NVIDIA CUDA
计 算 统 一 设 备 架 构
Device Architecture

参 考 手 册

Version 2.0

2 0 0 8 年 6 月

目录

1 RuntimeApiReference

名称

运行时API参考

说明

有两种级别的运行时API。

低级API（cuda_runtime_api.h）是一种C风格的接口，不需要使用nvcc进行编译。

高级API（cuda_runtime.h）是一种C++风格的接口，构建于低级API之上。它包装了部分低级API例程，使用过载、参考和默认参数。这些包装器可以通过C++代码使用，也可使用任何C++编译器进行编译。高级API还具有一些特定于CUDA的包装器，包装处理符号、纹理和设备功能的低级例程。这些包装器需要使用nvcc，因为它们依赖于编译器生成的代码。例如，只有使用nvcc编译的源代码才可以执行配置语法来调用内核。

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.1 DeviceManagement RT

名称
设备管理

说明

本节描述CUDA运行时应用程序编程接口。

cudaGetDeviceCount
cudaSetDevice
cudaGetDevice
cudaGetDeviceProperties
cudaChooseDevice

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.1.1 cudaGetDeviceCount

名称

cudaGetDeviceCount – 返回具有计算能力的设备的数量

概要

cudaError_t cudaGetDeviceCount(int* count)

说明

以*count形式返回可用于执行的计算能力大于等于1.0的设备数量。如果不存在此类设备，cudaGetDeviceCount ()将返回1，且设备0仅支持设备模拟模式。由于此设备能够模拟所有硬件特性，因此该设备将报告9999种主要和次要计算能力。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGetDevice，cudaSetDevice，cudaGetDeviceProperties，cudaChooseDevice

1.1.2 cudaSetDevice

名称
cudaSetDevice – 设置设备以供GPU执行使用

概要
cudaError_t cudaSetDevice(int dev)

说明
将dev记录为活动主线程将执行设备码的设备。

返回值
相关返回值：
cudaSuccess
cudaErrorInvalidDevice

注意，如果之前是异步启动，该函数可能返回错误码。

参见
cudaGetDeviceCount，cudaGetDevice，cudaGetDeviceProperties，cudaChooseDevice

1.1.3 cudaGetDevice

名称
cudaGetDevice – 返回当前使用的设备

概要
cudaError_t cudaGetDevice(int *dev)

说明
以*dev形式返回活动主线程执行设备码的设备。

返回值
相关返回值
cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见
cudaGetDeviceCount，cudaSetDevice，cudaGetDeviceProperties，cudaChooseDevice

1.1.4 cudaGetDeviceProperties

名称

cudaGetDeviceProperties – 返回关于计算设备的信息

概要

cudaError_t cudaGetDeviceProperties(struct cudaDeviceProp* prop，int dev)

说明

以*prop形式返回设备dev的属性。cudaDeviceProp结构定义如下：

```
struct cudaDeviceProp {
    char name [256];
    size_t totalGlobalMem;
    size_t sharedMemPerBlock;
    int regsPerBlock;
    int warpSize;
    size_t memPitch;
    int maxThreadsPerBlock;
    int maxThreadsDim [3];
    int maxGridSize [3];
    size_t totalConstMem;
    int major;
    int minor;
    int clockRate;
    size_t textureAlignment;
    int deviceOverlap;
    int multiProcessorCount;
}
```

其中：

name

用于标识设备的ASCII字符串；

totalGlobalMem

设备上可用的全局存储器的总量，以字节为单位；

sharedMemPerBlock

线程块可以使用的共享存储器的最大值，以字节为单位；多处理器上的所有线程块可以同时共享这些存储器；

regsPerBlock

线程块可以使用的32位寄存器的最大值；多处理器上的所有线程块可以同时共享这些寄存器；

warpSize

按线程计算的warp块大小；

memPitch

允许通过cudaMallocPitch()为包含存储器区域的存储器复制函数分配的最大间距（pitch），以字节为单位；

maxThreadsPerBlock

每个块中的最大线程数：

maxThreadsDim[3]

块各个维度的最大值：

maxGridSize[3]

网格各个维度的最大值；

totalConstMem

设备上可用的不变存储器总量，以字节为单位；

major，minor

定义设备计算能力的主要修订号和次要修订号；

clockRate

以千赫为单位的时钟频率；

textureAlignment

对齐要求；与textureAlignment字节对齐的纹理基址无需对纹理取样应用偏移；

deviceOverlap

如果设备可在主机和设备之间并发复制存储器，同时又能执行内核，则此值为 1；否则此值为 0；

multiProcessorCount

设备上多处理器的数量。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidDevice

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGetDevice Count，cuda GetDevice，cudaSetDevice，cuda ChooseDevice

1.1.5 cudaChooseDevice

名称

cudaChooseDevice – 选择最匹配标准的计算设备

概要

cudaError_t cudaChooseDevice(int* dev，const struct cudaDeviceProp* prop)

说明

以*dev的形式返回属性与*prop的匹配程度最高的设备。

返回值

相关返回值：

cudaSuccess cudaErrorInvalid Value

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGetDevice Count，cuda GetDevice，cudaSetDevice，cuda GetDeviceProperties

1.2 ThreadManagement RT

名称

线程管理

说明

本节描述CUDA运行时应用程序编程接口。

cuda ThreadSynchronize

cuda ThreadExit

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.2.1 cudaThreadSynchronize

名称

cudaThreadSynchronize – 等待计算设备完成

概要

cudaError_t cudaThreadSynchronize (void)

说明

在设备完成所有之前请求的任务之前，一直阻塞操作。如果之前的任务失败，cudaThreadSynchronize()将返回一个错误。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaThreadExit

1.2.2 cudaThreadExit

名称

cudaThreadExit – 从CUDA启动中退出并清除

概要

cudaError_t cudaThreadExit (void)

说明

显式清除与调用主线程有关的运行时相关资源。后续的任何API调用都将重新初始化运行时。在主线程退出时，将隐式调用cudaThreadExit ()。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaThreadSynchronize

1.3 StreamManagement RT

名称

流管理

说明

本节描述CUDA运行时应用程序编程接口。

cudaStreamCreate

cudaStreamQuery

cudaStreamSynchronize

cudaStreamDestroy

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.3.1 cudaStreamCreate

名称

cudaStreamCreate – 创建异步流

概要

cudaError_t cudaStreamCreate(cudaStream_t* stream)

说明

创建流。

返回值

相关返回值：

cudaSuccess cudaErrorInvalid Value

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaSt ream Query，cudaStreamSynchronize，cudaStreamDestroy

1.3.2 cudaStreamQuery

名称

cudaStreamQuery – 查询流的完成状态

概要

cudaError_t cudaStreamQuery (cudaStream_t stream)

说明

如果流中的所有操作均已完成，则返回cudaSuccess，否则返回cudaErrorNotReady。

返回值

相关返回值：

cudaSuccess

cudaErrorNotReady

cudaErrorInvalidResourceHandle

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaSt ream Create，cudaStreamDestroy，cudaStreamSynchronize

1.3.3 cudaStreamSynchronize

名称

cudaStreamSynchronize – 等待流任务完成

概要

cudaError_t cudaStreamSynchronize (cudaStream_t stream)

说明

在设备完成流中的所有操作之前，一直阻塞操作。

返回值

相关返回值：

cudaSuccess cudaErrorInvalidResourceHandle

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaSt ream Create，cudaStreamDestroy，cudaStream Query

1.3.4 cudaStreamDestroy

名称

cudaStreamDestroy – 销毁并清除流对象。

概要

cudaError_t cudaStreamDestroy (cudaStream_t stream)

说明

销毁流对象。

返回值

相关返回值：

cudaSuccess cudaErrorInvalidResourceHandle

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaSt ream Create，cudaStreamSychronize，cudaStreamDestroy

1.4 EventManagement RT

名称

事件管理

说明

本节描述CUDA运行时应用程序编程接口。

cudaEventCreate

cudaEventRecord

cudaEventQuery

cudaEventSynchronize

cudaEventDestroy

cudaEventElapsedTime

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.4.1 cudaEventCreate

名称

cudaEventCreate – 创建事件对象

概要

cudaError_t cudaEventCreate(cudaEvent_t* event)

说明

创建事件对象

返回值

相关返回值：

cudaSuccess
cudaErrorInitializationError
cudaErrorPriorLaunchFailure cudaErrorInvalid Value

cudaErrorMemoryAllocation

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaEventRecord，cudaEvent Query，cudaEventSynchronzie，cudaEventDestroy，cudaEventElapsed Time

1.4.2 cudaEventRecord

名称

cudaEventRecord – 记录事件

概要

cudaError_t cudaEventRecord(cudaEvent_t event，CUstream stream)

说明

记录事件。如果流非零，则在完成流中所有先前的操作之后记录事件；否则，将在完成CUDA上下文中所有先前的操作之后记录事件。由于此操作是异步的，因而必须使用cudaEventQuery()和/或cudaEventSynchronize()确定事件的实际记录时间。

如果之前已调用过cudaEventRecord()，而尚未记录事件，则此函数将返回cudaErrorInvalidValue。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalidResourceHandle

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaEvent Create，cudaEvent Query，cudaEventSynchronize，cudaEventDestroy，cudaEventElapsed Time

1.4.3 cudaEventQuery

名称

cudaEventQuery – 查询是否已经记录了事件

概要

cudaError_t cudaEventQuery(cudaEvent_t event)

说明

如果确实已经记录了事件，则返回cudaSuccess；如果尚未记录，则返回cudaErrorNotReady。如果尚未对此事件调用cudaEventRecord()，该函数将返回cudaErrorInvalidValue。

返回值

相关返回值：

cudaSuccess

cudaErrorNotReady

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalidValue

cudaErrorInvalidResourceHandle

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaEvent Create，cudaEventRecord，cudaEventSynchronize，cudaEventDestroy，cudaEventElapsed Time

1.4.4 cudaEventSynchronize

名称

cudaEventSynchronize – 等待事件被记录

概要

cudaError_t cudaEventSynchronize(cudaEvent_t event)

说明

在实际记录事件之前，一直阻塞操作。如果尚未为此事件调用cudaEventRecord()，该函数将返回cudaErrorInvalidValue。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError
cudaErrorPriorLaunchFailure cudaErrorInvalid
Value

cudaErrorInvalidResourceHandle

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaEventCreate，cudaEventRecord，cudaEvent Query，cudaEventDestroy，cudaEventElapsedTime

1.4.5 cudaEventDestroy

名称

cudaEventDestroy – 销毁事件对象

概要

cudaError_t cudaEventDestroy(cudaEvent_t event)

说明

销毁事件对象。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError

cudaErrorPriorLaunchFailure cudaErrorInvalidValue

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaEventCreate，cudaEventQuery，cudaEventSynchronize，cudaEventRecord，cudaEventElapsedTime

1.4.6 cudaEventElapsedTime

名称

cudaEventElapsedTime – 计算两次事件之间相差的时间

概要

cudaError_t cudaEventElapsedTime(float* time，cudaEvent_t start，cudaEvent_t end);

说明

计算两次事件之间相差的时间（以毫秒为单位，精度为0.5微秒）。如果尚未记录其中任何一个事件，此函数将返回cudaErrorInvalidValue。如果记录其中任何一个事件使用了非零流，则结果不确定。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalidValue

cudaErrorInvalidResourceHandle

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaEvent Create，cudaEvent Query，cudaEventSynchronize，cudaEventDestroy，cudaEventRecord

1.5 MemoryManagement RT

名称

存储器管理

说明

本节描述CUDA运行时应用程序编程接口。

cudaMalloc

cudaMallocPitch

cudaFree

cudaMallocArray

cudaFreeArray

cudaMallocHost

cudaFreeHost

cudaMemset

cudaMemset2D

cudaMemcpy

cudaMemcpy2D

cudaMemcpyToArray

cudaMemcpy2DToArray

cudaMemcpyFromArray

cudaMemcpy2DFromArray

cudaMemcpyArrayToArray

cudaMemcpy2DArrayToArray

cudaMemcpyToSymbol

cudaMemcpyFromSymbol

cudaGetSymbolAddress

cudaGetSymbolSize

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.5.1 cudaMalloc

名称

cudaMalloc – 在GPU上分配存储器

概要

cudaError_t cudaMalloc(void** devPtr，size_t count)

说明

向设备分配 count 字节的线性存储器，并以*devPtr的形式返回指向所分配存储器的指针。可针对任何类型的变量合理调整所分配的存储器。存储器不会被清除。如果出现错误，cudaMalloc()将返回cudaErrorMemoryAllocation。

返回值

相关返回值：

cudaSuccess cudaErrorMemoryAllocation

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMallocPitch，cudaFree，cudaMallocArray，cudaFreeArray，cudaMallocHost，cudaFreeHost

1.5.2 cudaMallocPitch

名称

cudaMallocPitch – 向GPU分配存储器

概要

cudaError_t cudaMallocPitch(void** devPtr，size_t* pitch，size_t widthInBytes，size_t height)

说明

向设备分配至少widthInBytes*height字节的线性存储器，并以*devPtr的形式返回指向所分配存储器的指针。该函数可以填充所分配的存储器，以确保在地址从一行更新到另一行时，给定行的对应指针依然满足对齐要求。cudaMallocPitch()以*pitch的形式返回间距，即所分配存储器的宽度，以字节为单位。间距用作存储器分配的一个独立参数，用于在2D数组内计算地址。如果给定一个T类型数组元素的行和列，可按如下方法计算地址：

```
T* pElement = (T*)((char*)BaseAddress + Row * pitch) + Column;
```

对于2D数组的分配，建议程序员考虑使用cudaMallocPitch()来执行间距分配。由于硬件中存在间距对齐限制，如果应用程序将在设备存储器的不同区域之间执行2D存储器复制（无论是线性存储器还是CUDA数组），这种方法将非常有用。

返回值

相关返回值：

cudaSuccess cudaErrorMemoryAllocation

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMalloc，cudaFree，cudaMallocArray，cudaFreeArray，cudaMallocHost，cudaFreeHost

1.5.3 cudaFree

名称

cudaFree – 释放GPU上的存储器

概要

cudaError_t cudaFree (void* devPtr)

说明

释放devPtr（必须在之前调用cudaMalloc()或cudaMallocPitch()时返回）指向的存储器空间。如果未返回或者之前已经调用过cudaFree(devPtr)，则返回一个错误。如果devPtr为0，则不执行任何操作。如果出现错误，cudaFree()将返回cudaErrorInvalid-DevicePointer。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidDevicePointer

cudaErrorInitializationError

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMalloc，cudaMallocPitch，cudaMallocArray，cudaFreeArray，cudaMallocHost，cudaFreeHost

1.5.4 cudaMallocArray

名称

cudaMallocArray – 向GPU分配数组

概要

cudaError_t cudaMallocArray(struct cudaArray** array，const struct cudaChannelFormatDesc* desc，size_t width，size_t height)

说明

根据cudaChannelFormatDesc结构desc分配CUDA数组，以*array的形式返回新CUDA数组的句柄。
cudaChannelFormatDesc定义如下：

```
struct cudaChannelFormatDesc { int x，y，z，
    w;
    enum cudaChannelFormatKind f; };
```

其中cudaChannelFormatKind是cudaChannelFormatKindSigned、cudaChannelFormatKindUnsigned或cudaChannelFormatKindFloat之一。

返回值

相关返回值：

cudaSuccess cudaErrorMemoryAllocation

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMalloc，cudaMallocPitch，cudaFree，cudaFreeArray，cudaMallocHost，cudaFreeHost

1.5.5 cudaFreeArray

名称

cudaFreeArray – 释放GPU上的数组

概要

cudaError_t cudaFreeArray(struct cudaArray* array)

说明

释放CUDA数组array。如果array为0，则不执行任何操作。

返回值

相关返回值：

cudaSuccess cudaErrorInitializationError

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMalloc，cudaMallocPitch，cudaFree，cudaMallocArray，cudaMallocHost，cudaFreeHost

1.5.6 cudaMallocHost

名称

cudaMallocHost – 向主机分配分页锁定的存储器

概要

cudaError_t cudaMallocHost(void** hostPtr，size_t size)

说明

分配size字节大小的分页锁定且设备可访问的主存储器。驱动程序会跟踪由此函数分配的虚拟存储器范围，自动加速对cudaMemcpy*()等函数的调用。由于设备可直接访问存储器，因而与malloc()等函数分配的可分页存储器相比，这种存储器在读取或写入时的带宽更高。使用cudaMallocHost()分配过多存储器可能导致系统性能降低，因为这会减少系统可用于分页的存储器数量。因而，最好少使用此函数，一般只用于为主机和设备之间的数据交换分配存储器。

返回值

相关返回值：

cudaSuccess cudaErrorMemoryAllocation

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMalloc，cudaMallocPitch，cudaFree，cudaMallocArray，cudaFreeArray，cudaFreeHost

1.5.7 cudaFreeHost

名称

cudaFreeHost – 释放分页锁定的存储器

概要

cudaError_t cudaFreeHost(void* hostPtr)

说明

释放hostPtr指向的存储器空间，之前的cudaMallocHost()调用必须返回hostPtr。

返回值

相关返回值：

cudaSuccess cudaErrorInitializationError

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMalloc，cudaMallocPitch，cudaFree，cudaMallocArray，cudaFreeArray，cudaMallocHost

1. NAME
5. 1.5.9 cudaMemset2D
8
cu cudaMemset2D - initializes or sets GPU memory to a value
da
Memset

名称

cudaMemset – 初始化（设置）GPU存储器的值

概要

cudaError_t cudaMemset(void* devPtr，int value，size_t count)

说明

使用固定字节值value来填充devPtr所指向存储器区域的前count个字节。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalid Value

cudaErrorInvalidDevicePointer

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemset2D，cudaMemset3D

1.5.9 cudaMemset2D

名称

cudaMemset2D -初始化（设置）GPU存储器的值

说明

将dstPtr指向的矩阵（共有height行，各行width字节）设置为指定值value。pitch是dstPtr指向的2D数组在存储器中的宽度（以字节为单位），其中包括添加到各行末尾的填充符。在cudaMallocPitch()已经传回所使用的间距时，此函数的执行速度最快。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalid Value

cudaErrorInvalidDevicePointer

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemset，cudaMemset3D

1.5.10 cudaMemcpy

名称

cudaMemcpy – 在GPU和主机之间复制数据

概要

cudaError_t cudaMemcpy(void* dst，const void* src，size_t count，enum cudaMemcpyKind kind)
cudaError_t cudaMemcpyAsync(void* dst，const void* src，size_t count，enum cudaMemcpyKind kind，cudaStream_t stream)

说明

从src指向的存储器区域中将count个字节复制到dst指向的存储器区域，其中kind是cudaMemcpyHostToHost、cudaMemcpyHostToDevice、cudaMemcpyDeviceToHost或cudaMemcpyDeviceToDevice之一，用于指定复制的方向。存储器区域不可重叠。调用cudaMemcpy()时，如果dst和src指针与复制的方向不匹配，则将导致不确定的行为。

cudaMemcpyAsync()是异步的，可选择传入非零流参数，从而将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy2D，cudaMemcpyToArray，cudaMemcpy2D ToArray，cudaMemcpyFromArray，cudaMemcpy2DFromArray，cudaMemcpyArray ToArray，cudaMemcpy2DArrayToArray，cudaMemcpyToSymbol，cudaMemcpyFromSymbol

1.5.11 cudaMemcpy2D

名称

cudaMemcpy2D – 在主机和设备之间复制数据

概要

cudaError_t cudaMemcpy2D(void* dst，size_t dpitch，const void* src，size_t spitch，size_t width，size_t height，enum cudaMemcpyKind kind)

cudaError_t cudaMemcpy2DAsync(void* dst，size_t dpitch，const void* src，size_t spitch，size_t width，size_t height，enum cudaMemcpyKind kind，cudaStream_t stream)

说明

从src指向的存储器区域中将一个矩阵（共height行，每行width字节）复制到dst指向的存储器区域，其中kind是cudaMemcpyHostToHost、cudaMemcpyHostToDevice、cudaMemcpyDeviceToHost或cudaMemcpyDeviceToDevice之一，用于指定复制的方向。Dpitch和spitch是dst和src指向的2D数组在存储器中的宽度（以字节为单位），其中包括添加到各行末尾的填充符。存储器区域不可重叠。调用cudaMemcpy2D()时，如果dst和src指针与复制方向不匹配，则将导致不确定的行为。如果dpitch或spitch超过允许的最大值，cudaMemcpy2D()将返回一个错误。

cudaMemcpy2DAsync()是异步的，可选择传入非零流参数，从而将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidPitchValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy，cudaMemcpyToArray，cudaMemcpy2DToArray，cudaMemcpyFromArray，cudaMemcpy2DFromArray，cudaMemcpyArray ToArray，cudaMemcpy2DArrayToArray，cudaMemcpyToSymbol，cudaMemcpyFromSymbol

1.5.12 cudaMemcpyToArray

名称

cudaMemcpyToArray – 在主机和设备间复制数据

概要

cudaError_t cudaMemcpyToArray(struct cudaArray* dstArray，size_t dstX，size_t dstY，const void* src，size_t count，enum cudaMemcpyKind kind)

cudaError_t cudaMemcpyToArrayAsync(struct cudaArray* dstArray，size_t dstX，size_t dstY，const void* src，size_t count，enum cudaMemcpyKind kind，cudaStream_t stream)

说明

从src指向的存储器区域内将count个字节复制到一个CUDA数组dstArray，该数组的左上角从(dstX，dstY)开始，其中kind是cudaMemcpyHostToHost、cudaMemcpyHost-ToDevice、cudaMemcpyDeviceToHost或cudaMemcpyDeviceToDevice之一，用于指定复制的方向。

cudaMemcpyToArrayAsync()是异步的，可选择传入非零流参数，从而将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy，cudaMemcpy2D，cudaMemcpy2DToArray，cudaMemcpyFromArray，cudaMemcpy2DFromArray，cudaMemcpyArray ToArray，cudaMemcpy2DArrayToArray，cudaMemcpyToSymbol，cudaMemcpyFromSymbol

1.5.13 cudaMemcpy2DToArray

名称

cudaMemcpy2DToArray – 在主机和设备间复制数据

概要

cudaError_t cudaMemcpy2DToArray(struct cudaArray* dstArray，size_t dstX，size_t dstY，const void* src，size_t spitch，size_t width，size_t height，enum cudaMemcpyKind kind); cudaError_t cudaMemcpy2DToArrayAsync(struct cudaArray* dstArray，size_t dstX，size_t dstY，const void*
src，size_t spitch，size_t width，size_t height，enum cudaMemcpyKind kind，cudaStream_t stream);

说明

从src指向的存储器区域内将一个矩阵（共height行，每行width字节）复制到CUDA数组dstArray，该数组的左上角从(dstX，dstY)开始，其中kind是cudaMemcpyHostToHost、cudaMemcpyHost-ToDevice、cudaMemcpyDeviceToHost或cudaMemcpyDeviceToDevice之一，用于指定复制的方向。spitch是src指向的2D数组在存储器中的宽度（以字节为单位），其中包括添加到各行末尾的填充符。如果spitch超过允许的最大值，cudaMemcpy2D()将返回一个错误。

cudaMemcpy2DToArrayAsync()是异步的，可选择传入非零流参数，从而将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidPitchValue

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy，cudaMemcpy2D，cudaMemcpy ToArray，cudaMemcpyFromArray，cudaMemcpy2DFromArray，cudaMemcpyArray ToArray，cudaMemcpy2DArrayToArray，cudaMemcpyToSymbol，cudaMemcpyFromSymbol

1.5.14 cudaMemcpyFromArray

名称

cudaMemcpyFromArray – 在主机和设备间复制数据

概要

cudaError_t cudaMemcpyFromArray(void* dst，const struct cudaArray* srcArray，size_t srcX，size_t srcY，size_t count，enum cudaMemcpyKind kind)

cudaError_t cudaMemcpyFromArrayAsync(void* dst，const struct cudaArray* srcArray，size_t srcX，size_t srcY，size_t count，enum cudaMemcpyKind kind，cudaStream_t stream)

说明

从一个CUDA数组dstArray（左上角以(srcX，srcY)开始）内将count个字节复制到dst指向的存储器区域。其中kind是 cudaMemcpyHostToHost 、 cudaMemcpyHost-ToDevice 、 cudaMemcpyDeviceToHost 或cudaMemcpyDeviceToDevice之一，用于指定复制的方向。

cudaMemcpyFromArrayAsync()是异步的，可选择传入非零流参数，从而将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy，cudaMemcpy2D，cudaMemcpyToArray，cudaMemcpy2D ToArray，cudaMemcpy2DFromArray，cudaMemcpyArray ToArray，cudaMemcpy2DArrayToArray，cudaMemcpyToSymbol，cudaMemcpyFromSymbol

1.5.15 cudaMemcpy2DFromArray

名称

cudaMemcpy2DFromArray – 在主机和设备间复制数据

概要

cudaError_t cudaMemcpy2DFromArray(void* dst，size_t dpitch，const struct cudaArray* srcArray，size_t srcX，size_t srcY，size_t width，size_t height，enum cudaMemcpyKind kind)

cudaError_t cudaMemcpy2DFromArrayAsync(void* dst，size_t dpitch，const struct cudaArray* srcArray，size_t srcX，size_t srcY，size_t width，size_t height，enum cudaMemcpyKind kind，cudaStream_t stream)

说明

从一个CUDA数组dstArray（左上角以(srcX，srcY)开始）内将一个矩阵（共height行，每行width字节）复制到dst指向的存储器区域，其中kind是cudaMemcpyHost-ToHost、cudaMemcpyHostToDevice、cudaMemcpyDeviceToHost或cudaMemcpyDeviceToDevice之一，用于指定复制的方向。dpitch是dst指向的2D数组在存储器中的宽度（以字节为单位），其中包括添加到各行末尾的填充符。如果dpitch超过允许的最大值，cudaMemcpy2D()将返回一个错误。

cudaMemcpy2DFromArrayAsync()是异步的，可选择传入非零流参数，从而将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidPitchValue

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy，cudaMemcpy2D，cudaMemcpy ToArray，cudaMemcpy2D ToArray，cudaMemcpyFromArray，cudaMemcpyArrayToArray，cudaMemcpy2DArrayToArray，cudaMemcpyToSymbol，cudaMemcpyFromSymbol

1. NAME
5. 1.5.17 cudaMemcpy2DArrayToArray
16
cu cudaMemcpy2DArrayToArray - copies data between host and device
da
MemcpyArrayToArray

名称

cudaMemcpyArrayToArray – 在主机和设备间复制数据

概要

cudaError_t cudaMemcpyArrayToArray(struct cudaArray* dstArray，size_t dstX，size_t dstY，const struct cudaArray* srcArray，size_t srcX，size_t srcY，size_t count，enum cudaMemcpyKind kind)

说明

从一个CUDA数组dstArray（左上角以(srcX，srcY)开始）内将count字节复制到另一个CUDA数组dstArray（左上角以(dstX，dstY)开始），其中kind是cudaMemcpyHostToHost、cudaMemcpyHostToDevice、cudaMemcpyDeviceToHost或cudaMemcpyDeviceToDevice之一，用于指定复制的方向。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy，cudaMemcpy2D，cudaMemcpy ToArray，cudaMemcpy2D ToArray，cudaMemcpyFromArray，cudaMemcpy2DFromArray，cudaMemcpy2DArray ToArray，cudaMemcpyToSymbol，cudaMemcpyFromSymbol

1.5.17 cudaMemcpy2DArrayToArray

名称

cudaMemcpy2DArrayToArray – 在主机和设备间复制数据

概要

cudaError_t cudaMemcpy2DArrayToArray(struct cudaArray* dstArray，size_t dstX，size_t dstY，const struct cudaArray* srcArray，size_t srcX，size_t srcY，size_t width，size_t height，enum cudaMemcpyKind kind)

说明

从一个CUDA数组dstArray（左上角以(srcX，srcY)开始）内将一个矩阵（共height行，每行width字节）复制到另一个CUDA数组dstArray（左上角以(dstX，dstY)开始），其中kind是cudaMemcpyHostToHost、cudaMemcpyHostToDevice、cudaMemcpyDeviceToHost或cudaMemcpyDeviceToDevice之一，用于指定复制的方向。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy，cudaMemcpy2D，cudaMemcpy ToArray，cudaMemcpy2D ToArray，cudaMemcpyFromArray，cudaMemcpy2DFromArray，cudaMemcpyArrayToArray，cudaMemcpy ToSymbol，cudaMemcpyFromSymbol

1.5.18 cudaMemcpyToSymbol

名称

cudaMemcpyToSymbol – 将主存储器的数据复制到GPU

概要

template < class T >

cudaError_t cudaMemcpyToSymbol(const T& symbol，const void* src，size_t count，size_t offset，enum cudaMemcpyKind kind)

说明

从src指向的存储器区域内将count个字节复制到offset字节所指向的存储器区域（从symbol符号开始）。存储器区域不可重叠。Symbol可以是位于全局存储器或不变存储器空间内的变量，也可以是一个指定全局存储器或不变存储器空间变量的字符串。Kind可以是cudaMemcpyHostToDevice或cudaMemcpyDevi ceToDevice。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidSymbol

cudaErrorInvalidDevicePointer

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy，cudaMemcpy2D，cudaMemcpy ToArray，cudaMemcpy2D ToArray，cudaMemcpyFromArray，cudaMemcpy2DFromArray，cudaMemcpyArrayToArray，cudaMemcpy2DArrayToArray，cudaMemcpyFromSymbol

1.5.19 cudaMemcpyFromSymbol

名称

cudaMemcpyFromSymbol – 将GPU的数据复制到主存储器

概要

template < class T >

cudaError_t cudaMemcpyFromSymbol(void *dst，const T& symbol，size_t count，size_t offset，enum cudaMemcpyKind kind)

说明

从offset字节所指向的存储器区域内（从symbol符号开始）将count个字节复制到dst指向的存储器区域。存储器区域不可重叠。Symbol可以是位于全局存储器或不变存储器空间内的变量，也可以是一个指定全局存储器或不变存储器空间变量的字符串。Kind可以是cudaMemcpyDeviceToHost或cudaMemcpyDevi ceToDevice。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidSymbol

cudaErrorInvalidDevicePointer

cudaErrorInvalidMemcpyDirection

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemcpy， cudaMemcpy2D， cudaMemcpy ToArray， cudaMemcpy2D ToArray， cudaMemcpyFromArray， cudaMemcpy2DFromArray，cudaMemcpyArrayToArray，cudaMemcpy2DArrayToArray，cudaMemcpyToSymbol

1.5.20 cudaGetSymbolAddress

名称

cudaGetSymbolAddress – 查找与CUDA符号关联的地址

概要

template < class T >

cudaError_t cudaGetSymbolAddress(void** devPtr，const T& symbol)

说明

以*devPtr的形式返回符号symbol在设备上的地址。Symbol可以是位于全局存储器或不变存储器空间内的变量，也可以是一个指定全局存储器或不变存储器空间变量的字符串。如果无法找到symbol，或未在全局存储器空间内声明symbol，*devPtr将保持不变，并返回一个错误。如果出现错误，cudaGetSymbolAddress()将返回cudaErrorInvalidSymbol。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidSymbol

cudaErrorAddressOfConstant

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGetSymbolSize

1.5.21 cudaGetSymbolSize

名称

cudaGetSymbolSize – 查找与CUDA符号关联的对象的大小

概要

template < class T >

cudaError_t cudaGetSymbolSize(size_t* size，const T& symbol)

说明

以*size的形式返回符号symbol的大小。Symbol可以是位于全局存储器或不变存储器空间内的变量，也可以是一个指定全局存储器或不变存储器空间变量的字符串。如果无法找到symbol，或未在全局或不变存储器空间内声明symbol，*size将保持不变，并返回一个错误。如果出现错误，cudaGetSymbolSize()将返回cudaErrorInvalidSymbol。

返回值

相关返回值：

cudaSuccess cudaErrorInvalidSymbol

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGetSymbolAddress

1.5.22 cudaMalloc3D

名称

cudaMalloc3D – 向GPU分配逻辑1D、2D或3D存储器对象

概要

```
struct cudaPitchedPtr {
    void *ptr;
    size_t   pitch;   size_t   xsize;
    size_t ysize;
};

struct cudaExtent { size_t width;
    size_t height; size_t depth;
};
```

cudaError_t cudaMalloc3D(struct cudaPitchedPtr* pitchDevPtr，struct cudaExtent extent)

说明

向设备分配至少width*height*depth个字节的线性存储器，并返回pitchedDevPtr，其中ptr是指向已分配存储器的指针。该函数可填充所分配的存储器，确保满足硬件对齐要求。pitchedDevPtr的pitch字段中返回的间距是所分配存储器的宽度，以字节为单位。

返回的cudaPitchedPtr包含额外的字段xsize和ysize，是所分配存储器的逻辑宽度和高度，等于程序员在分配期间提供的width和height范围参数。

对于2D或3D对象的分配，建议程序员使用cudaMalloc3D()或cudaMallocPitch()来执行分配。由于硬件中存在对齐限制，如果应用程序将执行涉及2D或3D对象的存储器复制（无论是线性存储器还是CUDA数组），这种方法将非常有用。

返回值

相关返回值：

cudaSuccess

cudaErrorMemoryAllocation

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMallocPitch，cudaFree，cudaMemcpy3D，cudaMemset3D，cudaMalloc3DArray，cudaMallocArray，cudaFreeArray，cudaMallocHost，cudaFreeHost

1.5.23 cudaMalloc3DArray

名称

cudaMalloc3DArray – 向GPU分配一个数组

概要

struct cudaExtent { size_t width; size_t height; size_t depth; };

cudaError_t cudaMalloc3DArray(struct cudaArray** arrayPtr，const struct cudaChannelFormatDesc* desc，struct cudaExtent extent)

说明

根据cudaChannelFormatDesc结构desc分配一个CUDA数组，并以*arrayPtr的形式返回新CUDA数组的句柄。cudaChannelFormatDesc定义如下：

```
struct cudaChannelFormatDesc { int x，y，
    z，w;
    enum cudaChannelFormatKind f; };
```

其中cudaChannelFormatKind是cudaChannelFormatKindSigned、cudaChannelFormatKindUnsigned或cudaChannelFormatKindFloat之一。

cudaMalloc3Darray能够分配1D、2D或3D数组。

- 如果height和depth范围都是0，则分配1D数组。对于1D数组而言，有效范围是{(1，8192)，0，0}。
- 如果仅有depth范围是0，则分配2D数组。对于2D数组而言，有效范围是{(1，65536)，(1，32768)，0}。
- 如果全部三项范围均非0，则分配3D数组。对于3D数组而言，有效范围是{(1，2048)，(1，2048)，(1，2048)}。

注意：原因在于区分范围限制有助于利用更高维度的退化数组（未用维度采用同样的设置）。例如，退化2D数组比1D数组支持的线性存储空间更多。

返回值

相关返回值：

cudaSuccess

cudaErrorMemoryAllocation

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMalloc3D，cudaMalloc，cudaMallocPitch，cudaFree，cudaFreeArray，cudaMallocHost，cudaFreeHost

1.5.24 cudaMemset3D

名称

cudaMemset3D – 初始化（设置）GPU存储器的值

概要

```
struct cudaPitchedPtr {
    void *ptr;
    size_t   pitch;   size_t
    xsize; size_t ysize;
};

struct cudaExtent { size_t width;
    size_t height; size_t
    depth;
};
```

cudaError_t cudaMemset3D(struct cudaPitchedPtr dstPitchPtr，int value，struct cudaExtent extent)

说明

初始化3D数组的各元素，将其设置为特定值value。dstPitchPtr定义了需要初始化的对象。dstPitchPtr的pitch字段是dstPitchPtr指向的3D数组在存储器中的宽度（以字节为单位），其中包括添加到各行末尾的填充符。xsize字段指定各行的逻辑宽度，以字节为单位；zsize字段指定2D片段的高度，以行为单位。

被初始化的区域的范围指定如下：宽度为width字节；高度为height行；深度为depth片。

如果区域的width大于等于dstPitchPtr的xsize，其执行速度将远远超过宽度小于xsize的区域。其次，如果区域的height等于dstPitchPtr的ysize，其执行速度将超过高度小于ysize的区域。

在使用cudaMalloc3D()分配了dstPitchPtr的情况下，此函数的执行速度最快。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaMemset，cudaMemset2D，cudaMalloc3D

1.5.25 cudaMemcpy3D

名称

cudaMemcpy3D – 在3D对象间复制数据

概要

```
struct cudaExtent {
size_t width，height，depth;
};
struct cudaPos {
size_t x，y，z;
};
struct cudaMemcpy3DParms {
struct cudaArray *srcArray;
struct cudaPos srcPos;
struct cudaPitchedPtr srcPtr;
struct cudaArray *dstArray;
struct cudaPos dstPos;
struct cudaPitchedPtr dstPtr;
struct cudaExtent extent;
enum cudaMemcpyKind kind;

};
```

cudaError_t cudaMemcpy3D (const struct cudaMemcpy3DP arms *p)

cudaError_t cudaMemcpy3DAsync (const struct cudaMemcpy3DParms *p，cudaStream_t stream)

说明

cudaMemcpy3D()在两个3D对象间复制数据。源对象和目标对象均可以是主存储器、设备存储器或CUDA数组。所执行的复制操作的源、目标、范围和类型由cudaMemcpy3Dparms结构体指定，在使用之前应将其初始化为0：

cudaMemcpy3DParms myParms = {0};

传递给cudaMemcpy3D()的结构体必须指定srcArray或srcPtr以及dstArray或dstPtr。传递多个非零源或目标将导致cudaMemcpy3D()返回一个错误。

srcPos和dstPos字段是源和目标对象的可选偏移，是在各对象元素的单位中定义的。主机或设备指针的元素应为无符号字符。对于CUDA数组，指针的所有维度都必须在[0，2048)的范围之内。

extent字段定义元素中传输区域的维度。如果一个CUDA数组参与复制，则范围将根据该数组元素进行定义。如果没有任何CUDA数组参与复制，则范围将以无符号字符元素定义。

kind字段定义复制的方向。它必须是cudaMemcpyHostToHost、cudaMemcpyHostToDevice、cudaMemcpyDeviceToHost或cudaMemcpyDeviceToDevice之一。

如果源和目标均为数组，且元素大小不同，则cudaMemcpy3D()将返回一个错误。

源和目标对象不可重叠。如果指定的源和目标对象重叠，则将导致不确定的行为。

如果srcPtr或dstPtr的间距超过允许的最大值，cudaMemcpy3D()将返回一个错误。使用cudaMalloc3D()分配的cudaPitchedPtr的间距总是有效的。

cudaMemcpy3DAsync()是一种异步复制操作，可选择传入非零流参数，从而将其关联到一个流。如果源或目标是主对象，则必须将其分配到cudaMallocHost()返回的分页锁定存储器中。如果传入了一个未使用cudaMallocHost()进行分配的存储器指针，它将返回一个错误。

返回值 cudaSuccess

参见

cudaMalloc3D，cudaMalloc3DArray，cudaMemset3D，cudaMemcpy，cudaMemcpyToArray，cudaMemcpy2DToArray，cudaMemcpyFromArray，cudaMemcpy2DFromArray，cudaMemcpyArrayToArray，cudaMemcpy2DArrayToArray，cudaMemcpyToSymbol，cudaMemcpyFromSymbol

1.6 TextureReferenceManagement RT

名称

纹理引用管理

说明

本节描述CUDA运行时应用程序编程接口。

低级API

高级API

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.6.1 LowLevelApi

名称

低级纹理API

说明

本节描述用于纹理的低级CUDA运行时应用程序编程接口

cudaCreateChannelDesc cudaGetChannelDesc
cudaGetTextureReference cudaBindTexture
cudaBindTextureToArray
cudaUnbindTexture
cudaGetTextureAlignmentOffset

参见

高级API

cudaCreateChannelDesc

名称

cudaCreateChannelDesc – 低级纹理API

概要

struct cudaChannelFormatDesc cudaCreateChannelDesc(int x，int y，int z，int w，enum cudaChannelFormatKind f);

说明

返回通道描述符，格式为f，各组件的位数为x、y、z和w。cudaChannelFormatDesc定义如下：

```
struct cudaChannelFormatDesc
    { int x，y，z，w;
    enum cudaChannelFormatKind
f; };
```

其中cudaChannelFormatKind是cudaChannelFormatKindSigned、cudaChannelFormatKindUnsigned或cudaChannelFormatKindFloat之一。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGetChannelDesc，cudaGetTextureReference，cudaBindTexture，cudaBindTextureToArray，cudaUnbind-Texture，cudaGetTextureAlignmentOffset

cudaGetChannelDesc

名称

cudaGetChannelDesc – 低级纹理API

概要

cudaError_t cudaGetChannelDesc(struct cudaChannelFormatDesc* desc，const struct cudaArray* array);

说明

以*desc的形式返回CUDA数组array的通道描述符。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaCreateChannelDesc，cudaGetTextureReference，cudaBindTexture，cudaBindTextureToArray，cudaUnbindTexture，cudaGetTextureAlignmentOffset

cudaGetTextureReference

名称

cudaGetTextureReference – 低级纹理API

概要

cudaError_t cudaGetTextureReference(struct textureReference** texRef，const char* symbol)

说明

以*texRef的形式返回与符号symbol定义的纹理引用有关的结构。

返回值

相关返回值：

cudaSuccess cudaErrorInvalidTexture

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaCreateChannelDesc，cudaGetChannelDesc，cudaBindTexture，cudaBindTextureToArray，cudaUnbindTexture，cudaGetTextureAlignmentOffset

cudaBindTexture

名称

cudaBindTexture – 低级纹理API

概要

cudaError_t cudaBindTexture(size_t* offset，const struct textureReference* texRef，const void* devPtr，const struct cudaChannelFormatDesc* desc，size_t size = UINT_MAX);

说明

将devPtr指向的存储器区域中的size个字节绑定到纹理引用texRef。desc描述从纹理中获取值时如何解释存储器。所有以前绑定到texRef的存储器都将解除绑定。

由于硬件对纹理基址实施对齐要求，cudaBindTexture()将以*offset的形式返回字节偏移，为了从所需存储器中读取，必须为所获取的纹理应用此偏移。此偏移必须除以texel大小，并传递给从纹理中读取的内核，以便应用于tex1Dfetch()函数。如果设备存储器指针是由cudaMalloc()返回的，则偏移必定为0，此时可将NULL作为offset参数传递。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidTexture

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaCreateChannelDesc，cudaGetChannelDesc，cudaGetTextureReference，cudaBindTextureToArray，cudaUnbindTexture，cudaGetTextureAlignmentOffset

cudaBindTextureToArray

名称

cudaBindTextureToArray – 低级纹理API

概要

cudaError_t cudaBindTextureToArray(const struct textureReference* texRef，const struct cudaArray* array，const struct cudaChannelFormatDesc* desc);

说明

将CUDA数组array绑定到纹理引用texRef。desc描述从纹理中获取值时如何解释存储器。所有以前绑定到texRef的CUDA数组都将解除绑定。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidTexture

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaCreateChannelDesc，cudaGetChannelDesc，cudaGetTextureReference，cudaBindTexture，cudaUnbindTexture，cudaGetTextureAlignmentOffset

cudaUnbindTexture

名称

cudaUnbindTexture – 低级纹理API

概要

cudaError_t cudaUnbindTexture(const struct textureReference* texRef);

说明

将绑定到texRef纹理引用的纹理解除绑定。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaCreateChannelDesc，cudaGetChannelDesc，cudaGetTextureReference，cudaBindTexture，cudaBindTextureToArray，cudaGetTextureAlignmentOffset

cudaGetTextureAlignmentOffset

名称

cudaGetTextureAlignmentOffset – 低级纹理API

概要

cudaError_t cudaGetTextureAlignmentOffset(size_t* offset，const struct textureReference* texRef);

说明

以*offset的形式返回在绑定纹理引用texRef时返回的偏移。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidTexture

cudaErrorInvalidTextureBinding

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaCreateChannelDesc，cudaGetChannelDesc，cudaGetTextureReference，cudaBindTexture，
cudaBindTextureToArray，cuda UnbindTexture

1.6.2 HighLevelApi

名称

高级纹理API

说明

本节描述用于结构的高级CUDA运行时应用程序编程接口

cudaCreateChannelDesc cudaBindTexture

cudaBindTextureToArray cudaUnbindTexture

参见

低级API

cudaCreateChannelDesc HL

名称

cudaCreateChannelDesc – 高级纹理API

概要

template < class T >

struct cudaChannelFormatDesc cudaCreateChannelDesc <T >();

说明

返回通道描述符，格式为f，各组件的位数为x、y、z和w。cudaChannelFormatDesc定义如下：

```
struct cudaChannelFormatDesc { int x，y，z，
    w;
    enum cudaChannelFormatKind f; };
```

其中cudaChannelFormatKind是cudaChannelFormatKindSigned、cudaChannelFormatKindUnsigned或cudaChannelFormatKindFloat之一。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaCreateChannelDesc HL

名称

cudaCreateChannelDesc – 高级纹理API

概要

template < class T，int dim，enum cudaTextureReadMode readMode >

static __inline__ __host__ cudaError_t cudaBindTexture(size_t* offset，const struct texture < T，dim，readMode >& texRef，const void* devPtr，const struct cudaChannelFormatDesc& desc，size_t size = UINT_MAX)

说明

将devPtr指向的存储器区域中的size个字节绑定到纹理引用texRef。desc描述从纹理中获取值时如何解释存储器。Offset参数是可选的偏移字节，这与低级cudaBindTexture()函数相同。所有以前绑定到texRef的存储器都将解除绑定。

template E<lt> class T，int dim，enum cudaTextureReadMode readMode E<gt> static __inline__
__host__ cudaError_t cudaBindTexture(
    size_t* offset,
    const struct texture E<lt> T，dim，readMode E<gt>& texRef，const void* devPtr,
    size_t size = UINT_MAX);

将devPtr指向的存储器区域中的size个字节绑定到纹理引用texRef。通道描述符继承自纹理引用类型。offset参数是可选的字符偏移，与低级cudaBindTexture()函数一节中所述相同。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidTexture

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaBindTextureToArray HL

名称

cudaBindTextureToArray – 高级纹理API

概要

template < class T，int dim，enum cudaTextureReadMode readMode >

static __inline__ __host__ cudaError_t cudaBindTextureToArray(const struct texture < T，dim，readMode >& texRef，const struct cudaArray* cuArray，const struct cudaChannelFormatDesc& desc)

说明

将CUDA数组array绑定到纹理引用texRef。Desc描述从纹理中获取值时如何解释存储器。所有以前绑定到texRef的CUDA数组都将解除绑定。

template E<lt> class T，int dim，enum cudaTextureReadMode readMode E<gt> static __inline__ __host__ cudaError_t cudaBindTextureToArray(const struct texture E<lt> T，dim，readMode E<gt>& texRef,
const struct cudaArray* cuArray);

将CUDA数组array绑定到纹理引用texRef。通道描述符继承自CUDA数组。所有以前绑定到texRef的CUDA数组都将解除绑定。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer

cudaErrorInvalidTexture

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaCreateChannelDesc HL

cudaUnbindTexture HL

名称

cudaUnbindTexture – 高级纹理API

概要

template < class T，int dim，enum cudaTextureReadMode readMode >

static __inline__ __host__ cudaError_t cudaUnbindTexture(const struct texture < T，dim，readMode >& texRef)

说明

将绑定到纹理引用texRef的纹理解除绑定。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

1.7 ExecutionControl RT

名称

执行控制

说明

本节描述CUDA运行时应用程序编程接口。

cudaConfigureCall

cudaLaunch

cudaSetupArgument

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.7.1 cudaConfigureCall

名称

cudaConfigureCall – 配置设备启动

概要

cudaError_t cudaConfigureCall(dim3 gridDim，dim3 blockDim，size_t sharedMem = 0，int tokens = 0)

说明

为要执行的设备调用指定网格和块大小，类似于执行配置语法。cudaConfigureCall()基于堆栈。每次调用都会将数据放入一个执行堆栈的顶端。此数据包含网格和线程块的大小以及针对调用的所有参数。

返回值

相关返回值：

cudaSuccess cudaErrorInvalidConfiguration

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaLaunch，cudaSetupArgument

1.7.2 cudaLaunch

名称

cudaLaunch – 启动设备函数

概要

template < class T > cudaError_t cudaLaunch(T entry)

说明

在设备上启动函数entry。entry可以是一个在设备上执行的函数，也可以是指定在设备上执行的函数的字符串。entry必须声明为全局函数。在cudaLaunch()之前必须存在cudaConfigureCall()调用，因为它将从执行堆栈中弹出cudaConfigureCall()放入的数据。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidDeviceFunction

cudaErrorInvalidConfiguration

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaConfigure Call，cudaSetupArgument

1.7.3 cudaSetupArgument

名称

cudaSetupArgument – 配置设备启动

概要

cudaError_t cudaSetupArgument(void* arg，size_t count，size_t offset) template < class T > cudaError_t

cudaSetupArgument(T arg，size_t offset)

说明

将arg指向的参数中的count个字节放到离区域传递参数offset个字节处，其中区域从offset 0开始。参数存储在执行堆栈顶端。cudaSetupArgument()之前必须存在cudaConfigureCall()调用。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaConfigureCall，cudaLaunch

1.8 OpenGlInteroperability RT

名称

OpenGL互操作性

说明

本节描述CUDA运行时应用程序编程接口。

cudaGLSetGLDevice

cudaGLRegisterBufferObject

cudaGLMapBufferObject

cudaGLUnmapBufferObject

cudaGLUnregisterBufferObject

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.8.1 cudaGLSetGLDevice

名称

cudaGLSetGLDevice – 设置CUDA设备以利用GL互操作性

概要

cudaError_t cudaGLSetGLDevice(int device);

说明

将dev记录为活动主线程执行设备码的设备。将线程记录为使用GL互操作性。

返回值

相关返回值：

cudaSuccess cudaErrorInvalidDevice

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGLRegisterBufferObject，cudaGLMapBufferObject，cudaGLUnmapBufferObject，cudaGLUnregisterBufferObject

1.8.2 cudaGLRegisterBufferObject

名称

cudaGLRegisterBufferObject - OpenGL互操作性

概要

cudaError_t cudaGLRegisterBufferObject (GLuint buff erObj)

说明

注册缓存对象（其ID为bufferObj）以便CUDA访问。必须先调用此函数CUDA才能映射缓存对象。注册后，缓存对象无法供任何OpenGL命令使用，只能作为OpenGL绘图命令的数据源。

返回值

相关返回值：

cudaSuccess cudaErrorNotInitialized

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGLSetGLDevice，cudaGLMapBufferObject，cudaGLUnmapBufferObject，cudaGLUnregisterBufferObject

1.8.3 cudaGLMapBufferObject

名称

cudaGLMapBufferObject – OpenGL互操作性

概要

cudaError_t cudaGLMapBufferObject(void** devPtr，GLuint buff erObj);

说明

将ID为bufferObj的缓存对象映射到CUDA地址空间内，并以*devPtr的形式返回所得映射的基址指针。

返回值

相关返回值：

cudaSuccess cudaErrorMapBufferObjectFailed

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGLSetGLDevice，cudaGLRegisterBufferObject，cudaGLUnmapBufferObject，cudaGLUnregisterBufferObject

1.8.4 cudaGLUnmapBufferObject

名称

cudaGLUnmapBufferObject – OpenGL互操作性

概要

cudaError_t cudaGLUnmapBufferObject(GLuint buff erObj);

说明

解除ID为bufferObj的缓存对象的映射，以便CUDA访问。

返回值

相关返回值：

cudaSuccess

cudaErrorInvalidDevicePointer

cudaErrorUnmapBufferObjectFailed

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGLSetGLDevice，cudaGLRegisterBufferObject，cudaGLMapBufferObject，cudaGLUnregisterBufferObject

1.8.5 cudaGLUnregisterBufferObject

名称

cudaGLUnregisterBufferObject – OpenGL互操作性

概要

cudaError_t cudaGLUnregisterBufferObject(GLuint buff erObj);

说明

注销ID为bufferObj的缓存对象，以便CUDA访问。

返回值

相关返回值：

cudaSuccess

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGLSetGLDevice，cudaGLRegisterBufferObject，cudaGLMapBufferObject，cudaGLUnmapBufferObject

1.9 Direct3dInteroperability RT

名称

Direct3D互操作性

说明

本节描述CUDA运行时应用程序编程接口。

cudaD3D9GetDevice

cudaD3D9SetDirect3DDevice

cudaD3D9GetDirect3DDevice

cudaD3D9RegisterResource

cudaD3D9UnregisterResource

cudaD3D9MapResources

cudaD3D9UnmapResources

cudaD3D9ResourceGetSurfaceDimensions

cudaD3D9ResourceSetMapFlags

cudaD3D9ResourceGetMappedPointer

cudaD3D9ResourceGetMappedSize

cudaD3D9ResourceGetMappedPitch

从CUDA 2.0起，以下函数被取消。在新开发工作中不应再继续使用这些函数。

cudaD3D9Begin

cudaD3D9End

cudaD3D9RegisterVertexBuffer

cudaD3D9MapVertexBuffer

cudaD3D9UnmapVertexBuffer

cudaD3D9UnregisterVertexBuffer

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.9.1 cudaD3D9SetDirect3DDevice

名称

cudaD3D9SetDirect3DDevice – 设置Direct3D设备，以便在此线程中用于互操作

概要

cudaError_t cudaD3D9SetDirect3DDevice(IDirect3DDevice9* pDxDevice);

说明

将pDxDevice记录为Direct3D设备，以在此主线程上利用Direct3D互操作性。为了使用Direct3D互操作性，必须在此线程执行其他CUDA运行时调用之前完成此调用。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalidValue

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaSetDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapFla cudaD3D9ResourceGetMappedPointer，cudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.2 cudaD3D9GetDirect3DDevice

名称

cudaD3D9GetDirect3DDevice – 获取据以创建当前CUDA上下文的Direct3D设备

概要

cudaError_t cudaD3D9GetDirect3DDevice(IDirect3DDevice9** ppDxDevice);

说明

以*ppDxDevice的形式返回cudaD3D9SetDirect3Ddevice中创建CUDA上下文所用的Direct3D设备。

返回值

cudaSuccess
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT

参见

cudaD3D9SetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapFlags，cudaD3D9ResourceGetMappedPointer, cudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.3 cudaD3D9RegisterResource

名称

cudaD3D9RegisterResource – 注册Direct3D资源，以便CUDA访问

概要

cudaError_t cudaD3D9RegisterResource(IDirect3DResource9* pResource，unsigned int Flags);

说明

注册Direct3D资源pResource，以便CUDA访问。

如果此调用成功，则应用程序能够在资源注销之前映射和解除映射此资源。此调用开销较高，不应在交互式应用程序的每一帧中进行调用。

pResource的类型必须为以下之一：

- IDirect3DVertexBuffer9：无备注。
- IDirect3DIndexBuffer9：无备注。
- IDirect3DSurface9：只有IDirect3DSurface9类型的独立对象才可显式共享。具体来说，不可直接注册独立的mipmap级别和立方体各个面的映射。要访问与纹理相关的独立表面，必须注册基本纹理对象。
- IDirect3DBaseTexture9：注册一个纹理时，CUDA可以访问与mipmap顶级相关的所有表面。在mipmap顶级以下的mipmap级别不可访问。

并非所有上述类型的Direct3D资源均可用于与CUDA进行互操作。下面列举了部分限制：

- 主呈现目标未向CUDA注册。
- 深度和模板表面未向CUDA注册。
- 分配为共享的资源未向CUDA注册。
- D3DPOOL_SYSTEMMEM中分配的任意资源未向CUDA注册。

Flags参数必须设置为cudaD3D9RegisterFlagsNone。

如果在此上下文内未初始化Direct3D互操作性，则将返回CUDA_ERROR_INVALID_CONTEXT。

如果pResource是不正确的类型（例如，是非独立的IDirect3DResource9或已经注册），则返回CUDA_ERROR_INVALID_HANDLE。如果pResource无法注册，则返回CUDA_ERROR_UNKNOWN。

返回值

cudaSuccess

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_OUT_OF_MEMORY

CUDA_ERROR_UNKNOWN

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapFlags，cudaD3D9ResourceGetMappedPointer，cudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.4 cudaD3D9UnregisterResource

名称

cudaD3D9UnregisterResource – 注销Direct3D资源

概要

cudaError_t cudaD3D9UnregisterResource(IDirect3DResource9* pResource);

说明

注销Direct3D资源pResource，除非再次注册，否则CUDA将无法再访问此资源。如果pResource没有注册，则返回CUDA_ERROR_INVALID_HANDLE。

返回值

cudaSuccess
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_HANDLE

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapFlags，cudaD3D9ResourceGetMappedPointer，cudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.5 cudaD3D9MapResources

名称

cudaD3D9MapResources – 映射Direct3D资源，以便CUDA访问

概要

cudaError_t cudaD3D9MapResources(unsigned int count，IDirect3DResource9 **ppResources);

说明

映射ppResources中的count Direct3D资源，以便CUDA访问。

解除映射之前，可在CUDA内核中访问ppResources中的资源。在CUDA映射了资源后，Direct3D不应访问任何资源。如果应用程序允许其访问，将导致不确定的结果。

此函数提供了同步保证，确保在cudaD3D9MapResources开始执行CUDA内核之前，cudaD3D9MapResources之前发出的任何Direct3D调用都将完成。

如果存在尚未注册与CUDA共同使用的ppResources，如果ppResources包含重复项，则将返回CUDA_ERROR_INVALID_HANDLE。如果有ppResources已经映射为CUDA访问，则返回CUDA_ERROR_ALREADY_MAPPED。

返回值

cudaSuccess

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_ALREADY_MAPPED

CUDA_ERROR_UNKNOWN

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapFlags，cudaD3D9ResourceGetMappedPointer，cudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.6 cudaD3D9UnmapResources

名称

cudaD3D9UnmapResources – 解除映射Direct3D资源

概要

cudaError_t cudaD3D9UnmapResources(unsigned int count，IDirect3DResource9** ppResources);

说明

解除映射ppResources中的count Direct3D资源。

此函数提供了同步保证，确保在cudaD3D9UnmapResources开始发出Direct3D调用之前，cudaD3D9UnmapResources之前发出的任何CUDA内核都将完成。
如果存在尚未注册与CUDA共同使用的ppResources，如果ppResources包含重复项，则将返回CUDA_ERROR_INVALID_HANDLE。如果有ppResources未映射为CUDA访问，则返回CUDA_ERROR_NOT_MAPPED。

返回值

cudaSuccess

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_MAPPED

CUDA_ERROR_UNKNOWN

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapFlags，cudaD3D9ResourceGetcudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.7 cudaD3D9ResourceSetMapFlags

名称

cudaD3D9ResourceSetMapFlags – 为映射Direct3D资源设置使用标志

概要

cudaError_t cudaD3D9ResourceSetMapFlags(IDirect3DResource9 *pResource，unsigned int Flags);

说明

为映射Direct3D资源pResource设置标志。

标志更改将在下一次映射pResource时生效。Flags参数可为以下任何值之一：

- cudaD3D9MapFlagsNone：指明没有任何关于使用此资源方式的提示。因此，假设此资源将由CUDA内核读取并写入。该值为默认值。
- cudaD3D9MapFlagsReadOnly：指明访问此资源的CUDA内核将不会写入此资源。
- cudaD3D9MapFlagsWriteDiscard：指明访问此资源的CUDA内核将不会读取此资源，且将改写该资源的所有内容，之前存储在此资源中的任何数据均不予保留。

如果pResource尚未注册与CUDA共同使用，则将返回CUDA_ERROR_INVALID_HANDLE。如果有pResource已映射为CUDA访问，则返回CUDA_ERROR_ALREADY_MAPPED。

返回值

cudaSuccess

CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE
CUDA_ERROR_INVALID_HANDLE
CUDA_ERROR_ALREADY_MAPPED
CUDA_ERROR_UNKNOWN

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceGetMapcudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.8 cudaD3D9ResourceGet SurfaceDimensions

名称

cudaD3D9ResourceGetSurfaceDimensions – 获取已注册表面的维度

概要

cudaError_t cudaD3D9ResourceGetSurfaceDimensions(size_t* pWidth，size_t* pHeight，size_t *pDepth，IUnknown* pResource，unsigned int Face，unsigned int Level);

说明

以*pWidth、*pHeight和*pDepth的形式返回对应于Face和Level的已映射Direct3D资源pResource的子资源维度。

对于抗锯齿表面来说，每像素可能包含多个样本，因而一个资源的维度可能大于Direct3D运行时报告的维度。

pWidth、pHeight和pDepth参数是可选的。对于2D表面而言，*pDepth返回的值是0。

如果pResource的类型不是IDirect3DBaseTexture9或IDirect3DSurface9，如果pResource尚未注册与CUDA共同使用，则将返回CUDA_ERROR_INVALID_HANDLE。

返回值

cudaSuccess

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceSetMapFlags，cudaD3D9ResourceGetMappedPoint，cudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.9 cudaD3D9ResourceGetMappedPointer

名称

cudaD3D9ResourceGetMappedPointer – 获取已映射CUDA资源的基址指针

概要

cudaError_t cudaD3D9ResourceGetMappedPointer(void** pDevPtr，IUnknown* pResource，unsigned int Face，unsigned int Level);

说明

以*pDevPtr的形式返回对应于Face和Level的已映射Direct3D资源pResource的子资源基址指针。每次映射pResource时，pDevPtr中设置的值都会发生变化。

如果pResource没有注册，则返回CUDA_ERROR_INVALID_HANDLE。如果pResource没有映射，则返回CUDA_ERROR_NOT_MAPPED。

如果pResource的类型是IDirect3DCubeTexture9，则Face必须是按D3DCUBEMAP_FACES类型枚举的值之一。对于其他所有类型，Face必须为0。如果Face无效，则返回CUDA_ERROR_IN_VALID。

如果pResource的类型是IDirect3DBaseTexture9，则Level必须对应于有效的mipmap级别。对于其他所有类型，Level必须为0。目前仅支持mipmap级别0。如果Level无效，则返回CUDA_ERROR_INVALID_VALUE。

返回值

cudaSuccess

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_MAPPED

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapF cudaD3D9ResourceGetMappedSize，cudaD3D9ResourceGetMappedPitch

1.9.10 cudaD3D9ResourceGetMappedSize

名称

cudaD3D9ResourceGetMappedSize – 获取已映射CUDA资源的大小

概要

cudaError_t cudaD3D9ResourceGetMappedSize(size_t* pSize，IDirect3DResource9* pResource);

说明

以*pSize的形式返回对应于Face和Level的已映射Direct3D资源pResource的子资源大小。每次映射pResource时，pSize中设置的值都会发生变化。

如果pResource尚未注册与CUDA一起使用，则返回CUDA_ERROR_INVALID_HANDLE。如果pResource尚未映射为CUDA访问，则返回CUDA_ERROR_NOT_MAPPED。

关于Face和Level参数的使用要求，请参见cudaD3D9ResourceGetMappedPointer。

返回值

cudaSuccess

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_MAPPED

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapF，cudaD3D9ResourceGetMappedPointer，cudaD3D9ResourceGetMappedPitch

1.9.11 cudaD3D9ResourceGetMappedPitch

名称

cudaD3D9ResourceGetMappedPitch – 获取已映射CUDA资源的间距

概要

cudaError_t cudaD3D9ResourceGetMappedPitch(size_t* pPitch，size_t* pPitchSlice，IDirect3DResource9* pResource，unsigned int Face，unsigned int Level);

说明

以*pPitch和*pPitchSlice的形式返回对应于Face和Level的已映射Direct3D资源的子资源间距和Z-slice间距。每次映射pResource时，pPitch和pPitchSlice中设置的值都会发生变化。

如果pResource不是IDirect3DBaseTexture9类型或其任意子类型，如果pResource尚未注册与CUDA一起使用，则将返回CUDA_ERROR_INVALID_HANDLE。如果pResource尚未映射为CUDA访问，则返回CUDA_ERROR_NOT_MAPPED。

关于Face和Level参数的使用要求，请参见cudaD3D9ResourceGetMappedPointer。参数pPitch和

pPitchSlice都是可选的，可设置为NULL。

间距和Z-slice间距值可用于计算表面上一个样本的位置，具体方法见下文。对于2D表面而言，从表面基址算起，位置x,y处的样本字节偏移为：

y*pitch + (bytes per pixel)*x

对于3D表面而言，从表面基址算起，位置x,y处的样本字节偏移为：

z*slicePitch + y*pitch + (bytes per pixel)*x

返回值

cudaSuccess

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_MAPPED

参见

cudaD3D9SetDirect3DDevice，cudaD3D9GetDirect3DDevice，cudaD3D9RegisterResource，cudaD3D9UnregisterResource，cudaD3D9MapResources，cudaD3D9UnmapResources，cudaD3D9ResourceGetSurfaceDimensions，cudaD3D9ResourceSetMapF cudaD3D9ResourceGetMappedPointer，cudaD3D9ResourceGetMappedSize

1.9.12 cudaD3D9Begin

名称

cudaD3D9Begin - D3D互操作性

概要

cudaError_t cudaD3D9Begin (IDirect3DDevice9* device)

说明

初始化与Direct3D设备device的互操性。必须首先调用此函数CUDA才能映射设备的对象。然后，在调用cuD3D9End()之前，应用程序可以调用Direct3D设备拥有的映射顶点缓存。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果使用cudaD3D9Begin为此CUDA上下文初始化了Direct3D互操作，则互操作将限制在被废弃的顶点缓存接口，调用CUDA 2.0及更新版本中引入的函数都将失败。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaD3D9End，cudaD3D9RegisterVertexBuffer，cudaD3D9MapVertexBuffer，cudaD3D9UnmapVertexBuffer，cudaD3D9UnregisterVertexBuffer，cudaD3D9GetDevice

1.9.13 cudaD3D9End

名称

cudaD3D9End - D3D互操作性

概要

cudaError_t cudaD3D9End(void);

说明

结束之前在cuD3D9Begin()中指定的与Direct3D设备的互操作。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError

cudaErrorPriorLaunchFailure cudaErrorInvalid

Value

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaD3D9Begin，cudaD3D9RegisterVertexBuffer，cudaD3D9MapVertexBuffer，cudaD3D9UnmapVertexBuffer，cudaD3D9UnregisterVertexBuffer，cudaD3D9GetDevice

1.9.14 cudaD3D9RegisterVertexBuffer

名称

cudaD3D9RegisterVertexBuffer - D3D互操作性

概要

cudaError_t cudaD3D9RegisterVertexBuffer(IDirect3DVertexBuffer9* VB);

说明

Registers the vertex buffer VB for access by CUDA.

该函数自 CUDA 2.0 起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalid Value

cudaErrorMemoryAllocation

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaD3D9Begin，cudaD3D9End，cudaD3D9MapVertexBuffer，cudaD3D9UnmapVertexBuffer，cudaD3D9UnregisterVertexBuff cudaD3D9GetDevice

1.9.15 cudaD3D9MapVertexBuffer

名称

cudaD3D9MapVertexBuffer - D3D互操作性

概要

cudaError_t cudaD3D9MapVertexBuffer (void** devPtr，IDirect3DVertexBuffer9* VB)

说明

将顶点缓存VB映射到当前CUDA上下文的地址空间中，并以*devPtr的形式返回最终映射的基址指针。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalid Value

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaD3D9Begin，cudaD3D9End，cudaD3D9RegisterVertexBuffer，cudaD3D9UnmapVertexBuffer，cudaD3D9UnregisterVertexB cudaD3D9GetDevice

1.9.16 cudaD3D9UnmapVertexBuffer

名称

cudaD3D9UnmapVertexBuffer - D3D互操作性

概要

cudaError_t cudaD3D9UnmapVertexBuffer(IDirect3DVertexBuffer9* VB);

说明

解除顶点缓存的映射，以便CUDA访问。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalid Value

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaD3D9Begin，cudaD3D9End，cudaD3D9RegisterVertexBuffer，cudaD3D9MapVertexBuffer，cudaD3D9UnregisterVertexBuff cudaD3D9GetDevice

1.9.17 cudaD3D9UnregisterVertexBuffer

名称

cudaD3D9UnregisterVertexBuffer - D3D互操作性

概要

cudaError_t cudaD3D9UnregisterVertexBuffer(IDirect3DVertexBuffer9* VB);

说明

注销顶点缓存VB，以便CUDA访问。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice
接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

cudaSuccess

cudaErrorNotYetImplemented

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalid Value

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaD3D9Begin，cudaD3D9End，cudaD3D9RegisterVertexBuffer，cudaD3D9MapVertexBuffer，cudaD3D9UnmapVertexBuffe
cudaD3D9GetDevice

1.9.18 cudaD3D9GetDevice

名称

cudaD3D9GetDevice - D3D互操作性

概要

cudaError_t cudaD3D9GetDevice(int* dev，const char* adapterName);

说明

以*dev的形式返回对应于从EnumDisplayDevices或IDirect3D9: :GetAdapterIdentifier()中获取的适配器名adapterName的设备。

返回值

相关返回值：

cudaSuccess

cudaErrorInitializationError

cudaErrorPriorLaunchFailure

cudaErrorInvalid Value

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaD3D9Begin，cudaD3D9End，cudaD3D9RegisterVertexBuffer，cudaD3D9MapVertexBuffer，cudaD3D9UnmapVertexBuffe cudaD3D9UnregisterVertexBuffer

1.10 ErrorHandling RT

名称

错误处理

说明

本节描述CUDA运行时应用程序编程接口。

cudaGetLastError

cudaGetErrorString

参见

设备管理，线程管理，流管理，事件管理，执行管理，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性，错误处理

1.10.1 cudaGetLastError

名称

cudaGetLastError – 返回运行时调用的最新错误

概要

cudaError_t cudaGetLastError(void)

说明

返回同一主线程中运行时调用所返回的最新错误，并将其重置为cudaSuccess。

返回值

相关返回值：

cudaSuccess
cudaErrorInitializationError cudaErrorLaunchFailure
cudaErrorPriorLaunchFailure
cudaErrorLaunchTimeout
cudaErrorLaunchOutOfResources
cudaErrorInvalidDeviceFunction
cudaErrorInvalidConfiguration
cudaErrorInvalidDevice

cudaErrorInvalidValue

cudaErrorInvalidDevicePointer cudaErrorInvalidTexture
cudaErrorInvalidTextureBinding
cudaErrorInvalidChannelDescriptor

cudaErrorTextureFetchFailed
cudaErrorTextureNotBound
cudaErrorSynchronizationError cudaErrorUnknown

cudaErrorInvalidResourceHandle
cudaErrorNotReady

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cudaGetErrorString，cudaError

1.10.2 cudaGetErrorString

名称

cudaGetErrorString - 返回错误中的消息字符串

概要

const char* cudaGetErrorString(cudaError_t error);

说明

返回错误码中的消息字符串

返回值

指向NULL终止字符串的char*指针

参见

cudaGetLastError

2 DriverApiReference

名称

驱动程序API接口

说明

本节描述低级CUDA驱动程序应用程序编程接口。

参见

Initialization，DeviceManagement，ContextManagement，ModuleManagement，StreamManagement，Event-Management，Execution Control，MemoryManagement，TextureReferenceManagement，OpenGlInteroperability，Direct3dInteroperability

2.1 Initialization

名称

驱动程序初始化

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuInit

参见

Initialization，DeviceManagement，ContextManagement，ModuleManagement，StreamManagement，Event-Management，ExecutionControl，MemoryManagement，TextureReferenceManagement，OpenGlInteroperability，Direct3dInteroperability

2.1.1 cuInit

名称

cuInit – 初始化CUDA驱动程序API

概要

CUresult cuInit(unsigned int Flags);

说明

初始化驱动程序API，必须首先调用此函数之后才能调用驱动程序API中的其他函数。目前，Flags参数必须为0。如果未调用cuInit()，驱动程序API中的任何函数都将返回CUDA_ERROR_NOT_INITIALIZED。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_NO_DEVICE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

2.2 DeviceManagement

名称

设备管理

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuDeviceComputeCapability

cuDeviceGet

cuDeviceGetAttribute

cuDeviceGetCount

cuDeviceGetName

cuDeviceGetProperties

cuDeviceTotalMem

参见

Initialization，DeviceManagement，ContextManagement，ModuleManagement，StreamManagement，Event-Management，ExecutionControl，MemoryManagement，TextureReferenceManagement，OpenGlInteroperability，Direct3dInteroperability

2.2.1 cuDeviceComputeCapability

名称

cuDeviceComputeCapability – 返回设备的计算能力

概要

CUresult cuDeviceComputeCapability(int* major，int* minor，CUdevice dev);

说明

以*major和*minor的形式返回定义设备dev计算能力的主要修订号和次要修订号。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE
CUDA_ERROR_INVALID_DEVICE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuDeviceGetAttribute，cuDeviceGetCount，cuDeviceGetName，cuDeviceGet，cuDeviceGetProperties，cuDeviceTotalMem

2.2.2 cuDeviceGet

名称

cuDeviceGet – 返回设备句柄

概要

CUresult cuDeviceGet(CUdevice* dev，int ordinal);

说明

以*dev的形式返回设备句柄，其序数范围为 [0，cuDeviceGet Count ()-1]。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE
CUDA_ERROR_INVALID_DEVICE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuDeviceComputeCapability，cuDeviceGetAttribute，cuDeviceGetCount，cuDeviceGetName，cuDeviceGetProperties，cuDeviceTotalMem

2.2.3 cuDeviceGetAttribute

名称

cuDeviceGetAttribute – 返回关于设备的信息

概要

CUresult cuDeviceGetAttribute(int* value，CUdevice_attribute attrib，CUdevice dev);

说明

以*value的形式返回设备dev上属性attrib的整型值。支持的属性包括：

- CU_DEVICE_ATTRIBUTE_MAX_THREADS_PER_BLOCK：每个块的最大线程数；
- CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_X：一个块的最大x维度；
- CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_Y：一个块的最大y维度；
- CU_DEVICE_ATTRIBUTE_MAX_BLOCK_DIM_Z：一个块的最大z维度；
- CU_DEVICE_ATTRIBUTE_MAX_GRID_DIM_X：一个网格的最大x维度；
- CU_DEVICE_ATTRIBUTE_MAX_GRID_DIM_Y：一个网格的最大y维度；
- CU_DEVICE_ATTRIBUTE_MAX_GRID_DIM_Z：一个网格的最大z维度
- CU_DEVICE_ATTRIBUTE_MAX_SHARED_MEMORY_PER_BLOCK：可用于线程块的最大共享存储器数量，以字节为单位；位于一个多处理器上的所有线程块可以同时共享这些存储器；
- CU_DEVICE_ATTRIBUTE_TOTAL_CONSTANT_MEMORY：设备上可用的不变存储器总量，以字节为单位；
- CU_DEVICE_ATTRIBUTE_WARP_SIZE：按线程计算的 warp 块大小；
- CU_DEVICE_ATTRIBUTE_MAX_PITCH：包含通过cuMemAllocPitch()分配的存储器区域的存储器复制函数所支持的最大间距，以字节为单位；
- CU_DEVICE_ATTRIBUTE_MAX_REGISTERS_PER_BLOCK：线程块可以使用的32位寄存器的最大值；位于一个多处理器上的所有线程块可以共享这些寄存器；
- CU_DEVICE_ATTRIBUTE_CLOCK_RATE：时钟频率，以千赫为单位；
- CU_DEVICE_ATTRIBUTE_ TEXTURE_ALIGNMENT：对齐要求；与textureAlignment字节对齐的纹理基址无需对纹理取样应用偏移；
- CU_DEVICE_ATTRIBUTE_GPU_OVERLAP：如果设备可在主机和设备之间并发复制存储器，同时又能执行内核，则此值为1；否则此值为0；
- CU_DEVICE_ATTRIBUTE_MULTIPROCESSOR_COUNT：设备上多处理器的数量。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_DEVICE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuDeviceComputeCapability，cuDeviceGetCount，cuDevice GetName，cuDeviceGet，cuDeviceGetProperties，cuDevice TotalMem

2.2.4 cuDeviceGetCount

名称

cuDeviceGetCount - 返回具有计算能力的设备的数量

概要

CUresult cuDeviceGetCount (int* count);

说明

以*count形式返回可用于执行的计算能力大于等于1.0的设备数。如果不存在此类设备，cudaGetDeviceCount()将返回0。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuDeviceComputeCapability，cuDeviceGetAttribute，cuDeviceGetName，cuDeviceGet，cuDeviceGetProperties，cuDeviceTotalMem

2.2.5 cuDeviceGetName

名称

cuDeviceGetName - 返回标识符字符串

概要

CUresult cuDeviceGetName(char* name，int len，CUdevice dev);

说明

返回一个ASCII字符串，在name指向的以NULL结尾的字符串中指定设备dev。len指定返回字符串的最大长度。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE
CUDA_ERROR_INVALID_DEVICE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuDeviceComputeCapability，cuDeviceGetAttribute，cuDeviceGetCount，cuDeviceGet，cuDevice GetProperties，cuDeviceTotalMem

2.2.6 cuDeviceGetProperties

名称

cuDeviceGetProperties – 获取设备属性

概要

CUresult cuDeviceGetProperties(CUdevprop* prop，CUdevice dev);

说明

以*prop形式返回设备dev的属性。 Udevprop结构定义如下：

```
typedef struct CUdevprop_st {
    int maxThreadsPerBlock;
    int maxThreadsDim[3];
    int maxGridSize [3];
    int sharedMemPerBlock;
    int totalConstantMemory;
    int SIMDWidth; int
    memPitch;
    int regsPerBlock; int clockRate;
    int textureAlign
} CUdevprop;
```

其中：

- maxThreadsPerBlock是每个块中的最大线程数；
- maxThreadsDim[3]是一个块各维度的最大值；
- maxGridSize[3]是一个网格各维度的最大值；
- sharedMemPerBlock是每个块可以使用的共享存储器的总量，以字节为单位；
- totalConstantMemory是设备上可用不变存储器的总量，以字节为单位；
- SIMDWidth是warp块的大小；
- memPitch是包含通过cuMemAllocPitch()分配的存储器区域的存储器复制函数所支持的最大间距，以字节为单位；
- regsPerBlock是每个块上可以使用的寄存器的总量；
- clockRate是时钟频率，以千赫为单位；
- textureAlign是对齐要求；与textureAlign字节对齐的纹理基址无需对纹理取样应用偏移。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_DEVICE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuDeviceComputeCapability，cuDeviceGetAttribute，cuDeviceGetCount，cuDeviceGetName，cuDeviceGet，cuDeviceTotalMem

2.2.7 cuDeviceTotalMem

名称

cuDeviceTotalMem – 返回设备上的存储器总量

概要

CUresult cuDeviceTotalMem(unsigned int* bytes，CUdevice dev);

说明

以*bytes的形式返回设备dev上可用存储器的总量，以字节为单位。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_DEVICE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuDeviceComputeCapability，cuDeviceGetAttribute，cuDeviceGetCount，cuDeviceGetName，cuDeviceGet，cuDeviceGetProperties

2.3 ContextManagement

名称

上下文管理

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuCtxAttach
cuCtxCre ate
cuCtxDestroy
cuCtxDetach
cuCtxGetDevice
cuCtxPop Current

cuCt xPushCurrent
cuCtxSynchronize

参见

初始化，设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.3.1 cuCtxAttach

名称

cuCtxAttach – 增加上下文使用计数

概要

CUresult cuCtxAttach(CUcontext* pCtx，unsigned int Flags);

说明

增加上下文的使用计数，并以*pCtx的形式传递回一个上下文句柄，在应用程序处理完该上下文之后，必须向cuCtxDetach()传递该上下文句柄。如果不存在对于线程来说最新的上下文，cuCtxAttach()将失败。

目前，Flags参数必须为0。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuCtxCreate，cuCtxDetach，cuCtxGetDevice，cuCtxSynchronize

2.3.2 cuCtxCreate

名称

cuCtxCreate – 创建CUDA上下文

概要

CUresult cuCtxCreate(CUcontext* pCtx，unsigned int Flags，CUdevice dev);

说明

创建一个新的CUDA上下文，并将其与调用线程关联起来。Flags参数将在下面介绍。该上下文是在使用计数为1的情况下创建，使用了上下文之后，cuCtxCreate()的调用方必须调用cuCtxDestroy()或cuCtxDetach()。如果上下文对于线程来说已经是最新状态，则替换为新创建的上下文，随后可以调用cuCtxPopCurrent()恢复。

在等待GPU返回结果时，Flags参数的两个LSB可以控制在API调用时拥有CUDA上下文的OS线程如何与OS调度程序交互。

- CU_CTX_SCHED_AUTO：Flags参数为0时的默认值，根据流程C中活动CUDA上下文的数量和系统P中逻辑处理器的数量进行探索。如果C大于P，则CUDA在等待GPU时将让给其他OS线程，否则在等待结果时，CUDA将不会让步，保持在处理器上的活动状态。
- CU_CTX_SCHED_SPIN：指示CUDA在等待GPU返回结果时保持活动。这可降低等待GPU的延迟，但如果CPU线程与CUDA线程并行执行工作，则可能会降低CPU线程的性能。
- CU_CTX_SCHED_YIELD：指示CUDA在等待GPU返回结果时放弃其线程。这可能会增加等待GPU的延迟，但如果过CPU线程与GPU并行执行工作，这能够提高CPU线程的性能。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_DEVICE

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

CUDA_ERROR_UNKNOWN

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuCtxAttach，cuCtxDetach，cuCtxDestroy，cuCtxPush Current，cuCtxPop Current

2.3.3 cuCtxDetach

名称

cuCtxDetach – 减少上下文的使用计数

概要

CUresult cuCtxDetach(CUcontext ctx);

说明

减少上下文使用计数，如果使用计数减少到0，则销毁上下文。上下文必须是cuCtxCreate()或cuCtxAttach()传递回的句柄，且必须对调用方线程保持最新。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuCtxCreate，cuCtxAttach，cuCtxDestroy，cuCtxPushCurrent，cuCtxPopCurrent

2.3.4 cuCtxGetDevice

名称

cuCtxGetDevice – 返回当前上下文的设备ID

概要

CUresult cuCtxGetDevice(CUdevice* device);

说明

以*device的形式返回当前上下文设备的序号。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuCtxCreate，cuCtxAttach，cuCtxDetach，cuCtxSynchronize

2.3.5 cuCtxPopCurrent

名称

cuCtxPopCurrent – 从当前CPU线程弹出当前CUDA上下文

概要

CUresult cuCtxPopCurrent (CUcontext *pctx);

说明

从CPU线程弹出当前CUDA上下文。CUDA上下文的使用计数必须为1。创建CUDA上下文时，其使用计数就是1，使用计数可通过cuCtxAttach()增加，可通过cuCtxDetach()减少。

如果成功，cuCtxPopCurrent()将以*pctx的形式传回上下文句柄。随后即可通过调用cuCtxPushCurrent()使上下文对其他CPU线程保持最新。

可通过调用cuCtxDestroy()来销毁不固定上下文。

如果在调用cuCtxCreate或cuCtxPushCurrent之前，上下文对于CPU线程是最新的，则此函数将再次使上下文对于CPU线程保持最新。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuCtxCreate，cuCtxAttach，cuCtxDetach，cuCtxDestroy，cuCtxPushCurrent

2.3.6 cuCtxPushCurrent

名称

cuCtxPushCurrent – 为CPU线程附加不固定上下文

概要

CUresult cuCtxPushCurrent (CUcontext ctx);

说明

将给定上下文压入当前上下文的CPU线程堆栈。指定上下文将成为CPU线程的当前上前文，从而影响根据当前上下文进行操作的所有CUDA函数。

可调用cuCtxDestroy()或cuCtxPopCurrent()，使之前的当前上下文再次成为最新。

上下文必须是“不固定的”，即不附加到任何线程。可通过调用cuCtxPopCurrent()使上下文成为不固定的。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuCtxCreate，cuCtxAttach，cuCtxDetach，cuCtxDestroy，cuCtxPopCurrent

2.3.7 cuCtxSynchronize

名称

cuCtxSynchronize – 阻塞操作，直至上下文的任务完成

概要

CUresult cuCtxSynchronize(void);

说明

在设备完成之前请求的所有任务之前阻塞操作。如果之前的某项任务失败，cuCtxSynchronize()将返回一个错误。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuCtxCreate，cuCtxAttach，cuCtxDetach，cuCtxGetDevice

2.4 ModuleManagement

名称

模块管理

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuModule Get Function

cuModuleGetGlobal

cuModuleGetTexRef

cuModuleLoad

cuModuleLoadData

cuModuleLoadFatBinary

cuModuleUnload

参见

I初始化，设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.4.1 cuModuleGetFunction

名称

cuModuleGetFunction – 返回函数句柄

概要

CUresult cuModuleGetFunction(CUfunction* func，CUmodule mod，const char* funcname);

说明

以*func的形式返回位于模块mod中名为funcname的函数的句柄。如果不存在具有此名称的函数，cuModuleGetFunction()将返回CUDA_ERROR_NOT_FOUND。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE
CUDA_ERROR_NOT_FOUND

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuModuleLoad，cuModuleLoadData，cuModuleLoadFatBinary，cuModuleUnload，cuModuleGetGlobal，cuModuleGetTexRef

2.4.2 cuModuleGetGlobal

名称

cuModuleGetGlobal – 从模块返回一个全局指针

概要

CUresult cuModuleGetGlobal(CUdeviceptr* devPtr，unsigned int* bytes，CUmodule mod，const char* globalname);

说明

分别以*devPtr和*bytes的形式返回模块mod中名称为globalname的全局变量的基址指针和大小。如果不存在具有此名称的变量，则cuModuleGetGlobal()将返回CUDA_ERROR_NOT_FOUND。参数devPtr和bytes都是可选的。如果其中任一参数的值为空，则该参数将被忽略。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_NOT_FOUND

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuModuleLoad，cuModuleLoadData，cuModuleLoadFatBinary，cuModuleUnload，cuModuleGetFunction，cuModuleGetTexRef

2.4.3 cuModuleGetTexRef

名称

cuModuleGetTexRef – 获取纹理引用的句柄

概要

CUresult cuModuleGetTexRef(CUtexref* texRef，CUmodule hmod，const char* texrefname);

说明

以*texref的形式返回模块mod中名称为texrefname的纹理引用。如果不存在具有此名称的纹理引用，cuModuleGetTexRef()将返回CUDA_ERROR_NOT_FOUND。此纹理引用句柄不应该被销毁，它将在模块卸载时被销毁。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE
CUDA_ERROR_NOT_FOUND

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuModuleLoad，cuModuleLoadData，cuModuleLoadFatBinary，cuModuleUnload，cuModuleGeFunction，cuModuleGetGlobal

2.4.4 cuModuleLoad

名称

cuModuleLoad – 载入计算模块

概要

CUresult cuModuleLoad(CUmodule* mod，const char* filename);

说明

接受一个文件名filename，将对应的模块mod载入当前上下文。CUDA驱动程序API不会试图延迟分配模块所需的资源，如果无法为模块所需的函数和数据分配存储器（不变存储器和全局存储器），cuModuleLoad()将失败。文件应为nvcc输出的cubin文件。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_NOT_FOUND

CUDA_ERROR_OUT_OF_MEMORY

CUDA_ERROR_FILE_NOT_FOUND

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuModuleLoadData，cuModuleLoadFatBinary，cuModuleUnload，cuModuleGetFunction，cuModuleGetGlobal，cuModuleGetTexRef

2.4.5 cuModuleLoadData

名称

cuModuleLoadData – 载入模块的数据

概要

CUresult cuModuleLoadData(CUmodule* mod，const void* image);

说明

获得指针image，并将对应模块mod载入当前上下文。指针的获得方法如下：映射一个cubin文件、将cubin文件作为文本字符串传递，或者将一个cubin对象整合到可执行资源之中，并利用WindowsâĂŹ FindResource()等操作系统调用来获取指针。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuModuleLoad，cuModuleLoadFatBinary，cuModuleUnload，cuModuleGet Function，cuModuleGetGlobal，cuModuleGetTexRef

2.4.6 cuModuleLoadFatBinary

名称

cuModuleLoadFatBinary – 载入一个胖二进制对象

概要

CUresult cuModuleLoadFatBinary(CUmodule* mod，const void* fatBin);

说明

获取指针fatBin，并将对应模块mod载入当前上下文。该指针表示一个胖二进制对象，这是多个cubin文件的集合，均表示相同的设备码，但针对不同的架构而编译和优化。当前尚无已归档的API可供程序员用于构建和使用胖二进制对象，因而，此函数是该版本CUDA的一个内部函数。可在nvcc文档中找到更多信息。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_NOT_FOUND

CUDA_ERROR_OUT_OF_MEMORY

CUDA_ERROR_NO_BINARY_FOR_GPU

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuModuleLoad，cuModuleLoadData，cuModuleUnload，cuModuleGetFunction，cuModuleGetGlobal，cuModuleGetTexRef

2.4.7 cuModuleUnload

名称

cuModuleUnload – 卸载一个模块

概要

CUresult cuModuleUnload(CUmodule mod);

说明

将模块mod从当前上下文中卸载。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuModuleLoad，cuModuleLoadData，cuModuleLoadFatBinary，cuModuleGetFunction，cuModuleGetGlobal，cuModuleGetTexRef

2.5 StreamManagement

名称

流管理

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuStreamCreate

cuStreamDestroy

cuStreamQuery

cuStreamSynchronize

参见

设备管理，设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.5.1 cuStreamCreate

名称

cuStreamCreate – 创建一个流

概要

CUresult cuStreamCreate(CUstream* stream，unsigned int flags);

说明

创建一个流。目前flags必须为0。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID_VALUE
CUDA_ERROR_OUT_OF_MEMORY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuSt ream Query，cuStreamSynchronize，cuStreamDestroy

2.5.2 cuStreamDestroy

名称

cuStreamDestroy – 销毁一个流

概要

CUresult cuStreamDestroy(CUstream stream);

说明

销毁流。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID_HANDLE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuSt reamCreate，cuStreamQuery，cuStreamSynchronize

2.5.3 cuStreamQuery

名称

cuStreamQuery – 确定计算流的状态

概要

CUresult cuStreamQuery(CUstream stream);

说明

如果流中的所有操作都已完成，返回CUDA_SUCCESS，否则返回CUDA_ERROR_NOT_READY。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_HANDLE
CUDA_ERROR_NOT_READY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuStreamCreate，cuStreamSynchronize，cuStreamDestroy

2.5.4 cuStreamSynchronize

名称

cuStreamSynchronize – 在流任务完成之前阻塞操作

概要

CUresult cuStreamSynchronize (CUstream stream);

说明

在设备完成流中的所有操作之前，阻塞操作。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID_HANDLE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuStreamCreate，cuStreamQuery，cuStreamDestroy

2.6 EventManagement

名称

事件管理

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuEventCreate

cuEventDestroy

cuEventElapsedTime

cuEventQuery

cuEventRecord

cuEventSynchronize

参见

初始化，设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.6.1 cuEventCreate

名称

cuEvent Create – 创建事件

概要

CUresult cuEventCreate(CUevent* event，unsigned int flags);

说明

创建事件。目前flags必须为0。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID_VALUE
CUDA_ERROR_OUT_OF_MEMORY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuEventRecord，cuEventQuery，cuEventSynchronize，cuEventDestroy，cuEventElapsedTime

2.6.2 cuEventDestroy

名称

cuEventDestroy – 销毁事件

概要

CUresult cuEventDestroy(CUevent event);

说明

销毁事件。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID_HANDLE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuEventCreate，cuEventRecord，cuEventQuery，cuEventSynchronize，cuEventElapsedTime

2.6.3 cuEventElapsedTime

名称

cuEventElapsedTime – 计算两次事件之间经过的时间

概要

CUresult cuEventDestroy(float* time，CUevent start，CUevent end);

说明

计算两次事件之间经过的时间（以毫秒为单位，分离度约为0.5微秒）。如果其中任何一个事件尚未被记录，此函数将返回CUDA_ERROR_INVALID_VALUE。如果其中任何一个事件使用非零流进行了记录，则结果不确定。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuEventCreate，cuEventRecord，cuEventQuery，cuEventSynchronize，cuEventDestroy

2.6.4 cuEventQuery

名称

cuEvent Query – 查询事件状态

概要

CUresult cuEventQuery (CUevent event);

说明

如果事件确实已被记录，返回CUDA_SUCCESS，否则返回CUDA_ERROR_NOT_READY。如果尚未在此事件上调用cuEventRecord()，该函数将返回CUDA_ERROR_INVALID_VALUE。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_READY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuEventCreate，cuEventRecord，cuEventSynchronize，cuEventDestroy，cuEventElapsedTime

2.6.5 cuEventRecord

名称

cuEvent Record – 记录事件

概要

CUresult cuEventRecord(CUevent event，CUstream stream);

说明

记录事件。如果stream非空，事件将在流中之前的所有操作完成后被记录；否则，它将在CUDA上下文中之前的所有操作完成后被记录。由于此操作是异步的，必须使用cuEventQuery()和/或cuEventSynchronize()来确定事件的实际记录时间。

如果之前已经调用过cuEventRecord()，而事件尚未被记录，则此函数将返回CUDA_ERROR_INVALID_VALUE。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_HANDLE
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuEventCreate，cuEventQuery，cuEventSynchronize，cuEventDestroy，cuEventElapsedTime

2.6.6 cuEventSynchronize

名称

cuEvent Synchronize – 等待事件完成

概要

CUresult cuEventSynchronize (CUevent event);

说明

在事件被记录之前阻塞操作。如果尚未对此事件调用cuEventRecord()，该函数将返回CUDA_ERROR_INVALID_VALUE。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_HANDLE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuEventCreate，cuEventRecord，cuEventQuery，cuEventDestroy，cuEventElapsedTime

2.7 ExecutionControl

名称

执行控制

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuLaunch

cuLaunchG rid

cuParamSetSize

cuParamSetTexRef

cuParamSetf

cuParamSeti

cuParamSetv

cuFuncSetBlockShape

cuFuncSetSharedSize

参见

初始化，设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.7.1 cuLaunch

名称

cuLaunch – 启动CUDA函数

概要

CUresult cuLaunch(CUfunction func);

说明

在块的1ÃŮ1网格上调用内核func。该块包含此前通过调用cuFuncSetBlockShape()指定的多个线程。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_LAUNCH_INCOMPATIBLE_TEXTURING

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetBlockShape，cuFuncSetSharedSize，cuParamSetSize，cuParamSeti，cuParamSetf，cuParamSetv，cuParamSetTexRef，cuLaunchGrid

2.7.2 cuLaunchGrid

名称

cuLaunchGrid – 启动CUDA函数

概要

CUresult cuLaunchGrid(CUfunction func，int grid_width，int grid_height);

CUresult cuLaunchGridAsync(CUfunction func，int grid_width，int grid_height，CUstream stream);

说明

在大小为grid_width x grid_height的块网格上调用内核。每个块包含此前通过调用cuFuncSetBlockShape()指定的多个线程。

cuLaunchGridAsync()是可选的，可通过传入非零stream参数关联到一个流。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_LAUNCH_INCOMPATIBLE_TEXTURING

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetBlockShape，cuFuncSetSharedSize，cuParamSetSize，cuParamSeti，cuParamSetf，cuParamSetv，cuParamSetTexRef，cuLaunch

2.7.3 cuParamSetSize

名称

cuParamSetSize – 设置函数的参数大小

概要

CUresult cuParamSetSize(CUfunction func，unsigned int numbytes);

说明

通过numbytes设置函数func的函数参数所需的总大小，以字节为单位。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetBlockShape，cuFuncSetSharedSize，cuParamSeti，cuParamSetf，cuParamSetv，cuParamSetTexRef，cuLaunch，cuLaunchGrid

2.7.4 cuParamSetTexRef

名称

cuParamSetTexRef – 为函数的参数列表添加纹理引用

概要

CUresult cuParamSetTexRef(CUfunction func，int texunit，CUtexref texRef);

说明

将CUDA数组或线性存储器绑定到纹理引用texRef，texRef可作为纹理供设备程序使用。在该版本的CUDA中，纹理引用必须通过cuModuleGetTexRef()获取，textunit参数必须设置为CU_PARAM_TR_DEFAULT。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetBlockShape，cuFuncSetSharedSize，cuParamSetSize，cuParamSeti，cuParamSetf，cuParamSetv，cuLaunch，cuLaunchGrid

2.7.5 cuParamSetf

名称

cuParamSetf – 为函数的参数列表添加浮点参数

概要

CUresult cuParamSetf(CUfunction func，int offset，float value);

说明

设置一个浮点参数，该参数将在下一次调用对应于func的内核时指定。Offset是字节偏移量。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetBlockShape，cuFuncSetSharedSize，cuParamSetSize，cuParamSeti，cuParamSetv，cuParamSetTexRef，cuLaunch，cuLaunchGrid

2.7.6 cuParamSeti

名称

cuParamSeti – 为函数的参数列表添加一个整型参数

概要

CUresult cuParamSeti(CUfunction func，int offset，unsigned int value);

说明

设置一个整型参数，该参数将在下一次调用对应于func的内核时指定。Offset是字节偏移量。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetBlockShape，cuFuncSetSharedSize，cuParamSetSize，cuParamSetf，cuParamSetv，cuParamSetTexRef，cuLaunch，cuLaunchGrid

2.7.7 cuParamSetv

名称

cuParamSetv – 为函数的参数列表添加随机数据

概要

CUresult cuParamSetv(CUfunction func，int offset，void* ptr，unsigned int numbytes);

说明

将随机数量的数据复制到对应于func的内核的参数空间之中。Offset是字节偏移量。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetBlockShape，cuFuncSetSharedSize，cuParamSetSize，cuParamSeti，cuParamSetf，cuParamSetTexRef，cuLaunch，cuLaunchGrid

2.7.8 cuFuncSetBlockShape

名称

cuFuncSetBlockShape – 为函数设置块维度

概要

CUresult cuFuncSetBlockShape(CUfunction func，int x，int y，int z);

说明

指定在func给定的内核启动时所创建的线程块的X、Y和Z维度。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_HANDLE
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetSharedSize，cuParamSetSize，cuParamSeti，cuParamSetf，cuParamSetv，cuParamSetTexRef，cuLaunch，cuLaunchGrid

2.7.9 cuFuncSetSharedSize

名称

cuFuncSetSharedSize – 为函数设置共享存储器的大小

概要

CUresult cuFuncSetSharedSize(CUfunction func，unsigned int bytes);

说明

通过bytes设置在func给定的内核启动时各线程块可以使用的共享存储器数量。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_HANDLE
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuFuncSetBlockShape，cuParamSetSize，cuParamSeti，cuParamSetf，cuParamSetv，cuParamSetTexRef，cuLaunch，cuLaunchGrid

2.8 MemoryManagement

名称

存储器管理

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuArrayCreate

cuArray3Dcreate

cuArrayDestroy

cuArrayGetDescriptor

cuArray3DGetDescriptor

cuMemAlloc

cuMemAllocHost

cuMemAllocPitch

cuMemFree

cuMemFreeHost

cuMemGetAddressRange

cuMemGetInfo

cuMemcpy2D

cuMemcpy3D

cuMemcpyAtoA

cuMemcpyAtoD

cuMemcpyAtoH

cuMemcpyDtoA

cuMemcpyDtoD

cuMemcpyDtoH

cuMemcpyHtoA

cuMemcpyHtoD

cuMemset

cuMemset2D

参见

初始化，设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.8.1 cuArrayCreate

名称

cuArrayCreate – 创建1D或2D CUDA数组

概要

CUresult cuArrayCreate(CUarray* array，const CUDA_ARRAY_DESCRIPTOR* desc);

说明

根据CUDA_ARRAY_DESCRIPTOR结构体desc创建一个CUDA数组，并以*array的形式返回新CUDA数组的句柄。CUDA_ARRAY_DESCRIPTOR结构体定义如下：

```
typedef struct {
    unsigned int Width;
    unsigned int Height;
    CUarray_format Format; unsigned int
  NumChannels; }
  CUDA_ARRAY_DESCRIPTOR;
```

其中：

- Width和Height是CUDA数组的宽度和高度（以元素为单位）；如果高度为0，则该CUDA数组是一维数组，否则是二维数组；
- NumChannels指定每个CUDA数组元素的打包组件的数量，可能是1、2或4；
- Format指定元素的格式；CUarray_format定义如下：

```
typedef enum CUarray_format_enum
    { CU_AD_FORMAT_UNSIGNED_INT8 = 0x01，
    CU_AD_FORMAT_UNSIGNED_INT16 = 0x02，
    CU_AD_FORMAT_UNSIGNED_INT32 = 0x03，
    CU_AD_FORMAT_SIGNED_INT8 = 0x08，
    CU_AD_FORMAT_SIGNED_INT16 = 0x09，
    CU_AD_FORMAT_SIGNED_INT32 = 0x0a，
    CU_AD_FORMAT_HALF = 0x 10,
    CU_AD_FORMAT_FLOAT = 0x20
} CUarray_format;
```

下面是几个CUDA数组描述的例子：

- 描述具有2048个浮点的CUDA数组：

```
CUDA_ARRAY_DESCRIPTOR desc; desc.Format =
CU_AD_FORMAT_FLOAT;
```

```
desc.NumChannels = 1; desc.Width =
2048; desc.Height = 1;
```

- 描述64 x 64 CUDA浮点数组：

```
CUDA_ARRAY_DESCRIPTOR desc; desc.Format =
CU_AD_FORMAT_FLOAT; desc.NumChannels = 1;
desc.Width = 64;
desc.Height = 64;
```

- 描述包含64位、4x16位16进制浮点数据的width x height CUDA数组：

```
CUDA_ARRAY_DESCRIPTOR desc;
desc.FormatFlags = CU_AD_FORMAT_HALF;
desc.NumChannels = 4; desc.Width =
width; desc.Height = height;
```

- 描述包含16位元素的width x height CUDA数组，每个元素都包含两个8位的无符号字符：

```
CUDA_ARRAY_DESCRIPTOR arrayDesc;
desc.FormatFlags = CU_AD_FORMAT_UNSIGNED_INTS;
desc.NumChannels = 2; desc.Width =
width; desc.Height = height;
```

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

CUDA_ERROR_UNKNOWN

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemGetInfo，cuMemAlloc，cuMemAllocPitch，cuMemFree，cuMemAllocHost，cuMemFreeHost，cuMemGetAddressRange，cuArrayGetDescriptor，cuArrayDestroy，cuMemset，cuMemset2D

2.8.2 cuArrayDestroy

名称

cuArrayDestroy – 销毁CUDA数组

概要

CUresult cuArrayDestroy(CUarray array);

说明

销毁CUDA数组array。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID_HANDLE CUDA_ERROR_ARRAY_IS_MAPPED

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemGetInfo，cuMemAlloc，cuMemAllocPitch，cuMemFree，cuMemAllocHost，cuMemFreeHost，cuMemGetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuMemset，cuMemset2D

2.8.3 cuArrayGetDescriptor

名称

cuArrayGetDescriptor – 获取1D或2D CUDA数组的描述符

概要

CUresult cuArrayGetDescriptor (CUDA_ARRAY_DESCRIPTOR* arrayDesc，CUarray array)

说明

以*arrayDesc的形式返回1D或2D CUDA数组array的格式描述符和维度描述符。这对于已传递了CUDA数组的子例程来说非常有用，但需要知道CUDA数组的参数，以便进行验证或用于其他目的。如果数组是3D的，此函数将返回CUDA_ERROR_INVALID_VALUE。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE CUDA_ERROR_INVALID_HANDLE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuArrayCreate，cuArray3DCreate，cuArray3DGetDescriptor，cuArrayDestroy

2.8.4 cuMemAlloc

名称

cuMemAlloc – 分配设备存储器

概要

CUresult cuMemAlloc(CUdeviceptr* devPtr，unsigned int count);

说明

向设备分配count字节的线性存储器，并以*devPtr的形式返回指向已分配存储器的指针。已分配的存储器可与任何类型的变量对齐。存储器不会被清除。如果count为0，cuMemAlloc()将返回CUDA_ERROR_INVALID_VALUE。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemGetInfo，cuMemAllocPitch，cuMemFree，cuMemAllocHost，cuMemFreeHost，cuMemGetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset，cuMemset2D

2.8.5 cuMemAllocHost

名称

cuMemAllocHost – 分配分页锁定的主存储器

概要

CUresult cuMemAllocHost(void** hostPtr，unsigned int count);

说明

分配count字节的主存储器，这部分存储器是分页锁定的，可由设备访问。驱动程序会跟踪由此函数分配的虚拟存储器，自动加速对cudaMemcpy()等函数的调用。由于设备可直接访问存储器，因而与malloc()等函数分配的可分页存储器相比，在读取或写入时的带宽更高。使用cudaMallocHost()分配过多存储器可能导致系统性能降级，因为这会减少系统可用于分页的存储器数量。因而，最好少使用此函数，一般只用于为主机和设备之间的数据交换分配存储器。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemGetInfo，cuMemAlloc，cuMemAllocPitch，cuMemFree，cuMemFreeHost，cuMemGetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset，cuMemset2D

2.8.6 cuMemAllocPitch

名称

cuMemAllocPitch – 分配设备存储器

概要

CUresult cuMemAllocPitch(CUdeviceptr* devPtr，unsigned int* pitch，unsigned int widthInBytes，unsigned int height，unsigned int elementSizeBytes);

说明

向设备分配至少widthInBytes*height字节的线性存储器，并以*devPtr的形式返回指向已分配存储器的指针。函数可填充所分配的存储器，确保在地址从一行更新到另一行时，给定行中的对应指针依然满足对齐要求。elementSizeBytes指定可在此存储器范围上执行的最大读取和写入的规模。elementSizeBytes可以是4、8、16（因为合并存储器事务不可能存在其他数据规模）。如果elementSizeBytes比内核的实际读/写规模小，内核将正确运行，但有可能速度较慢。cudaMallocPitch()以*pitch的形式返回间距，也就是所分配存储器的宽度（以字节为单位）。间距旨在用作存储器分配的一个独立参数，用于在2D数组内计算地址。给定类型T数组元素的行和列，可按如下方法计算地址：

```
T* pElement = (T*)((char*)BaseAddress + Row * Pitch) + Column;
```

cuMemAllocPitch()返回的间距在所有条件下都可保证在cuMemcpy2D()中有效。对于2D数组的分配，建议程序员考虑使用cudaMallocPitch()来执行间距分配。由于硬件中存在间距对齐限制，如果应用程序将在设备存储器的不同区域之间执行2D存储器复制（无论是线性存储器还是CUDA数组），这种方法将非常有用。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemGetInfo，cuMemAlloc，cuMemFree，cuMemAllocHost，cuMemFreeHost，cuMemGetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset，cuMemset2D

2.8.7 cuMemFree

名称

cuMemFree – 释放设备存储器

概要

CUresult cuMemFree(CUdeviceptr devPtr);

说明

释放devPtr指向的存储器空间，该devPtr必须在之前的cuMemMalloc()或cuMemMallocPitch()调用时返回。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMem GetInfo，cuMemAlloc，cuMemAllocPitch，cuMemAllocHost，cuMemFreeHost，cuMemGetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset，cuMemset2D

2.8.8 cuMemFreeHost

名称

cuMemFreeHost – 释放分页锁定的主存储器

概要

CUresult cuMemFreeHost(void* hostPtr);

说明

释放hostPtr指向的存储器空间，该hostPtr必须在之前的cudaMallocHost()调用时返回。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMem GetInfo，cuMemAlloc，cuMemAllocPitch，cuMemFree，cuMemAllocHost，cuMem GetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset，cuMemset2D

2.8.9 cuMemGetAddressRange

名称

cuMemGetAddressRange – 获取关于存储器分配的信息

概要

CUresult cuMemGetAddressRange(CUdeviceptr* basePtr，unsigned int* size，CUdeviceptr devPtr);

说明

以*basePtr的形式返回cuMemAlloc()或cuMemAllocPitch()所分配存储器的基址，并以*size的形式返回其大小，这些函数包含输入指针devPtr。参数basePtr和size都是可选的。如果其中任何一个为空，则将被忽略。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemGetInfo，cuMemAlloc，cuMemAllocPitch，cuMemFree，cuMemAllocHost，cuMemFreeHost，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset，cuMemset2D

2.8.10 cuMemGetInfo

名称

cuMemGetInfo – 获取存储器的空闲量和总量

概要

CUresult cuMemGetInfo(unsigned int* free，unsigned int* total);

说明

分别以*free和*total的形式返回可供CUDA上下文分配的存储器空闲量和总量，以字节为单位。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemAlloc，cuMemAllocPitch，cuMemFree，cuMemAllocHost，cuMemFreeHost，cuMemGetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset，cuMemset2D

2.8.11 cuMemcpy2D

名称

cuMemcpy2D – 为2D数组复制存储器

概要

CUresult cuMemcpy2D (const CUDA_MEMCPY2D* copyParam);

CUresult cuMemcpy2DUnaligned (const CUDA_MEMCPY2D* copyParam);

CUresult cuMemcpy2DAsync(const CUDA_MEMCPY2D* copyParam，CUstream stream);

说明

根据copyParam中指定的参数执行2D存储器复制。CUDA_MEMCPY2D结构体定义如下：

```
typedef struct CUDA_MEMCPY2D_st { unsigned
    int srcXInBytes，srcY; CUmemorytype
    srcMemoryType; const void *srcHost;
    CUdeviceptr srcDevice;
    CUarray srcArray;
    unsigned int srcPitch;
    unsigned int dstXInBytes，dstY; CUmemorytype
    dstMemoryType; void *dstHost;
    CUdeviceptr dstDevice;
    CUarray dstArray;
    unsigned int dstPitch;
    unsigned int WidthInBytes; unsigned int
    Height;
} CUDA_MEMCPY2D;
```

其中：

- srcMemoryType和dstMemoryType分别指定源和目标的存储器类型。Cumemorytype_enum定义如下：

```
typedef enum CUmemorytype_enum
    { CU_MEMORYTYPE_HOST = 0x01，
    CU_MEMORYTYPE_DEVICE = 0x02，
    CU_MEMORYTYPE_ARRAY = 0x03
} CUmemorytype;
```

如果srcMemoryType是CU_MEMORYTYPE_HOST，srcHost和srcPitch指定源数据的（主机）基址和每行要应用的字节。srcArray被忽略。

如果srcMemoryType是CU_MEMORYTYPE_DEVICE，srcDevice和srcPitch指定源数据的（设备）基址和每行要应用的字节。srcArray被忽略。

如果srcMemoryType是CU_MEMORYTYPE_ARRAY，srcArray指定源数据的句柄。srcHost、srcDevice和srcPitch被忽略。

如果dstMemoryType是CU_MEMORYTYPE_HOST，dstHost和dstPitch指定目标数据的（主机）基址和每行要应用的字节。dstArray被忽略。

如果dstMemoryType是CU_MEMORYTYPE_DEVICE，dstDevice和dstPitch指定目标数据的（设备）基址和每行要应用的字节。dstArray被忽略。

如果dstMemoryType是CU_MEMORYTYPE_ARRAY，dstArray指定目标数据的句柄。dstHost、dstDevice和dstPitch被忽略。

- srcXInBytes和srcY为复制指定源数据的基址。对于主机指针，开始地址为：

```
void* Start = (void*)((char*)srcHost+srcY*srcPitch + srcXInBytes);
```

对于设备指针，开始地址为

```
CUdeviceptr Start = srcDevice+srcY*srcPitch+srcXInBytes;
```

对于CUDA数组，srcXInBytes必须能被数组的元素数量整除。

- dstXInBytes和dstY为复制指定目标数据的基址。对于主机指针，基址为：

```
void* dstStart = (void*)((char*)dstHost+dstY*dstPitch + dstXInBytes);
```

对于设备指针，开始地址为

```
CUdeviceptr dstStart = dstDevice+dstY*dstPitch+dstXInBytes;
```

对于CUDA数组，dstXInBytes必须能被数组的元素大小整除。

- WidthInBytes和Height指定要执行的2D复制的宽度（以字节为单位）和高度。任何间距都必须大于等于WidthInBytes。

如果任何间距大于允许的最大值（CU_DEVICE_ATTRIBUTE_MAX_PITCH），cuMemcpy2D()将返回一个错误，cuMemAllocPitch()传递回的间距对cuMemcpy2D()总是有效的。在设备存储器储器复制中（device ? device，CUDA array? device，CUDA array? CUDA array），cuMemcpy2D()可能会因为cuMemAllocPitch()未计算间距而失败。cuMemcpy2DUnaligned()没有这样的限制，但在cuMemcpy2D()会返回错误码的情况下，其运行速度可能会非常缓慢。

cuMemcpy2DAsync()是异步的，可通过传入非零stream参数关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyHtoD，cuMemcpyDtoH，cuMemcpyDtoD，cuMemcpyDtoA，cuMemcpyAtoD，cuMemcpyAtoH，cuMemcpyHtoA，cuMemcpyAtoA，cuMemcpy3D

2.8.12 cuMemcpy3D

名称

cuMemcpy3D – 为3D数组复制存储器

概要

CUresult cuMemcpy3D (const CUDA_MEMCPY3D* copyParam);

CUresult cuMemcpy3DAsync(const CUDA_MEMCPY3D* copyParam，CUstream stream);

说明

根据copyParam指定的参数执行3D存储器复制。CUDA_MEMCPY3D结构体定义如下：

```
typedef struct CUDA_MEMCPY3D_st {
unsigned int srcXInBytes，srcY，srcZ;
unsigned int srcLOD;
CUmemorytype srcMemoryType;
const void *srcHost;
CUdeviceptr srcDevice;
CUarray srcArray;
unsigned int srcPitch; // ignored when src is array
unsigned int srcHeight; // ignored when src is array; may be 0 if Depth==1
unsigned int dstXInBytes，dstY，dstZ;
unsigned int dstLOD;
CUmemorytype dstMemoryType;
void *dstHost;
CUdeviceptr dstDevice;
CUarray dstArray;
unsigned int dstPitch; // ignored when dst is array
unsigned int dstHeight; // ignored when dst is array; may be 0 if Depth==1
unsigned int WidthInBytes;
unsigned int Height;
unsigned int Depth;
} CUDA_MEMCPY3D;

CUresult CUDAAPI cuMemcpy3D( const CUDA_MEMCPY3D *pCopy );
```

其中：

- srcMemoryType和dstMemoryType分别指定源和目标存储器的类型。Cumemorytype_enum定义如下：

  typedef enum CUmemorytype_enum {

```
    CU_MEMORYTYPE_HOST =
    0x01，
    CU_MEMORYTYPE_DEVICE =
    0x02，
    CU_MEMORYTYPE_ARRAY =
    0x03
} CUmemorytype;
```

如果srcMemoryType是CU_MEMORYTYPE_HOST，srcHost、srcPitch和srcHeight指定源数据的（主机）基址和3D数组各2D部分的高度。srcArray被忽略。

如果srcMemoryType是CU_MEMORYTYPE_DEVICE，srcDevice、srcPitch和srcHeight指定源数据的（设备）基址、每行的字节数和3D数组中每个2D部分的高度。srcArray被忽略。

如果srcMemoryType是CU_MEMORYTYPE_ARRAY，srcArray指定源数据的句柄。srcHost、srcDevice、srcPitch和srcHeight被忽略。

如果dstMemoryType是CU_MEMORYTYPE_HOST，dstHost和dstPitch指定目标数据的（主机）基址、每行的字节数和3D数组中每个2D部分的高度。dstArray被忽略。

如果dstMemoryType是CU_MEMORYTYPE_DEVICE，dstDevice和dstPitch指定目标数据的（设备）基址、每行的字节数和3D数组中每个2D部分的高度。dstArray被忽略。

如果dstMemoryType是CU_MEMORYTYPE_ARRAY，dstArray指定目标设备的句柄。dstHost、dstDevice、dstPitch和dstHeight被忽略。

- srcXInBytes、srcY和srcZ指定复制源数据的基址。

对于主机指针，开始地址为：

```
void* Start = (void*)((char*)srcHost+(srcZ*srcHeight+srcY)*srcPitch + srcXInBytes);
```

对于设备指针，开始地址为

```
CUdeviceptr Start = srcDevice+(srcZ*srcHeight+srcY)*srcPitch+srcXInBytes;
```

对于CUDA数组，srcXInBytes必须能被数组元素大小整除。

- dstXInBytes、dstY和dstZ指定复制目标数据的基址。

对于主机指针，基址为：

```
void* dstStart = (void*)((char*)dstHost+(dstZ*dstHeight+dstY)*dstPitch + dstXInBytes);
```

对于设备指针，开始地址为

```
CUdeviceptr dstStart = dstDevice+(dstZ*dstHeight+dstY)*dstPitch+dstXInBytes;
```

对于CUDA数组，dstXInBytes必须可被数组元素的大小整除。

- WidthInBytes、Height和Depth指定要执行的3D复制的宽度（以字节为单位）、高度和深度。任何间距都必须大于等于WidthInBytes。

如果任何间距超过所允许的最大值（CU_DEVICE_ATTRIBUTE_MAX_PITCH），cuMemcpy3D()将返回一个错误。

cuMemcpy3DAsync()是异步的，可通过传入非零stream参数将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

CUDA_MEMCPY3D结构体的srcLOD和dstLOD成员必须设置为0。

返回值

CUDA SUCCESS

参见

cuMemcpyHtoD，cuMemcpyDtoH，cuMemcpyDtoD，cuMemcpyDtoA，cuMemcpyAtoD，cuMemcpyAtoH，cuMemcpyHtoA，cuMemcpyAtoA，cuMemcpy2D，cuMemcpy2DAsync

2.8.13 cuMemcpyAtoA

名称

cuMemcpyAtoA – 在数组之间复制存储器

概要

CUresult cuMemcpyAtoA(CUarray dstArray，unsigned int dstIndex，CUarray srcArray，unsigned int srcIndex，unsigned int count);

说明

从一个1D CUDA数组复制到另一个1D数组。dstArray和srcArray分别指定复制的目标和源CUDA数组的句柄。dstIndex和srcIndex指定CUDA数组的目标和源索引。对于CUDA数组，这些值的范围是[0，Width-1]，它们不是字节偏移量。count是要复制的字节数。CUDA数组中的元素大小的格式不必相同，但元素必须具有相同的大小，count必须能被这个大小整除。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyHtoD，cuMemcpyDtoH，cuMemcpyDtoD，cuMemcpyDtoA，cuMemcpyAtoD，cuMemcpyAtoH，cuMemcpyHtoA，cuMemcpy2D

2.8.14 cuMemcpyAtoD

名称

cuMemcpyAtoD – 将存储器从数组复制到设备

概要

CUresult cuMemcpyAtoD(CUdeviceptr dstDevPtr，CUarray srcArray，unsigned int srcIndex，unsigned int count);

说明

从1D CUDA数组复制到设备存储器。dstDevPtr指定目标的基址指针，必须与CUDA数组元素对齐。srcArray和srcIndex指定CUDA数组的句柄以及表示复制起始位置数组元素的索引（以数组元素的形式表示）。Count表示要复制的字节数，必须能被数组元素大小整除。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyHtoD，cuMemcpyDtoH，cuMemcpyDtoD，cuMemcpyDtoA，cuMemcpyAtoH，cuMemcpyHtoA，cuMemcpyAtoA，cuMemcpy2D

2.8.15 cuMemcpyAtoH

名称

cuMempcyAtoH – 将存储器从数组复制到主机

概要

CUresult cuMemcpyAtoH(void* dstHostPtr，CUarray srcArray，unsigned int srcIndex，unsigned int count);

CUresult cuMemcpyAtoHAsync(void* dstHostPtr，CUarray srcArray，unsigned int srcIndex，unsigned int count，CUstream stream);

说明

从1D CUDA数组复制到主存储器。dstHostPtr指定目标的基址指针。srcArray和srcIndex指定CUDA数组句柄和源数据的起始索引。Count指定要复制的字节数。

cuMemcpyAtoHAsync()是异步的，可通过传入一个非零的stream参数将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyHtoD，cuMemcpyDtoH，cuMemcpyDtoD，cuMemcpyDtoA，cuMemcpyAtoD，cuMemcpyHtoA，cuMemcpyAtoA，cuMemcpy2D

2.8.16 cuMemcpyDtoA

名称

cuMemcpyDtoA – 将存储器从设备复制到数组

概要

CUresult cuMemcpyDtoA(CUarray dstArray，unsigned int dstIndex，CUdeviceptr srcDevPtr，unsigned int count);

说明

从设备存储器复制到1D CUDA数组。dstArray和dstIndex指定CUDA数组句柄和目标数据的起始索引。srcDevPtr指定源的基址指针。Count指定要复制的字节数。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyHtoD，cuMemcpyDtoH，cuMemcpyDtoD，cuMemcpyAtoD，cuMemcpyAtoH，cuMemcpyHtoA，cuMemcpyAtoA，cuMemcpy2D

2.8.17 cuMemcpyDtoD

名称

cuMemcpyDtoD – 设备间的存储器复制

概要

CUresult cuMemcpyDtoD(CUdeviceptr dstDevPtr，CUdeviceptr srcDevPtr，unsigned int count);

说明

在设备存储器之间进行复制。dstDevice和srcDevPtr分别是目标和源的基址指针。Count指定要复制的字节数。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyHtoD，cuMemcpyDtoH，cuMemcpyDtoA，cuMemcpyAtoD，cuMemcpyAtoH，cuMemcpyHtoA，cuMemcpyAtoA，cuMemcpy2D

2.8.18 cuMemcpyDtoH

名称

cuMemcpyDtoH – 将存储器从设备复制到主机

概要

CUresult cuMemcpyDtoH(void* dstHostPtr，CUdeviceptr srcDevPtr，unsigned int count);

CUresult cuMemcpyDtoHAsync(void* dstHostPtr，CUdeviceptr srcDevPtr，unsigned int count，CUstream stream);

说明

从设备复制到主机存储器。dstHostPtr和srcDevPtr分别指定源和目标的基址指针。Count指定要复制的字节数。

MemcpyDtoHAsync()是异步的，可通过传递一个非零的stream参数将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyHtoD，cuMemcpyDtoD，cuMemcpyDtoA，cuMemcpyAtoD，cuMemcpyAtoH，cuMemcpyHtoA，cuMemcpyAtoA，cuMemcpy2D

2.8.19 cuMemcpyHtoA

名称

cuMemcpyHtoA – 将存储器从主机复制到数组

概要

CUresult cuMemcpyHtoA(CUarray dstArray，unsigned int dstIndex，const void *srcHostPtr，unsigned int count);

CUresult cuMemcpyHtoAAsync(CUarray dstArray，unsigned int dstIndex，const void *srcHostPtr，unsigned int count，CUstream stream);

说明

从主机存储器复制到一个1D CUDA数组。dstArray和dstIndex指定CUDA数组句柄和目标数据的起始索引。srcHostPtr指定源的基址。Count指定要复制的字节数。

cuMemcpyHtoAAsync()是异步的，可通过传递一个非零的stream参数将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyHtoD，cuMemcpyDtoH，cuMemcpyDtoD，cuMemcpyDtoA，cuMemcpyAtoD，cuMemcpyAtoH，cuMemcpyAtoA，cuMemcpy2D

2.8.20 cuMemcpyHtoD

名称

cuMemcpyHtoD – 将存储器从主机复制到设备

概要

CUresult cuMemcpyHtoD(CUdeviceptr dstDevPtr，const void *srcHostPtr，unsigned int count);

CUresult cuMemcpyHtoDAsync(CUdeviceptr dstDevPtr，const void *srcHostPtr，unsigned int count，CUstream stream);

说明

从主机存储器复制到设备存储器。dstDevPtr和srcHostPtr分别指定目标和源的基址。Count指定要复制的字节数。

cuMemcpyHtoDAsync()是异步的，可通过传递一个非零的stream参数将其关联到一个流。它仅对分页锁定的主存储器有效，如果传入指向可分页存储器的指针，那么将返回一个错误。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemcpyDtoH，cuMemcpyDtoD，cuMemcpyDtoA，cuMemcpyAtoD，cuMemcpyAtoH，cuMemcpyHtoA，cuMemcpyAtoA，cuMemcpy2D

2.8.21 cuMemset

名称

cuMemset – 初始化设备存储器

概要

CUresult cuMemsetD8(CUdeviceptr dstDevPtr，unsigned char value，unsigned int count); CUresult CUMemsetD16(CUdeviceptr dstDevPtr，unsigned short value，unsigned int count); CUresult CUMemsetD32(CUdeviceptr dstDevPtr，unsigned int value，unsigned int count);

说明

将计数范围为8位、16位或32位值的存储器设置为指定值value。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemGetInfo，cuMemAlloc，cuMemAllocPitch，cuMemFree，cuMemAllocHost，cuMemFreeHost，cuMemGetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset2D

2.8.21 cuMemset

名称

cuMemset – 初始化设备存储器

概要

CUresult cuMemsetD2D8(CUdeviceptr dstDevPtr，unsigned int dstPitch，unsigned char value，unsigned int width，unsigned int height);

CUresult cuMemsetD2D16(CUdeviceptr dstDevPtr，unsigned int dstPitch，unsigned short value，unsigned int width，unsigned int height);

CUresult cuMemsetD2D32(CUdeviceptr dstDevPtr，unsigned int dstPitch，unsigned int value，unsigned int width，unsigned int height);

说明

将宽度范围为8位、16位或32位值的2D存储器设置为指定值value，height指定要设置的行数、dstPitch指定各行间的字节数。在所用间距是由cuMemAllocPitch()传递回的值时，这些函数的执行速度最快。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuMemGetInfo，cuMemAlloc，cuMemAllocPitch，cuMemFree，cuMemAllocHost，cuMemFreeHost，cuMemGetAddressRange，cuArrayCreate，cuArrayGetDescriptor，cuArrayDestroy，cuMemset

2.9 TextureReferenceManagement

名称

纹理引用管理

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuTexRefCreate

cuTexRefDestroy

cuTexRefGetAddress

cuTexRefGetAddressMode

cuTexRefGetArray

cuTexRefGetFilterMode

cuTexRefGetFlags

cuTexRefGetFormat

cuTexRefSetAddress

cuTexRefSetAddressMode

cuTexRefSetArray

cuTexRefSetFilterMode

cuTexRefSetFlags

cuTexRefSetFormat

参见

初始化，设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.9.1 cuTexRefCreate

名称

cuTexRefCreate – 创建纹理引用

概要

CUresult cuTexRefCreate(CUtexref* texRef);

说明

创建纹理引用，并以*texRef的形式返回其句柄。创建完成后，应用程序必须调用cuTexRefSetArray()或cuTexRefSetAddress()将引用关联到所分配的存储器。可以使用其他纹理引用函数指定在通过纹理引用读取存储器时使用的格式和互操作（寻址、过滤等）。要为给定函数关联纹理引用和结构序号，应用程序应调用cuParamSetTexRef()。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefDestroy， cuTexRefSetArray， cuTexRefSetAddress， cuTexRefSetFormat， cuTexRefSetAddressMode，cuTexRefSetFilterMode， cuTexRefSetFlags， cuTexRefGetAddress， cuTexRefGetArray， cuTexRefGetAddressMode，cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.2 cuTexRefDestroy

名称

cuTexRefDestroy – 销毁纹理引用

概要

CUresult cuTexRefDestroy(CUtexref texRef);

说明

销毁纹理引用。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate ， cuTexRefSetArray ， cuTexRefSetAddress ， cuTexRefSetFormat ， cuTexRefSetAddressMode ， cuTexRefSetFilterMode ， cuTexRefSetFlags ， cuTexRefGetAddress ， cuTexRefGetArray ， cuTexRefGetAddressMode ， cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.3 cuTexRefGetAddress

名称

cuTexRefGetAddress – 获取与纹理引用关联的地址

概要

CUresult cuTexRefGetAddress(CUdeviceptr* devPtr，CUtexref texRef);

说明

以*devPtr的形式返回绑定到纹理引用texRef的基址，如果纹理引用未绑定到任何设备存储器区域，则返回CUDA_ERROR_INVALID_VALUE。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate， cuTexRefDestroy， cuTexRefSetArray， cuTexRefSetAddress， cuTexRefSetFormat，cuTexRefSetAddressMode ， cuTexRefSetFilterMode ， cuTexRefSetFlags ， cuTexRefGetArray ，cuTexRefGetAddressMode，cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.4 cuTexRefGetAddressMode

名称

cuTexRefGetAddressMode – 获取一个纹理引用所用的寻址模式

概要

CUresult cuTexRefGetAddressMode(CUaddress_mode* mode，CUtexref texRef，int dim);

说明

以*mode的形式返回对应于纹理引用texRef的维度dim的寻址模式。目前，dim的有效值仅有0和1。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate ， cuTexRefDestroy ， cuTexRefSetArray ， cuTexRefSetAddress ， cuTexRefSetFormat ， cuTexRef-SetAddressMode ， cuTexRefSetFilterMode ， cuTexRefSetFlags ， cuTexRefGetAddress ， cuTexRefGetArray ， cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.5 cuTexRefGetArray

名称

cuTexRefGetArray – 获取绑定到一个纹理引用的数组

概要

CUresult cuTexRefGetArray(CUarray* array，CUtexref texRef);

说明

以*array的形式返回纹理引用texRef绑定的CUDA数组，如果纹理引用未绑定任何CUDA数组，则返回CUDA_ERROR_INVALID_VALUE。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate， cuTexRefDestroy， cuTexRefSetArray， cuTexRefSetAddress， cuTexRefSetFormat，cuTexRefSetAddressMode， cuTexRefSetFilterMode， cuTexRefSetFlags， cuTexRefGetAddress，cuTexRefGetAddressMode，cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.6 cuTexRefGetFilterMode

名称

cuTexRefGetFilterMode – 获取纹理引用所用的过滤模式

概要

CUresult cuTexRefGetFilterMode(CUfilter_mode* mode，CUtexref texRef);

说明

以*mode的形式返回纹理引用texRef的过滤模式。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate ， cuTexRefDestroy ， cuTexRefSetArray ， cuTexRefSetAddress ， cuTexRefSetFormat ， cuTexRef-SetAddressMode ， cuTexRefSetFilterMode ， cuTexRefSetFlags ， cuTexRefGetAddress ， cuTexRefGetArray ， cuTexRefGetAddressMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.7 cuTexRefGetFlags

名称

cuTexRefGetFlags – 获取一个纹理引用所用的标记

概要

CUresult cuTexRefGetFlags(unsigned int* flags，CUtexref texRef);

说明

以*flags的形式返回纹理引用texRef的标记。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate ， cuTexRefDestroy ， cuTexRefSetArray ， cuTexRefSetAddress ， cuTexRefSetFormat ， cuTexRefSetAddressMode ， cuTexRefSetFilterMode ， cuTexRefSetFlags ， cuTexRefGetAddress ， cuTexRefGetArray ， cuTexRefGetAddressMode，cuTexRefGetFilterMode，cuTexRefGetFormat

2.9.8 cuTexRefGetFormat

名称

cuTexRefGetFormat – 获取纹理引用所用的格式

概要

CUresult cuTexRefGetFormat(CUarray_format* format，int* numPackedComponents，CUtexref texRef);

说明

以*format和*numPackedComponents的形式返回绑定到纹理引用texRef的CUDA数组组件的格式和数量。如果格式或numPackedComponents为空，则将被忽略。

返回值

相关返回值：

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate ， cuTexRefDestroy ， cuTexRefSetArray ， cuTexRefSetAddress ， cuTexRefSetFormat ，cuTexRefSetAddressMode，cuTexRefSetFilterMode，cuTexRefSetFlags，cuTexRefGetAddress，cuTexRefGetArray，cuTexRefGetAddressMode，cuTexRefGetFilterMode，cuTexRefGetFlags

2.9.9 cuTexRefSetAddress

名称

cuTexRefSetAddress – 将地址作为纹理引用绑定

概要

CUresult cuTexRefSetAddress(unsigned int* byteOff set，CUtexref texRef，CUdeviceptr devPtr，int bytes);

说明

将线性地址区域绑定到纹理引用texRef。此前与纹理引用相关的任何地址或CUDA数组状态都将被此函数替换。所有以前绑定到texRef的存储器都将解除绑定。

由于硬件对纹理基址实施对齐要求，cuTexRefSetAddress()将以*offset的形式返回字节偏移量，为了从所需存储器中读取，必须对纹理获取应用此偏移。此偏移必须除以texel大小，并传递给从纹理中进行读取的内核，使之可应用于tex1Dfetch()函数。

如果设备存储器指针是由cudaMemAlloc()返回的，则偏移必定为0，此时可将NULL作为offset参数传递。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate ， cuTexRefDestroy ， cuTexRefSetArray ， cuTexRefSetFormat ， cuTexRefSetAddressMode ， cuTexRefSetFilterMode ， cuTexRefSetFlags ， cuTexRefGetAddress ， cuTexRefGetArray ， cuTexRefGetAddressMode ， cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.10 cuTexRefSetAddressMode

名称

cuTexRefSetAddressMode – 为纹理引用设置寻址模式

概要

CUresult cuTexRefSetAddressMode(CUtexref texRef，int dim，CUaddress_mode mode);

说明

为纹理引用texRef的给定维度指定寻址模式mode。如果dim为0，寻址模式将应用于用来从纹理中获取内容的函数的第一个参数。如果dim为1，则应用于第二个参数，依此类推。CUaddress_mode定义如下：

```
typedef enum CUaddress_mode_enum
    { CU_TR_ADDRESS_MODE_WRAP = 0，
    CU_TR_ADDRESS_MODE_CLAMP = 1，
    CU_TR_ADDRESS_MODE_MIRROR = 2,
} CUaddress_mode;
```

注意，如果texRef绑定到线性存储器，则此调用没有任何效果。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate，cuTexRefDestroy，cuTexRefSetArray，cuTexRefSetAddress，cuTexRefSetFormat，cuTexRefSetFilterMode，cuTexRefSetFlags， cuTexRefGetAddress， cuTexRefGetArray， cuTexRefGetAddressMode， cu-TexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.11 cuTexRefSet Array

名称

cuTexRefSetArray – 将数组绑定到纹理引用

概要

CUresult cuTexRefSetArray(CUtexref texRef，CUarray array，unsigned int flags);

说明

将CUDA数组array绑定到纹理引用texRef。此前与纹理引用关联的任何地址或CUDA数组状态都将被此函数替换。Flags必须设置为CU_TRSA_OVERRIDE_FORMAT。所有以前绑定到texRef的CUDA数组都将解除绑定。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate ， cuTexRefDestroy ， cuTexRefSetAddress ， cuTexRefSetFormat ， cuTexRefSetAddressMode ， cuTexRefSetFilterMode ， cuTexRefSetFlags ， cuTexRefGetAddress ， cuTexRefGetArray ， cuTexRefGetAddressMode ， cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.12 cuTexRefSetFilterMode

名称

cuTexRefSetFilterMode – 为纹理引用设置模式

概要

CUresult cuTexRefSetFilterMode(CUtexref texRef，CUfilter_mode mode);

说明

指定过滤模式mode，该模式将在通过纹理引用texRef读取存储器时使用。CUfilter_mode_enum定义如下：

```
typedef enum CUfilter_mode_enum {
    CU_TR_FILTER_MODE_POINT = 0,
    CU_TR_FILTER_MODE_LINEAR = 1 }
CUfilter_mode;
```

请注意，如果texRef绑定到线性存储器，则此调用没有任何效果。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate，cuTexRefDestroy，cuTexRefSetArray，cuTexRefSetAddress，cuTexRefSetFormat，cuTexRefSetAddressMode，cuTexRefSetFlags，cuTexRefGetAddress，cuTexRefGetArray，cuTexRefGetAddressMode，cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.13 cuTexRefSetFlags

名称

cuTexRefSetFlags – 为纹理引用设置标记

概要

CUresult cuTexRefSetFlags(CUtexref texRef，unsigned int Flags);

说明

指定可选标记，控制通过纹理引用返回的数据的行为。有效的标记包括：

- CU_TRSF_READ_AS_INTEGER，阻止将整型数据转变为[0，1]范围内的浮点数据的默认行为；
- CU_TRSF_NORMALIZED_COORDINATES，阻止使用纹理坐标范围[0，Dim)的默认行为，其中的Dim是CUDA数组的宽度或高度。与此不同，使用结构坐标[0，1.0)可以引用数组维度的整个宽度

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRef Create，cuTexRefDestroy，cuTexRefSetArray，cuTexRefSetAddress，cuTexRefSetFormat，cuTexRefSetAddressMode，cuTexRefSetFilterMode，cuTexRefGetAddress，cuTexRefGetArray，cuTexRefGetAddressMode，cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.9.14 cuTexRefSet Format

名称

cuTexRefSetFormat – 为纹理引用设置格式

概要

CUresult cuTexRefSetFormat(CUtexref texRef，CUarray_format format，int numPackedComponents)

说明

指定要由纹理引用texRef读取的数据的格式。format和numPackedComponents正是CUDA_ARRAY_DESCRIPTOR结构体中的Format和NumChannels成员。它们指定了各组件的元素和每个数组元素的组件数量。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuTexRefCreate，cuTexRefDestroy，cuTexRefSetArray，cuTexRefSetAddress，cuTexRefSetAddressMode，cuTexRefSetFilterMode，cuTexRefSetFlags，cuTexRefGetAddress，cuTexRefGetArray，cuTexRefGetAddressMode，cuTexRefGetFilterMode，cuTexRefGetFormat，cuTexRefGetFlags

2.10 OpenGlInteroperability

名称

OpenGL互操作性

说明

本节描述低级CUDA驱动程序应用程序编程接口。

cuGLCtxCreate

cuGLInit

cuGLMapBufferObject

cuGLRegisterBufferObject

cuGLUnmapBufferObject

cuGLUnregisterBufferObject

参见

初始化，设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.10.1 cuGLCtxCreate

名称

cuGLCtxCreate – 为与OpenGL互操作创建一个CUDA上下文

概要

CUresult cuGLCtxCreate(CUcontext *pCtx，unsigned int Flags，CUdevice device);

说明

创建一个新的CUDA上下文、初始化OpenGL互操作性并将CUDA上下文与调用方线程关联。在执行其他任何OpenGL互操作之前，必须调用此函数。如果所需的OpenGL驱动程序设施不可用，它可能会失败。关于Flags参数的使用，请参见cuCtxCreate。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuCtxCreate，cuGLInitcuGLRegisterBufferObject，cuGLMapBufferObject，cuGLUnmapBufferObject，cuGLUnregisterBufferObject

2.10.2 cuGLInit

名称

cuGLInit – 初始化GL互操作性

概要

CUresult cuGLInit(void);

说明

初始化OpenGL互操作性。在执行其他任何OpenGL互操作之前都必须调用此函数。如果所需的OpenGL驱动程序设施不可用，它可能会失败。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_UNKNOWN

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuGLCtxCreate，cuGLRegisterBufferObject，cuGLMapBufferObject，cuGLUnmapBufferObject，cuGLUnregisterBufferObject

2.10.3 cuGLMapBufferObject

名称

cuGLMapBufferObject – 映射一个GL缓存对象

概要

CUresult cuGLMapBufferObject(CUdeviceptr* devPtr，unsigned int* size，GLuint buff erObj);

说明

将ID为bufferObj的缓存对象映射到当前CUDA上下文的地址空间，并以*devPtr和*size的形式返回所得映射的基址指针和大小。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_MAP_FAILED

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuGLCtxCreate，cuGLInit，cuGLRegisterBufferObject，cuGLUnmapBufferObject，cuGLUnregisterBufferObject

2.10.4 cuGLRegisterBufferObject

名称

cuGLRegisterBufferObject – 注册一个GL缓存对象

概要

CUresult cuGLRegisterBufferObject(GLuint buff erObj);

说明

注册ID为bufferObj的缓存对象，以便CUDA访问。在CUDA映射任何缓存对象之前都必须调用此函数。完成注册后，缓存对象只能作为OpenGL绘图命令的数据源使用，不能被其他OpenGL命令使用。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_ALREADY_MAPPED

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuGLCtxCreate，cuGLInit，cuGLMapBufferObject，cuGLUnmapBufferObject，cuGLUnregisterBufferObject

2.10.5 cuGLUnmapBufferObject

名称

cuGLUnmapBufferObject - 解除GL缓存对象的映射

概要

CUresult cuGLUnmapBufferObject(GLuint buff erObj);

说明

解除ID为bufferObj的缓存对象的映射，以便CUDA访问。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuGLCtxCreate，cuGLInit，cuGLRegisterBufferObject，cuGLMapBufferObject，cuGLUnregisterBufferObject

2.10.6 cuGLUnregisterBufferObject

名称

cuGLUnregisterBufferObject – 注销GL缓存对象

概要

CUresult cuGLUnregisterBufferObject(GLuint buff erObj);

说明

注销注册ID为bufferObj的缓存对象，以便CUDA访问。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuGLCtxCreate，cuGLInit，cuGLRegisterBufferObject，cuGLMapBufferObject，cuGLUnmapBufferObject

2.11 Direct3dInteroperability

名称

Direct3D互操作性

说明

本节描述低级CUDA驱动程序应用程序编程接口中的Direct3D互操作性。

cuD3D9GetDevice

cuD3D9CtxCreate

cuD3D9GetDirect3DDevice

cuD3D9RegisterResource

cuD3D9UnregisterResource

cuD3D9MapResources

cuD3D9UnmapResources

cuD3D9ResourceGetSurfaceDimensions

cuD3D9ResourceSetMapFlags

cuD3D9ResourceGetMappedPointer

cuD3D9ResourceGetMappedSize

cuD3D9ResourceGetMappedPitch

在CUDA 2.0中，以下函数已被废弃。不应在新开发工作中继续使用这些函数。

cuD3D9Begin

cuD3D9End

cuD3D9MapVertexBuffer

cuD3D9RegisterVertexBuffer

cuD3D9UnmapVertexBuffer

cuD3D9UnregisterVertexBuffer

参见

初始化,设备管理，上下文管理，模块管理，流管理，事件管理，执行控制，存储器管理，纹理引用管理，OpenGL互操作性，Direct3D互操作性

2.11.1 cuD3D9CtxCreate

名称

cuD3D9CtxCreate – 为与Direct3D互操作创建一个CUDA上下文

概要

CUresult cuD3D9CtxCreate(CUcontext* pCtx，CUdevice* pCuDevice，unsigned int Flags，IDirect3DDevice9* pDxDevice);

说明

创建一个新的CUDA上下文，为此上下文支持与Direct3D设备pDxDevice的互操作，并将所创建的CUDA上下文关联到调用方线程。如果非空，将使用创建此CUDA上下文的Cudevice填充pCuDevice指向的Cudevice。关于Flags参数的使用，请参见cuCtxCreate。此上下文仅在Direct3D设备被销毁时才能发挥作用。来自该设备的Direct3D资源可在此CUDA上下文的整个生命周期内注册和映射。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

CUDA_ERROR_UNKNOWN

参见

cuD3D9GetDirect3DDevice，cuD3D9RegisterResource，cuD3D9UnregisterResource，cuD3D9MapResources，cuD3D9UnmapResources，cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9ResourceGetMappe，cuD3D9ResourceGetMappedSize，cuD3D9ResourceGetMappedPitch

2.11.2 cuD3D9GetDirect3DDevice

名称

cuD3D9GetDirect3DDevice – 获取据以创建当前CUDA上下文的Direct3D设备

概要

CUresult cuD3D9GetDirect3DDevice(IDirect3DDevice9** ppDxDevice);

说明

以*ppDxDevice的形式返回cuD3D9CtxCreate中据以创建此CUDA上下文的Direct3D设备。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

参见

cuD3D9CtxCreate，cuD3D9RegisterResource，cuD3D9UnregisterResource，cuD3D9MapResources，cuD3D9UnmapResources，cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9ResourceGetMappedPointer，cuD3D9ResourceGetMappedSize，cuD3D9ResourceGetMappedPitch

2.11.3 cuD3D9RegisterResource

名称

cuD3D9RegisterResource – 注册Direct3D资源，以便CUDA访问

概要

CUresult cuD3D9RegisterResource(IDirect3DResource9* pResource，unsigned int Flags);

说明

注册Direct3D资源pResource，以便CUDA访问。

如果此调用成功，则应用程序将能够在资源注销之前映射和解除映射此资源。此调用开销较高，不应在交互式应用程序的每一帧中进行调用。

pResource的类型必须为以下之一：

- IDirect3DVertexBuffer9：无备注。
- IDirect3DIndexBuffer9：无备注。
- IDirect3DSurface9：只有IDirect3DSurface9类型的独立对象才可以显式共享。具体来说，不可以直接注册单个mipmap级别和立方体的各面。要访问与纹理相关的独立表面，必须注册基本纹理对象。
- IDirect3DBaseTexture9：注册一个纹理时，CUDA可以访问与mipmap顶级相关的所有表面。不可以访问在mipmap顶级以下的mipmap级别。

并非所有上述类型的Direct3D资源均可用于CUDA互操作。下面列举了部分限制：

- 主呈现目标未向CUDA注册。
- 深度和模板表面未向CUDA注册。
- 分配为共享的资源未向CUDA注册。
- D3DPOOL_SYSTEMMEM中分配的任意资源未向CUDA注册。

Flags参数必须设置为cudaD3D9RegisterFlagsNone。

如果在此上下文内未初始化Direct3D互操作性，则将返回CUDA_ERROR_INVALID_CONTEXT。

如果pResource是不正确的类型（例如，是非独立的IDirect3DResource9或已经注册），则返回CUDA_ERROR_INVALID_HANDLE。如果pResource无法注册，则返回CUDA_ERROR_UNKNOWN。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_OUT_OF_MEMORY

CUDA_ERROR_UNKNOWN

参见

cuD3D9CtxCreate，cuD3D9GetDirect3DDevice，cuD3D9UnregisterResource，cuD3D9MapResources，
cuD3D9UnmapResource，cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，
cuD3D9ResourceGetMappedPointer，cuD3D9ResourceGetMappedSize，cuD3D9ResourceGetMappedPitch

2.11.4 cuD3D9UnregisterResource

名称

cuD3D9UnregisterResource – 注销Direct3D资源

概要

CUresult cuD3D9UnregisterResource(IDirect3DResource9* pResource);

说明

注销Direct3D资源pResource，使之无法再由CUDA访问，除非再次注册。如果pResource没有注册，则返回CUDA_ERROR_INVALID_HANDLE。

返回值

CUDA_SUCCESS
CUDA_ERROR_DEINITIALIZED
CUDA_ERROR_NOT_INITIALIZED
CUDA_ERROR_INVALID_CONTEXT
CUDA_ERROR_INVALID_HANDLE

参见

cuD3D9CtxCreate，cuD3D9GetDirect3DDevice，cuD3D9RegisterResource，cuD3D9MapResources，cuD3D9UnmapResources，cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9ResourceGetMappedPointer，cuD3D9ResourceGetMappedSize，cuD3D9ResourceGetMappedPitch

2.11.5 cuD3D9MapResources

名称

cuD3D9MapResources - 映射Direct3D资源，以便CUDA访问

概要

CUresult cuD3D9MapResources(unsigned int count，IDirect3DResource9 **ppResources);

说明

映射ppResources中的count Direct3D资源，以便CUDA访问。

在解除映射之前，可在CUDA内核中访问ppResources中的资源。在CUDA映射了资源后，Direct3D不应访问任何资源。如果应用程序允许其访问，则将导致不确定的结果。

此函数提供了同步保证，确保在cudaD3D9MapResources开始执行CUDA内核之前，cudaD3D9MapResources之前发出的任何Direct3D调用都将完成。

如果存在尚未注册与CUDA共同使用的ppResources，如果ppResources包含重复项，则将返回CUDA_ERROR_INVALID_HANDLE。如果有ppResources映射为CUDA访问，则返回CUDA_ERROR_ALREADY_MAPPED。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_ALREADY_MAPPED

CUDA_ERROR_UNKNOWN

参见

cuD3D9CtxCreate，cuD3D9GetDirect3DDevice，cuD3D9RegisterResource，cuD3D9UnregisterResource，cuD3D9UnmapResourc， cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9ResourceGetMappedPointer，cuD3D9ResourceGetMappedSize，cuD3D9ResourceGetMappedPitch

2.11.6 cuD 3D9UnmapResources

名称

cuD3D9UnmapResources - 解除映射Direct3D资源

概要

CUresult cuD3D9UnmapResources(unsigned int count，IDirect3DResource9** ppResources);

说明

解除映射ppResources中的count Direct3D资源。

此函数提供了同步保证，确保在cudaD3D9UnmapResources开始发出Direct3D调用之前，cudaD3D9UnmapResources之前发出的任何CUDA内核都将完成。

如果存在尚未注册与CUDA共同使用的ppResources，如果ppResources包含重复项，则将返回CUDA_ERROR_INVALID_HANDLE。如果有ppResources未注册为CUDA访问，则返回CUDA_ERROR_NOT_MAPPED。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_MAPPED

CUDA_ERROR_UNKNOWN

参见

cuD3D9CtxCreate，cuD3D9 GetDirect3DDevice，cuD3D9RegisterResource，cuD3D9UnregisterResource，cuD3D9MapResources，cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9ResourceGetMappedPointer，cuD3D9ResourceGetMappedSize，cuD3D9ResourceGetMappedPitch

2.11.7 cuD3D9ResourceSetMapFlags

名称

cuD3D9ResourceSetMapFlags - 为映射Direct3D资源设置使用标志

概要

CUresult cuD3D9ResourceSetMapFlags(IDirect3DResource9 *pResource，unsigned int Flags);

说明

为映射Direct3D资源pResource设置标志。

标志更改将在下一次映射pResource时生效。Flags参数可为以下任何值之一：

- cudaD3D9MapFlagsNone：指明没有任何关于此资源如何使用的提示。因而，假设此资源将由CUDA内核读取并写入。该值为默认值。

- cudaD3D9MapFlagsReadOnly：指明访问此资源的CUDA内核将不会写入此资源。

- cudaD3D9MapFlagsWriteDiscard：指明访问此资源的CUDA内核将不会读取此资源，且将改写该资源的所有内容，之前存储在此资源中的任何数据均不予保留。

如果pResource尚未注册与CUDA共同使用，则将返回CUDA_ERROR_INVALID_HANDLE。如果pResource已映射为由CUDA访问，则返回CUDA_ERROR_ALREADY_MAPPED。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT CUDA_ERROR_INVALID

_VALUE CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_ALREADY_MAPPED CUDA_ERROR_UNKNOWN

参见

uD3D9CtxCreate，cuD3D9GetDirect3DDevice，cuD3D9RegisterResource

2.11.8 cuD3D9ResourceGetSurfaceDimensions

名称

cuD3D9ResourceGetSurfaceDimensions - 获取已注册表面的维度

概要

CUresult cuD3D9ResourceGetSurfaceDimensions(unsigned int* pWidth，unsigned int* pHeight，unsigned int *pDepth，IUnknown* pResource，unsigned int Face，unsigned int Level);

说明

以*pWidth、*pHeight和*pDepth的形式返回对应于Face和Level的已映射Direct3D资源pResource的子资源维度。

对于抗锯齿表面来说，每个像素可能包含多个样本，因而一个资源的维度因数可能大于Direct3D运行时报告的维度。

pWidth、pHeight和pDepth参数是可选的。对于2D表面而言，*pDepth返回的值是0。

如果pResource的类型不是IDirect3DBaseTexture9或IDirect3DSurface9，如果pResource尚未注册与CUDA共同使用，则将返回CUDA_ERROR_INVALID_HANDLE。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

参见

cuD3D9CtxCreate ， cuD3D9GetDirect3DDevice ， cuD3D9RegisterResource ， cuD3D9UnregisterResource ， cuD3D9MapResources cuD3D9UnmapResources ， cuD3D9ResourceSetMapFlags ， cuD3D9ResourceGetMappedPointer ， cuD3D9ResourceGetMappedSiz cuD3D9ResourceGetMappedPitch

2.11.9 cuD3D9ResourceGetMappedPointer

名称

cuD3D9ResourceGetMappedPointer - 获取已映射CUDA资源的基址指针

概要

CUresult cuD3D9ResourceGetMappedPointer(CUdeviceptr* pDevPtr，IUnknown* pResource，unsigned int Face，unsigned int Level);

说明

以*pDevPtr的形式返回对应于Face和Level的已映射Direct3D资源pResource的子资源基址指针。每次映射pResource时，pDevPtr中设置的值都会发生更改。

如果pResource没有注册，则返回CUDA_ERROR_INVALID_HANDLE。如果pResource没有映射，则返回CUDA_ERROR_NOT_MAPPED。

如果pResource的类型是IDirect3DCubeTexture9，则Face必须是按D3DCUBEMAP_FACES类型枚举的值之一。对于其他所有类型，Face必须为0。如果Face无效，则返回CUDA_ERROR_INVALID_VALUE。

如果pResource的类型是IDirect3DBaseTexture9，则Level必须对应于有效的mipmap级别。对于其他所有类型，Level必须为0。目前仅支持mipmap级别0。如果Level无效，则返回CUDA_ERROR_INVALID_VALUE。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_MAPPED

参见

cuD3D9CtxCreate ， cuD3D9GetDirect3DDevice ， cuD3D9RegisterResource ， cuD3D9UnregisterResource ， cuD3D9MapResources cuD3D9UnmapResources，cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9ResourceGetMappe cuD3D9ResourceGetMappedPitch

2.11.10 cuD3D9ResourceGetMappedSize

名称

cuD3D9ResourceGetMappedSize - 获取已映射CUDA资源的大小

概要

CUresult cuD3D9ResourceGetMappedSize(unsigned int* pSize，IDirect3DResource9* pResource);

说明

以*pSize的形式返回对应于Face和Level的已映射Direct3D资源pResource的子资源大小。每次映射pResource时，pSize中设置的值都会发生变化。

如果pResource尚未注册与CUDA一起使用，则返回CUDA_ERROR_INVALID_HANDLE。如果pResource尚未映射为CUDA访问，则返回CUDA_ERROR_NOT_MAPPED。

关于Face和Level参数的使用要求，请参见cudaD3D9ResourceGetMappedPointer。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_MAPPED

参见

cuD3D9CtxCreate ， cuD3D9GetDirect3DDevice ， cuD3D9RegisterResource ， cuD3D9UnregisterResource ， cuD3D9MapResources cuD3D9UnmapResources，cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9ResourceGetMappe cuD3D9ResourceGetMappedPitch

2.11.11 cuD3D9ResourceGetMappedPitch

名称

cuD3D9ResourceGetMappedPitch - 获取已映射CUDA资源的间距

概要

CUresult cuD3D9ResourceGetMappedPitch(unsigned int* pPitch，unsigned int* pPitchSlice，IDirect3DResourc pResource，unsigned int Face，unsigned int Level);

说明

以*pPitch和*pPitchSlice的形式返回对应于Face和Level的已映射Direct3D资源的子资源间距和Z-slice间距。每次映射pResource时，pPitch和pPitchSlice中设置的值都会发生变化。

如果pResource不是IDirect3DBaseTexture9类型或其任意子类型，如果pResource尚未注册与CUDA一起使用，则将返回CUDA_ERROR_INVALID_HANDLE。如果pResource没有映射为CUDA访问，则将返回CUDA_ERROR_NOT_MAPPED。

关于Face和Level参数的使用要求，请参见cudaD3D9ResourceGetMappedPointer。参数pPitch和pPitchSlice都是可选的，可设置为NULL。

间距和Z-slice间距值可用于计算表面上一个样本的位置，具体方法如下。对于2D表面，从表面基址算起，位置x,y处的样本字节偏移量为：

y*pitch + (bytes per pixel)*x

对于3D表面，从表面基址算起，位置x,y处的样本字节偏移量为：

z*slicePitch + y*pitch + (bytes per pixel)*x

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_INVALID_HANDLE

CUDA_ERROR_NOT_MAPPED

参见

cuD3D9CtxCreate ， cuD3D9GetDirect3DDevice ， cuD3D9RegisterResource ， cuD3D9 UnregisterResource ，cuD3D9MapResources cuD3D9 UnmapResources，cuD3D9Resource GetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9Resource GetMappe cuD3D9ResourceGetMappedSize

2.11.12 cuD3D9Begin

名称

cuD3D9Begin – 初始化Direct3D互操作性

概要

CUresult cuD3D9Begin(IDirect3DDevice9* device);

说明

初始化与Direct3D设备device的互操性。必须首先调用此函数之后CUDA才能映射device的对象。随后，在调用cuD3D9End()之前，应用程序可以调用Direct3D设备拥有的顶点缓存。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果使用cudaD3D9Begin为此CUDA上下文初始化了Direct3D互操作，则互操作将限制为被废弃的顶点缓存接口，调用CUDA 2.0及更新版本中引入的函数都将失败。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_UNKNOWN

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuD3D9End，cuD3D9RegisterVertexBuffer，cuD3D9MapVertexBuffer，cuD3D9UnmapVertexBuffer，cuD3D9UnregisterVertexB cuD3D9GetDevice

2.11.13 cuD3D9End

名称

cuD3D9End – 结束Direct3D互操作性

概要

CUresult cuD3D9End(void);

说明

结束之前在cuD3D9Begin()中指定的与Direct3D设备的互操作。
该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuD3D9Begin，cuD3D9RegisterVertexBuffer，cuD3D9MapVertexBuffer，cuD3D9UnmapVertexBuffer，cuD3D9UnregisterVertex cuD3D9GetDevice

2.11.14 cuD3D9GetDevice

名称

cuD3D9GetDevice – 获取适配器的设备号

概要

CUresult cuD3D9GetDevice(CUdevice* dev，const char* adapterName);

说明

以*dev的形式返回对应于从EnumDisplayDevices或IDirect3D9: :GetAdapterIdentifier()获取的适配器名adapterName的设备。

返回值

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_NOT_FOUND

CUDA_ERROR_UNKNOWN

参见

cuD3D9CtxCreate ， cuD3D9GetDirect3DDevice ， cuD3D9RegisterResource ， cuD3D9UnregisterResource ，cuD3D9MapResources cuD3D9UnmapResources，cuD3D9ResourceGetSurfaceDimensions，cuD3D9ResourceSetMapFlags，cuD3D9ResourceGetMappe cuD3D9ResourceGetMappedSize，cuD3D9ResourceGetMappedPitch

2.11.15 cuD3D9MapVertexBuffer

名称

cuD3D9MapVertexBuffer – 映射Direct3D缓存

概要

CUresult cuD3D9MapVertexBuffer(CUdeviceptr* devPtr，unsigned int* size，IDirect3DVertexBuffer9* VB);

说明

将顶点缓存VB映射到当前CUDA上下文的地址空间中，并以*devPtr和*size的形式返回所得映射的基址指针和大小。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuD3D9Begin，cuD3D9End，cuD3D9RegisterVertexBuffer，cuD3D9UnmapVertexBuffer，cuD3D9UnregisterVertexBuffer，cuD3D9GetDevice

2.11.16 cuD3D9RegisterVertexBuffer

名称

cuD3D9RegisterVertexBuffer – 注册Direct3D缓存

概要

CUresult cuD3D9RegisterVertexBuffer(IDirect3DVertexBuffer9* VB);

说明

注册Direct3D顶点缓存VB，以便CUDA访问。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_OUT_OF_MEMORY

CUDA_ERROR_UNKNOWN

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuD3D9Begin，cuD3D9End，cuD3D9MapVertexBuffer，cuD3D9UnmapVertexBuffer，cuD3D9UnregisterVertexBuffer，cuD3D9GetDevice

2.11.17 cuD3D9UnmapVertexBuffer

名称

cuD3D9UnmapVertexBuffer – 解除Direct3D缓存的映射

概要

CUresult cuD3D9UnmapVertexBuffer(IDirect3DVertexBuffer9* VB);

说明

解除顶点缓存VB的映射，以便CUDA访问。

该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

CUDA_ERROR_UNKNOWN

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuD3D9Begin，cuD3D9End，cuD3D9RegisterVertexBuffer，cuD3D9MapVertexBuffer，cuD3D9UnregisterVertexBuffer，cuD3D9GetDevice

2.11.18 cuD3D9UnregisterVertexBuffer

名称
cuD3D9UnregisterVertexBuffer – 注销Direct3D缓存

概要

CUresult cuD3D9UnregisterVertexBuffer(IDirect3DVertexBuffer9* VB);

说明
注销顶点缓存VB，以便CUDA访问。
该函数自CUDA 2.0起已废弃，不应该在新的开发中使用。如果通过非废弃的cuduD3D9SetDirect3Ddevice接口为此CUDA上下文初始化Directo3D互操作，则此函数将失败。

返回值

相关返回值：

CUDA_SUCCESS

CUDA_ERROR_DEINITIALIZED

CUDA_ERROR_NOT_INITIALIZED

CUDA_ERROR_INVALID_CONTEXT

CUDA_ERROR_INVALID_VALUE

注意，如果之前是异步启动，该函数可能返回错误码。

参见

cuD3D9Begin，cuD3D9End，cuD3D9RegisterVertexBuffer，cuD3D9MapVertexBuffer，cuD3D9UnmapVertexBuffer，cuD3D9GetDevice

3 AtomicFunctions

名称

原子函数

说明

仅可在设备函数中使用的原子函数。

备注

只有计算能力在1.1以上的设备才支持32位原子操作。只有计算能力在1.2以上的设备才支持64位原子操作。

参见

ArithmeticFunctions，BitwiseFunctions

3.1 ArithmeticFunctions

名称

算术函数

说明

本节介绍原子算术函数。

atomicAdd
atomicSub
atomicExch
atomicMin
atomicMax
atomicInc
atomicDec
atomicCAS

参见

BitwiseFunctions

3.1.1 atomicAdd

名称

atomicAdd – 原子加法

概要

int atomicAdd(int* address，int val);

unsigned int atomicAdd(unsigned int* address，unsigned int val);

unsigned long long int atomicAdd(unsigned long long int* address，unsigned long long int val);

说明

读取位于全局存储器中address地址处的32位或64位词old，计算(old + val)，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持32位原子操作。 只有计算能力在1.2以上的设备才支持32位原子操作。

参见

atomicSub，atomicExch，atomicMin，atomicMax，atomicInc，atomicDec，atomicCAS

3.1.2 atomicSub

名称

atomicSub – 原子减法

概要

int atomicSub(int* address，int val);

unsigned int atomicSub(unsigned int* address，unsigned int val);

说明

读取位于全局存储器中address地址处的32位或64位词old，计算(old - val)，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持原子操作。

参见

atomicAdd，atomicExch，atomicMin，atomicMax，atomicInc，atomicDec，atomic CAS

3.1.3 atomicExch

名称

atomicExch – 原子交换

概要

int atomicExch(int* address，int val);

unsigned int atomicExch(unsigned int* address，unsigned int val);

unsigned long long int atomicExch(unsigned long long int* address，unsigned long long int val);

说明

读取位于全局存储器中address地址处的32位或64位词old，并将val存储到全局存储器的同一地址中。这两个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持32位原子操作。 只有计算能力在1.2以上的设备才支持32位原子操作。

参见

atomicAdd，atomicSub，atomicMin，atomicMax，atomicInc，atomicDec，atomic CAS

3.1.4 atomicMin

名称

atomicMin – 原子最小值

概要

int atomicMin(int* address，int val);

unsigned int atomicMin(unsigned int* address，unsigned int val);

说明

读取位于全局存储器中address地址处的32位词old，计算old和val中的最小值，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持原子操作。

参见

atomicAdd，atomicSub，atomicExch，atomicMax，atomicInc，atomicDec，atomicCAS

3.1.5 atomicMax

名称

atomicMax – 原子最大值

概要

int atomicMax(int* address，int val);

unsigned int atomicMax(unsigned int* address，unsigned int val);

说明

读取位于全局存储器中address地址处的32位词old，计算old和val中的最大值，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持原子操作。

参见

atomicAdd，atomicSub，atomicExch，atomicMin，atomicInc，atomicDec，atomicCAS

3.1.6 atomicInc

名称

atomicInc – 原子递增

概要

unsigned int atomicInc(unsigned int* address，unsigned int val);

说明

读取位于全局存储器中address地址处的32位词old，计算((old >= val) ? 0 : (old+1))，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持原子操作。

参见

atomicAdd，atomicSub，atomicExch，atomicMin，atomicMax，atomicDec，atomic CAS

3.1.7 atomicDec

名称

atomicDec – 原子递减

概要

unsigned int atomicDec(unsigned int* address，unsigned int val);

说明

读取位于全局存储器中address地址处的32位词old，计算(((old == 0)，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持原子操作。

参见

atomicAdd，atomicSub，atomicExch，atomicMin，atomicMax，atomicInc，atomic CAS

3.1.8 atomicCAS

名称

atomicCAS – 原子比较与交换

概要

int atomicCAS(int* address，int compare，int val);

unsigned int atomicCAS(unsigned int* address，unsigned int compare，unsigned int val);

unsigned long long int atomicCAS(unsigned long long int* address，unsigned long long int compare，unsigned long long int val);

说明

读取位于全局存储器中address地址处的32位或64位词old，计算(old == compare ? val : old)，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old（比较与交换）。

备注

只有计算能力在1.1以上的设备才支持32位原子操作。 只有计算能力在1.2以上的设备才支持32位原子操作。

参见

atomicAdd，atomicSub，atomicExch，atomicMin，atomicMax，atomicInc，atomicDec

3.2 BitwiseFunctions

名称

按位函数

说明

本节介绍按位函数。

atomicAnd
atomic Or
atomicXor

参见

ArithmeticFunctions

3.2.1 atomicAnd

名称

atomicAnd – 原子按位与

概要

int atomicAnd(int* address，int val);

unsigned int atomicAnd(unsigned int* address，unsigned int val);

说明

读取位于全局存储器中address地址处的32位词old，计算(old & val)，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持原子操作。

参见

atomic Or，atomicXor

3.2.2 atomicOr

名称
atomicOr – 原子按位或

概要

int atomicOr(int* address，int val);

unsigned int atomicOr(unsigned int* address，unsigned int val);

说明

读取位于全局存储器中address地址处的32位词old，计算(old | val)，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持原子操作。

参见

atomicAnd，atomicXor

3.2.3 atomicXor

名称

atomicXor – 原子按位异或

概要

int atomicXor(int* address，int val);

unsigned int atomicXor(unsigned int* address，unsigned int val);

说明

读取位于全局存储器中address地址处的32位词old，计算(old ^ val)，并将结果存储到全局存储器的同一地址中。这三个操作在一个原子事务中执行。函数将返回old。

备注

只有计算能力在1.1以上的设备才支持原子操作。

参见

atomicAnd，atomic Or

索引