTKresult UPTKModuleUnload (UPTKmodule hmod)

卸载模块。

参数

`hmod` - 要卸载的模块

返回

UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_NOT_PERMITTED

描述

从当前上下文卸载模块 `hmod`。尝试卸载从库管理 API（如 `UPTKLibraryGetModule`）获取的模块将返回 `UPTK_ERROR_NOT_PERMITTED`。

注意

此函数可能还会返回来自先前异步启动的错误代码。 此调用后使用该句柄是未定义行为。

2.5.4 UPTKModuleGetFunction

UPTKresult UPTKModuleGetFunction (UPTKfunction hfunc, UPTKmodule hmod, const char* name)*

返回函数句柄。

参数

`hfunc` - 返回的函数句柄

`hmod` - 要从中检索函数的模块

name - 要检索的函数名称

返回

UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_NOT_FOUND

描述

在 *hfunc 中返回位于模块 hmod 中的名称 name 的函数的句柄。如果不存在该名称的函数，UPTKModuleGetFunction() 返回 UPTK_ERROR_NOT_FOUND。

注意

此函数可能还会返回来自先前异步启动的错误代码。

2.5.5 UPTKModuleGetGlobal

UPTKresult UPTKModuleGetGlobal (UPTKdeviceptr dptr, size_t* bytes, UPTKmodule hmod, const char* name)*

从模块返回全局指针。

参数

dptr - 返回的全局设备指针

bytes - 返回的全局大小（字节）

hmod - 要从中检索全局量的模块

name - 要检索的全局量名称

返回

UPTK_SUCCESS, UPTK_ERROR_DEINITialized, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_NOT_FOUND

描述

在 *dptr 和 *bytes 中返回位于模块 hmod 中的名称 name 的全局量的基指针和大小。如果不存在该名称的变量，UPTKModuleGetGlobal() 返回 UPTK_ERROR_NOT_FOUND。参数 dptr 或 bytes 之一（不能同时）可以为 NULL，在这种情况下它将被忽略。

注意

此函数可能还会返回来自先前异步启动的错误代码。

2.5.6 UPTKLinkCreate

UPTKresult UPTKLinkCreate (unsigned int numOptions, UPTKjit_option options, void** optionValues, UPTKlinkState* stateOut)*

创建一个挂起的 JIT 链接器调用。

参数

`numOptions` - 选项数组的大小

`options` - 链接器和编译器选项数组

`optionValues` - 选项值数组，每个都转换为

`void **` `stateOut` - 成功时，这将包含一个 `UPTKlinkState` 以指定和完成此操作

返回

UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_OUT_OF_MEMORY, UPTK_ERROR_JIT_COMPILER_NOT_FOUND

描述

如果调用成功，调用者拥有返回的 `UPTKlinkState`，最终应使用 `UPTKLinkDestroy` 销毁它。设备代码机器大小（32 或 64 位）将与调用应用程序匹配。

2.5.7 UPTKLinkDestroy

UPTKresult UPTKLinkDestroy (UPTKlinkState state)

销毁 JIT 链接器调用的状态。

参数

`state` - 链接器调用的状态对象

返回

UPTK_SUCCESS, UPTK_ERROR_INVALID_HANDLE

描述

销毁链接器状态。

2.5.8 UPTKLinkAddData

UPTKresult UPTKLinkAddData (UPTKlinkState state, UPTKjitInputType type, void data, size_t size, const char* name, unsigned int numOptions, UPTKjit_option* options, void** optionValues)*

向挂起的链接器调用添加输入。

参数

`state` - 挂起的链接器操作。

`type` - 输入数据的类型。

`data` - 输入数据。

`size` - 输入数据的长度。

`name` - 此输入在日志消息中的可选名称。

`numOptions` - 选项的大小。

`options` - 仅适用于此输入的选项（覆盖来自 `UPTKLinkCreate` 的选项）。

`optionValues` - 选项值数组，每个都转换为 `void *`。

返回

UPTK_SUCCESS, UPTK_ERROR_INVALID_HANDLE, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_INVALID_IMAGE, UPTK_ERROR_INVALID_PTX, UPTK_ERROR_UNSUPPORTED_PTX_VERSION, UPTK_ERROR_OUT_OF_MEMORY, UPTK_ERROR_NO_BINARY_FOR_GPU

描述

数据的所有权由调用者保留。此调用返回后，不保留对任何输入的引用。

此方法仅接受编译器选项，如果数据必须从 PTX 编译，则使用这些选项，并且不接受任何 `UPTK_JIT_WALL_TIME`、`UPTK_JIT_INFO_LOG_BUFFER`、`UPTK_JIT_ERROR_LOG_BUFFER`、`UPTK_JIT_TARGET_FROM_UPTKCONTEXT` 或 `UPTK_JIT_TARGET`。

2.5.9 UPTKLinkComplete

*UPTKresult UPTKLinkComplete (UPTKlinkState state, void** cubinOut, size_t* sizeOut)*

完成挂起的链接调用。

参数

`` `state` `` – 挂起的链接调用

`` `UPTKbinOut` `` – 成功时指向输出映像

`` `sizeOut` `` – 可选参数，接收生成映像的大小

返回

`UPTK_SUCCESS`, `UPTK_ERROR_INVALID_HANDLE`, `UPTK_ERROR_OUT_OF_MEMORY`

描述

完成挂起的链接操作并返回链接设备代码的 `cubin` 映像

2.6 上下文功能接口

2.6.1 UPTKCtxCreate

UPTKresult UPTKCtxCreate (UPTKcontext pctx, UPTKctxCreateParams* ctxCreateParams, unsigned int flags, UPTKdevice dev)*

创建 UPTK 上下文。

参数

`pctx` – 返回新上下文的上下文句柄

`ctxCreateParams` – 上下文创建参数

`flags` – 上下文创建标志

`dev` – 用于创建上下文的设备

返回

UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_DEVICE, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_NOT_SUPPORTED, UPTK_ERROR_OUT_OF_MEMORY, UPTK_ERROR_UNKNOWN

描述

创建一个新的 UPTK 上下文并将其与调用线程关联。上下文的默认使用计数为 1，调用者在使用完毕后必须调用 UPTKCtxDestroy()。如果线程当前已存在一个上下文，则该上下文将被新创建的上下文替换，并可通过后续调用 UPTKCtxPopUPTKrrent() 恢复。

注意

此函数也可能返回先前异步启动时的错误代码。

2.6.2 UPTKCtxDestroy

UPTKresult UPTKCtxDestroy (UPTKcontext ctx)

销毁 UPTK 上下文。

参数

`ctx` – 需要销毁的上下文

返回

UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT, UPTK_ERROR_INVALID_VALUE

描述

销毁由 `ctx` 指定的 `UPTK` 上下文。调用者必须确保在 `UPTKCtxDestroy()` 执行期间不会出现任何使用 `ctx` 的 `API` 调用。

注意

此函数也可能返回先前异步启动时的错误代码。

2.6.3 UPTKCtxGetLimit

UPTKresult UPTKCtxGetLimit (size_t pvalue, UPTKlimit limit)*

返回资源限制。

参数

`pvalue` – 返回的限制大小

`limit` – 限制查询

返回

`UPTK_SUCCESS, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_UNSUPPORTED_LIMIT`

描述

返回 `*pvalue` 中当前限制的大小。支持的 `UPTKlimit` 值包括：

`UPTK_LIMIT_STACK_SIZE`

`UPTK_LIMIT_PRINTF_FIFO_SIZE`

`UPTK_LIMIT_MALLOC_HEAP_SIZE`

`UPTK_LIMIT_DEV_RUNTIME_SYNC_DEPTH`

`UPTK_LIMIT_DEV_RUNTIME_PENDING_LAUNCH_COUNT`

`UPTK_LIMIT_MAX_L2_FETCH_GRANULARITY`

`UPTK_LIMIT_PERSISTING_L2_CACHE_SIZE`

2.6.4 UPTKCtxSetCurrent

UPTKresult UPTKCtxSetCurrent(UPTKcontext ctx)

将指定的 `UPTK` 上下文绑定到调用 `CPU` 线程。

参数

`ctx` – 要绑定到调用 `CPU` 线程的上下文

返回

`UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT`

描述

将指定的 `UPTK` 上下文绑定到调用 `CPU` 线程。如果 `ctx` 为 `NULL`，则解除先前绑定的 `UPTK` 上下文。

2.6.5 UPTKCtxSetLimit

UPTKresult UPTKCtxSetLimit (UPTKlimit limit, size_t value)

设置资源限制。

参数

`limit` - 限制设置

`value` - 限制的大小

返回

`UPTK_SUCCESS, UPTK_ERROR_INVALID_VALUE, UPTK_ERROR_UNSUPPORTED_LIMIT, UPTK_ERROR_OUT_OF_MEMORY, UPTK_ERROR_INVALID_CONTEXT`

描述

设置限制值是应用程序请求更新上下文维护的当前限制值。驱动程序可以根据硬件要求自由修改请求的值。

2.6.6 UPTKCtxSynchronize

UPTKresult UPTKCtxSynchronize (void)

阻塞当前上下文中的任务完成。

返回

`UPTK_SUCCESS, UPTK_ERROR_DEINITIALIZED, UPTK_ERROR_NOT_INITIALIZED, UPTK_ERROR_INVALID_CONTEXT`

描述

阻塞直到当前上下文完成所有先前请求的任务。

2.7 图功能接口

2.7.1 UPTKGraphCreate

__host__UPTKError_t UPTKGraphCreate (UPTKGraph_t pGraph, unsigned int flags)*

创建一个图。

参数

pGraph - 返回新创建的图

flags - 图创建标志，必须为 0

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorMemoryAllocation

描述

创建一个空图，并通过 pGraph 返回。

注意

图对象不是线程安全的。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.7.2 UPTKGraphClone

__host__UPTKError_t UPTKGraphClone (UPTKGraph_t pGraphClone, UPTKGraph_t originalGraph)*

克隆一个图。

参数

pGraphClone - 返回新创建的克隆图

originalGraph - 要克隆的图

返回

UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorMemoryAllocation

描述

此函数创建 originalGraph 的副本，并在 pGraphClone 中返回它。所有参数都被复制到克隆的图中。原始图可以在此调用后被修改，而不会影响克隆。

原始图中的子图节点会被递归复制到克隆中。

注意

图对象不是线程安全的。此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.7.3 UPTKGraphDestroy

__host__UPTKError_t UPTKGraphDestroy (UPTKGraph_t graph)

销毁一个图。

参数

graph - 要销毁的图

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

销毁由 graph 指定的图及其所有节点。

注意

图对象不是线程安全的。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。 此调用后使用该句柄是未定义行为。

2.7.4 UPTKGraphAddDependencies

__host__UPTKError_t UPTKGraphAddDependencies (UPTKGraph_t graph, const UPTKGraphNode_t from, const UPTKGraphNode_t* to, size_t numDependencies)*

向图中添加依赖边。

参数

graph - 要添加依赖项的图

from - 提供依赖项的节点数组

to - 依赖节点数组

numDependencies - 要添加的依赖项数量

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

要添加的依赖项数量由 numDependencies 定义。pFrom 和 pTo 中对应索引的元素定义了一个依赖项。pFrom 和 pTo 中的每个节点必须属于 graph。

如果 numDependencies 为 0，则忽略 pFrom 和 pTo 中的元素。指定已存在的依赖项将返回错误。

注意

图对象不是线程安全的。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.7.5 UPTKGraphRemoveDependencies

__host__UPTKError_t UPTKGraphRemoveDependencies (UPTKGraph_t graph, const UPTKGraphNode_t from, const UPTKGraphNode_t* to, size_t numDependencies)*

从图中移除依赖边。

参数

graph - 要从中移除依赖项的图

from - 提供依赖项的节点数组

to - 依赖节点数组

numDependencies - 要移除的依赖项数量

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

要移除的依赖项数量由 numDependencies 定义。pFrom 和 pTo 中对应索引的元素定义了一个依赖项。pFrom 和 pTo 中的每个节点必须属于 graph。

如果 numDependencies 为 0，则忽略 pFrom 和 pTo 中的元素。指定不存在的依赖项将返回错误。

注意

图对象不是线程安全的。此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或

UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.7.6 UPTKGraphInstantiate

__host__UPTKError_t UPTKGraphInstantiate (UPTKGraphExec_t pGraphExec, UPTKGraph_t graph, unsigned long long flags = 0)*

从图创建可执行图。

参数

pGraphExec - 返回实例化的图

graph - 要实例化的图

flags - 控制实例化的标志。

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

将 graph 实例化为可执行图。该图将针对任何先前未验证的结构约束或节点内约束进行验证。如果实例化成功，则在 pGraphExec 中返回实例化图的句柄。

flags 参数控制实例化和后续图启动的行为。有效标志为：
UPTKGraphInstantiateFlagAutoFreeOnLaunch：配置包含内存分配节点的图，以便在图重新启动前自动释放任何未释放的内存分配。
UPTKGraphInstantiateFlagDeviceLaunch：配置图以便从设备启动。如果传递此标志，返回的可执行图句柄可用于从主机和设备启动图。此标志不能与 UPTKGraphInstantiateFlagAutoFreeOnLaunch 结合使用。
UPTKGraphInstantiateFlagUseNodePriority：导致图在执行期间使用来自每节点属性的优先级，而不是启动流的优先级。优先级仅在内核节点上可用，并且是在流捕获期间从流优先级复制的。

如果 graph 包含任何分配或释放节点，则该图最多只能存在一个可执行图。在通过 UPTKGraphExecDestroy 销毁第一个可执行图之前，尝试实例化第二个可执行图将导致错误。如果 graph 包含任何设备可更新内核节点，同样适用。

为在设备上启动而实例化的图具有不适用于主机图的额外限制： 图的节点必须驻留在单个设备上。 图只能包含内核节点、内存复制节点、内存设置节点和子图节点。操作特定的限制概述如下。 内核节点： 不允许使用 UPTK 动态并行性。 只要未使用 MPS，就允许协同启动。 内存复制节点：仅允许涉及设备内存和/或固定的设备映射主机内存的复制。 不允许涉及 UPTK 数组的复制。 两个操作数必须可从当前设备访问，并且当前设备必须与图中其他节点的设备匹配。 如果图不是为在设备上启动而实例化，但包含从多个设备调用设备端 UPTKGraphLaunch() 的内核，这将导致错误。

注意

图对象不是线程安全的。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.7.7 UPTKGraphLaunch

__host__UPTKError_t UPTKGraphLaunch (UPTKGraphExec_t graphExec, UPTKStream_t stream)

在流中启动可执行图。

参数

graphExec - 要启动的可执行图

stream - 要在其中启动图的流

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

在 stream 中执行 graphExec。一次只能执行一个 graphExec 实例。每次启动都排序在 stream 中任何先前工作以及 graphExec 的任何先前启动之后。要并发执行图，必须多次实例化到多个可执行图中。

如果 graphExec 创建的任何分配仍未释放（来自先前的启动）并且 graphExec 不是使用 UPTKGraphInstantiateFlagAutoFreeOnLaunch 实例化的，则启动将失败并返回 UPTKErrorInvalidValue。

注意

图对象不是线程安全的。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.7.8 UPTKGraphExecUpdate

__host__UPTKError_t UPTKGraphExecUpdate (UPTKGraphExec_t hGraphExec, UPTKGraph_t hGraph, UPTKGraphExecUpdateResultInfo resultInfo)*

检查可执行图是否可以使用图进行更新，并在可能的情况下执行更新。

参数

`hGraphExec` - 要更新的实例化图

`hGraph` - 包含更新参数的图

`resultInfo` - 错误信息结构

返回

UPTKSuccess, UPTKErrorGraphExecUpdateFailure,

描述

使用拓扑相同的图 `hGraph` 中的节点参数更新由 `hGraphExec` 指定的实例化图中的节点参数。

注意

图对象不是线程安全的。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 `UPTKErrorInitializationError`、`UPTKErrorInsufficientDriver` 或 `UPTKErrorNoDevice`。 正如 `UPTKStreamAddCallback` 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 `UPTKErrorNotPermitted` 作为诊断信息。

2.7.9 UPTKGraphAddNode

__host__ UPTKError_t UPTKGraphAddNode (UPTKGraphNode_t pGraphNode, UPTKGraph_t graph, const UPTKGraphNode_t* pDependencies, const UPTKGraphEdgeData* dependencyData, size_t numDependencies, UPTKGraphNodeParams* nodeParams)*

向图中添加任意类型的节点。

参数

`` `pGraphNode` `` – 返回新创建的节点

`` `graph` `` – 要添加节点的图

`` `pDependencies` `` – 节点的依赖项

`` `dependencyData` `` – 依赖项的可选边数据

`` `numDependencies` `` – 依赖项的数量

`` `nodeParams` `` – 节点的规范

返回

`UPTKSuccess, UPTKErrorInvalidValue, UPTKErrorInvalidDeviceFunction, UPTKErrorNotSupported`

描述

在 graph 中创建一个由 nodeParams 描述的新节点，具有通过 pDependencies 指定的 numDependencies 个依赖项。

2.7.10 UPTKGraphDestroyNode

__host__ UPTKError_t UPTKGraphDestroyNode (UPTKGraphNode_t node)

从图中移除节点。

参数

`node` – 要移除的节点

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

从其图中移除 node。此操作还会切断其他节点对 node 的依赖关系。

2.7.11 UPTKGraphExecDestroy

__host__ UPTKError_t UPTKGraphExecDestroy (UPTKGraphExec_t graphExec)

销毁可执行图。

参数

`graphExec` – 要销毁的可执行图

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

销毁由 graphExec 指定的可执行图。

2.7.12 UPTKGraphGetNodes

__host__ UPTKError_t UPTKGraphGetNodes (UPTKGraph_t graph, UPTKGraphNode_t nodes, size_t* numNodes)*

返回图的节点。

参数

`graph` – 要查询的图

`nodes` – 指向返回节点的指针

`numNodes` – 节点数量

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

返回 graph 的节点列表。

2.7.13 UPTKGraphNodeGetType

__host__ UPTKError_t UPTKGraphNodeGetType (UPTKGraphNode_t node, UPTKGraphNodeType pType)*

返回节点的类型。

参数

`node` – 要查询的节点

`pType` – 指向返回节点类型的指针

返回

UPTKSuccess, UPTKErrorInvalidValue

描述

返回 node 的节点类型到 pType。

2.8 调度功能接口

2.8.1 UPTKLaunchKernel

__host__UPTKError_t UPTKLaunchKernel (const void func, dim3 gridDim, dim3 blockDim, void** args, size_t sharedMem, UPTKStream_t stream)*

启动设备函数。

参数

func - 设备函数符号 gridDim - 网格维度 blockDim - 块维度 args - 参数 sharedMem - 共享内存 stream - 流标识符

返回

UPTKSuccess, UPTKErrorInvalidDeviceFunction, UPTKErrorInvalidConfiguration, UPTKErrorLaunchFailure, UPTKErrorLaunchTimeout, UPTKErrorLaunchOutOfResources, UPTKErrorSharedObjectInitFailed, UPTKErrorInvalidPtx, UPTKErrorUnsupportedPtxVersion,

UPTKErrorNoKernelImageForDevice, UPTKErrorJitCompilerNotFound, UPTKErrorJitCompilationDisabled

描述

该函数在 gridDim (gridDim.x * gridDim.y * gridDim.z) 的网格上调用内核 func。每个块包含 blockDim (blockDim.x * blockDim.y * blockDim.z) 个线程。

如果内核有 N 个参数，则 args 应指向包含 N 个指针的数组。每个指针，从 args[0] 到 args[N - 1]，指向实际参数将被复制的内存区域。

对于模板化函数，按如下方式传递函数符号：
func_name<template_arg_0,...,template_arg_N>

sharedMem 设置每个线程块可用的动态共享内存量。

stream 指定调用所关联的流。

注意

此函数使用标准默认流语义。 此函数可能还会返回来自先前异步启动的错误代码。 如果此调用尝试初始化内部 UPTK RT 状态，此函数可能还会返回 UPTKErrorInitializationError、UPTKErrorInsufficientDriver 或 UPTKErrorNoDevice。 正如 UPTKStreamAddCallback 所指定的，不能从回调中调用任何 UPTK 函数。在这种情况下，可能会（但不保证）返回 UPTKErrorNotPermitted 作为诊断信息。

2.8.2 UPTKFuncGetAttributes

__host__UPTKError_t UPTKFuncGetAttributes (UPTKFuncAttributes attr, const void* func)*

查找给定函数的属性。

参数

attr - 返回指向函数属性的指针

func - 设备函数符号

返回

UPTKSuccess, UPTKErrorInvalidDeviceFunction

描述

此函数获取通过 func 指定的函数的属性。func 是一个设备函数符号，必须声明为 __global__ 函数。获取的属性放置在 attr 中。如果指定的函数不存在，则返回 UPTKErrorInvalidDeviceFunction。对于模板化函数，按如下方式传递函数符号：
func_name<template_arg_0,...,template_arg_N>

某些函数属性（如 maxThreadsPerBlock）可能会根据当前使用的设备而变化。

注意

此函数可能还会返回来自先前异步启动的错误代码。

2.8.3 UPTKFuncSetAttribute

__host__ UPTKError_t UPTKFuncSetAttribute (T func, UPTKFuncAttribute attr, int value)*

为给定函数设置属性。

参数

`func` - 函数指针

`attr` - 要设置的属性

`value` - 要设置的值

返回

UPTKSuccess, UPTKErrorInvalidDeviceFunction, UPTKErrorInvalidValue

描述

设置通过 func 指定的函数的属性。

2.8.4 UPTKLaunchCooperativeKernelMultiDevice

__host__ UPTKError_t UPTKLaunchCooperativeKernelMultiDevice (UPTKLaunchParams launchParamsList, unsigned int numDevices, unsigned int flags = 0)*

在多个设备上启动设备函数，线程块可以在执行时协作和同步。

参数

`launchParamsList` - 每个设备的启动参数列表

`numDevices` - `launchParamsList` 数组的大小

`flags` - 控制启动行为的标志

返回

UPTKSuccess, UPTKErrorInvalidDeviceFunction,
UPTKErrorInvalidConfiguration, UPTKErrorLaunchFailure,
UPTKErrorLaunchTimeout, UPTKErrorLaunchOutOfResources,
UPTKErrorCooperativeLaunchTooLarge, UPTKErrorSharedObjectInitFailed

描述

在 launchParamsList 数组中指定的多个设备上启动内核，这些内核可以协作和同步执行。

2.9 错误信息功能接口

2.9.1 UPTKGetErrorName

__host__ __device__ const char UPTKGetErrorName (UPTKError_t error)*

返回错误代码枚举名称的字符串表示。

参数

`error` - 要转换为字符串的错误代码

返回

指向以 NULL 结尾的字符串的 char* 指针

描述

返回包含枚举中错误代码名称的字符串。

2.9.2 UPTKGetErrorString

__host__ __device__ const char UPTKGetErrorString (UPTKError_t error)*

返回错误代码的描述字符串。

参数

`error` - 要转换为字符串的错误代码

返回

指向以 NULL 结尾的字符串的 char* 指针

描述

返回错误代码的描述字符串。

2.9.3 UPTKGetLastError

__host__ __device__ UPTKError_t UPTKGetLastError (void)

返回运行时调用的最后一个错误。

返回

UPTKSuccess, UPTKErrorMissingConfiguration, UPTKErrorMemoryAllocation,
UPTKErrorInitializationError, UPTKErrorLaunchFailure,
UPTKErrorLaunchTimeout, UPTKErrorLaunchOutOfResources,
UPTKErrorInvalidDeviceFunction, UPTKErrorInvalidConfiguration,
UPTKErrorInvalidDevice, UPTKErrorInvalidValue,
UPTKErrorInvalidPitchValue, UPTKErrorInvalidSymbol,
UPTKErrorUnmapBufferObjectFailed, UPTKErrorInvalidDevicePointer,
UPTKErrorInvalidTexture, UPTKErrorInvalidTextureBinding,

UPTKErrorInvalidChannelDescriptor, UPTKErrorInvalidMemcpyDirection,
UPTKErrorInvalidFilterSetting, UPTKErrorInvalidNormSetting,
UPTKErrorUnknown, UPTKErrorInvalidResourceHandle,
UPTKErrorInsufficientDriver, UPTKErrorNoDevice,
UPTKErrorSetOnActiveProcess, UPTKErrorStartupFailure,
UPTKErrorInvalidPtx, UPTKErrorUnsupportedPtxVersion,
UPTKErrorNoKernelImageForDevice, UPTKErrorJitCompilerNotFound,
UPTKErrorJitCompilationDisabled

描述

返回同一主机线程中 UPTK 运行时库实例中任何运行时调用产生的最后一个错误，并将其重置为 UPTKSuccess。