
cudawrappers

Release 0.4.0

cudawrappers developers

Jun 23, 2023

API DOCUMENTATION

1	Requirements	3
2	Usage	5
2.1	Usage examples	5
3	Used by	7
4	Alternatives	9
4.1	cuda-api-wrappers	9
4.2	cuda-wrapper	9
4.3	CudaPlusPlus	9
5	Contributing	11
6	Developer documentation	13
7	Class Hierarchy	15
8	File Hierarchy	17
9	Full API	19
9.1	Namespaces	19
9.2	Classes and Structs	20
9.3	Functions	30
10	Indices and tables	33
	Index	35

This library is a C++ wrapper for the Nvidia C libraries (e.g. CUDA driver, nvrtc, cuFFT etc.). The main purposes are:

1. *easier resource management*, leading to *lower risk of programming errors*;
2. *better fault handling* (through exceptions);
3. *more compact user code*.

Originally, the API enforced RAII to even further reduce the risk of faulty code, but enforcing RAII and compatibility with (unmanaged) objects obtained outside this API are mutually exclusive.

REQUIREMENTS

Software	Minimum version
CUDA	10.0 or later
CMake	3.17 or later
gcc	9.3 or later
OS	Linux distro (amd64)

Hardware	Type
GPU architecture	NVIDIA PASCAL or newer

USAGE

We use CMake in this project, so you can clone and build this library with the following steps:

```
git clone https://github.com/nlesc-recruit/cudawrappers
cd cudawrappers
cmake -S . -B build
make -C build
```

This command will create a build folder, compile the code and generate the libraries `libcudawrappers-*.so` in the build directory. For more details on the building requirements and on testing, check the developer documentation.

To install to `~/local`, use

```
git clone https://github.com/nlesc-recruit/cudawrappers
cd cudawrappers
cmake -DCMAKE_INSTALL_PREFIX=$HOME/.local -S . -B build
make -C build
make -C build install
```

2.1 Usage examples

You can include the cudawrappers library in your own projects in various ways. We have created a few repositories with example setups to get you started:

1. [usage-example-git-submodules](#) Example project that uses the cudawrappers library as a dependency by using git submodules on its source tree.
2. [usage-example-locally-installed](#) Example project that uses the cudawrappers library as a dependency by having it locally installed.
3. [usage-example-cmake-pull](#) Example project that uses the cudawrappers library as a dependency by having cmake pull it in from github.
4. there are also examples available in the examples directory, they can be compiled by providing the flag `-DCUDAWRAPPERS_BUILD_EXAMPLES=ON` to the cmake command shown above.

USED BY

This section aims to provide an overview of projects that use this repo's library (or something very similar), e.g. through git submodules or by including copies of this library in their source tree:

1. <https://git.astron.nl/RD/dedisp/>
2. <https://git.astron.nl/RD/idg>
3. <https://git.astron.nl/RD/tensor-core-correlator>

ALTERNATIVES

This section provides an overview of similar tools in this space, and how they are different.

4.1 cuda-api-wrappers

url: <https://github.com/eyalroz/cuda-api-wrappers>

- Aims to provide wrappers for the CUDA runtime API
- Development has slowed a bit recently
- Has 1 or 2 main developers
- Has gained quite a bit of attention (e.g. 440 stars; 57 forks)

The project is planning to support more of the Driver API (for fine-grained control of CUDA devices) and NVRTC API (for runtime compilation of kernels); there is a release candidate ([v0.5.0-rc1](#)). It doesn't provide support for cuFFT and cuBLAS though.

4.2 cuda-wrapper

url: <https://github.com/halmd-org/cuda-wrapper>

- Aims to provide a C++ wrapper for the CUDA Driver and Runtime APIs

4.3 CudaPlusPlus

url: <https://github.com/apardyl/cudaplusplus>

- Aims to provide a C++ wrapper for the CUDA Driver API
- Project appears inactive

CONTRIBUTING

See CONTRIBUTING for a guide on how to contribute.

DEVELOPER DOCUMENTATION

See [README.dev.md](#) for documentation on setting up your development environment.

CLASS HIERARCHY

FILE HIERARCHY

9.1 Namespaces

9.1.1 Namespace cu

Contents

- *Classes*
- *Functions*

Classes

- *Class Array*
- *Class Context*
- *Class Device*
- *Class DeviceMemory*
- *Class Error*
- *Class Event*
- *Class Function*
- *Class HostMemory*
- *Class Module*
- *Class Stream*
- *Template Class Wrapper*

Functions

- *Function cu::checkCudaCall*
- *Function cu::driverGetVersion*
- *Function cu::init*
- *Function cu::memcpyHtoD*

9.1.2 Namespace nvrtc

Contents

- *Classes*
- *Functions*

Classes

- *Class Error*
- *Class Program*

Functions

- *Function nvrtc::checkNvrtcCall*

9.1.3 Namespace nvtx

Contents

- *Classes*

Classes

- *Class Marker*

9.2 Classes and Structs

9.2.1 Class Array

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- `public cu::Wrapper< CUarray >` (*Template Class Wrapper*)

Class Documentation

class **Array** : public cu::Wrapper<CUarray>

Public Functions

inline **Array**(unsigned width, CUarray_format format, unsigned numChannels)

inline **Array**(unsigned width, unsigned height, CUarray_format format, unsigned numChannels)

inline **Array**(unsigned width, unsigned height, unsigned depth, CUarray_format format, unsigned numChannels)

inline explicit **Array**(CUarray &array)

9.2.2 Class Context

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- `public cu::Wrapper< CUcontext >` (*Template Class Wrapper*)

Class Documentation

class **Context** : public cu::Wrapper<CUcontext>

Public Functions

inline **Context**(int flags, *Device* &device)

inline explicit **Context**(CUcontext context)

inline unsigned **getApiVersion**() const

inline void **setCurrent**() const

inline void **pushCurrent**()

inline void **setSharedMemConfig**(CUsharedconfig config)

Public Static Functions

```
static inline CUfunc_cache getCacheConfig()

static inline void setCacheConfig(CUfunc_cache config)

static inline Context getCurrent()

static inline Context popCurrent()

static inline Device getDevice()

static inline size_t getLimit(CUlimit limit)

template<CUlimit limit>
static inline size_t getLimit()

static inline void setLimit(CUlimit limit, size_t value)

template<CUlimit limit>
static inline void setLimit(size_t value)

static inline void synchronize()
```

9.2.3 Class Device

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUdevice > (*Template Class Wrapper*)

Class Documentation

```
class Device : public cu::Wrapper<CUdevice>
```

Public Functions

```
inline explicit Device(int ordinal)

inline int getAttribute(CUdevice_attribute attribute) const

template<CUdevice_attribute attribute>
inline int getAttribute() const

inline std::string getName() const

inline size_t totalMem() const

inline std::pair<unsigned, bool> primaryCtxGetState() const
```

```
inline void primaryCtxReset()
Context primaryCtxRetain()
inline void primaryCtxSetFlags(unsigned flags)
```

Public Static Functions

```
static inline int getCount()
```

9.2.4 Class DeviceMemory

- Defined in file_include_cudaWrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUdeviceptr > (*Template Class Wrapper*)

Class Documentation

```
class DeviceMemory : public cu::Wrapper<CUdeviceptr>
```

Public Functions

```
inline explicit DeviceMemory(size_t size)
inline explicit DeviceMemory(CUdeviceptr ptr)
inline explicit DeviceMemory(const HostMemory &hostMemory)
void zero(size_t size)
void zero(size_t size, Stream &stream)
inline const void *parameter() const
```

9.2.5 Class Error

- Defined in file_include_cudaWrappers_cu.hpp

Inheritance Relationships

Base Type

- public exception

Class Documentation

class **Error** : public exception

Public Functions

```
inline explicit Error(CUsresult result)
const char *what() const noexcept override
inline operator CUsresult() const
```

9.2.6 Class Event

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUevent > (*Template Class Wrapper*)

Class Documentation

class **Event** : public cu::Wrapper<CUevent>

Public Functions

```
inline explicit Event(unsigned int flags = CU_EVENT_DEFAULT)
inline explicit Event(CUevent &event)
inline float elapsedTime(const Event &start) const
inline void query() const
inline void record()
inline void record(Stream&)
inline void synchronize()
```

9.2.7 Class Function

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CUfunction > (*Template Class Wrapper*)

Class Documentation

class **Function** : public cu::Wrapper<CUfunction>

Public Functions

inline **Function**(const *Module* &module, const char *name)

inline explicit **Function**(CUfunction &function)

inline int **getAttribute**(CUfunction_attribute attribute)

inline void **setCacheConfig**(CUfunc_cache config)

9.2.8 Class HostMemory

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< void * > (*Template Class Wrapper*)

Class Documentation

class **HostMemory** : public cu::Wrapper<void*>

Public Functions

```
inline explicit HostMemory(size_t size, unsigned int flags = 0)

template<typename T>
inline operator  T*()
```

9.2.9 Class Module

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CModule > (*Template Class Wrapper*)

Class Documentation

```
class Module : public cu::Wrapper<CModule>
```

Public Functions

```
inline explicit Module(const char *file_name)
inline explicit Module(const void *data)
inline explicit Module(CModule &module)
inline CUdeviceptr getGlobal(const char *name) const
```

9.2.10 Class Stream

- Defined in file_include_cudawrappers_cu.hpp

Inheritance Relationships

Base Type

- public cu::Wrapper< CStream > (*Template Class Wrapper*)

Class Documentation

class **Stream** : public cu::Wrapper<CUstream>

Public Functions

```

inline explicit Stream(unsigned int flags = CU_STREAM_DEFAULT)
inline explicit Stream(CUstream stream)
inline void memcpyHtoHAsync(void *dstPtr, const void *srcPtr, size_t size)
inline void memcpyHtoDAsync(CUdeviceptr devPtr, const void *hostPtr, size_t size)
inline void memcpyDtoHAsync(void *hostPtr, CUdeviceptr devPtr, size_t size)
inline void memcpyDtoDAsync(CUdeviceptr dstPtr, CUdeviceptr srcPtr, size_t size)
inline void launchKernel(Function &function, unsigned gridX, unsigned gridY, unsigned gridZ, unsigned
    blockX, unsigned blockY, unsigned blockZ, unsigned sharedMemBytes, const
    std::vector<const void*> &parameters)
inline void query()
inline void synchronize()
inline void wait(Event &event)
inline void addCallback(CUstreamCallback callback, void *userData, unsigned int flags = 0)
inline void record(Event &event)
inline void batchMemOp(unsigned count, CUstreamBatchMemOpParams *paramArray, unsigned flags)
inline void waitValue32(CUdeviceptr addr, cuuint32_t value, unsigned flags) const
inline void writeValue32(CUdeviceptr addr, cuuint32_t value, unsigned flags)

```

9.2.11 Template Class Wrapper

- Defined in file_include_cudawrappers_cu.hpp

Class Documentation

template<typename T>

class **Wrapper**

Public Functions

```
inline operator T() const

inline operator T()

inline bool operator==(const Wrapper<T> &other)

inline bool operator!=(const Wrapper<T> &other)
```

Protected Functions

```
Wrapper() = default

inline Wrapper(const Wrapper<T> &other)

inline Wrapper(Wrapper<T> &&other)

inline explicit Wrapper(T &obj)
```

Protected Attributes

```
T _obj = {}

std::shared_ptr<T> manager
```

9.2.12 Class Error

- Defined in file_include_cudawrappers_nvrtc.hpp

Inheritance Relationships

Base Type

- public exception

Class Documentation

```
class Error : public exception
```


Public Functions

```
inline explicit Error(nvrtcResult result)

const char *what() const noexcept override

inline operator  nvrtcResult() const
```

9.2.13 Class Program

- Defined in file_include_cudawrappers_nvrtc.hpp

Class Documentation

class **Program**

Public Functions

```
inline Program(const std::string &src, const std::string &name, const std::vector<std::string> &headers =
               std::vector<std::string>(), const std::vector<std::string> &includeNames =
               std::vector<std::string>())

inline explicit Program(const std::string &filename)

inline ~Program()

inline void compile(const std::vector<std::string> &options)

inline std::string getPTX()

inline std::string getLog()
```

9.2.14 Class Marker

- Defined in file_include_cudawrappers_nvtx.hpp

Class Documentation

class **Marker**

Public Types

enum **Color**

Values:

enumerator **red**

enumerator **green**

enumerator **blue**

enumerator **yellow**

enumerator **black**

Public Functions

explicit **Marker**(const char *message, unsigned color = 0xff00ff00)

Marker(const char *message, *Marker::Color* color)

void **start**()

void **end**()

9.3 Functions

9.3.1 Function `cu::checkCudaCall`

- Defined in `file_include_cudawrappers_cu.hpp`

Function Documentation

inline void `cu::checkCudaCall`(CUresult result)

9.3.2 Function `cu::driverGetVersion`

- Defined in `file_include_cudawrappers_cu.hpp`

Function Documentation

inline int `cu::driverGetVersion()`

9.3.3 Function `cu::init`

- Defined in `file_include_cudawrappers_cu.hpp`

Function Documentation

inline void `cu::init`(unsigned flags = 0)

9.3.4 Function `cu::memcpyHtoD`

- Defined in `file_include_cudawrappers_cu.hpp`

Function Documentation

inline void `cu::memcpyHtoD`(CUdeviceptr dst, const void *src, size_t size)

9.3.5 Function `nVRTC::checkNVRTCcall`

- Defined in `file_include_cudawrappers_nVRTC.hpp`

Function Documentation

inline void `nVRTC::checkNVRTCcall`(nVRTCResult result)

INDICES AND TABLES

- `genindex`
- `search`

C

cu::Array (C++ class), 21
 cu::Array::Array (C++ function), 21
 cu::checkCudaCall (C++ function), 30
 cu::Context (C++ class), 21
 cu::Context::Context (C++ function), 21
 cu::Context::getApiVersion (C++ function), 21
 cu::Context::getCacheConfig (C++ function), 22
 cu::Context::getCurrent (C++ function), 22
 cu::Context::getDevice (C++ function), 22
 cu::Context::getLimit (C++ function), 22
 cu::Context::popCurrent (C++ function), 22
 cu::Context::pushCurrent (C++ function), 21
 cu::Context::setCacheConfig (C++ function), 22
 cu::Context::setCurrent (C++ function), 21
 cu::Context::setLimit (C++ function), 22
 cu::Context::setSharedMemConfig (C++ function), 21
 cu::Context::synchronize (C++ function), 22
 cu::Device (C++ class), 22
 cu::Device::Device (C++ function), 22
 cu::Device::getAttribute (C++ function), 22
 cu::Device::getCount (C++ function), 23
 cu::Device::getName (C++ function), 22
 cu::Device::primaryCtxGetState (C++ function), 22
 cu::Device::primaryCtxReset (C++ function), 22
 cu::Device::primaryCtxRetain (C++ function), 23
 cu::Device::primaryCtxSetFlags (C++ function), 23
 cu::Device::totalMem (C++ function), 22
 cu::DeviceMemory (C++ class), 23
 cu::DeviceMemory::DeviceMemory (C++ function), 23
 cu::DeviceMemory::parameter (C++ function), 23
 cu::DeviceMemory::zero (C++ function), 23
 cu::driverGetVersion (C++ function), 31
 cu::Error (C++ class), 24
 cu::Error::Error (C++ function), 24
 cu::Error::operator CUresult (C++ function), 24
 cu::Error::what (C++ function), 24
 cu::Event (C++ class), 24

cu::Event::elapsedTime (C++ function), 24
 cu::Event::Event (C++ function), 24
 cu::Event::query (C++ function), 24
 cu::Event::record (C++ function), 24
 cu::Event::synchronize (C++ function), 24
 cu::Function (C++ class), 25
 cu::Function::Function (C++ function), 25
 cu::Function::getAttribute (C++ function), 25
 cu::Function::setCacheConfig (C++ function), 25
 cu::HostMemory (C++ class), 25
 cu::HostMemory::HostMemory (C++ function), 26
 cu::HostMemory::operator T* (C++ function), 26
 cu::init (C++ function), 31
 cu::memcpyHtoD (C++ function), 31
 cu::Module (C++ class), 26
 cu::Module::getGlobal (C++ function), 26
 cu::Module::Module (C++ function), 26
 cu::Stream (C++ class), 27
 cu::Stream::addCallback (C++ function), 27
 cu::Stream::batchMemOp (C++ function), 27
 cu::Stream::launchKernel (C++ function), 27
 cu::Stream::memcpyDtoDAsync (C++ function), 27
 cu::Stream::memcpyDtoHAsync (C++ function), 27
 cu::Stream::memcpyHtoDAsync (C++ function), 27
 cu::Stream::memcpyHtoHAsync (C++ function), 27
 cu::Stream::query (C++ function), 27
 cu::Stream::record (C++ function), 27
 cu::Stream::Stream (C++ function), 27
 cu::Stream::synchronize (C++ function), 27
 cu::Stream::wait (C++ function), 27
 cu::Stream::waitValue32 (C++ function), 27
 cu::Stream::writeValue32 (C++ function), 27
 cu::Wrapper (C++ class), 27
 cu::Wrapper::_obj (C++ member), 28
 cu::Wrapper::manager (C++ member), 28
 cu::Wrapper::operator T (C++ function), 28
 cu::Wrapper::operator!= (C++ function), 28
 cu::Wrapper::operator== (C++ function), 28
 cu::Wrapper::Wrapper (C++ function), 28

N

nvrtc::checkNvrtcCall (C++ function), 31

nvrtc::Error (C++ *class*), 28
 nvrtc::Error::Error (C++ *function*), 29
 nvrtc::Error::operator nvrtcResult (C++ *function*), 29
 nvrtc::Error::what (C++ *function*), 29
 nvrtc::Program (C++ *class*), 29
 nvrtc::Program::~~Program (C++ *function*), 29
 nvrtc::Program::compile (C++ *function*), 29
 nvrtc::Program::getLog (C++ *function*), 29
 nvrtc::Program::getPTX (C++ *function*), 29
 nvrtc::Program::Program (C++ *function*), 29
 nvtx::Marker (C++ *class*), 29
 nvtx::Marker::Color (C++ *enum*), 30
 nvtx::Marker::Color::black (C++ *enumerator*), 30
 nvtx::Marker::Color::blue (C++ *enumerator*), 30
 nvtx::Marker::Color::green (C++ *enumerator*), 30
 nvtx::Marker::Color::red (C++ *enumerator*), 30
 nvtx::Marker::Color::yellow (C++ *enumerator*), 30
 nvtx::Marker::end (C++ *function*), 30
 nvtx::Marker::Marker (C++ *function*), 30
 nvtx::Marker::start (C++ *function*), 30