CUDA Toolkit CUPTI User's Guide

DA-05679-001_v01 | September 2012

Document Change History

Ver	Date	Resp	Reason for change
v01	2011/1/19	DG	Initial revision for CUDA Tools SDK 4.0
v02	2012/1/5	DG	Revisions for CUDA Tools SDK 4.1
v03	2012/2/13	DG	Revisions for CUDA Tools SDK 4.2
v04	2012/5/1	DG	Revisions for CUDA Toolkit 5.0

CUPTI Reference

CUPTI Version

Defines

- #define CUPTI_API_VERSION 3

  The API version for this implementation of CUPTI.

Functions

- CUptiResult cuptiGetVersion (uint32_t *version)

  Get the CUPTI API version.

Detailed Description

Function and macro to determine the CUPTI version.

Define Documentation

#define CUPTI_API_VERSION 3

The API version for this implementation of CUPTI. This define along with cuptiGetVersion can be used to dynamically detect if the version of CUPTI compiled against matches the version of the loaded CUPTI library.

v1 : CUDAToolsSDK 4.0 v2 : CUDAToolsSDK 4.1 v3 : CUDA Toolkit 5.0

Function Documentation

CUptiResult cuptiGetVersion (uint32_t * version)

Return the API version in `*version`.

Parameters:

version Returns the version

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_INVALID_PARAMETER if `version` is NULL

See also:

CUPTI_API_VERSION

CUPTI Result Codes

Enumerations

▶ enum CUptiResult {

CUPTI_SUCCESS = 0,

CUPTI_ERROR_INVALID_PARAMETER = 1,

CUPTI_ERROR_INVALID_DEVICE = 2,

CUPTI_ERROR_INVALID_CONTEXT = 3,

CUPTI_ERROR_INVALID_EVENT_DOMAIN_ID = 4,

CUPTI_ERROR_INVALID_EVENT_ID = 5,

CUPTI_ERROR_INVALID_EVENT_NAME = 6,

CUPTI_ERROR_INVALID_OPERATION = 7,

CUPTI_ERROR_OUT_OF_MEMORY = 8,

CUPTI_ERROR_HARDWARE = 9,

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT = 10,

CUPTI_ERROR_API_NOT_IMPLEMENTED = 11,

CUPTI_ERROR_MAX_LIMIT_REACHED = 12,

CUPTI_ERROR_NOT_READY = 13,

CUPTI_ERROR_NOT_COMPATIBLE = 14,

CUPTI_ERROR_NOT_INITIALIZED = 15,

CUPTI_ERROR_INVALID_METRIC_ID = 16,

CUPTI_ERROR_INVALID_METRIC_NAME = 17,

CUPTI_ERROR_QUEUE_EMPTY = 18,

CUPTI_ERROR_INVALID_HANDLE = 19,

CUPTI_ERROR_INVALID_STREAM = 20,

CUPTI_ERROR_INVALID_KIND = 21,

CUPTI_ERROR_INVALID_EVENT_VALUE = 22,

CUPTI_ERROR_DISABLED = 23,

CUPTI_ERROR_INVALID_MODULE = 24,

CUPTI_ERROR_UNKNOWN = 999 }

CUPTI result codes.

Functions

- CUptiResult cuptiGetResultString (CUptiResult result, const char **str)

  Get the descriptive string for a CUptiResult.

Detailed Description

Error and result codes returned by CUPTI functions.

Enumeration Type Documentation

enum **CUptiResult**

Error and result codes returned by CUPTI functions.

Enumerator:

CUPTI_SUCCESS No error.

CUPTI_ERROR_INVALID_PARAMETER One or more of the parameters is invalid.

CUPTI_ERROR_INVALID_DEVICE The device does not correspond to a valid CUDA device.

CUPTI_ERROR_INVALID_CONTEXT The context is NULL or not valid.

CUPTI_ERROR_INVALID_EVENT_DOMAIN_ID The event domain id is invalid.

CUPTI_ERROR_INVALID_EVENT_ID The event id is invalid.

CUPTI_ERROR_INVALID_EVENT_NAME The event name is invalid.

CUPTI_ERROR_INVALID_OPERATION The current operation cannot be performed due to dependency on other factors.

CUPTI_ERROR_OUT_OF_MEMORY Unable to allocate enough memory to perform the requested operation.

CUPTI_ERROR_HARDWARE The performance monitoring hardware could not be reserved or some other hardware error occurred.

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT The output buffer size is not sufficient to return all requested data.

CUPTI_ERROR_API_NOT_IMPLEMENTED	API is not implemented.
CUPTI_ERROR_MAX_LIMIT_REACHED	The maximum limit is reached.
CUPTI_ERROR_NOT_READY	The object is not yet ready to perform the requested operation.
CUPTI_ERROR_NOT_COMPATIBLE	The current operation is not compatible with the current state of the object
CUPTI_ERROR_NOT_INITIALIZED	CUPTI is unable to initialize its connection to the CUDA driver.
CUPTI_ERROR_INVALID_METRIC_ID	The metric id is invalid.
CUPTI_ERROR_INVALID_METRIC_NAME	The metric name is invalid.
CUPTI_ERROR_QUEUE_EMPTY	The queue is empty.
CUPTI_ERROR_INVALID_HANDLE	Invalid handle (internal?).
CUPTI_ERROR_INVALID_STREAM	Invalid stream.
CUPTI_ERROR_INVALID_KIND	Invalid kind.
CUPTI_ERROR_INVALID_EVENT_VALUE	Invalid event value.
CUPTI_ERROR_DISABLED	CUPTI is disabled due to conflicts with other enabled profilers
CUPTI_ERROR_INVALID_MODULE	Invalid module.
CUPTI_ERROR_UNKNOWN	An unknown internal error has occurred.

Function Documentation

CUptiResult cuptiGetResultString (**CUptiResult** result, const char ** str)

Return the descriptive string for a CUptiResult in `*str`.

Note:

Thread-safety: this function is thread safe.

Parameters:

result	The result to get the string for
str	Returns the string

Return values:

CUPTI_SUCCESS	on success
CUPTI_ERROR_INVALID_PARAMETER	if `str` is NULL or `result` is not a valid CUptiResult

CUPTI Activity API

Data Structures

- struct CUpti_Activity

  The base activity record.

- struct CUpti_ActivityAPI

  The activity record for a driver or runtime API invocation.

- struct CUpti_ActivityBranch

  The activity record for source level result branch.

- struct CUpti_ActivityContext

  The activity record for a context.

- struct CUpti_ActivityDevice

  The activity record for a device.

- struct CUpti_ActivityEvent

  The activity record for a CUPTI event.

- struct CUpti_ActivityGlobalAccess

  The activity record for source-level global access.

- struct CUpti_ActivityKernel

  The activity record for kernel.

- struct CUpti_ActivityMarker

  The activity record providing a marker which is an instantaneous point in time.

- struct CUpti_ActivityMarkerData

  The activity record providing detailed information for a marker.

- struct CUpti_ActivityMemcpy

  The activity record for memory copies.

- struct CUpti_ActivityMemset

The activity record for memset.

- struct CUpti_ActivityMetric

  The activity record for a CUPTI metric.

- struct CUpti_ActivityName

  The activity record providing a name.

- union CUpti_ActivityObjectKindId

  Identifiers for object kinds as specified by CUpti_ActivityObjectKind.

- struct CUpti_ActivityOverhead

  The activity record for CUPTI and driver overheads.

- struct CUpti_ActivitySourceLocator

  The activity record for source locator.

Defines

- #define CUPTI_SOURCE_LOCATOR_ID_UNKNOWN 0

Enumerations

- enum CUpti_ActivityComputeApiKind {
  CUPTI_ACTIVITY_COMPUTE_API_UNKNOWN = 0,
  CUPTI_ACTIVITY_COMPUTE_API_CUDA = 1 }

  The kind of a compute API.

- enum CUpti_ActivityFlag {
  CUPTI_ACTIVITY_FLAG_NONE = 0,
  CUPTI_ACTIVITY_FLAG_DEVICE_CONCURRENT_KERNELS = 1 << 0,
  CUPTI_ACTIVITY_FLAG_MEMCPY_ASYNC = 1 << 0,
  CUPTI_ACTIVITY_FLAG_MARKER_INSTANTANEOUS = 1 << 0,
  CUPTI_ACTIVITY_FLAG_MARKER_START = 1 << 1,
  CUPTI_ACTIVITY_FLAG_MARKER_END = 1 << 2,

CUPTI_ACTIVITY_FLAG_MARKER_COLOR_NONE = 1 << 0,

CUPTI_ACTIVITY_FLAG_MARKER_COLOR_ARGB = 1 << 1,

CUPTI_ACTIVITY_FLAG_GLOBAL_ACCESS_KIND_SIZE_MASK = 0xFF << 0,

CUPTI_ACTIVITY_FLAG_GLOBAL_ACCESS_KIND_LOAD = 1 << 8,

CUPTI_ACTIVITY_FLAG_GLOBAL_ACCESS_KIND_CACHED = 1 << 9 }

Flags associated with activity records.

- enum CUpti_ActivityKind {

CUPTI_ACTIVITY_KIND_INVALID = 0,

CUPTI_ACTIVITY_KIND_MEMCPY = 1,

CUPTI_ACTIVITY_KIND_MEMSET = 2,

CUPTI_ACTIVITY_KIND_KERNEL = 3,

CUPTI_ACTIVITY_KIND_DRIVER = 4,

CUPTI_ACTIVITY_KIND_RUNTIME = 5,

CUPTI_ACTIVITY_KIND_EVENT = 6,

CUPTI_ACTIVITY_KIND_METRIC = 7,

CUPTI_ACTIVITY_KIND_DEVICE = 8,

CUPTI_ACTIVITY_KIND_CONTEXT = 9,

CUPTI_ACTIVITY_KIND_CONCURRENT_KERNEL = 10,

CUPTI_ACTIVITY_KIND_NAME = 11,

CUPTI_ACTIVITY_KIND_MARKER = 12,

CUPTI_ACTIVITY_KIND_MARKER_DATA = 13,

CUPTI_ACTIVITY_KIND_SOURCE_LOCATOR = 14,

CUPTI_ACTIVITY_KIND_GLOBAL_ACCESS = 15,

CUPTI_ACTIVITY_KIND_BRANCH = 16,

CUPTI_ACTIVITY_KIND_OVERHEAD = 17 }

The kinds of activity records.

- enum CUpti_ActivityMemcpyKind {

CUPTI_ACTIVITY_MEMCPY_KIND_UNKNOWN = 0,

CUPTI_ACTIVITY_MEMCPY_KIND_HTOD = 1,

CUPTI_ACTIVITY_MEMCPY_KIND_DTOH = 2,

CUPTI_ACTIVITY_MEMCPY_KIND_HTOA = 3,

CUPTI_ACTIVITY_MEMCPY_KIND_ATOH = 4,

CUPTI_ACTIVITY_MEMCPY_KIND_ATOA = 5,

CUPTI_ACTIVITY_MEMCPY_KIND_ATOD = 6,

CUPTI_ACTIVITY_MEMCPY_KIND_DTOA = 7,

CUPTI_ACTIVITY_MEMCPY_KIND_DTOD = 8,

CUPTI_ACTIVITY_MEMCPY_KIND_HTOH = 9 }

The kind of a memory copy, indicating the source and destination targets of the copy.

- enum CUpti_ActivityMemoryKind {

CUPTI_ACTIVITY_MEMORY_KIND_UNKNOWN = 0,

CUPTI_ACTIVITY_MEMORY_KIND_PAGEABLE = 1,

CUPTI_ACTIVITY_MEMORY_KIND_PINNED = 2,

CUPTI_ACTIVITY_MEMORY_KIND_DEVICE = 3,

CUPTI_ACTIVITY_MEMORY_KIND_ARRAY = 4 }

The kinds of memory accessed by a memory copy.

- enum CUpti_ActivityObjectKind {

CUPTI_ACTIVITY_OBJECT_UNKNOWN = 0,

CUPTI_ACTIVITY_OBJECT_PROCESS = 1,

CUPTI_ACTIVITY_OBJECT_THREAD = 2,

CUPTI_ACTIVITY_OBJECT_DEVICE = 3,

CUPTI_ACTIVITY_OBJECT_CONTEXT = 4,

CUPTI_ACTIVITY_OBJECT_STREAM = 5 }

The kinds of activity objects.

- enum CUpti_ActivityOverheadKind {

CUPTI_ACTIVITY_OVERHEAD_UNKNOWN = 0,

CUPTI_ACTIVITY_OVERHEAD_DRIVER_COMPILER = 1,

CUPTI_ACTIVITY_OVERHEAD_CUPTI_BUFFER_FLUSH = 1<<16,

CUPTI_ACTIVITY_OVERHEAD_CUPTI_INSTRUMENTATION = 2<<16,

CUPTI_ACTIVITY_OVERHEAD_CUPTI_RESOURCE = 3<<16 }

The kinds of activity overhead.

Functions

- CUptiResult cuptiActivityDequeueBuffer (CUcontext context, uint32_t streamId, uint8_t **buffer, size_t *validBufferSizeBytes)

  Dequeue a buffer containing activity records.

- CUptiResult cuptiActivityDisable (CUpti_ActivityKind kind)

  Disable collection of a specific kind of activity record.

- CUptiResult cuptiActivityDisableContext (CUcontext context, CUpti_ActivityKind kind)

  Disable collection of a specific kind of activity record for a context.

- CUptiResult cuptiActivityEnable (CUpti_ActivityKind kind)

  Enable collection of a specific kind of activity record.

- CUptiResult cuptiActivityEnableContext (CUcontext context, CUpti_ActivityKind kind)

  Enable collection of a specific kind of activity record for a context.

- CUptiResult cuptiActivityEnqueueBuffer (CUcontext context, uint32_t streamId, uint8_t *buffer, size_t bufferSizeBytes)

  Queue a buffer for activity record collection.

- CUptiResult cuptiActivityGetNextRecord (uint8_t *buffer, size_t validBufferSizeBytes, CUpti_Activity **record)

  Iterate over the activity records in a buffer.

- CUptiResult cuptiActivityGetNumDroppedRecords (CUcontext context, uint32_t streamId, size_t *dropped)

  Get the number of activity records that were dropped from a queue because of insufficient buffer space.

- CUptiResult cuptiActivityQueryBuffer (CUcontext context, uint32_t streamId, size_t *validBufferSizeBytes)

  Query the status of the buffer at the head of a queue.

- CUptiResult cuptiGetDeviceId (CUcontext context, uint32_t *deviceId)

  Get the ID of a device.

- CUptiResult cuptiGetStreamId (CUcontext context, CUstream stream, uint32_t *streamId)

  Get the ID of a stream.

- CUptiResult cuptiGetTimestamp (uint64_t *timestamp)

  Get the CUPTI timestamp.

Detailed Description

Functions, types, and enums that implement the CUPTI Activity API.

Define Documentation

#define CUPTI_SOURCE_LOCATOR_ID_UNKNOWN 0

The source-locator ID that indicates an unknown source location. There is not an actual CUpti_ActivitySourceLocator object corresponding to this value.

Enumeration Type Documentation

enum **CUpti_ActivityComputeApiKind**

Enumerator:

CUPTI_ACTIVITY_COMPUTE_API_UNKNOWN	The compute API is not known.
CUPTI_ACTIVITY_COMPUTE_API_CUDA	The compute APIs are for CUDA.

enum **CUpti_ActivityFlag**

Activity record flags. Flags can be combined by bitwise OR to associated multiple flags with an activity record. Each flag is specific to a certain activity kind, as noted below.

Enumerator:

CUPTI_ACTIVITY_FLAG_NONE	Indicates the activity record has no flags.

CUPTI_ACTIVITY_FLAG_DEVICE_CONCURRENT_KERNELS Indicates the activity represents a device that supports concurrent kernel execution. Valid for CUPTI_ACTIVITY_KIND_DEVICE.

CUPTI_ACTIVITY_FLAG_MEMCPY_ASYNC Indicates the activity represents an asychronous memcpy operation. Valid for CUPTI_ACTIVITY_KIND_MEMCPY.

CUPTI_ACTIVITY_FLAG_MARKER_INSTANTANEOUS Indicates the activity represents an instantaneous marker. Valid for CUPTI_ACTIVITY_KIND_MARKER.

CUPTI_ACTIVITY_FLAG_MARKER_START Indicates the activity represents a region start marker. Valid for CUPTI_ACTIVITY_KIND_MARKER.

CUPTI_ACTIVITY_FLAG_MARKER_END Indicates the activity represents a region end marker. Valid for CUPTI_ACTIVITY_KIND_MARKER.

CUPTI_ACTIVITY_FLAG_MARKER_COLOR_NONE Indicates the activity represents a marker that does not specify a color. Valid for CUPTI_ACTIVITY_KIND_MARKER_DATA.

CUPTI_ACTIVITY_FLAG_MARKER_COLOR_ARGB Indicates the activity represents a marker that specifies a color in alpha-red-green-blue format. Valid for CUPTI_ACTIVITY_KIND_MARKER_DATA.

CUPTI_ACTIVITY_FLAG_GLOBAL_ACCESS_KIND_SIZE_MASK The number of bytes requested by each thread Valid for CUpti_ActivityGlobalAccess.

CUPTI_ACTIVITY_FLAG_GLOBAL_ACCESS_KIND_LOAD If bit in this flag is set, the access was load, else it is a store access. Valid for CUpti_ActivityGlobalAccess.

CUPTI_ACTIVITY_FLAG_GLOBAL_ACCESS_KIND_CACHED If this bit in flag is set, the load access was cached else it is uncached. Valid for CUpti_ActivityGlobalAccess.

enum **CUpti_ActivityKind**

Each activity record kind represents information about a GPU or an activity occurring on a CPU or GPU. Each kind is associated with a activity record structure that holds the information associated with the kind.

See also:

CUpti_Activity
CUpti_ActivityAPI
CUpti_ActivityContext
CUpti_ActivityDevice
CUpti_ActivityEvent

CUpti_ActivityKernel
CUpti_ActivityMemcpy
CUpti_ActivityMemset
CUpti_ActivityMetric
CUpti_ActivityName
CUpti_ActivityMarker
CUpti_ActivityMarkerData
CUpti_ActivitySourceLocator
CUpti_ActivityGlobalAccess
CUpti_ActivityBranch
CUpti_ActivityOverhead

Enumerator:

CUPTI_ACTIVITY_KIND_INVALID The activity record is invalid.

CUPTI_ACTIVITY_KIND_MEMCPY A host<->host, host<->device, or device<->device memory copy. The corresponding activity record structure is CUpti_ActivityMemcpy.

CUPTI_ACTIVITY_KIND_MEMSET A memory set executing on the GPU. The corresponding activity record structure is CUpti_ActivityMemset.

CUPTI_ACTIVITY_KIND_KERNEL A kernel executing on the GPU. The corresponding activity record structure is CUpti_ActivityKernel.

CUPTI_ACTIVITY_KIND_DRIVER A CUDA driver API function execution. The corresponding activity record structure is CUpti_ActivityAPI.

CUPTI_ACTIVITY_KIND_RUNTIME A CUDA runtime API function execution. The corresponding activity record structure is CUpti_ActivityAPI.

CUPTI_ACTIVITY_KIND_EVENT An event value. The corresponding activity record structure is CUpti_ActivityEvent.

CUPTI_ACTIVITY_KIND_METRIC A metric value. The corresponding activity record structure is CUpti_ActivityMetric.

CUPTI_ACTIVITY_KIND_DEVICE Information about a device. The corresponding activity record structure is CUpti_ActivityDevice.

CUPTI_ACTIVITY_KIND_CONTEXT Information about a context. The corresponding activity record structure is CUpti_ActivityContext.

CUPTI_ACTIVITY_KIND_CONCURRENT_KERNEL A (potentially concurrent) kernel executing on the GPU. The corresponding activity record structure is CUpti_ActivityKernel.

CUPTI_ACTIVITY_KIND_NAME Thread, device, context, etc. name. The corresponding activity record structure is CUpti_ActivityName.

CUPTI_ACTIVITY_KIND_MARKER Instantaneous, start, or end marker.

CUPTI_ACTIVITY_KIND_MARKER_DATA Extended, optional, data about a marker.

CUPTI_ACTIVITY_KIND_SOURCE_LOCATOR Source information about source level result. The corresponding activity record structure is CUpti_ActivitySourceLocator.

CUPTI_ACTIVITY_KIND_GLOBAL_ACCESS Results for source-level global acccess. The corresponding activity record structure is CUpti_ActivityGlobalAccess.

CUPTI_ACTIVITY_KIND_BRANCH Results for source-level branch. The corresponding activity record structure is CUpti_ActivityBranch.

CUPTI_ACTIVITY_KIND_OVERHEAD Overhead activity records. The corresponding activity record structure is CUpti_ActivityOverhead.

enum **CUpti_ActivityMemcpyKind**

Each kind represents the source and destination targets of a memory copy. Targets are host, device, and array.

Enumerator:

CUPTI_ACTIVITY_MEMCPY_KIND_UNKNOWN The memory copy kind is not known.

CUPTI_ACTIVITY_MEMCPY_KIND_HTOD A host to device memory copy.

CUPTI_ACTIVITY_MEMCPY_KIND_DTOH A device to host memory copy.

CUPTI_ACTIVITY_MEMCPY_KIND_HTOA A host to device array memory copy.

CUPTI_ACTIVITY_MEMCPY_KIND_ATOH A device array to host memory copy.

CUPTI_ACTIVITY_MEMCPY_KIND_ATOA A device array to device array memory copy.

CUPTI_ACTIVITY_MEMCPY_KIND_ATOD A device array to device memory copy.

CUPTI_ACTIVITY_MEMCPY_KIND_DTOA A device to device array memory copy.

CUPTI_ACTIVITY_MEMCPY_KIND_DTOD A device to device memory copy.

CUPTI_ACTIVITY_MEMCPY_KIND_HTOH A host to host memory copy.

enum **CUpti_ActivityMemoryKind**

Each kind represents the type of the source or destination memory accessed by a memory copy.

Enumerator:

CUPTI_ACTIVITY_MEMORY_KIND_UNKNOWN The source or destination memory kind is unknown.

CUPTI_ACTIVITY_MEMORY_KIND_PAGEABLE The source or destination memory is pageable.

CUPTI_ACTIVITY_MEMORY_KIND_PINNED The source or destination memory is pinned.

CUPTI_ACTIVITY_MEMORY_KIND_DEVICE The source or destination memory is on the device.

CUPTI_ACTIVITY_MEMORY_KIND_ARRAY The source or destination memory is an array.

enum **CUpti_ActivityObjectKind**

See also:

CUpti_ActivityObjectKindId

Enumerator:

CUPTI_ACTIVITY_OBJECT_UNKNOWN The object kind is not known.

CUPTI_ACTIVITY_OBJECT_PROCESS A process.

CUPTI_ACTIVITY_OBJECT_THREAD A thread.

CUPTI_ACTIVITY_OBJECT_DEVICE A device.

CUPTI_ACTIVITY_OBJECT_CONTEXT A context.

CUPTI_ACTIVITY_OBJECT_STREAM A stream.

enum **CUpti_ActivityOverheadKind**

Enumerator:

CUPTI_ACTIVITY_OVERHEAD_UNKNOWN The overhead kind is not known.

CUPTI_ACTIVITY_OVERHEAD_DRIVER_COMPILER Compiler(JIT) overhead.

CUPTI_ACTIVITY_OVERHEAD_CUPTI_BUFFER_FLUSH Activity buffer flush overhead.

CUPTI_ACTIVITY_OVERHEAD_CUPTI_INSTRUMENTATION CUPTI instrumentation overhead.

CUPTI_ACTIVITY_OVERHEAD_CUPTI_RESOURCE CUPTI resource creation and destruction overhead.

Function Documentation

CUptiResult cuptiActivityDequeueBuffer (CUcontext context, uint32_t streamId, uint8_t ** buffer, size_t * validBufferSizeBytes)

Remove the buffer from the head of the specified queue. See cuptiActivityEnqueueBuffer() for description of queues. Calling this function transfers ownership of the buffer from CUPTI. CUPTI will no add any activity records to the buffer after it is dequeued.

Parameters:

context The context, or NULL to dequeue from the global queue

streamId The stream ID

buffer Returns the dequeued buffer

validBufferSizeBytes Returns the number of bytes in the buffer that contain activity records

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_PARAMETER if `buffer` or `validBufferSizeBytes` are NULL

CUPTI_ERROR_QUEUE_EMPTY the queue is empty, `buffer` returns NULL and `validBufferSizeBytes` returns 0

CUptiResult cuptiActivityDisable (**CUpti_ActivityKind** kind)

Disable collection of a specific kind of activity record. Multiple kinds can be disabled by calling this function multiple times. By default all activity kinds are disabled for collection.

Parameters:

kind The kind of activity record to stop collecting

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUptiResult cuptiActivityDisableContext (CUcontext context, **CUpti_ActivityKind** kind)

Disable collection of a specific kind of activity record for a context. This setting done by this API will supercede the global settings for activity records. Multiple kinds can be enabled by calling this function multiple times.

Parameters:

context The context for which activity is to be disabled

kind The kind of activity record to stop collecting

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUptiResult cuptiActivityEnable (**CUpti_ActivityKind** kind)

Enable collection of a specific kind of activity record. Multiple kinds can be enabled by calling this function multiple times. By default all activity kinds are disabled for collection.

Parameters:

kind The kind of activity record to collect

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_NOT_COMPATIBLE if the activity kind cannot be enabled

CUptiResult cuptiActivityEnableContext (CUcontext context, **CUpti_ActivityKind** kind)

Enable collection of a specific kind of activity record for a context. This setting done by this API will supercede the global settings for activity records enabled by cuptiActivityEnable Multiple kinds can be enabled by calling this function multiple times.

Parameters:

context The context for which activity is to be enabled

kind The kind of activity record to collect

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_NOT_COMPATIBLE if the activity kind cannot be enabled

CUptiResult cuptiActivityEnqueueBuffer (CUcontext context, uint32_t streamId, uint8_t * buffer, size_t bufferSizeBytes)

Queue a buffer for activity record collection. Calling this function transfers ownership of the buffer to CUPTI. The buffer should not be accessed or modified until ownership is regained by calling cuptiActivityDequeueBuffer().

There are three types of queues:

Global Queue: The global queue collects all activity records that are not associated with a valid context. All device and API activity records are collected in the global queue. A buffer is enqueued in the global queue by specifying `context` == NULL.

Context Queue: Each context queue collects activity records associated with that context that are not associated with a specific stream or that are associated with the default stream. A buffer is enqueued in a context queue by specifying the context and a `streamId` of 0.

Stream Queue: Each stream queue collects memcpy, memset, and kernel activity records associated with the stream. A buffer is enqueued in a stream queue by specifying a context and a non-zero stream ID.

Multiple buffers can be enqueued on each queue, and buffers can be enqueue on multiple queues.

When a new activity record needs to be recorded, CUPTI searches for a non-empty queue to hold the record in this order: 1) the appropriate stream queue, 2) the appropriate context queue. If the search does not find any queue with a buffer then the activity record is dropped. If the search finds a queue containing a buffer, but that buffer is full, then the activity record is dropped and the dropped record count for the queue is incremented. If the search finds a queue containing a buffer with space available to hold the record, then the record is recorded in the buffer.

At a minimum, one or more buffers must be queued in the global queue and context queue at all times to avoid dropping activity records. Global queue will not store any activity records for gpu activity(kernel, memcpy, memset). It is also necessary to enqueue at least one buffer in the context queue of each context as it is created. The stream queues are optional and can be used to reduce or eliminate application perturbations caused by the need to process or save the activity records returned in the buffers. For example, if a stream queue is used, that queue can be flushed when the stream is synchronized.

Parameters:

context The context, or NULL to enqueue on the global queue

streamId The stream ID

buffer The pointer to user supplied buffer for storing activity records.The buffer must be at least 8 byte aligned, and the size of the buffer must be at least 1024 bytes.

bufferSizeBytes The size of the buffer, in bytes. The size of the buffer must be at least

1024 bytes.

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_PARAMETER if `buffer` is NULL, does not have alignment of at least 8 bytes, or is not at least 1024 bytes in size

CUptiResult cuptiActivityGetNextRecord (uint8_t * buffer, size_t validBufferSizeBytes, **CUpti_Activity** ** record)

This is a helper function to iterate over the activity records in a buffer. A buffer of activity records is typically obtained by using the cuptiActivityDequeueBuffer() function.

An example of typical usage:

```
CUpti_Activity *record = NULL;
CUptiResult status = CUPTI_SUCCESS;
  do {
     status = cuptiActivityGetNextRecord(buffer, validSize, &record);
     if(status == CUPTI_SUCCESS) {
          // Use record here...
     }
     else if (status == CUPTI_ERROR_MAX_LIMIT_REACHED)
         break;
     else {
         goto Error;
     }
   } while (1);
```

Parameters:

buffer The buffer containing activity records

record Inputs the previous record returned by cuptiActivityGetNextRecord and returns the next activity record from the buffer. If input value if NULL, returns the first activity record in the buffer.

validBufferSizeBytes The number of valid bytes in the buffer.

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_MAX_LIMIT_REACHED if no more records in the buffer

CUPTI_ERROR_INVALID_PARAMETER if `buffer` is NULL.

CUptiResult cuptiActivityGetNumDroppedRecords (CUcontext context, uint32_t streamId, size_t * dropped)

Get the number of records that were dropped from a queue because all the buffers in the queue are full. See cuptiActivityEnqueueBuffer() for description of queues. Calling this function does not transfer ownership of the buffer. The dropped count maintained for the queue is reset to zero when this function is called.

Parameters:

context The context, or NULL to get dropped count from global queue

streamId The stream ID

dropped The number of records that were dropped since the last call to this function.

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_PARAMETER if `dropped` is NULL

CUptiResult cuptiActivityQueryBuffer (CUcontext context, uint32_t streamId, size_t * validBufferSizeBytes)

Query the status of buffer at the head in the queue. See cuptiActivityEnqueueBuffer() for description of queues. Calling this function does not transfer ownership of the buffer.

Parameters:

context The context, or NULL to query the global queue

streamId The stream ID

validBufferSizeBytes Returns the number of bytes in the buffer that contain activity records

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_PARAMETER if `buffer` or `validBufferSizeBytes` are NULL

CUPTI_ERROR_MAX_LIMIT_REACHED if buffer is full

CUPTI_ERROR_QUEUE_EMPTY the queue is empty, `validBufferSizeBytes` returns 0

CUptiResult cuptiGetDeviceId (CUcontext context, uint32_t * deviceId)

If `context` is NULL, returns the ID of the device that contains the currently active context. If `context` is non-NULL, returns the ID of the device which contains that context. Operates in a similar manner to cudaGetDevice() or cuCtxGetDevice() but may be called from within callback functions.

Parameters:

context The context, or NULL to indicate the current context.

deviceId Returns the ID of the device that is current for the calling thread.

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE if unable to get device ID

CUPTI_ERROR_INVALID_PARAMETER if `deviceId` is NULL

CUptiResult cuptiGetStreamId (CUcontext context, CUstream stream, uint32_t * streamId)

Get the ID of a stream. The stream ID is unique within a context (i.e. all streams within a context will have unique stream IDs).

Parameters:

context If non-NULL then the stream is checked to ensure that it belongs to this context. Typically this parameter should be null.

stream The stream

streamId Returns a context-unique ID for the stream

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_STREAM if unable to get stream ID, or if `context` is non-NULL and `stream` does not belong to the context

CUPTI_ERROR_INVALID_PARAMETER if `streamId` is NULL

See also:

cuptiActivityEnqueueBuffer
cuptiActivityDequeueBuffer

CUptiResult cuptiGetTimestamp (uint64_t * timestamp)

Returns a timestamp normalized to correspond with the start and end timestamps reported in the CUPTI activity records. The timestamp is reported in nanoseconds.

Parameters:

timestamp	Returns the CUPTI timestamp

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_INVALID_PARAMETER if `timestamp` is NULL

CUpti_Activity Type Reference

The base activity record.

Data Fields

- CUpti_ActivityKind kind

Detailed Description

The activity API uses a CUpti_Activity as a generic representation for any activity. The 'kind' field is used to determine the specific activity kind, and from that the CUpti_Activity object can be cast to the specific activity record type appropriate for that kind.

Note that all activity record types are padded and aligned to ensure that each member of the record is naturally aligned.

See also:

CUpti_ActivityKind

Field Documentation

CUpti_ActivityKind CUpti_Activity::kind

The kind of this activity.

CUpti_ActivityAPI Type Reference

The activity record for a driver or runtime API invocation.

Data Fields

- CUpti_CallbackId cbid
- uint32_t correlationId
- uint64_t end
- CUpti_ActivityKind kind
- uint32_t processId
- uint32_t returnValue
- uint64_t start
- uint32_t threadId

Detailed Description

This activity record represents an invocation of a driver or runtime API (CUPTI_ACTIVITY_KIND_DRIVER and CUPTI_ACTIVITY_KIND_RUNTIME).

Field Documentation

CUpti_CallbackId CUpti_ActivityAPI::cbid

The ID of the driver or runtime function.

uint32_t **CUpti_ActivityAPI::correlationId**

The correlation ID of the driver or runtime CUDA function. Each function invocation is assigned a unique correlation ID that is identical to the correlation ID in the memcpy, memset, or kernel activity record that is associated with this function.

uint64_t **CUpti_ActivityAPI::end**

The end timestamp for the function, in ns.

CUpti_ActivityKind CUpti_ActivityAPI::kind

The activity record kind, must be CUPTI_ACTIVITY_KIND_DRIVER or CUPTI_ACTIVITY_KIND_RUNTIME.

uint32_t CUpti_ActivityAPI::processId

The ID of the process where the driver or runtime CUDA function is executing.

uint32_t CUpti_ActivityAPI::returnValue

The return value for the function. For a CUDA driver function with will be a CUresult value, and for a CUDA runtime function this will be a cudaError_t value.

uint64_t CUpti_ActivityAPI::start

The start timestamp for the function, in ns.

uint32_t CUpti_ActivityAPI::threadId

The ID of the thread where the driver or runtime CUDA function is executing.

CUpti_ActivityDevice Type Reference

The activity record for a device.

Data Fields

- uint32_t computeCapabilityMajor
- uint32_t computeCapabilityMinor
- uint32_t constantMemorySize
- uint32_t coreClockRate
- CUpti_ActivityFlag flags
- uint64_t globalMemoryBandwidth
- uint64_t globalMemorySize
- uint32_t id
- CUpti_ActivityKind kind
- uint32_t l2CacheSize
- uint32_t maxBlockDimX
- uint32_t maxBlockDimY
- uint32_t maxBlockDimZ
- uint32_t maxBlocksPerMultiprocessor
- uint32_t maxGridDimX
- uint32_t maxGridDimY
- uint32_t maxGridDimZ
- uint32_t maxIPC
- uint32_t maxRegistersPerBlock
- uint32_t maxSharedMemoryPerBlock
- uint32_t maxThreadsPerBlock
- uint32_t maxWarpsPerMultiprocessor
- const char * name
- uint32_t numMemcpyEngines
- uint32_t numMultiprocessors
- uint32_t numThreadsPerWarp

Detailed Description

This activity record represents information about a GPU device (CUPTI_ACTIVITY_KIND_DEVICE).

Field Documentation

uint32_t **CUpti_ActivityDevice::computeCapabilityMajor**

Compute capability for the device, major number.

uint32_t **CUpti_ActivityDevice::computeCapabilityMinor**

Compute capability for the device, minor number.

uint32_t **CUpti_ActivityDevice::constantMemorySize**

The amount of constant memory on the device, in bytes.

uint32_t **CUpti_ActivityDevice::coreClockRate**

The core clock rate of the device, in kHz.

CUpti_ActivityFlag CUpti_ActivityDevice::flags

The flags associated with the device.

See also:

CUpti_ActivityFlag

uint64_t **CUpti_ActivityDevice::globalMemoryBandwidth**

The global memory bandwidth available on the device, in kBytes/sec.

uint64_t **CUpti_ActivityDevice::globalMemorySize**

The amount of global memory on the device, in bytes.

uint32_t **CUpti_ActivityDevice::id**

The device ID.

CUpti_ActivityKind CUpti_ActivityDevice::kind

The activity record kind, must be CUPTI_ACTIVITY_KIND_DEVICE.

uint32_t **CUpti_ActivityDevice::l2CacheSize**

The size of the L2 cache on the device, in bytes.

uint32_t **CUpti_ActivityDevice::maxBlockDimX**

Maximum allowed X dimension for a block.

uint32_t **CUpti_ActivityDevice::maxBlockDimY**

Maximum allowed Y dimension for a block.

uint32_t **CUpti_ActivityDevice::maxBlockDimZ**

Maximum allowed Z dimension for a block.

uint32_t **CUpti_ActivityDevice::maxBlocksPerMultiprocessor**

Maximum number of blocks that can be present on a multiprocessor at any given time.

uint32_t **CUpti_ActivityDevice::maxGridDimX**

Maximum allowed X dimension for a grid.

uint32_t **CUpti_ActivityDevice::maxGridDimY**

Maximum allowed Y dimension for a grid.

uint32_t **CUpti_ActivityDevice::maxGridDimZ**

Maximum allowed Z dimension for a grid.

uint32_t **CUpti_ActivityDevice::maxIPC**

The maximum "instructions per cycle" possible on each device multiprocessor.

uint32_t **CUpti_ActivityDevice::maxRegistersPerBlock**

Maximum number of registers that can be allocated to a block.

uint32_t **CUpti_ActivityDevice::maxSharedMemoryPerBlock**

Maximum amount of shared memory that can be assigned to a block, in bytes.

uint32_t **CUpti_ActivityDevice::maxThreadsPerBlock**

Maximum number of threads allowed in a block.

uint32_t **CUpti_ActivityDevice::maxWarpsPerMultiprocessor**

Maximum number of warps that can be present on a multiprocessor at any given time.

const char* **CUpti_ActivityDevice::name**

The device name. This name is shared across all activity records representing instances of the device, and so should not be modified.

uint32_t **CUpti_ActivityDevice::numMemcpyEngines**

Number of memory copy engines on the device.

uint32_t **CUpti_ActivityDevice::numMultiprocessors**

Number of multiprocessors on the device.

uint32_t **CUpti_ActivityDevice::numThreadsPerWarp**

The number of threads per warp on the device.

CUpti_ActivityEvent Type Reference

The activity record for a CUPTI event.

Data Fields

- uint32_t correlationId
- CUpti_EventDomainID domain
- CUpti_EventID id
- CUpti_ActivityKind kind
- uint64_t value

Detailed Description

This activity record represents the collection of a CUPTI event value (CUPTI_ACTIVITY_KIND_EVENT). This activity record kind is not produced by the activity API but is included for completeness and ease-of-use. Profile frameworks built on top of CUPTI that collect event data may choose to use this type to store the collected event data.

Field Documentation

uint32_t **CUpti_ActivityEvent::correlationId**

The correlation ID of the event. Use of this ID is user-defined, but typically this ID value will equal the correlation ID of the kernel for which the event was gathered.

CUpti_EventDomainID CUpti_ActivityEvent::domain

The event domain ID.

CUpti_EventID CUpti_ActivityEvent::id

The event ID.

CUpti_ActivityKind CUpti_ActivityEvent::kind

The activity record kind, must be CUPTI_ACTIVITY_KIND_EVENT.

uint64_t **CUpti_ActivityEvent::value**

The event value.

CUpti_ActivityKernel Type Reference

The activity record for kernel.

Data Fields

- int32_t blockX
- int32_t blockY
- int32_t blockZ
- uint8_t cacheConfigExecuted
- uint8_t cacheConfigRequested
- uint32_t contextId
- uint32_t correlationId
- uint32_t deviceId
- int32_t dynamicSharedMemory
- uint64_t end
- int32_t gridX
- int32_t gridY
- int32_t gridZ
- CUpti_ActivityKind kind
- uint32_t localMemoryPerThread
- uint32_t localMemoryTotal
- const char * name
- uint32_t pad
- uint16_t registersPerThread
- void * reserved0
- uint32_t runtimeCorrelationId
- uint64_t start
- int32_t staticSharedMemory
- uint32_t streamId

Detailed Description

This activity record represents a kernel execution (CUPTI_ACTIVITY_KIND_KERNEL and CUPTI_ACTIVITY_KIND_CONCURRENT_KERNEL).

Field Documentation

int32_t **CUpti_ActivityKernel::blockX**

The X-dimension block size for the kernel.

int32_t **CUpti_ActivityKernel::blockY**

The Y-dimension block size for the kernel.

int32_t **CUpti_ActivityKernel::blockZ**

The Z-dimension grid size for the kernel.

uint8_t **CUpti_ActivityKernel::cacheConfigExecuted**

The cache configuration used for the kernel. The value is one of the CUfunc_cache enumeration values from cuda.h.

uint8_t **CUpti_ActivityKernel::cacheConfigRequested**

The cache configuration requested by the kernel. The value is one of the CUfunc_cache enumeration values from cuda.h.

uint32_t **CUpti_ActivityKernel::contextId**

The ID of the context where the kernel is executing.

uint32_t **CUpti_ActivityKernel::correlationId**

The correlation ID of the kernel. Each kernel execution is assigned a unique correlation ID that is identical to the correlation ID in the driver API activity record that launched the kernel.

uint32_t **CUpti_ActivityKernel::deviceId**

The ID of the device where the kernel is executing.

int32_t **CUpti_ActivityKernel::dynamicSharedMemory**

The dynamic shared memory reserved for the kernel, in bytes.

uint64_t **CUpti_ActivityKernel::end**

The end timestamp for the kernel execution, in ns.

int32_t **CUpti_ActivityKernel::gridX**

The X-dimension grid size for the kernel.

int32_t **CUpti_ActivityKernel::gridY**

The Y-dimension grid size for the kernel.

int32_t **CUpti_ActivityKernel::gridZ**

The Z-dimension grid size for the kernel.

CUpti_ActivityKind CUpti_ActivityKernel::kind

The activity record kind, must be CUPTI_ACTIVITY_KIND_KERNEL or CUPTI_ACTIVITY_KIND_CONCURRENT_KERNEL.

uint32_t **CUpti_ActivityKernel::localMemoryPerThread**

The amount of local memory reserved for each thread, in bytes.

uint32_t **CUpti_ActivityKernel::localMemoryTotal**

The total amount of local memory reserved for the kernel, in bytes.

const char∗ **CUpti_ActivityKernel::name**

The name of the kernel. This name is shared across all activity records representing the same kernel, and so should not be modified.

uint32_t **CUpti_ActivityKernel::pad**

Undefined. Reserved for internal use.

uint16_t **CUpti_ActivityKernel::registersPerThread**

The number of registers required for each thread executing the kernel.

void* **CUpti_ActivityKernel::reserved0**

Undefined. Reserved for internal use.

uint32_t **CUpti_ActivityKernel::runtimeCorrelationId**

The runtime correlation ID of the kernel. Each kernel execution is assigned a unique runtime correlation ID that is identical to the correlation ID in the runtime API activity record that launched the kernel.

uint64_t **CUpti_ActivityKernel::start**

The start timestamp for the kernel execution, in ns.

int32_t **CUpti_ActivityKernel::staticSharedMemory**

The static shared memory allocated for the kernel, in bytes.

uint32_t **CUpti_ActivityKernel::streamId**

The ID of the stream where the kernel is executing.

CUpti_ActivityMemcpy Type Reference

The activity record for memory copies.

Data Fields

- uint64_t bytes
- uint32_t contextId
- uint8_t copyKind
- uint32_t correlationId
- uint32_t deviceId
- uint8_t dstKind
- uint64_t end
- uint8_t flags
- CUpti_ActivityKind kind
- void * reserved0
- uint32_t runtimeCorrelationId
- uint8_t srcKind
- uint64_t start
- uint32_t streamId

Detailed Description

This activity record represents a memory copy (CUPTI_ACTIVITY_KIND_MEMCPY).

Field Documentation

uint64_t **CUpti_ActivityMemcpy::bytes**

The number of bytes transferred by the memory copy.

uint32_t **CUpti_ActivityMemcpy::contextId**

The ID of the context where the memory copy is occurring.

uint8_t **CUpti_ActivityMemcpy::copyKind**

The kind of the memory copy, stored as a byte to reduce record size.

See also:

CUpti_ActivityMemcpyKind

uint32_t **CUpti_ActivityMemcpy::correlationId**

The correlation ID of the memory copy. Each memory copy is assigned a unique correlation ID that is identical to the correlation ID in the driver API activity record that launched the memory copy.

uint32_t **CUpti_ActivityMemcpy::deviceId**

The ID of the device where the memory copy is occurring.

uint8_t **CUpti_ActivityMemcpy::dstKind**

The destination memory kind read by the memory copy, stored as a byte to reduce record size.

See also:

CUpti_ActivityMemoryKind

uint64_t **CUpti_ActivityMemcpy::end**

The end timestamp for the memory copy, in ns.

uint8_t **CUpti_ActivityMemcpy::flags**

The flags associated with the memory copy.

See also:

CUpti_ActivityFlag

CUpti_ActivityKind CUpti_ActivityMemcpy::kind

The activity record kind, must be CUPTI_ACTIVITY_KIND_MEMCPY.

void* CUpti_ActivityMemcpy::reserved0

Undefined. Reserved for internal use.

uint32_t CUpti_ActivityMemcpy::runtimeCorrelationId

The runtime correlation ID of the memory copy. Each memory copy is assigned a unique runtime correlation ID that is identical to the correlation ID in the runtime API activity record that launched the memory copy.

uint8_t CUpti_ActivityMemcpy::srcKind

The source memory kind read by the memory copy, stored as a byte to reduce record size.

See also:

CUpti_ActivityMemoryKind

uint64_t CUpti_ActivityMemcpy::start

The start timestamp for the memory copy, in ns.

uint32_t CUpti_ActivityMemcpy::streamId

The ID of the stream where the memory copy is occurring.

CUpti_ActivityMemset Type Reference

The activity record for memset.

Data Fields

- uint64_t bytes
- uint32_t contextId
- uint32_t correlationId
- uint32_t deviceId
- uint64_t end
- CUpti_ActivityKind kind
- void * reserved0
- uint32_t runtimeCorrelationId
- uint64_t start
- uint32_t streamId
- uint32_t value

Detailed Description

This activity record represents a memory set operation (CUPTI_ACTIVITY_KIND_MEMSET).

Field Documentation

uint64_t **CUpti_ActivityMemset::bytes**

The number of bytes being set by the memory set.

uint32_t **CUpti_ActivityMemset::contextId**

The ID of the context where the memory set is occurring.

uint32_t **CUpti_ActivityMemset::correlationId**

The correlation ID of the memory set. Each memory set is assigned a unique correlation ID that is identical to the correlation ID in the driver API activity record that launched the memory set.

uint32_t **CUpti_ActivityMemset::deviceId**

The ID of the device where the memory set is occurring.

uint64_t **CUpti_ActivityMemset::end**

The end timestamp for the memory set, in ns.

CUpti_ActivityKind CUpti_ActivityMemset::kind

The activity record kind, must be CUPTI_ACTIVITY_KIND_MEMSET.

void* **CUpti_ActivityMemset::reserved0**

Undefined. Reserved for internal use.

uint32_t **CUpti_ActivityMemset::runtimeCorrelationId**

The runtime correlation ID of the memory set. Each memory set is assigned a unique runtime correlation ID that is identical to the correlation ID in the runtime API activity record that launched the memory set.

uint64_t **CUpti_ActivityMemset::start**

The start timestamp for the memory set, in ns.

uint32_t **CUpti_ActivityMemset::streamId**

The ID of the stream where the memory set is occurring.

uint32_t **CUpti_ActivityMemset::value**

The value being assigned to memory by the memory set.

CUpti_ActivityMetric Type Reference

The activity record for a CUPTI metric.

Data Fields

- uint32_t correlationId
- CUpti_MetricID id
- CUpti_ActivityKind kind
- uint32_t pad
- CUpti_MetricValue value

Detailed Description

This activity record represents the collection of a CUPTI metric value (CUPTI_ACTIVITY_KIND_METRIC). This activity record kind is not produced by the activity API but is included for completeness and ease-of-use. Profile frameworks built on top of CUPTI that collect metric data may choose to use this type to store the collected metric data.

Field Documentation

uint32_t **CUpti_ActivityMetric::correlationId**

The correlation ID of the metric. Use of this ID is user-defined, but typically this ID value will equal the correlation ID of the kernel for which the metric was gathered.

CUpti_MetricID CUpti_ActivityMetric::id

The metric ID.

CUpti_ActivityKind CUpti_ActivityMetric::kind

The activity record kind, must be CUPTI_ACTIVITY_KIND_METRIC.

uint32_t **CUpti_ActivityMetric::pad**

Undefined. Reserved for internal use.

CUpti_MetricValue CUpti_ActivityMetric::value

The metric value.

CUPTI Callback API

Data Structures

- struct CUpti_CallbackData

  Data passed into a runtime or driver API callback function.

- struct CUpti_NvtxData

  Data passed into a NVTX callback function.

- struct CUpti_ResourceData

  Data passed into a resource callback function.

- struct CUpti_SynchronizeData

  Data passed into a synchronize callback function.

Typedefs

- typedef void(∗ CUpti_CallbackFunc)(void ∗userdata, CUpti_CallbackDomain domain, CUpti_CallbackId cbid, const void ∗cbdata)

  Function type for a callback.

- typedef uint32_t CUpti_CallbackId

  An ID for a driver API, runtime API, resource or synchronization callback.

- typedef CUpti_CallbackDomain ∗ CUpti_DomainTable

  Pointer to an array of callback domains.

- typedef struct CUpti_Subscriber_st ∗ CUpti_SubscriberHandle

  A callback subscriber.

Enumerations

- enum CUpti_ApiCallbackSite {

  CUPTI_API_ENTER = 0,

CUPTI_API_EXIT = 1 }

Specifies the point in an API call that a callback is issued.

- enum CUpti_CallbackDomain {
  CUPTI_CB_DOMAIN_INVALID = 0,
  CUPTI_CB_DOMAIN_DRIVER_API = 1,
  CUPTI_CB_DOMAIN_RUNTIME_API = 2,
  CUPTI_CB_DOMAIN_RESOURCE = 3,
  CUPTI_CB_DOMAIN_SYNCHRONIZE = 4,
  CUPTI_CB_DOMAIN_NVTX = 5 }

  Callback domains.

- enum CUpti_CallbackIdResource {
  CUPTI_CBID_RESOURCE_INVALID = 0,
  CUPTI_CBID_RESOURCE_CONTEXT_CREATED = 1,
  CUPTI_CBID_RESOURCE_CONTEXT_DESTROY_STARTING = 2,
  CUPTI_CBID_RESOURCE_STREAM_CREATED = 3,
  CUPTI_CBID_RESOURCE_STREAM_DESTROY_STARTING = 4 }

  Callback IDs for resource domain.

- enum CUpti_CallbackIdSync {
  CUPTI_CBID_SYNCHRONIZE_INVALID = 0,
  CUPTI_CBID_SYNCHRONIZE_STREAM_SYNCHRONIZED = 1,
  CUPTI_CBID_SYNCHRONIZE_CONTEXT_SYNCHRONIZED = 2 }

  Callback IDs for synchronization domain.

Functions

- CUptiResult cuptiEnableAllDomains (uint32_t enable, CUpti_SubscriberHandle subscriber)

  Enable or disable all callbacks in all domains.

- CUptiResult cuptiEnableCallback (uint32_t enable, CUpti_SubscriberHandle subscriber, CUpti_CallbackDomain domain, CUpti_CallbackId cbid)

Enable or disabled callbacks for a specific domain and callback ID.

- ▶ CUptiResult cuptiEnableDomain (uint32_t enable, CUpti_SubscriberHandle subscriber, CUpti_CallbackDomain domain)

  Enable or disabled all callbacks for a specific domain.

- ▶ CUptiResult cuptiGetCallbackName (CUpti_CallbackDomain domain, uint32_t cbid, const char ∗∗name)

  Get the name of a callback for a specific domain and callback ID.

- ▶ CUptiResult cuptiGetCallbackState (uint32_t ∗enable, CUpti_SubscriberHandle subscriber, CUpti_CallbackDomain domain, CUpti_CallbackId cbid)

  Get the current enabled/disabled state of a callback for a specific domain and function ID.

- ▶ CUptiResult cuptiSubscribe (CUpti_SubscriberHandle ∗subscriber, CUpti_CallbackFunc callback, void ∗userdata)

  Initialize a callback subscriber with a callback function and user data.

- ▶ CUptiResult cuptiSupportedDomains (size_t ∗domainCount, CUpti_DomainTable ∗domainTable)

  Get the available callback domains.

- ▶ CUptiResult cuptiUnsubscribe (CUpti_SubscriberHandle subscriber)

  Unregister a callback subscriber.

Detailed Description

Functions, types, and enums that implement the CUPTI Callback API.

Typedef Documentation

typedef void(∗ **CUpti_CallbackFunc**)(void ∗userdata, **CUpti_CallbackDomain** domain, **CUpti_CallbackId** cbid, const void ∗cbdata)

Function type for a callback. The type of the data passed to the callback in `cbdata` depends on the `domain`. If `domain` is CUPTI_CB_DOMAIN_DRIVER_API or CUPTI_CB_DOMAIN_RUNTIME_API the type of `cbdata` will be

CUpti_CallbackData. If `domain` is CUPTI_CB_DOMAIN_RESOURCE the type of `cbdata` will be CUpti_ResourceData. If `domain` is CUPTI_CB_DOMAIN_SYNCHRONIZE the type of `cbdata` will be CUpti_SynchronizeData. If `domain` is CUPTI_CB_DOMAIN_NVTX the type of `cbdata` will be CUpti_NvtxData.

Parameters:

userdata	User data supplied at subscription of the callback
domain	The domain of the callback
cbid	The ID of the callback
cbdata	Data passed to the callback.

typedef uint32_t **CUpti_CallbackId**

An ID for a driver API, runtime API, resource or synchronization callback. Within a driver API callback this should be interpreted as a CUpti_driver_api_trace_cbid value (these values are defined in cupti_driver_cbid.h). Within a runtime API callback this should be interpreted as a CUpti_runtime_api_trace_cbid value (these values are defined in cupti_runtime_cbid.h). Within a resource API callback this should be interpreted as a CUpti_CallbackIdResource value. Within a synchronize API callback this should be interpreted as a CUpti_CallbackIdSync value.

Enumeration Type Documentation

enum **CUpti_ApiCallbackSite**

Specifies the point in an API call that a callback is issued. This value is communicated to the callback function via CUpti_CallbackData::callbackSite.

Enumerator:

CUPTI_API_ENTER	The callback is at the entry of the API call.
CUPTI_API_EXIT	The callback is at the exit of the API call.

enum **CUpti_CallbackDomain**

Callback domains. Each domain represents callback points for a group of related API functions or CUDA driver activity.

Enumerator:

CUPTI_CB_DOMAIN_INVALID	Invalid domain.

CUPTI_CB_DOMAIN_DRIVER_API Domain containing callback points for all driver API functions.

CUPTI_CB_DOMAIN_RUNTIME_API Domain containing callback points for all runtime API functions.

CUPTI_CB_DOMAIN_RESOURCE Domain containing callback points for CUDA resource tracking.

CUPTI_CB_DOMAIN_SYNCHRONIZE Domain containing callback points for CUDA synchronization.

CUPTI_CB_DOMAIN_NVTX Domain containing callback points for NVTX API functions.

enum **CUpti_CallbackIdResource**

Callback IDs for resource domain, CUPTI_CB_DOMAIN_RESOURCE. This value is communicated to the callback function via the `cbid` parameter.

Enumerator:

CUPTI_CBID_RESOURCE_INVALID Invalid resource callback ID.

CUPTI_CBID_RESOURCE_CONTEXT_CREATED A new context has been created.

CUPTI_CBID_RESOURCE_CONTEXT_DESTROY_STARTING A context is about to be destroyed.

CUPTI_CBID_RESOURCE_STREAM_CREATED A new stream has been created.

CUPTI_CBID_RESOURCE_STREAM_DESTROY_STARTING A stream is about to be destroyed.

enum **CUpti_CallbackIdSync**

Callback IDs for synchronization domain, CUPTI_CB_DOMAIN_SYNCHRONIZE. This value is communicated to the callback function via the `cbid` parameter.

Enumerator:

CUPTI_CBID_SYNCHRONIZE_INVALID Invalid synchronize callback ID.

CUPTI_CBID_SYNCHRONIZE_STREAM_SYNCHRONIZED Stream synchronization has completed for the stream.

CUPTI_CBID_SYNCHRONIZE_CONTEXT_SYNCHRONIZED Context synchronization has completed for the context.

Function Documentation

CUptiResult cuptiEnableAllDomains (uint32_t enable, **CUpti_SubscriberHandle** subscriber)

Enable or disable all callbacks in all domains.

Note:

Thread-safety: a subscriber must serialize access to cuptiGetCallbackState, cuptiEnableCallback, cuptiEnableDomain, and cuptiEnableAllDomains. For example, if cuptiGetCallbackState(sub, d, ∗) and cuptiEnableAllDomains(sub) are called concurrently, the results are undefined.

Parameters:

enable New enable state for all callbacks in all domain. Zero disables all callbacks, non-zero enables all callbacks.

subscriber - Handle to callback subscription

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_NOT_INITIALIZED if unable to initialized CUPTI

CUPTI_ERROR_INVALID_PARAMETER if `subscriber` is invalid

CUptiResult cuptiEnableCallback (uint32_t enable, **CUpti_SubscriberHandle** subscriber, **CUpti_CallbackDomain** domain, **CUpti_CallbackId** cbid)

Enable or disabled callbacks for a subscriber for a specific domain and callback ID.

Note:

Thread-safety: a subscriber must serialize access to cuptiGetCallbackState, cuptiEnableCallback, cuptiEnableDomain, and cuptiEnableAllDomains. For example, if cuptiGetCallbackState(sub, d, c) and cuptiEnableCallback(sub, d, c) are called concurrently, the results are undefined.

Parameters:

enable New enable state for the callback. Zero disables the callback, non-zero enables the callback.

subscriber - Handle to callback subscription

domain The domain of the callback

cbid The ID of the callback

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_NOT_INITIALIZED if unable to initialized CUPTI

CUPTI_ERROR_INVALID_PARAMETER if `subscriber`, `domain` or `cbid` is invalid.

CUptiResult cuptiEnableDomain (uint32_t enable, **CUpti_SubscriberHandle** subscriber, **CUpti_CallbackDomain** domain)

Enable or disabled all callbacks for a specific domain.

Note:

Thread-safety: a subscriber must serialize access to cuptiGetCallbackState, cuptiEnableCallback, cuptiEnableDomain, and cuptiEnableAllDomains. For example, if cuptiGetCallbackEnabled(sub, d, ∗) and cuptiEnableDomain(sub, d) are called concurrently, the results are undefined.

Parameters:

enable New enable state for all callbacks in the domain. Zero disables all callbacks, non-zero enables all callbacks.

subscriber - Handle to callback subscription

domain The domain of the callback

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_NOT_INITIALIZED if unable to initialized CUPTI

CUPTI_ERROR_INVALID_PARAMETER if `subscriber` or `domain` is invalid

CUptiResult cuptiGetCallbackName (**CUpti_CallbackDomain** domain, uint32_t cbid, const char ∗∗ name)

Returns a pointer to the name c_string in ∗∗`name`.

Note:

Names are available only for the DRIVER and RUNTIME domains.

Parameters:

domain The domain of the callback

cbid The ID of the callback

name Returns pointer to the name string on success, NULL otherwise

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_INVALID_PARAMETER if `name` is NULL, or if `domain` or `cbid` is invalid.

CUptiResult cuptiGetCallbackState (uint32_t * enable, **CUpti_SubscriberHandle** subscriber, **CUpti_CallbackDomain** domain, **CUpti_CallbackId** cbid)

Returns non-zero in `*enable` if the callback for a domain and callback ID is enabled, and zero if not enabled.

Note:

Thread-safety: a subscriber must serialize access to cuptiGetCallbackState, cuptiEnableCallback, cuptiEnableDomain, and cuptiEnableAllDomains. For example, if cuptiGetCallbackState(sub, d, c) and cuptiEnableCallback(sub, d, c) are called concurrently, the results are undefined.

Parameters:

enable Returns non-zero if callback enabled, zero if not enabled

subscriber Handle to the initialize subscriber

domain The domain of the callback

cbid The ID of the callback

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_NOT_INITIALIZED if unable to initialized CUPTI

CUPTI_ERROR_INVALID_PARAMETER if `enabled` is NULL, or if `subscriber`, `domain` or `cbid` is invalid.

CUptiResult cuptiSubscribe (**CUpti_SubscriberHandle** * subscriber, **CUpti_CallbackFunc** callback, void * userdata)

Initializes a callback subscriber with a callback function and (optionally) a pointer to user data. The returned subscriber handle can be used to enable and disable the callback for specific domains and callback IDs.

Note:

Only a single subscriber can be registered at a time.

This function does not enable any callbacks.
Thread-safety: this function is thread safe.

Parameters:

subscriber Returns handle to initialize subscriber

callback The callback function

userdata A pointer to user data. This data will be passed to the callback function via the `userdata` paramater.

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_NOT_INITIALIZED if unable to initialize CUPTI

CUPTI_ERROR_MAX_LIMIT_REACHED if there is already a CUPTI subscriber

CUPTI_ERROR_INVALID_PARAMETER if `subscriber` is NULL

CUptiResult cuptiSupportedDomains (size_t ∗ domainCount, **CUpti_DomainTable** ∗ domainTable)

Returns in ∗`domainTable` an array of size ∗`domainCount` of all the available callback domains.

Note:

Thread-safety: this function is thread safe.

Parameters:

domainCount Returns number of callback domains

domainTable Returns pointer to array of available callback domains

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_NOT_INITIALIZED if unable to initialize CUPTI

CUPTI_ERROR_INVALID_PARAMETER if `domainCount` or `domainTable` are NULL

CUptiResult cuptiUnsubscribe (**CUpti_SubscriberHandle** subscriber)

Removes a callback subscriber so that no future callbacks will be issued to that subscriber.

Note:

Thread-safety: this function is thread safe.

Parameters:

subscriber Handle to the initialize subscriber

Return values:

CUPTI_SUCCESS on success

CUPTI_ERROR_NOT_INITIALIZED if unable to initialized CUPTI

CUPTI_ERROR_INVALID_PARAMETER if `subscriber` is NULL or not initialized

CUpti_CallbackData Type Reference

Data passed into a runtime or driver API callback function.

Data Fields

- CUpti_ApiCallbackSite callbackSite
- CUcontext context
- uint32_t contextUid
- uint64_t * correlationData
- uint32_t correlationId
- const char * functionName
- const void * functionParams
- void * functionReturnValue
- const char * symbolName

Detailed Description

Data passed into a runtime or driver API callback function as the `cbdata` argument to CUpti_CallbackFunc. The `cbdata` will be this type for `domain` equal to CUPTI_CB_DOMAIN_DRIVER_API or CUPTI_CB_DOMAIN_RUNTIME_API. The callback data is valid only within the invocation of the callback function that is passed the data. If you need to retain some data for use outside of the callback, you must make a copy of that data. For example, if you make a shallow copy of CUpti_CallbackData within a callback, you cannot dereference `functionParams` outside of that callback to access the function parameters. `functionName` is an exception: the string pointed to by `functionName` is a global constant and so may be accessed outside of the callback.

Field Documentation

CUpti_ApiCallbackSite CUpti_CallbackData::callbackSite

Point in the runtime or driver function from where the callback was issued.

CUcontext **CUpti_CallbackData::context**

Driver context current to the thread, or null if no context is current. This value can change from the entry to exit callback of a runtime API function if the runtime initializes a context.

uint32_t **CUpti_CallbackData::contextUid**

Unique ID for the CUDA context associated with the thread. The UIDs are assigned sequentially as contexts are created and are unique within a process.

uint64_t* **CUpti_CallbackData::correlationData**

Pointer to data shared between the entry and exit callbacks of a given runtime or drive API function invocation. This field can be used to pass 64-bit values from the entry callback to the corresponding exit callback.

uint32_t **CUpti_CallbackData::correlationId**

The activity record correlation ID for this callback. For a driver domain callback (i.e. `domain` CUPTI_CB_DOMAIN_DRIVER_API) this ID will equal the correlation ID in the CUpti_ActivityAPI record corresponding to the CUDA driver function call. For a runtime domain callback (i.e. `domain` CUPTI_CB_DOMAIN_RUNTIME_API) this ID will equal the correlation ID in the CUpti_ActivityAPI record corresponding to the CUDA runtime function call. Within the callback, this ID can be recorded to correlate user data with the activity record. This field is new in 4.1.

const char* **CUpti_CallbackData::functionName**

Name of the runtime or driver API function which issued the callback. This string is a global constant and so may be accessed outside of the callback.

const void* **CUpti_CallbackData::functionParams**

Pointer to the arguments passed to the runtime or driver API call. See generated_cuda_runtime_api_meta.h and generated_cuda_meta.h for structure definitions for the parameters for each runtime and driver API function.

void* **CUpti_CallbackData::functionReturnValue**

Pointer to the return value of the runtime or driver API call. This field is only valid within the exit::CUPTI_API_EXIT callback. For a runtime API `functionReturnValue` points to a `cudaError_t`. For a driver API `functionReturnValue` points to a `CUresult`.

const char* **CUpti_CallbackData::symbolName**

Name of the symbol operated on by the runtime or driver API function which issued the callback. This entry is valid only for driver and runtime launch callbacks, where it returns the name of the kernel.

CUpti_ResourceData Type Reference

Data passed into a resource callback function.

Data Fields

- CUcontext context
- void * resourceDescriptor
- CUstream stream

Detailed Description

Data passed into a resource callback function as the `cbdata` argument to CUpti_CallbackFunc. The `cbdata` will be this type for `domain` equal to CUPTI_CB_DOMAIN_RESOURCE. The callback data is valid only within the invocation of the callback function that is passed the data. If you need to retain some data for use outside of the callback, you must make a copy of that data.

Field Documentation

CUcontext **CUpti_ResourceData::context**

For CUPTI_CBID_RESOURCE_CONTEXT_CREATED and CUPTI_CBID_RESOURCE_CONTEXT_DESTROY_STARTING, the context being created or destroyed. For CUPTI_CBID_RESOURCE_STREAM_CREATED and CUPTI_CBID_RESOURCE_STREAM_DESTROY_STARTING, the context containing the stream being created or destroyed.

void* **CUpti_ResourceData::resourceDescriptor**

Reserved for future use.

CUstream **CUpti_ResourceData::stream**

For CUPTI_CBID_RESOURCE_STREAM_CREATED and CUPTI_CBID_RESOURCE_STREAM_DESTROY_STARTING, the stream being created or destroyed.

CUpti_SynchronizeData Type Reference

Data passed into a synchronize callback function.

Data Fields

- CUcontext context
- CUstream stream

Detailed Description

Data passed into a synchronize callback function as the `cbdata` argument to CUpti_CallbackFunc. The `cbdata` will be this type for `domain` equal to CUPTI_CB_DOMAIN_SYNCHRONIZE. The callback data is valid only within the invocation of the callback function that is passed the data. If you need to retain some data for use outside of the callback, you must make a copy of that data.

Field Documentation

CUcontext **CUpti_SynchronizeData::context**

The context of the stream being synchronized.

CUstream **CUpti_SynchronizeData::stream**

The stream being synchronized.

CUPTI Event API

Data Structures

- struct CUpti_EventGroupSet

  A set of event groups.

- struct CUpti_EventGroupSets

  A set of event group sets.

Defines

- #define CUPTI_EVENT_OVERFLOW ((uint64_t)0xFFFFFFFFFFFFFFFFULL)

  The overflow value for a CUPTI event.

Typedefs

- typedef uint32_t CUpti_EventDomainID

  ID for an event domain.

- typedef void * CUpti_EventGroup

  A group of events.

- typedef uint32_t CUpti_EventID

  ID for an event.

Enumerations

- enum CUpti_DeviceAttribute {
  CUPTI_DEVICE_ATTR_MAX_EVENT_ID = 1,
  CUPTI_DEVICE_ATTR_MAX_EVENT_DOMAIN_ID = 2,
  CUPTI_DEVICE_ATTR_GLOBAL_MEMORY_BANDWIDTH = 3,

CUPTI_DEVICE_ATTR_INSTRUCTION_PER_CYCLE = 4,

CUPTI_DEVICE_ATTR_INSTRUCTION_THROUGHPUT_SINGLE_PRECISION = 5 }

Device attributes.

- enum CUpti_EventAttribute {

CUPTI_EVENT_ATTR_NAME = 0,

CUPTI_EVENT_ATTR_SHORT_DESCRIPTION = 1,

CUPTI_EVENT_ATTR_LONG_DESCRIPTION = 2,

CUPTI_EVENT_ATTR_CATEGORY = 3 }

Event attributes.

- enum CUpti_EventCategory {

CUPTI_EVENT_CATEGORY_INSTRUCTION = 0,

CUPTI_EVENT_CATEGORY_MEMORY = 1,

CUPTI_EVENT_CATEGORY_CACHE = 2,

CUPTI_EVENT_CATEGORY_PROFILE_TRIGGER = 3 }

An event category.

- enum CUpti_EventCollectionMode {

CUPTI_EVENT_COLLECTION_MODE_CONTINUOUS = 0,

CUPTI_EVENT_COLLECTION_MODE_KERNEL = 1 }

Event collection modes.

- enum CUpti_EventDomainAttribute {

CUPTI_EVENT_DOMAIN_ATTR_NAME = 0,

CUPTI_EVENT_DOMAIN_ATTR_INSTANCE_COUNT = 1,

CUPTI_EVENT_DOMAIN_ATTR_TOTAL_INSTANCE_COUNT = 3 }

Event domain attributes.

- enum CUpti_EventGroupAttribute {

CUPTI_EVENT_GROUP_ATTR_EVENT_DOMAIN_ID = 0,

CUPTI_EVENT_GROUP_ATTR_PROFILE_ALL_DOMAIN_INSTANCES = 1,

CUPTI_EVENT_GROUP_ATTR_USER_DATA = 2,

CUPTI_EVENT_GROUP_ATTR_NUM_EVENTS = 3,
CUPTI_EVENT_GROUP_ATTR_EVENTS = 4,
CUPTI_EVENT_GROUP_ATTR_INSTANCE_COUNT = 5 }

Event group attributes.

- enum CUpti_ReadEventFlags { CUPTI_EVENT_READ_FLAG_NONE = 0 }

  Flags for cuptiEventGroupReadEvent an cuptiEventGroupReadAllEvents.

Functions

- CUptiResult cuptiDeviceEnumEventDomains (CUdevice device, size_t ∗arraySizeBytes, CUpti_EventDomainID ∗domainArray)

  Get the event domains for a device.

- CUptiResult cuptiDeviceGetAttribute (CUdevice device, CUpti_DeviceAttribute attrib, size_t ∗valueSize, void ∗value)

  Read a device attribute.

- CUptiResult cuptiDeviceGetEventDomainAttribute (CUdevice device, CUpti_EventDomainID eventDomain, CUpti_EventDomainAttribute attrib, size_t ∗valueSize, void ∗value)

  Read an event domain attribute.

- CUptiResult cuptiDeviceGetNumEventDomains (CUdevice device, uint32_t ∗numDomains)

  Get the number of domains for a device.

- CUptiResult cuptiDeviceGetTimestamp (CUcontext context, uint64_t ∗timestamp)

  Read a device timestamp.

- CUptiResult cuptiEnumEventDomains (size_t ∗arraySizeBytes, CUpti_EventDomainID ∗domainArray)

  Get the event domains available on any device.

- CUptiResult cuptiEventDomainEnumEvents (CUpti_EventDomainID eventDomain, size_t ∗arraySizeBytes, CUpti_EventID ∗eventArray)

  Get the events in a domain.

- CUptiResult cuptiEventDomainGetAttribute (CUpti_EventDomainID eventDomain, CUpti_EventDomainAttribute attrib, size_t *valueSize, void *value)

  Read an event domain attribute.

- CUptiResult cuptiEventDomainGetNumEvents (CUpti_EventDomainID eventDomain, uint32_t *numEvents)

  Get number of events in a domain.

- CUptiResult cuptiEventGetAttribute (CUpti_EventID event, CUpti_EventAttribute attrib, size_t *valueSize, void *value)

  Get an event attribute.

- CUptiResult cuptiEventGetIdFromName (CUdevice device, const char *eventName, CUpti_EventID *event)

  Find an event by name.

- CUptiResult cuptiEventGroupAddEvent (CUpti_EventGroup eventGroup, CUpti_EventID event)

  Add an event to an event group.

- CUptiResult cuptiEventGroupCreate (CUcontext context, CUpti_EventGroup *eventGroup, uint32_t flags)

  Create a new event group for a context.

- CUptiResult cuptiEventGroupDestroy (CUpti_EventGroup eventGroup)

  Destroy an event group.

- CUptiResult cuptiEventGroupDisable (CUpti_EventGroup eventGroup)

  Disable an event group.

- CUptiResult cuptiEventGroupEnable (CUpti_EventGroup eventGroup)

  Enable an event group.

- CUptiResult cuptiEventGroupGetAttribute (CUpti_EventGroup eventGroup, CUpti_EventGroupAttribute attrib, size_t *valueSize, void *value)

  Read an event group attribute.

- CUptiResult cuptiEventGroupReadAllEvents (CUpti_EventGroup eventGroup, CUpti_ReadEventFlags flags, size_t *eventValueBufferSizeBytes, uint64_t

∗eventValueBuffer, size_t ∗eventIdArraySizeBytes, CUpti_EventID ∗eventIdArray, size_t ∗numEventIdsRead)

Read the values for all the events in an event group.

- CUptiResult cuptiEventGroupReadEvent (CUpti_EventGroup eventGroup, CUpti_ReadEventFlags flags, CUpti_EventID event, size_t ∗eventValueBufferSizeBytes, uint64_t ∗eventValueBuffer)

  Read the value for an event in an event group.

- CUptiResult cuptiEventGroupRemoveAllEvents (CUpti_EventGroup eventGroup)

  Remove all events from an event group.

- CUptiResult cuptiEventGroupRemoveEvent (CUpti_EventGroup eventGroup, CUpti_EventID event)

  Remove an event from an event group.

- CUptiResult cuptiEventGroupResetAllEvents (CUpti_EventGroup eventGroup)

  Zero all the event counts in an event group.

- CUptiResult cuptiEventGroupSetAttribute (CUpti_EventGroup eventGroup, CUpti_EventGroupAttribute attrib, size_t valueSize, void ∗value)

  Write an event group attribute.

- CUptiResult cuptiEventGroupSetsCreate (CUcontext context, size_t eventIdArraySizeBytes, CUpti_EventID ∗eventIdArray, CUpti_EventGroupSets ∗∗eventGroupPasses)

  For a set of events, get the grouping that indicates the number of passes and the event groups necessary to collect the events.

- CUptiResult cuptiEventGroupSetsDestroy (CUpti_EventGroupSets ∗eventGroupSets)

  Destroy a CUpti_EventGroupSets object.

- CUptiResult cuptiGetNumEventDomains (uint32_t ∗numDomains)

  Get the number of event domains available on any device.

- CUptiResult cuptiSetEventCollectionMode (CUcontext context, CUpti_EventCollectionMode mode)

  Set the event collection mode.

Detailed Description

Functions, types, and enums that implement the CUPTI Event API.

Define Documentation

#define CUPTI_EVENT_OVERFLOW ((uint64_t)0xFFFFFFFFFFFFFFFFULL)

The CUPTI event value that indicates an overflow.

Typedef Documentation

typedef uint32_t **CUpti_EventDomainID**

ID for an event domain. An event domain represents a group of related events. A device may have multiple instances of a domain, indicating that the device can simultaneously record multiple instances of each event within that domain.

typedef void* **CUpti_EventGroup**

An event group is a collection of events that are managed together. All events in an event group must belong to the same domain.

typedef uint32_t **CUpti_EventID**

An event represents a countable activity, action, or occurrence on the device.

Enumeration Type Documentation

enum **CUpti_DeviceAttribute**

CUPTI device attributes. These attributes can be read using cuptiDeviceGetAttribute.

Enumerator:

CUPTI_DEVICE_ATTR_MAX_EVENT_ID Number of event IDs for a device. Value is a uint32_t.

CUPTI_DEVICE_ATTR_MAX_EVENT_DOMAIN_ID Number of event domain IDs for a device. Value is a uint32_t.

CUPTI_DEVICE_ATTR_GLOBAL_MEMORY_BANDWIDTH Get global memory bandwidth in Kbytes/sec. Value is a uint64_t.

CUPTI_DEVICE_ATTR_INSTRUCTION_PER_CYCLE Get theoretical instructions per cycle. Value is a uint32_t.

CUPTI_DEVICE_ATTR_INSTRUCTION_THROUGHPUT_SINGLE_PRECISION Get theoretical number of single precision instructions that can be executed per second. Value is a uint64_t.

enum **CUpti_EventAttribute**

Event attributes. These attributes can be read using cuptiEventGetAttribute.

Enumerator:

CUPTI_EVENT_ATTR_NAME Event name. Value is a null terminated const c-string.

CUPTI_EVENT_ATTR_SHORT_DESCRIPTION Short description of event. Value is a null terminated const c-string.

CUPTI_EVENT_ATTR_LONG_DESCRIPTION Long description of event. Value is a null terminated const c-string.

CUPTI_EVENT_ATTR_CATEGORY Category of event. Value is CUpti_EventCategory.

enum **CUpti_EventCategory**

Each event is assigned to a category that represents the general type of the event. A event's category is accessed using cuptiEventGetAttribute and the CUPTI_EVENT_ATTR_CATEGORY attribute.

Enumerator:

CUPTI_EVENT_CATEGORY_INSTRUCTION An instruction related event.

CUPTI_EVENT_CATEGORY_MEMORY A memory related event.

CUPTI_EVENT_CATEGORY_CACHE A cache related event.

CUPTI_EVENT_CATEGORY_PROFILE_TRIGGER A profile-trigger event.

enum **CUpti_EventCollectionMode**

The event collection mode determines the period over which the events within the enabled event groups will be collected.

Enumerator:

CUPTI_EVENT_COLLECTION_MODE_CONTINUOUS Events are collected for the entire duration between the cuptiEventGroupEnable and cuptiEventGroupDisable calls. This is the default mode.

CUPTI_EVENT_COLLECTION_MODE_KERNEL Events are collected only for the durations of kernel executions that occur between the cuptiEventGroupEnable and cuptiEventGroupDisable calls. Event collection begins when a kernel execution begins, and stops when kernel execution completes. If multiple kernel executions occur between the cuptiEventGroupEnable and cuptiEventGroupDisable calls then the event values must be read after each kernel launch if those events need to be associated with the specific kernel launch.

enum **CUpti_EventDomainAttribute**

Event domain attributes. Except where noted, all the attributes can be read using either cuptiDeviceGetEventDomainAttribute or cuptiEventDomainGetAttribute.

Enumerator:

CUPTI_EVENT_DOMAIN_ATTR_NAME Event domain name. Value is a null terminated const c-string.

CUPTI_EVENT_DOMAIN_ATTR_INSTANCE_COUNT Number of instances of the domain for which event counts will be collected. The domain may have additional instances that cannot be profiled (see CUPTI_EVENT_DOMAIN_ATTR_TOTAL_INSTANCE_COUNT). Can be read only with cuptiDeviceGetEventDomainAttribute. Value is a uint32_t.

CUPTI_EVENT_DOMAIN_ATTR_TOTAL_INSTANCE_COUNT Total number of instances of the domain, including instances that cannot be profiled. Use CUPTI_EVENT_DOMAIN_ATTR_INSTANCE_COUNT to get the number of instances that can be profiled. Can be read only with cuptiDeviceGetEventDomainAttribute. Value is a uint32_t.

enum **CUpti_EventGroupAttribute**

Event group attributes. These attributes can be read using cuptiEventGroupGetAttribute. Attributes marked [rw] can also be written using cuptiEventGroupSetAttribute.

Enumerator:

CUPTI_EVENT_GROUP_ATTR_EVENT_DOMAIN_ID The domain to which the event group is bound. This attribute is set when the first event is added to the group. Value is a CUpti_EventDomainID.

CUPTI_EVENT_GROUP_ATTR_PROFILE_ALL_DOMAIN_INSTANCES
[rw] Profile all the instances of the domain for this eventgroup. This feature can be used to get load balancing across all instances of a domain. Value is an integer.

CUPTI_EVENT_GROUP_ATTR_USER_DATA [rw] Reserved for user data.

CUPTI_EVENT_GROUP_ATTR_NUM_EVENTS Number of events in the group. Value is a uint32_t.

CUPTI_EVENT_GROUP_ATTR_EVENTS Enumerates events in the group. Value is a pointer to buffer of size sizeof(CUpti_EventID) ∗ num_of_events in the eventgroup. num_of_events can be queried using CUPTI_EVENT_GROUP_ATTR_NUM_EVENTS.

CUPTI_EVENT_GROUP_ATTR_INSTANCE_COUNT Number of instances of the domain bound to this event group that will be counted. Value is a uint32_t.

enum **CUpti_ReadEventFlags**

Flags for cuptiEventGroupReadEvent an cuptiEventGroupReadAllEvents.

Enumerator:

CUPTI_EVENT_READ_FLAG_NONE No flags.

Function Documentation

CUptiResult cuptiDeviceEnumEventDomains (CUdevice device, size_t ∗ arraySizeBytes, **CUpti_EventDomainID** ∗ domainArray)

Returns the event domains IDs in `domainArray` for a device. The size of the `domainArray` buffer is given by ∗`arraySizeBytes`. The size of the `domainArray` buffer must be at least `numdomains` ∗ sizeof(CUpti_EventDomainID) or else all domains will not be returned. The value returned in ∗`arraySizeBytes` contains the number of bytes returned in `domainArray`.

Note:

Thread-safety: this function is thread safe.

Parameters:

device The CUDA device

arraySizeBytes The size of `domainArray` in bytes, and returns the number of bytes written to `domainArray`

domainArray Returns the IDs of the event domains for the device

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE

CUPTI_ERROR_INVALID_PARAMETER if `arraySizeBytes` or `domainArray` are NULL

CUptiResult cuptiDeviceGetAttribute (CUdevice device, **CUpti_DeviceAttribute** attrib, size_t * valueSize, void * value)

Read a device attribute and return it in *`value`.

Note:

Thread-safety: this function is thread safe.

Parameters:

device The CUDA device

attrib The attribute to read

valueSize Size of buffer pointed by the value, and returns the number of bytes written to `value`

value Returns the value of the attribute

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE

CUPTI_ERROR_INVALID_PARAMETER if `valueSize` or `value` is NULL, or if `attrib` is not a device attribute

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT For non-c-string attribute values, indicates that the `value` buffer is too small to hold the attribute value.

CUptiResult cuptiDeviceGetEventDomainAttribute (CUdevice device, **CUpti_EventDomainID** eventDomain, **CUpti_EventDomainAttribute** attrib, size_t * valueSize, void * value)

Returns an event domain attribute in *`value`. The size of the `value` buffer is given by *`valueSize`. The value returned in *`valueSize` contains the number of bytes returned in `value`.

If the attribute value is a c-string that is longer than ∗`valueSize`, then only the first ∗`valueSize` characters will be returned and there will be no terminating null byte.

Note:

> **Thread-safety**: this function is thread safe.

Parameters:

device The CUDA device

eventDomain ID of the event domain

attrib The event domain attribute to read

valueSize The size of the `value` buffer in bytes, and returns the number of bytes written to `value`

value Returns the attribute's value

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE

CUPTI_ERROR_INVALID_EVENT_DOMAIN_ID

CUPTI_ERROR_INVALID_PARAMETER if `valueSize` or `value` is NULL, or if `attrib` is not an event domain attribute

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT For non-c-string attribute values, indicates that the `value` buffer is too small to hold the attribute value.

CUptiResult cuptiDeviceGetNumEventDomains (CUdevice device, uint32_t ∗ numDomains)

Returns the number of domains in `numDomains` for a device.

Note:

> **Thread-safety**: this function is thread safe.

Parameters:

device The CUDA device

numDomains Returns the number of domains

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE

CUPTI_ERROR_INVALID_PARAMETER if `numDomains` is NULL

CUptiResult cuptiDeviceGetTimestamp (CUcontext context, uint64_t * timestamp)

Returns the device timestamp in `*timestamp`. The timestamp is reported in nanoseconds and indicates the time since the device was last reset.

Note:

Thread-safety: this function is thread safe.

Parameters:

context A context on the device from which to get the timestamp

timestamp Returns the device timestamp

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_CONTEXT

CUPTI_ERROR_INVALID_PARAMETER is `timestamp` is NULL

CUptiResult cuptiEnumEventDomains (size_t * arraySizeBytes, **CUpti_EventDomainID** * domainArray)

Returns all the event domains available on any CUDA-capable device. Event domain IDs are returned in `domainArray`. The size of the `domainArray` buffer is given by `*arraySizeBytes`. The size of the `domainArray` buffer must be at least `numDomains` * sizeof(CUpti_EventDomainID) or all domains will not be returned. The value returned in `*arraySizeBytes` contains the number of bytes returned in `domainArray`.

Note:

Thread-safety: this function is thread safe.

Parameters:

arraySizeBytes The size of `domainArray` in bytes, and returns the number of bytes written to `domainArray`

domainArray Returns all the event domains

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_INVALID_PARAMETER if `arraySizeBytes` or `domainArray` are NULL

CUptiResult cuptiEventDomainEnumEvents (**CUpti_EventDomainID** eventDomain, size_t ∗ arraySizeBytes, **CUpti_EventID** ∗ eventArray)

Returns the event IDs in `eventArray` for a domain. The size of the `eventArray` buffer is given by ∗`arraySizeBytes`. The size of the `eventArray` buffer must be at least `numdomainevents` ∗ sizeof(CUpti_EventID) or else all events will not be returned. The value returned in ∗`arraySizeBytes` contains the number of bytes returned in `eventArray`.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventDomain ID of the event domain

arraySizeBytes The size of `eventArray` in bytes, and returns the number of bytes written to `eventArray`

eventArray Returns the IDs of the events in the domain

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_EVENT_DOMAIN_ID

CUPTI_ERROR_INVALID_PARAMETER if `arraySizeBytes` or `eventArray` are NULL

CUptiResult cuptiEventDomainGetAttribute (**CUpti_EventDomainID** eventDomain, **CUpti_EventDomainAttribute** attrib, size_t ∗ valueSize, void ∗ value)

Returns an event domain attribute in ∗`value`. The size of the `value` buffer is given by ∗`valueSize`. The value returned in ∗`valueSize` contains the number of bytes returned in `value`.

If the attribute value is a c-string that is longer than ∗`valueSize`, then only the first ∗`valueSize` characters will be returned and there will be no terminating null byte.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventDomain ID of the event domain

attrib The event domain attribute to read

valueSize The size of the `value` buffer in bytes, and returns the number of bytes written to `value`

value Returns the attribute's value

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_EVENT_DOMAIN_ID

CUPTI_ERROR_INVALID_PARAMETER if `valueSize` or `value` is NULL, or if `attrib` is not an event domain attribute

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT For non-c-string attribute values, indicates that the `value` buffer is too small to hold the attribute value.

CUptiResult cuptiEventDomainGetNumEvents (**CUpti_EventDomainID** eventDomain, uint32_t * numEvents)

Returns the number of events in `numEvents` for a domain.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventDomain ID of the event domain

numEvents Returns the number of events in the domain

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_EVENT_DOMAIN_ID

CUPTI_ERROR_INVALID_PARAMETER if `numEvents` is NULL

CUptiResult cuptiEventGetAttribute (**CUpti_EventID** event, **CUpti_EventAttribute** attrib, size_t * valueSize, void * value)

Returns an event attribute in *`value`. The size of the `value` buffer is given by *`valueSize`. The value returned in *`valueSize` contains the number of bytes returned in `value`.

If the attribute value is a c-string that is longer than *`valueSize`, then only the first *`valueSize` characters will be returned and there will be no terminating null byte.

Note:

Thread-safety: this function is thread safe.

Parameters:

event ID of the event

attrib The event attribute to read

valueSize The size of the `value` buffer in bytes, and returns the number of bytes written to `value`

value Returns the attribute's value

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_EVENT_ID

CUPTI_ERROR_INVALID_PARAMETER if `valueSize` or `value` is NULL, or if `attrib` is not an event attribute

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT For non-c-string attribute values, indicates that the `value` buffer is too small to hold the attribute value.

CUptiResult cuptiEventGetIdFromName (CUdevice device, const char ∗ eventName, **CUpti_EventID** ∗ event)

Find an event by name and return the event ID in ∗`event`.

Note:

Thread-safety: this function is thread safe.

Parameters:

device The CUDA device

eventName The name of the event to find

event Returns the ID of the found event or undefined if unable to find the event

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE

CUPTI_ERROR_INVALID_EVENT_NAME if unable to find an event with name `eventName`. In this case ∗`event` is undefined

CUPTI_ERROR_INVALID_PARAMETER if `eventName` or `event` are NULL

CUptiResult cuptiEventGroupAddEvent (CUpti_EventGroup eventGroup, CUpti_EventID event)

Add an event to an event group. The event add can fail for a number of reasons:

- The event group is enabled
- The event does not belong to the same event domain as the events that are already in the event group
- Device limitations on the events that can belong to the same group
- The event group is full

Note:

Thread-safety: this function is thread safe.

Parameters:

eventGroup The event group

event The event to add to the group

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_EVENT_ID

CUPTI_ERROR_OUT_OF_MEMORY

CUPTI_ERROR_INVALID_OPERATION if `eventGroup` is enabled

CUPTI_ERROR_NOT_COMPATIBLE if `event` belongs to a different event domain than the events already in `eventGroup`, or if a device limitation prevents `event` from being collected at the same time as the events already in `eventGroup`

CUPTI_ERROR_MAX_LIMIT_REACHED if `eventGroup` is full

CUPTI_ERROR_INVALID_PARAMETER if `eventGroup` is NULL

CUptiResult cuptiEventGroupCreate (CUcontext context, CUpti_EventGroup * eventGroup, uint32_t flags)

Creates a new event group for `context` and returns the new group in *`eventGroup`.

Note:

`flags` are reserved for future use and should be set to zero.
Thread-safety: this function is thread safe.

Parameters:

context The context for the event group

eventGroup Returns the new event group

flags Reserved - must be zero

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_CONTEXT

CUPTI_ERROR_OUT_OF_MEMORY

CUPTI_ERROR_INVALID_PARAMETER if `eventGroup` is NULL

CUptiResult cuptiEventGroupDestroy (**CUpti_EventGroup** eventGroup)

Destroy an `eventGroup` and free its resources. An event group cannot be destroyed if it is enabled.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventGroup The event group to destroy

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_OPERATION if the event group is enabled

CUPTI_ERROR_INVALID_PARAMETER if eventGroup is NULL

CUptiResult cuptiEventGroupDisable (**CUpti_EventGroup** eventGroup)

Disable an event group. Disabling an event group stops collection of events contained in the group.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventGroup The event group

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_HARDWARE

CUPTI_ERROR_INVALID_PARAMETER if `eventGroup` is NULL

CUptiResult cuptiEventGroupEnable (**CUpti_EventGroup** eventGroup)

Enable an event group. Enabling an event group zeros the value of all the events in the group and then starts collection of those events.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventGroup The event group

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_HARDWARE

CUPTI_ERROR_NOT_READY if `eventGroup` does not contain any events

CUPTI_ERROR_NOT_COMPATIBLE if `eventGroup` cannot be enabled due to other already enabled event groups

CUPTI_ERROR_INVALID_PARAMETER if `eventGroup` is NULL

CUptiResult cuptiEventGroupGetAttribute (**CUpti_EventGroup** eventGroup, **CUpti_EventGroupAttribute** attrib, size_t * valueSize, void * value)

Read an event group attribute and return it in `*value`.

Note:

Thread-safety: this function is thread safe but client must guard against simultaneous destruction or modification of `eventGroup` (for example, client must guard against simultaneous calls to cuptiEventGroupDestroy, cuptiEventGroupAddEvent, etc.), and must guard against simultaneous destruction of the context in which `eventGroup` was created (for example, client must guard against simultaneous calls to cudaDeviceReset, cuCtxDestroy, etc.).

Parameters:

eventGroup The event group

attrib The attribute to read

valueSize Size of buffer pointed by the value, and returns the number of bytes written to `value`

value Returns the value of the attribute

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_PARAMETER if `valueSize` or `value` is NULL, or if `attrib` is not an eventgroup attribute

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT For non-c-string attribute values, indicates that the `value` buffer is too small to hold the attribute value.

CUptiResult cuptiEventGroupReadAllEvents (**CUpti_EventGroup** eventGroup, **CUpti_ReadEventFlags** flags, size_t * eventValueBufferSizeBytes, uint64_t * eventValueBuffer, size_t * eventIdArraySizeBytes, **CUpti_EventID** * eventIdArray, size_t * numEventIdsRead)

Read the values for all the events in an event group. The event values are returned in the `eventValueBuffer` buffer. `eventValueBufferSizeBytes` indicates the size of `eventValueBuffer`. The buffer must be at least (sizeof(uint64) * number of events in group) if CUPTI_EVENT_GROUP_ATTR_PROFILE_ALL_DOMAIN_INSTANCES is not set on the group containing the events. The buffer must be at least (sizeof(uint64) * number of domain instances * number of events in group) if CUPTI_EVENT_GROUP_ATTR_PROFILE_ALL_DOMAIN_INSTANCES is set on the group.

The data format returned in `eventValueBuffer` is:

- domain instance 0: event0 event1 ... eventN
- domain instance 1: event0 event1 ... eventN
- ...
- domain instance M: event0 event1 ... eventN

The event order in `eventValueBuffer` is returned in `eventIdArray`. The size of `eventIdArray` is specified in `eventIdArraySizeBytes`. The size should be at least (sizeof(CUpti_EventID) * number of events in group).

If any instance of any event counter overflows, the value returned for that event instance will be CUPTI_EVENT_OVERFLOW.

The only allowed value for `flags` is CUPTI_EVENT_READ_FLAG_NONE.

Reading events from a disabled event group is not allowed. After being read, an event's value is reset to zero.

Note:

> **Thread-safety**: this function is thread safe but client must guard against simultaneous destruction or modification of `eventGroup` (for example, client must guard against simultaneous calls to cuptiEventGroupDestroy, cuptiEventGroupAddEvent, etc.), and must guard against simultaneous destruction of the context in which `eventGroup` was created (for example, client must guard against simultaneous calls to cudaDeviceReset, cuCtxDestroy, etc.). If cuptiEventGroupResetAllEvents is called simultaneously with this function, then returned event values are undefined.

Parameters:

- eventGroup The event group
- flags Flags controlling the reading mode
- eventValueBufferSizeBytes The size of `eventValueBuffer` in bytes, and returns the number of bytes written to `eventValueBuffer`
- eventValueBuffer Returns the event values
- eventIdArraySizeBytes The size of `eventIdArray` in bytes, and returns the number of bytes written to `eventIdArray`
- eventIdArray Returns the IDs of the events in the same order as the values return in eventValueBuffer.
- numEventIdsRead Returns the number of event IDs returned in `eventIdArray`

Return values:

- CUPTI_SUCCESS
- CUPTI_ERROR_NOT_INITIALIZED
- CUPTI_ERROR_HARDWARE
- CUPTI_ERROR_INVALID_OPERATION if `eventGroup` is disabled
- CUPTI_ERROR_INVALID_PARAMETER if `eventGroup`, `eventValueBufferSizeBytes`, `eventValueBuffer`, `eventIdArraySizeBytes`, `eventIdArray` or `numEventIdsRead` is NULL

CUptiResult cuptiEventGroupReadEvent (**CUpti_EventGroup** eventGroup, **CUpti_ReadEventFlags** flags, **CUpti_EventID** event, size_t * eventValueBufferSizeBytes, uint64_t * eventValueBuffer)

Read the value for an event in an event group. The event value is returned in the `eventValueBuffer` buffer. `eventValueBufferSizeBytes` indicates the size of the `eventValueBuffer` buffer. The buffer must be at least sizeof(uint64) if CUPTI_EVENT_GROUP_ATTR_PROFILE_ALL_DOMAIN_INSTANCES is not set on the group containing the event. The buffer must be at least (sizeof(uint64) ∗ number of domain instances) if CUPTI_EVENT_GROUP_ATTR_PROFILE_ALL_DOMAIN_INSTANCES is set on the group.

If any instance of an event counter overflows, the value returned for that event instance will be CUPTI_EVENT_OVERFLOW.

The only allowed value for `flags` is CUPTI_EVENT_READ_FLAG_NONE.

Reading an event from a disabled event group is not allowed. After being read, an event's value is reset to zero.

Note:

Thread-safety: this function is thread safe but client must guard against simultaneous destruction or modification of `eventGroup` (for example, client must guard against simultaneous calls to cuptiEventGroupDestroy, cuptiEventGroupAddEvent, etc.), and must guard against simultaneous destruction of the context in which `eventGroup` was created (for example, client must guard against simultaneous calls to cudaDeviceReset, cuCtxDestroy, etc.). If cuptiEventGroupResetAllEvents is called simultaneously with this function, then returned event values are undefined.

Parameters:

eventGroup The event group

flags Flags controlling the reading mode

event The event to read

eventValueBufferSizeBytes The size of `eventValueBuffer` in bytes, and returns the number of bytes written to `eventValueBuffer`

eventValueBuffer Returns the event value(s)

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_EVENT_ID

CUPTI_ERROR_HARDWARE

CUPTI_ERROR_INVALID_OPERATION if `eventGroup` is disabled

CUPTI_ERROR_INVALID_PARAMETER if `eventGroup`, `eventValueBufferSizeBytes` or `eventValueBuffer` is NULL

CUptiResult cuptiEventGroupRemoveAllEvents (**CUpti_EventGroup** eventGroup)

Remove all events from an event group. Events cannot be removed if the event group is enabled.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventGroup The event group

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_OPERATION if `eventGroup` is enabled

CUPTI_ERROR_INVALID_PARAMETER if `eventGroup` is NULL

CUptiResult cuptiEventGroupRemoveEvent (**CUpti_EventGroup** eventGroup, **CUpti_EventID** event)

Remove `event` from the an event group. The event cannot be removed if the event group is enabled.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventGroup The event group

event The event to remove from the group

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_EVENT_ID

CUPTI_ERROR_INVALID_OPERATION if `eventGroup` is enabled

CUPTI_ERROR_INVALID_PARAMETER if `eventGroup` is NULL

CUptiResult cuptiEventGroupResetAllEvents (**CUpti_EventGroup** eventGroup)

Zero all the event counts in an event group.

Note:

> **Thread-safety**: this function is thread safe but client must guard against simultaneous destruction or modification of `eventGroup` (for example, client must guard against simultaneous calls to cuptiEventGroupDestroy, cuptiEventGroupAddEvent, etc.), and must guard against simultaneous destruction of the context in which `eventGroup` was created (for example, client must guard against simultaneous calls to cudaDeviceReset, cuCtxDestroy, etc.).

Parameters:

eventGroup The event group

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_HARDWARE

CUPTI_ERROR_INVALID_PARAMETER if `eventGroup` is NULL

CUptiResult cuptiEventGroupSetAttribute (**CUpti_EventGroup** eventGroup, **CUpti_EventGroupAttribute** attrib, size_t valueSize, void * value)

Write an event group attribute.

Note:

> **Thread-safety**: this function is thread safe.

Parameters:

eventGroup The event group

attrib The attribute to write

valueSize The size, in bytes, of the value

value The attribute value to write

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_PARAMETER if `valueSize` or `value` is NULL, or if `attrib` is not an event group attribute, or if `attrib` is not a writable attribute

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT Indicates that the `value` buffer is too small to hold the attribute value.

CUptiResult cuptiEventGroupSetsCreate (CUcontext context, size_t eventIdArraySizeBytes, **CUpti_EventID** ∗ eventIdArray, **CUpti_EventGroupSets** ∗∗ eventGroupPasses)

The number of events that can be collected simultaneously varies by device and by the type of the events. When events can be collected simultaneously, they may need to be grouped into multiple event groups because they are from different event domains. This function takes a set of events and determines how many passes are required to collect all those events, and which events can be collected simultaneously in each pass.

The CUpti_EventGroupSets returned in `eventGroupPasses` indicates how many passes are required to collect the events with the `numSets` field. Within each event group set, the `sets` array indicates the event groups that should be collected on each pass.

Note:

Thread-safety: this function is thread safe, but client must guard against another thread simultaneously destroying `context`.

Parameters:

context The context for event collection

eventIdArraySizeBytes Size of `eventIdArray` in bytes

eventIdArray Array of event IDs that need to be grouped

eventGroupPasses Returns a CUpti_EventGroupSets object that indicates the number of passes required to collect the events and the events to collect on each pass

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_CONTEXT

CUPTI_ERROR_INVALID_EVENT_ID

CUPTI_ERROR_INVALID_PARAMETER if `eventIdArray` or `eventGroupPasses` is NULL

CUptiResult cuptiEventGroupSetsDestroy (**CUpti_EventGroupSets** ∗ eventGroupSets)

Destroy a CUpti_EventGroupSets object.

Note:

Thread-safety: this function is thread safe.

Parameters:

eventGroupSets The object to destroy

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_OPERATION if any of the event groups contained in the sets is enabled

CUPTI_ERROR_INVALID_PARAMETER if `eventGroupSets` is NULL

CUptiResult cuptiGetNumEventDomains (uint32_t * numDomains)

Returns the total number of event domains available on any CUDA-capable device.

Note:

Thread-safety: this function is thread safe.

Parameters:

numDomains Returns the number of domains

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_INVALID_PARAMETER if `numDomains` is NULL

CUptiResult cuptiSetEventCollectionMode (CUcontext context, **CUpti_EventCollectionMode** mode)

Set the event collection mode for a `context`. The `mode` controls the event collection behavior of all events in event groups created in the `context`.

Note:

Thread-safety: this function is thread safe.

Parameters:

context The context

mode The event collection mode

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_CONTEXT

CUPTI Metric API

Data Structures

- union CUpti_MetricValue

  A metric value.

Typedefs

- typedef uint32_t CUpti_MetricID

  ID for a metric.

Enumerations

- enum CUpti_MetricAttribute {
  CUPTI_METRIC_ATTR_NAME = 0,
  CUPTI_METRIC_ATTR_SHORT_DESCRIPTION = 1,
  CUPTI_METRIC_ATTR_LONG_DESCRIPTION = 2,
  CUPTI_METRIC_ATTR_CATEGORY = 3,
  CUPTI_METRIC_ATTR_VALUE_KIND = 4,
  CUPTI_METRIC_ATTR_EVALUATION_MODE = 5 }

  Metric attributes.

- enum CUpti_MetricCategory {
  CUPTI_METRIC_CATEGORY_MEMORY = 0,
  CUPTI_METRIC_CATEGORY_INSTRUCTION = 1,
  CUPTI_METRIC_CATEGORY_MULTIPROCESSOR = 2,
  CUPTI_METRIC_CATEGORY_CACHE = 3,
  CUPTI_METRIC_CATEGORY_TEXTURE = 4 }

  A metric category.

- enum CUpti_MetricEvaluationMode {
  CUPTI_METRIC_EVALUATION_MODE_PER_INSTANCE = 1,

CUPTI_METRIC_EVALUATION_MODE_AGGREGATE = 1 << 1 }

A metric evaluation mode.

▶ enum CUpti_MetricValueKind {
CUPTI_METRIC_VALUE_KIND_DOUBLE = 0,
CUPTI_METRIC_VALUE_KIND_UINT64 = 1,
CUPTI_METRIC_VALUE_KIND_PERCENT = 2,
CUPTI_METRIC_VALUE_KIND_THROUGHPUT = 3,
CUPTI_METRIC_VALUE_KIND_INT64 = 4 }

Kinds of metric values.

Functions

▶ CUptiResult cuptiDeviceEnumMetrics (CUdevice device, size_t ∗arraySizeBytes, CUpti_MetricID ∗metricArray)

Get the metrics for a device.

▶ CUptiResult cuptiDeviceGetNumMetrics (CUdevice device, uint32_t ∗numMetrics)

Get the number of metrics for a device.

▶ CUptiResult cuptiEnumMetrics (size_t ∗arraySizeBytes, CUpti_MetricID ∗metricArray)

Get all the metrics available on any device.

▶ CUptiResult cuptiGetNumMetrics (uint32_t ∗numMetrics)

Get the total number of metrics available on any device.

▶ CUptiResult cuptiMetricCreateEventGroupSets (CUcontext context, size_t metricIdArraySizeBytes, CUpti_MetricID ∗metricIdArray, CUpti_EventGroupSets ∗∗eventGroupPasses)

For a set of metrics, get the grouping that indicates the number of passes and the event groups necessary to collect the events required for those metrics.

▶ CUptiResult cuptiMetricEnumEvents (CUpti_MetricID metric, size_t ∗eventIdArraySizeBytes, CUpti_EventID ∗eventIdArray)

Get the events required to calculating a metric.

- CUptiResult cuptiMetricGetAttribute (CUpti_MetricID metric, CUpti_MetricAttribute attrib, size_t ∗valueSize, void ∗value)

  Get a metric attribute.

- CUptiResult cuptiMetricGetIdFromName (CUdevice device, const char ∗metricName, CUpti_MetricID ∗metric)

  Find an metric by name.

- CUptiResult cuptiMetricGetNumEvents (CUpti_MetricID metric, uint32_t ∗numEvents)

  Get number of events required to calculate a metric.

- CUptiResult cuptiMetricGetValue (CUdevice device, CUpti_MetricID metric, size_t eventIdArraySizeBytes, CUpti_EventID ∗eventIdArray, size_t eventValueArraySizeBytes, uint64_t ∗eventValueArray, uint64_t timeDuration, CUpti_MetricValue ∗metricValue)

  Calculate the value for a metric.

Detailed Description

Functions, types, and enums that implement the CUPTI Metric API.

Typedef Documentation

typedef uint32_t **CUpti_MetricID**

A metric provides a measure of some aspect of the device.

Enumeration Type Documentation

enum **CUpti_MetricAttribute**

Metric attributes describe properties of a metric. These attributes can be read using cuptiMetricGetAttribute.

Enumerator:

CUPTI_METRIC_ATTR_NAME Metric name. Value is a null terminated const c-string.

CUPTI_METRIC_ATTR_SHORT_DESCRIPTION Short description of metric. Value is a null terminated const c-string.

CUPTI_METRIC_ATTR_LONG_DESCRIPTION Long description of metric. Value is a null terminated const c-string.

CUPTI_METRIC_ATTR_CATEGORY Category of the metric. Value is of type CUpti_MetricCategory.

CUPTI_METRIC_ATTR_VALUE_KIND Value type of the metric. Value is of type CUpti_MetricValueKind.

CUPTI_METRIC_ATTR_EVALUATION_MODE Metric evaluation mode. Value is of type CUpti_MetricEvaluationMode.

enum **CUpti_MetricCategory**

Each metric is assigned to a category that represents the general type of the metric. A metric's category is accessed using cuptiMetricGetAttribute and the CUPTI_METRIC_ATTR_CATEGORY attribute.

Enumerator:

CUPTI_METRIC_CATEGORY_MEMORY A memory related metric.

CUPTI_METRIC_CATEGORY_INSTRUCTION An instruction related metric.

CUPTI_METRIC_CATEGORY_MULTIPROCESSOR A multiprocessor related metric.

CUPTI_METRIC_CATEGORY_CACHE A cache related metric.

CUPTI_METRIC_CATEGORY_TEXTURE A texture related metric.

enum **CUpti_MetricEvaluationMode**

A metric can be evaluated per hardware instance to know the load balancing across instances of a domain or the metric can be evaluated in aggregate mode when the events involved in metric evaluation are from different event domains. It might be possible to evaluate some metrics in both modes for convenience. A metric's evaluation mode is accessed using CUpti_MetricEvaluationMode and the CUPTI_METRIC_ATTR_EVALUATION_MODE attribute.

Enumerator:

CUPTI_METRIC_EVALUATION_MODE_PER_INSTANCE If the metric evaluation mode is per instance, then the event value passed to cuptiMetricGetValue should contain value for an instance of the domain. Also in this mode, cuptiMetricGetValue should be called for all available instances of the domain to get overall status.

CUPTI_METRIC_EVALUATION_MODE_AGGREGATE If the metric evaluation mode is aggregate, then the event value passed to cuptiMetricGetValue should be aggregated value of an event for all instances of the domain. In this mode, cuptiMetricGetValue should be called only once.

enum **CUpti_MetricValueKind**

Metric values can be one of several different kinds. Corresponding to each kind is a member of the CUpti_MetricValue union. The metric value returned by cuptiMetricGetValue should be accessed using the appropriate member of that union based on its value kind.

Enumerator:

CUPTI_METRIC_VALUE_KIND_DOUBLE The metric value is a 64-bit double.

CUPTI_METRIC_VALUE_KIND_UINT64 The metric value is a 64-bit unsigned integer.

CUPTI_METRIC_VALUE_KIND_PERCENT The metric value is a percentage represented by a 64-bit double. For example, 57.5% is represented by the value 57.5.

CUPTI_METRIC_VALUE_KIND_THROUGHPUT The metric value is a throughput represented by a 64-bit integer. The unit for throughput values is bytes/second.

CUPTI_METRIC_VALUE_KIND_INT64 The metric value is a 64-bit signed integer.

Function Documentation

CUptiResult cuptiDeviceEnumMetrics (CUdevice device, size_t * arraySizeBytes, **CUpti_MetricID** * metricArray)

Returns the metric IDs in `metricArray` for a device. The size of the `metricArray` buffer is given by *`arraySizeBytes`. The size of the `metricArray` buffer must be at least `numMetrics` * sizeof(CUpti_MetricID) or else all metric IDs will not be returned. The value returned in *`arraySizeBytes` contains the number of bytes returned in `metricArray`.

Parameters:

device The CUDA device

arraySizeBytes The size of `metricArray` in bytes, and returns the number of bytes written to `metricArray`

metricArray Returns the IDs of the metrics for the device

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE

CUPTI_ERROR_INVALID_PARAMETER if `arraySizeBytes` or `metricArray` are NULL

CUptiResult cuptiDeviceGetNumMetrics (CUdevice device, uint32_t ∗ numMetrics)

Returns the number of metrics available for a device.

Parameters:

device The CUDA device

numMetrics Returns the number of metrics available for the device

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE

CUPTI_ERROR_INVALID_PARAMETER if `numMetrics` is NULL

CUptiResult cuptiEnumMetrics (size_t ∗ arraySizeBytes, **CUpti_MetricID** ∗ metricArray)

Returns the metric IDs in `metricArray` for all CUDA-capable devices. The size of the `metricArray` buffer is given by ∗`arraySizeBytes`. The size of the `metricArray` buffer must be at least `numMetrics` ∗ sizeof(CUpti_MetricID) or all metric IDs will not be returned. The value returned in ∗`arraySizeBytes` contains the number of bytes returned in `metricArray`.

Parameters:

arraySizeBytes The size of `metricArray` in bytes, and returns the number of bytes written to `metricArray`

metricArray Returns the IDs of the metrics

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_INVALID_PARAMETER if `arraySizeBytes` or `metricArray` are NULL

CUptiResult cuptiGetNumMetrics (uint32_t ∗ numMetrics)

Returns the total number of metrics available on any CUDA-capable devices.

Parameters:

numMetrics Returns the number of metrics

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_INVALID_PARAMETER if `numMetrics` is NULL

CUptiResult cuptiMetricCreateEventGroupSets (CUcontext context, size_t metricIdArraySizeBytes, **CUpti_MetricID** ∗ metricIdArray, **CUpti_EventGroupSets** ∗∗ eventGroupPasses)

For a set of metrics, get the grouping that indicates the number of passes and the event groups necessary to collect the events required for those metrics.

See also:

cuptiEventGroupSetsCreate for details on event group set creation.

Parameters:

context The context for event collection

metricIdArraySizeBytes Size of the metricIdArray in bytes

metricIdArray Array of metric IDs

eventGroupPasses Returns a CUpti_EventGroupSets object that indicates the number of passes required to collect the events and the events to collect on each pass

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_CONTEXT

CUPTI_ERROR_INVALID_METRIC_ID

CUPTI_ERROR_INVALID_PARAMETER if `metricIdArray` or `eventGroupPasses` is NULL

CUptiResult cuptiMetricEnumEvents (**CUpti_MetricID** metric, size_t ∗ eventIdArraySizeBytes, **CUpti_EventID** ∗ eventIdArray)

Gets the event IDs in `eventIdArray` required to calculate a `metric`. The size of the `eventIdArray` buffer is given by ∗`eventIdArraySizeBytes` and must be at least `numEvents` ∗ sizeof(CUpti_EventID) or all events will not be returned. The value returned in ∗`eventIdArraySizeBytes` contains the number of bytes returned in `eventIdArray`.

Parameters:

metric ID of the metric

eventIdArraySizeBytes The size of `eventIdArray` in bytes, and returns the number of bytes written to `eventIdArray`

eventIdArray Returns the IDs of the events required to calculate `metric`

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_METRIC_ID

CUPTI_ERROR_INVALID_PARAMETER if `eventIdArraySizeBytes` or `eventIdArray` are NULL.

CUptiResult cuptiMetricGetAttribute (**CUpti_MetricID** metric, **CUpti_MetricAttribute** attrib, size_t ∗ valueSize, void ∗ value)

Returns a metric attribute in ∗`value`. The size of the `value` buffer is given by ∗`valueSize`. The value returned in ∗`valueSize` contains the number of bytes returned in `value`.

If the attribute value is a c-string that is longer than ∗`valueSize`, then only the first ∗`valueSize` characters will be returned and there will be no terminating null byte.

Parameters:

metric ID of the metric

attrib The metric attribute to read

valueSize The size of the `value` buffer in bytes, and returns the number of bytes written to `value`

value Returns the attribute's value

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_METRIC_ID

CUPTI_ERROR_INVALID_PARAMETER if `valueSize` or `value` is NULL, or if `attrib` is not a metric attribute

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT For non-c-string attribute values, indicates that the `value` buffer is too small to hold the attribute value.

CUptiResult cuptiMetricGetIdFromName (CUdevice device, const char * metricName, **CUpti_MetricID** * metric)

Find a metric by name and return the metric ID in `*metric`.

Parameters:

device The CUDA device

metricName The name of metric to find

metric Returns the ID of the found metric or undefined if unable to find the metric

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_DEVICE

CUPTI_ERROR_INVALID_METRIC_NAME if unable to find a metric with name `metricName`. In this case `*metric` is undefined

CUPTI_ERROR_INVALID_PARAMETER if `metricName` or `metric` are NULL.

CUptiResult cuptiMetricGetNumEvents (**CUpti_MetricID** metric, uint32_t * numEvents)

Returns the number of events in `numEvents` that are required to calculate a metric.

Parameters:

metric ID of the metric

numEvents Returns the number of events required for the metric

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_METRIC_ID

CUPTI_ERROR_INVALID_PARAMETER if `numEvents` is NULL

CUptiResult cuptiMetricGetValue (CUdevice device, **CUpti_MetricID** metric, size_t eventIdArraySizeBytes, **CUpti_EventID** * eventIdArray, size_t eventValueArraySizeBytes, uint64_t * eventValueArray, uint64_t timeDuration, **CUpti_MetricValue** * metricValue)

Use the events collected for a metric to calculate the metric value. Metric value evaluation depends on the evaluation mode CUpti_MetricEvaluationMode that the metric supports.

If a metric has evaluation mode as CUPTI_METRIC_EVALUATION_MODE_PER_INSTANCE, then it assumes that the input event value is for one domain instance. If a metric has evaluation mode as CUPTI_METRIC_EVALUATION_MODE_AGGREGATE, it assumes that input event values are normalized to represent all domain instances on a device. For the most accurate metric collection, the events required for the metric should be collected for all profiled domain instances. For example, to collect all instances of an event, set the CUPTI_EVENT_GROUP_ATTR_PROFILE_ALL_DOMAIN_INSTANCES attribute on the group containing the event to 1. The normalized value for the event is then: (`sum_event_values` ∗ `totalInstanceCount`) / `instanceCount`, where `sum_event_values` is the summation of the event values across all profiled domain instances, `totalInstanceCount` is obtained from querying CUPTI_EVENT_DOMAIN_ATTR_TOTAL_INSTANCE_COUNT and `instanceCount` is obtained from querying CUPTI_EVENT_GROUP_ATTR_INSTANCE_COUNT (or CUPTI_EVENT_DOMAIN_ATTR_INSTANCE_COUNT).

Parameters:

device The CUDA device that the metric is being calculated for

metric The metric ID

eventIdArraySizeBytes The size of `eventIdArray` in bytes

eventIdArray The event IDs required to calculate `metric`

eventValueArraySizeBytes The size of `eventValueArray` in bytes

eventValueArray The normalized event values required to calculate `metric`. The values must be order to match the order of events in `eventIdArray`

timeDuration The duration over which the events were collected, in ns

metricValue Returns the value for the metric

Return values:

CUPTI_SUCCESS

CUPTI_ERROR_NOT_INITIALIZED

CUPTI_ERROR_INVALID_METRIC_ID

CUPTI_ERROR_INVALID_OPERATION

CUPTI_ERROR_PARAMETER_SIZE_NOT_SUFFICIENT if the eventIdArray does not contain all the events needed for metric

CUPTI_ERROR_INVALID_EVENT_VALUE if any of the event values required for the metric is CUPTI_EVENT_OVERFLOW

CUPTI_ERROR_NOT_COMPATIBLE if the computed metric value cannot be represented in the metric's value type. For example, if the metric value type is unsigned and the computed metric value is negative

CUPTI_ERROR_INVALID_PARAMETER if `metricValue`, `eventIdArray` or `eventValueArray` is NULL