CUDA Samples

Reference Manual

TRM-06704-001_v11.3 | May 2021

Table of Contents

List of Tables

Chapter 1. Release Notes

This section describes the release notes for the CUDA Samples only. For the release notes for the whole CUDA Toolkit, please see CUDA Toolkit Release Notes.

1.1. CUDA 11.2

- FreeImage is no longer distributed with the CUDA Samples. On Windows, see the Dependencies section for more details on how to set up FreeImage. On Linux, it is recommended to install FreeImage with your distribution's package manager.

1.2. CUDA 11.1

- Added `2_Graphics/simpleVulkanMMAP`. Demonstrates Vulkan CUDA Interop via cuMemMap APIs where CUDA buffer is imported in vulkan.
- Added `7_CUDALibraries/watershedSegmentationNPP`. Demonstrates how to use the NPP watershed segmentation function.
- Added `7_CUDALibraries/batchedLabelMarkersAndLabelCompressionNPP`. Demonstrates how to use the NPP label markers generation and label compression functions based on a Union Find (UF) algorithm including both single image and batched image versions.
- Deprecated Visual Studio 2015 support for all Windows supported samples.
- Dropped Visual Studio 2012, 2013 support from all the Windows supported samples.

1.3. CUDA 11.0

- Added `0_Simple/globalToShmemAsyncCopy`. Demonstrates asynchronous copy of data from global to shared memory using cuda pipeline. Also demonstrates arrive-wait barrier for synchronization.
- Added `0_Simple/simpleAttributes`. Demonstrates the stream attributes that affect L2 locality.

- Added `0_Simple/dmmaTensorCoreGemm`. Demonstrates double precision GEMM computation using the WMMA API for double precision employing the Tensor Cores. Also makes use of asynchronous copy from global to shared memory using cuda pipeline which leads to further performance gain.
- Added `0_Simple/bf16TensorCoreGemm`. Demonstrates __nv_bfloat16 (e8m7) GEMM computation using the WMMA API for __nv_bfloat16 employing the Tensor Cores. Also makes use of asynchronous copy from global to shared memory using cuda pipeline which leads to further performance gain.
- Added `0_Simple/tf32TensorCoreGemm`. Demonstrates tf32 (e8m10) GEMM computation using the WMMA API for tf32 employing the Tensor Cores. Also makes use of asynchronous copy from global to shared memory using cuda pipeline which leads to further performance gain.
- Added `0_Simple/simpleAWBarrier`. Demonstrates the arrive wait barriers.
- Added warp aggregated atomic multi bucket increments kernel using labeled_partition cooperative groups in `6_Advanced/warpAggregatedAtomicsCG` which can be used on compute capability 7.0 and above GPU architectures.
- Added `0_Simple/binaryPartitionCG`. Demonstrates binary_partition cooperative groups creation and usage in divergent path.
- Added `6_Advanced/cudaCompressibleMemory`. Demonstrates compressible memory allocation using cuMemMap API.
- Removed `7_CUDALibraries/nvgraph_Pagerank`, `7_CUDALibraries/nvgraph_SemiRingSpMV`, `7_CUDALibraries/nvgraph_SpectralClustering`, `7_CUDALibraries/nvgraph_SSSP` as the NVGRAPH library is dropped from CUDA Toolkit 11.0.
- Added two new reduction kernels in `6_Advanced/reduction` one which demonstrates reduce_add_sync intrinstic supported on compute capability 8.0 and another which uses cooperative_groups::reduce function which does thread_block_tile level reduction introduced from CUDA 11.0
- Added windows support to `6_Advanced/c++11_cuda`.

1.4. CUDA 10.2

- Added `6_Advanced/jacobiCudaGraphs`. Demonstrates Instantiated CUDA Graph Update usage.
- Added `0_Simple/memMapIPCDrv`. Demonstrates Inter Process Communication using cuMemMap APIs with one process per GPU for computation.
- Added `0_Simple/vectorAddMMAP`. Demonstrates how cuMemMap API allows the user to specify the physical properties of their memory while retaining the contiguous nature of their access, thus not requiring a change in their program structure.
- Added `0_Simple/simpleDrvRuntime`. Demonstrates how CUDA Driver and Runtime APIs can work together to load cuda fatbinary of vector add kernel.

- Added `0_Simple/cudaNvSci`. Demonstrates CUDA-NvSciBuf/NvSciSync Interop.

1.5. CUDA 10.1 Update 2

- Added `3_Imaging/vulkanImageCUDA`. Demonstrates how to perform Vulkan Image-CUDA Interop.
- Added `7_CUDALibraries/nvJPEG_encoder`. Demonstrates encoding of jpeg images using NVJPEG Library.
- Added Windows support to `7_CUDALibraries/nvJPEG`.
- Removed DirectX SDK (June 2010 or newer) installation requirement, all the DirectX-CUDA samples now use DirectX from Windows SDK shipped with Microsoft Visual Studio 2012 or higher

1.6. CUDA 10.1 Update 1

- Added `3_Imaging/NV12toBGRandResize`. Demonstrates how to convert and resize NV12 frames to BGR planars frames using CUDA in batch.
- Added Visual Studio 2019 support to all the samples.

1.7. CUDA 10.1

- Added `0_Simple/immaTensorCoreGemm`. Demonstrates integer GEMM computation using the Warp Matrix Multiply and Accumulate (WMMA) API for integers employing the Tensor Cores.
- Added `2_Graphics/simpleD3D12`. Demonstrates Direct3D12 interoperability with CUDA.
- Added `7_CUDALibraries/nvJPEG`. Demonstrates single and batched decoding of jpeg images using NVJPEG Library.
- Added `7_CUDALibraries/conjugateGradientCudaGraphs`. Demonstrates conjugate gradient solver on GPU using CUBLAS/CUSPARSE library calls captured and called using CUDA Graph APIs.
- Updated `0_Simple/simpleIPC` to work on Windows OS as well with TCC enabled GPUs.

1.8. CUDA 10.0

- Added `1_Utilities/UnifiedMemoryPerf`. Demonstrates the performance comparision of Unified Memory and other types of memory like zero copy buffers, pageable, pagelocked memory on a single GPU.

- Added `2_Graphics/simpleVulkan`. Demonstrates the Vulkan-CUDA Interop. CUDA imports the Vulkan vertex buffer and operates on it to create sinewave, and synchronizes with Vulkan through vulkan semaphores imported by CUDA.
- Added `0_Simple/simpleCudaGraphs`. Demonstrates how to use CUDA Graphs through Graphs APIs and Stream Capture APIs.
- Removed `3_Imaging/cudaDecodeGL`, `3_Imaging/cudaDecodeD3D9` as the cuvid library is dropped from CUDA Toolkit 10.0.
- Removed `6_Advanced/cdpLUDecomposition`, `7_CUDALibraries/simpleDevLibCUBLAS` as the CUBLAS Device library is dropped from CUDA Toolkit 10.0.

1.9. CUDA 9.2

- Added `7_CUDALibraries/boundSegmentsNPP`. Demonstrates nppiLabelMarkers to generate connected region segment labels.
- Added `6_Advanced/conjugateGradientMultiDeviceCG`. Demonstrates a conjugate gradient solver on multiple GPUs using Multi Device Cooperative Groups, also uses Unified Memory optimized using prefetching and usage hints.
- Updated `0_Simple/fp16ScalarProduct` to use fp16 native operators for half2 and other fp16 features, it also compare results of using native vs intrinsics fp16 operations.

1.10. CUDA 9.0

- Added `7_CUDALibraries/nvgraph_SpectralClustering`. Demonstrates Spectral Clustering using NVGRAPH Library.
- Added `6_Advanced/warpAggregatedAtomicsCG`. Demonstrates warp aggregated atomics using Cooperative Groups.
- Added `6_Advanced/reductionMultiBlockCG`. Demonstrates single pass reduction using Multi Block Cooperative Groups.
- Added `6_Advanced/conjugateGradientMultiBlockCG`. Demonstrates a conjugate gradient solver on GPU using Multi Block Cooperative Groups.
- Added Cooperative Groups(CG) support to several samples notable ones to name are `6_Advanced/cdpQuadtree`, `6_Advanced/cdpAdvancedQuicksort`, `6_Advanced/threadFenceReduction`, `3_Imaging/dxtc`, `4_Finance/MonteCarloMultiGPU`, `0_Simple/matrixMul_nvrtc`.
- Added `0_Simple/simpleCooperativeGroups`. Illustrates basic usage of Cooperative Groups within the thread block.
- Added `0_Simple/cudaTensorCoreGemm`. Demonstrates a GEMM computation using the Warp Matrix Multiply and Accumulate (WMMA) API introduced in CUDA 9, as well as the new Tensor Cores introduced in the Volta chip family.
- Updated `0_Simple/simpleVoteIntrinsics` to use newly added *_sync equivalent of the vote intrinsics _any, _all.

- Updated `6_Advanced/shfl_scan` to use newly added *_sync equivalent of the shfl intrinsics.

1.11. CUDA 8.0

- Added `7_CUDALibraries/FilterBorderControlNPP`. Demonstrates how any border version of an NPP filtering function can be used in the most common mode (with border control enabled), can be used to duplicate the results of the equivalent non-border version of the NPP function, and can be used to enable and disable border control on various source image edges depending on what portion of the source image is being used as input.
- Added `7_CUDALibraries/cannyEdgeDetectorNPP`. Demonstrates the recommended parameters to use with the nppiFilterCannyBorder_8u_C1R Canny Edge Detection image filter function. This function expects a single channel 8-bit grayscale input image. You can generate a grayscale image from a color image by first calling nppiColorToGray() or nppiRGBToGray(). The Canny Edge Detection function combines and improves on the techniques required to produce an edge detection image using multiple steps.
- Added `7_CUDALibraries/cuSolverSp_LowlevelCholesky`. Demonstrates Cholesky factorization using cuSolverSP's low level APIs.
- Added `7_CUDALibraries/cuSolverSp_LowlevelQR`. Demonstrates QR factorization using cuSolverSP's low level APIs.
- Added `7_CUDALibraries/BiCGStab`. Demonstrates Bi-Conjugate Gradient Stabilized (BiCGStab) iterative method for nonsymmetric and symmetric positive definite linear systems using CUSPARSE and CUBLAS
- Added `7_CUDALibraries/nvgraph_Pagerank`. Demonstrates Page Rank computation using nvGRAPH Library.
- Added `7_CUDALibraries/nvgraph_SemiRingSpMV`. Demonstrates Semi-Ring SpMV using nvGRAPH Library.
- Added `7_CUDALibraries/nvgraph_SSSP`. Demonstrates Single Source Shortest Path(SSSP) computation using nvGRAPH Library.
- Added `7_CUDALibraries/simpleCUBLASXT`. Demonstrates simple example to use CUBLAS-XT library.
- Added `6_Advanced/c++11_cuda`. Demonstrates C++11 feature support in CUDA.
- Added `1_Utilities/topologyQuery`. Demonstrates how to query the topology of a system with multiple GPU.
- Added `0_Simple/fp16ScalarProduct`. Demonstrates scalar product calculation of two vectors of FP16 numbers.
- Added `0_Simple/systemWideAtomics`. Demonstrates system wide atomic instructions on migratable memory.
- Removed `0_Simple/template_runtime`. Its purpose is served by `0_Simple/template`.

1.12. CUDA 7.5

- Added `7_CUDALibraries/cuSolverDn_LinearSolver`. Demonstrates how to use the CUSOLVER library for performing dense matrix factorization using cuSolverDN's LU, QR and Cholesky factorization functions.
- Added `7_CUDALibraries/cuSolverRf`. Demonstrates how to use cuSolverRF, a sparse re-factorization package of the CUSOLVER library.
- Added `7_CUDALibraries/cuSolverSp_LinearSolver`. Demonstrates how to use cuSolverSP which provides sparse set of routines for sparse matrix factorization.
- The `2_Graphics/simpleD3D9`, `2_Graphics/simpleD3D9Texture`, `3_Imaging/cudaDecodeD3D9`, and `5_Simulations/fluidsD3D9` samples have been modified to use the Direct3D 9Ex API instead of the Direct3D 9 API.
- The `7_CUDALibraries/grabcutNPP` and `7_CUDALibraries/imageSegmentationNPP` samples have been removed. These samples used the NPP graphcut APIs, which have been deprecated in CUDA 7.5.

1.13. CUDA 7.0

- Removed support for Windows 32-bit builds.
- The Makefile x86_64=1 and ARMv7=1 options have been deprecated. Please use TARGET_ARCH to set the targeted build architecture instead.
- The Makefile GCC option has been deprecated. Please use HOST_COMPILER to set the host compiler instead.
- The CUDA Samples are no longer shipped as prebuilt binaries on Windows. Please use VS Solution files provided to build respective executable.
- Added `0_Simple/clock_nvrtc`. Demonstrates how to compile clock function kernel at runtime using libNVRTC to measure the performance of kernel accurately.
- Added `0_Simple/inlinePTX_nvrtc`. Demonstrates compilation of CUDA kernel having PTX embedded at runtime using libNVRTC.
- Added `0_Simple/matrixMul_nvrtc`. Demonstrates compilation of matrix multiplication CUDA kernel at runtime using libNVRTC.
- Added `0_Simple/simpleAssert_nvrtc`. Demonstrates compilation of CUDA kernel having assert() at runtime using libNVRTC.
- Added `0_Simple/simpleAtomicIntrinsics_nvrtc`. Demonstrates compilation of CUDA kernel performing atomic operations at runtime using libNVRTC.
- Added `0_Simple/simpleTemplates_nvrtc`. Demonstrates compilation of templatized dynamically allocated shared memory arrays CUDA kernel at runtime using libNVRTC.
- Added `0_Simple/simpleVoteIntrinsics_nvrtc`. Demonstrates compilation of CUDA kernel which uses vote intrinsics at runtime using libNVRTC.
- Added `0_Simple/vectorAdd_nvrtc`. Demonstrates compilation of CUDA kernel performing vector addition at runtime using libNVRTC.

- Added `4_Finance/binomialOptions_nvrtc`. Demonstrates runtime compilation using libNVRTC of CUDA kernel which evaluates fair call price for a given set of European options under binomial model.
- Added `4_Finance/BlackScholes_nvrtc`. Demonstrates runtime compilation using libNVRTC of CUDA kernel which evaluates fair call and put prices for a given set of European options by Black-Scholes formula.
- Added `4_Finance/quasirandomGenerator_nvrtc`. Demonstrates runtime compilation using libNVRTC of CUDA kernel which implements Niederreiter Quasirandom Sequence Generator and Inverse Cumulative Normal Distribution functions for the generation of Standard Normal Distributions.

1.14. CUDA 6.5

- Added `7_CUDALibraries/cuHook`. Demonstrates how to build and use an intercept library with CUDA.
- Added `7_CUDALibraries/simpleCUFFT_callback`. Demonstrates how to compute a 1D-convolution of a signal with a filter using a user-supplied CUFFT callback routine, rather than a separate kernel call.
- Added `7_CUDALibraries/simpleCUFFT_MGPU`. Demonstrates how to compute a 1D-convolution of a signal with a filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain on Multiple GPUs.
- Added `7_CUDALibraries/simpleCUFFT_2d_MGPU`. Demonstrates how to compute a 2D-convolution of a signal with a filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain on Multiple GPUs.
- Removed `3_Imaging/cudaEncode`. Support for the CUDA Video Encoder (NVCUVENC) has been removed.
- Removed `4_Finance/ExcelCUDA2007`. The topic will be covered in a blog post at Parallel Forall.
- Removed `4_Finance/ExcelCUDA2010`. The topic will be covered in a blog post at Parallel Forall.
- The `4_Finance/binomialOptions` sample is now restricted to running on GPUs with SM architecture 2.0 or greater.
- The `4_Finance/quasirandomGenerator` sample is now restricted to running on GPUs with SM architecture 2.0 or greater.
- The `7_CUDALibraries/boxFilterNPP` sample now demonstrates how to use the static NPP libraries on Linux and Mac.
- The `7_CUDALibraries/conjugateGradient` sample now demonstrates how to use the static CUBLAS and CUSPARSE libraries on Linux and Mac.
- The `7_CUDALibraries/MersenneTwisterGP11213` sample now demonstrates how to use the static CURAND library on Linux and Mac.

1.15. CUDA 6.0

- New featured samples that support a new CUDA 6.0 feature called UVM-Lite
- Added `0_Simple/UnifiedMemoryStreams` - new CUDA sample that demonstrates the use of OpenMP and CUDA streams with Unified Memory on a single GPU.
- Added `1_Utilities/p2pBandwidthTestLatency` - new CUDA sample that demonstrates how measure latency between pairs of GPUs with P2P enabled and P2P disabled.
- Added `6_Advanced/StreamPriorities` - This sample demonstrates basic use of the new CUDA 6.0 feature stream priorities.
- Added `7_CUDALibraries/ConjugateGradientUM` - This sample implements a conjugate gradient solver on GPU using cuBLAS and cuSPARSE library, using Unified Memory.

1.16. CUDA 5.5

- Linux makefiles have been updated to generate code for the AMRv7 architecture. Only the ARM hard-float floating point ABI is supported. Both native ARMv7 compilation and cross compilation from x86 is supported
- Performance improvements in CUDA toolkit for Kepler GPUs (SM 3.0 and SM 3.5)
- Makefiles projects have been updated to properly find search default paths for OpenGL, CUDA, MPI, and OpenMP libraries for all OS Platforms (Mac, Linux x86, Linux ARM).
- Linux and Mac project Makefiles now invoke NVCC for building and linking projects.
- Added `0_Simple/cppOverload` - new CUDA sample that demonstrates how to use C++ overloading with CUDA.
- Added `6_Advanced/cdpBezierTessellation` - new CUDA sample that demonstrates an advanced method of implementing Bezier Line Tessellation using CUDA Dynamic Parallelism. Requires compute capability 3.5 or higher.
- Added `7_CUDALibrariess/jpegNPP` - new CUDA sample that demonstrates how to use NPP for JPEG compression on the GPU.
- CUDA Samples now have better integration with Nsight Eclipse IDE.
- `6_Advanced/ptxjit` sample now includes a new API to demonstrate PTX linking at the driver level.

1.17. CUDA 5.0

- New directory structure for CUDA samples. Samples are classified accordingly to categories: `0_Simple`, `1_Utilities`, `2_Graphics`, `3_Imaging`, `4_Finance`, `5_Simulations`, `6_Advanced`, and `7_CUDALibraries`
- Added `0_Simple/simpleIPC` - CUDA Runtime API sample is a very basic sample that demonstrates Inter Process Communication with one process per GPU for computation. Requires Compute Capability 2.0 or higher and a Linux Operating System.

- Added `0_Simple/simpleSeparateCompilation` - demonstrates a CUDA 5.0 feature, the ability to create a GPU device static library and use it within another CUDA kernel. This example demonstrates how to pass in a GPU device function (from the GPU device static library) as a function pointer to be called. Requires Compute Capability 2.0 or higher.
- Added `2_Graphics/bindlessTexture` - demonstrates use of `cudaSurfaceObject`, `cudaTextureObject`, and MipMap support in CUDA. Requires Compute Capability 3.0 or higher.
- Added `3_Imaging/stereoDisparity` - demonstrates how to compute a stereo disparity map using SIMD SAD (Sum of Absolute Difference) intrinsics. Requires Compute Capability 2.0 or higher.
- Added `0_Simple/cdpSimpleQuicksort` - demonstrates a simple quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added `0_Simple/cdpSimplePrint` - demonstrates simple printf implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added `6_Advanced/cdpLUDecomposition` - demonstrates LU Decomposition implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added `6_Advanced/cdpAdvancedQuicksort` - demonstrates an advanced quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added `6_Advanced/cdpQuadtree` - demonstrates Quad Trees implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.
- Added `7_CUDALibraries/simpleDevLibCUBLAS` - implements a simple cuBLAS function calls that call GPU device API library running cuBLAS functions. cuBLAS device code functions take advantage of CUDA Dynamic Parallelism and requires compute capability of 3.5 or higher.

1.18. CUDA 4.2

- Added `segmentationTreeThrust` - demonstrates a method to build image segmentation trees using Thrust. This algorithm is based on Boruvka's MST algorithm.

1.19. CUDA 4.1

- Added `MersenneTwisterGP11213` - implements Mersenne Twister GP11213, a pseudorandom number generator using the `cuRAND` library.
- Added `HSOpticalFlow` - When working with image sequences or video it's often useful to have information about objects movement. Optical flow describes apparent motion of objects in image sequence. This sample is a Horn-Schunck method for optical flow written using CUDA.
- Added `volumeFiltering` - demonstrates basic volume rendering and filtering using 3D textures.

- Added `simpleCubeMapTexture` - demonstrates how to use `texcubemap` fetch instruction in a CUDA C program.
- Added `simpleAssert` - demonstrates how to use GPU assert in a CUDA C program.
- Added `grabcutNPP` - CUDA implementation of Rother et al. GrabCut approach using the 8 neighborhood `NPP` Graphcut primitive introduced in CUDA 4.1. (C. Rother, V. Kolmogorov, A. Blake. *GrabCut: Interactive Foreground Extraction Using Iterated Graph Cuts. ACM Transactions on Graphics (SIGGRAPH'04)*, 2004).

Chapter 2. Getting Started

The CUDA Samples are an educational resource provided to teach CUDA programming concepts. The CUDA Samples are not meant to be used for performance measurements.

For system requirements and installation instructions, please refer to the Linux Installation Guide and the Windows Installation Guide.

2.1. Getting CUDA Samples

Windows

On Windows, the CUDA Samples are installed using the CUDA Toolkit Windows Installer. By default, the CUDA Samples are installed in:

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v11.3\
```

The installation location can be changed at installation time.

Linux

On Linux, to install the CUDA Samples, the CUDA toolkit must first be installed. See the Linux Installation Guide for more information on how to install the CUDA Toolkit.

Then the CUDA Samples can be installed by running the following command, where <target_path> is the location where to install the samples:

```
$ cuda-install-samples-11.3.sh <target_path>
```

2.2. Building Samples

Windows

The Windows samples are built using the Visual Studio IDE. Solution files (.sln) are provided for each supported version of Visual Studio, using the format:

```
*_vs<version>.sln - for Visual Studio <version>
```

Complete samples solution files exist at:

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v11.3\
```

Each individual sample has its own set of solution files at:

```
C:\ProgramData\NVIDIA Corporation\CUDA Samples\v11.3\<sample_dir>\
```

To build/examine all the samples at once, the complete solution files should be used. To build/examine a single sample, the individual sample solution files should be used.

Linux

The Linux samples are built using makefiles. To use the makefiles, change the current directory to the sample directory you wish to build, and run `make`:

```
$ cd <sample_dir>
$ make
```

The samples makefiles can take advantage of certain options:

- **TARGET_ARCH=<arch>** - cross-compile targeting a specific architecture. Allowed architectures are x86_64, armv7l, aarch64, sbsa, and ppc64le.

  By default, TARGET_ARCH is set to HOST_ARCH. On a x86_64 machine, not setting TARGET_ARCH is the equivalent of setting TARGET_ARCH=x86_64.

  ```
  $ make TARGET_ARCH=x86_64
  $ make TARGET_ARCH=armv7l
  $ make TARGET_ARCH=aarch64
  $ make TARGET_ARCH=sbsa
  $ make TARGET_ARCH=ppc64le
  ```

  See here for more details.
- **dbg=1** - build with debug symbols

  ```
  $ make dbg=1
  ```
- **SMS="A B ..."** - override the SM architectures for which the sample will be built, where `"A B ..."` is a space-delimited list of SM architectures. For example, to generate SASS for SM 35 and SM 50, use `SMS="35 50"`.

  ```
  $ make SMS="35 50"
  ```
- **HOST_COMPILER=<host_compiler>** - override the default g++ host compiler. See the Linux Installation Guide for a list of supported host compilers.

  ```
  $ make HOST_COMPILER=g++
  ```

2.3. CUDA Cross-Platform Samples

This section describes the options used to build cross-platform samples. `TARGET_ARCH=<arch>` and `TARGET_OS=<os>` should be chosen based on the supported targets shown below. `TARGET_FS=<path>` can be used to point nvcc to libraries and headers used by the sample.

Table 1. Supported Target Arch/OS Combinations

		TARGET OS		
		linux	android	qnx
TARGET ARCH	x86_64	YES	NO	NO
	aarch64	YES	YES	YES
	sbsa	YES	NO	NO

TARGET_ARCH

The target architecture must be specified when cross-compiling applications. If not specified, it defaults to the host architecture. Allowed architectures are:

- `x86_64` - 64-bit x86 CPU architecture
- `aarch64` - 64-bit ARM CPU architecture, like that found on Jetson TX1 onwards
- `sbsa` - 64-bit ARM Server CPU architecture

TARGET_OS

The target OS must be specified when cross-compiling applications. If not specified, it defaults to the host OS. Allowed OSes are:

- `linux` - for any Linux distributions
- `android` - for any supported device running Android
- `qnx` - for any supported device running QNX

TARGET_FS

The most reliable method to cross-compile the CUDA Samples is to use the TARGET_FS variable. To do so, mount the target's filesystem on the host, say at `/mnt/target`. This is typically done using `exportfs`. In cases where `exportfs` is unavailable, it is sufficient to copy the target's filesystem to `/mnt/target`. To cross-compile a sample, execute:

```
$ make TARGET_ARCH=<arch> TARGET_OS=<os> TARGET_FS=/mnt/target
```

Cross Compiling to Embedded ARM architectures

While cross compiling the samples from x86_64 installation to embedded ARM architectures, that is, `aarch64` or `armv7l`, if you intend to run the executable on tegra GPU then `SMS` variable need to override SM architectures to the tegra GPU through `SMS=<TEGRA_GPU_SM_ARCH>`, where `<TEGRA_GPU_SM_ARCH>` is the SM architecture of tegra GPU on which you want the generated binary to run on. For instance it can be `SMS="53 62 72"`. Note you can also add SM arch of discrete GPU to this list <TEGRA_GPU_SM_ARCH> if you intend to run on embedded board having discrete GPU as well. To cross compile a sample, execute:

```
$ make TARGET_ARCH=<arch> TARGET_OS=<os> SMS=<TEGRA_GPU_SM_ARCHS> TARGET_FS=/mnt/
target
```

Copying Libraries

If the TARGET_FS option is not available, the libraries used should be copied from the target system to the host system, say at `/opt/target/libs`. If the sample uses GL, the GL headers must also be copied, say at `/opt/target/include`. The linker must then be told where the libraries are with the `-rpath-link` and/or `-L` options. To ignore unresolved symbols from some libraries, use the `--unresolved-symbols` option as shown below. `SAMPLE_ENABLED` should be used to force the sample to build. For example, to cross-compile a sample which uses such libraries, execute:

```
$ make TARGET_ARCH=<arch> TARGET_OS=<os> \
           EXTRA_LDFLAGS="-rpath-link=/opt/target/libs -L/opt/target/libs --
unresolved-symbols=ignore-in-shared-libs" \
           EXTRA_CCFLAGS="-I /opt/target/include" \
           SAMPLE_ENABLED=1
```

2.4. Using CUDA Samples to Create Your Own CUDA Projects

2.4.1. Creating CUDA Projects for Windows

Creating a new CUDA Program using the CUDA Samples infrastructure is easy. We have provided a template project that you can copy and modify to suit your needs. Just follow these steps:

(<category> refers to one of the following folders: 0_Simple, 1_Utilities, 2_Graphics, 3_Imaging, 4_Finance, 5_Simulations, 6_Advanced, 7_CUDALibraries.)

1. Copy the content of:
   ```
   C:\ProgramData\NVIDIA Corporation\CUDA Samples\v11.3\<category>\template
   ```
   to a directory of your own:
   ```
   C:\ProgramData\NVIDIA Corporation\CUDA Samples\v11.3\<category>\myproject
   ```
2. Edit the filenames of the project to suit your needs.
3. Edit the *.sln, *.vcproj and source files.
   Just search and replace all occurrences of template with myproject.
4. Build the 64-bit, release or debug configurations using:
   ```
   myproject_vs<version>.sln
   ```
5. Run myproject.exe from the release or debug directories located in:
   ```
   C:\ProgramData\NVIDIA Corporation\CUDA Samples\v11.3\bin\win64\[release|debug]
   ```
6. Now modify the code to perform the computation you require.
   See the *CUDA Programming Guide* for details of programming in CUDA.

2.4.2. Creating CUDA Projects for Linux

> **Note:** The default installation folder <SAMPLES_INSTALL_PATH> is NVIDIA_CUDA_11.3_Samples and <category> is one of the following: 0_Simple, 1_Utilities, 2_Graphics, 3_Imaging, 4_Finance, 5_Simulations, 6_Advanced, 7_CUDALibraries.

Creating a new CUDA Program using the NVIDIA CUDA Samples infrastructure is easy. We have provided a template project that you can copy and modify to suit your needs. Just follow these steps:

1. Copy the template project:
   ```
   cd <SAMPLES_INSTALL_PATH>/<category>
   cp -r template <myproject>
   ```

```
cd <SAMPLES_INSTALL_PATH>/<category>
```

2. Edit the filenames of the project to suit your needs:

   ```
   mv template.cu myproject.cu
   mv template_cpu.cpp myproject_cpu.cpp
   ```

3. Edit the `Makefile` and source files.

   Just search and replace all occurrences of `template` with `myproject`.

4. Build the project as (release):

   ```
   make
   ```

   To build the project as (debug), use "make dbg=1":

   ```
   make dbg=1
   ```

5. Run the program:

   ```
   ../../bin/x86_64/linux/release/myproject
   ```

6. Now modify the code to perform the computation you require.

   See the *CUDA Programming Guide* for details of programming in CUDA.

Chapter 3. Samples Reference

This document contains a complete listing of the code samples that are included with the NVIDIA CUDA Toolkit. It describes each code sample, lists the minimum GPU specification, and provides links to the source code and white papers if available.

The code samples are divided into the following categories:

Simple Reference
: Basic CUDA samples for beginners that illustrate key concepts with using CUDA and CUDA runtime APIs.

Utilities Reference
: Utility samples that demonstrate how to query device capabilities and measure GPU/CPU bandwidth.

Graphics Reference
: Graphical samples that demonstrate interoperability between CUDA and OpenGL or DirectX.

Imaging Reference
: Samples that demonstrate image processing, compression, and data analysis.

Finance Reference
: Samples that demonstrate parallel algorithms for financial computing.

Simulations Reference
: Samples that illustrate a number of simulation algorithms implemented with CUDA.

Advanced Reference
: Samples that illustrate advanced algorithms implemented with CUDA.

Cudalibraries Reference
: Samples that illustrate how to use CUDA platform libraries (NPP, NVJPEG, NVGRAPH cuBLAS, cuFFT, cuSPARSE, cuSOLVER and cuRAND).

3.1. Simple Reference

asyncAPI

This sample uses CUDA streams and events to overlap execution on CPU and GPU.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events
Supported OSes	Linux, Windows

bf16TensorCoreGemm - bfloat16 Tensor Core GEMM

A CUDA sample demonstrating __nv_bfloat16 (e8m7) GEMM computation using the Warp Matrix Multiply and Accumulate (WMMA) API introduced with CUDA 11 in Ampere chip family tensor cores for faster matrix operations. This sample also uses async copy provided by cuda pipeline interface for gmem to shmem async loads which improves kernel performance and reduces register presssure.

Supported SM Architecture	SM 8.0, SM 8.6
CUDA API	cudaMallocManaged, cudaDeviceSynchronize, cudaFuncSetAttribute, cudaEventCreate, cudaEventRecord, cudaEventSynchronize, cudaEventElapsedTime, cudaFree
Key Concepts	Matrix Multiply, WMMA, Tensor Cores
Supported OSes	Linux, Windows

binaryPartitionCG - Binary Partition Cooperative Groups

This sample is a simple code that illustrates binary partition cooperative groups and reduce within the thread block.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Cooperative Groups
Supported OSes	Linux, Windows

cdpSimplePrint - Simple Print (CUDA Dynamic Parallelism)

This sample demonstrates simple printf implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux, Windows

cdpSimpleQuicksort - Simple Quicksort (CUDA Dynamic Parallelism)

This sample demonstrates simple quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux, Windows

clock - Clock

This example shows how to use the clock function to measure the performance of block of threads of a kernel accurately.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows

clock_nvrtc - Clock libNVRTC

This example shows how to use the clock function using libNVRTC to measure the performance of block of threads of a kernel accurately.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuMemAlloc, cuLaunchKernel, cuMemcpyHtoD, cuMemFree
Key Concepts	Performance Strategies, Runtime Compilation
Supported OSes	Linux, Windows

cppIntegration - C++ Integration

This example demonstrates how to integrate CUDA into an existing C++ application, i.e. the CUDA entry point on host side is only a function which is called from C++ code and only the file containing this function is compiled with nvcc. It also demonstrates that vector types can be used from cpp.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Supported OSes	Linux, Windows

cppOverload

This sample demonstrates how to use C++ function overloading on the GPU.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaFuncSetCacheConfig, cudaFuncGetAttributes
Key Concepts	C++ Function Overloading, CUDA Streams and Events
Supported OSes	Linux, Windows

cudaNvSci - CUDA NvSciBuf/NvSciSync Interop

This sample demonstrates CUDA-NvSciBuf/NvSciSync Interop. Two CPU threads import the NvSciBuf and NvSciSync into CUDA to perform two image processing algorithms on a ppm image - image rotation in 1st thread & rgba to grayscale conversion of rotated image in 2nd thread.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVSCI
Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaImportExternalMemory, cudaExternalMemoryGetMappedBuffer, cudaExternalMemoryGetMappedMipmappedArray, cudaImportExternalSemaphore, cudaSignalExternalSemaphoresAsync, cudaWaitExternalSemaphoresAsync, cudaDestroyExternalSemaphore, cudaDestroyExternalMemory
Key Concepts	CUDA NvSci Interop, Data Parallel Algorithms, Image Processing
Supported OSes	Linux

cudaOpenMP

This sample demonstrates how to use OpenMP API to write an application for multiple GPUs.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	OpenMP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Systems Integration, OpenMP, Multithreading
Supported OSes	Linux, Windows

cudaTensorCoreGemm - CUDA Tensor Core GEMM

CUDA sample demonstrating a GEMM computation using the Warp Matrix Multiply and Accumulate (WMMA) API introduced in CUDA 9. This sample demonstrates the use of the new CUDA WMMA API employing the Tensor Cores introcuced in the Volta chip family for faster matrix operations. In addition to that, it demonstrates the use of the new CUDA function attribute cudaFuncAttributeMaxDynamicSharedMemorySize that allows the application to reserve an extended amount of shared memory than it is available by default.

Supported SM Architecture	SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallocManaged, cudaDeviceSynchronize, cudaFuncSetAttribute, cudaEventCreate, cudaEventRecord, cudaEventSynchronize, cudaEventElapsedTime, cudaFree
Key Concepts	Matrix Multiply, WMMA, Tensor Cores
Supported OSes	Linux, Windows

dmmaTensorCoreGemm - Double Precision Tensor Core GEMM

CUDA sample demonstrates double precision GEMM computation using the Double precision Warp Matrix Multiply and Accumulate (WMMA) API introduced with CUDA 11 in Ampere chip family tensor cores for faster matrix operations. This sample also uses async copy provided by cuda pipeline interface for gmem to shmem async loads which improves kernel performance and reduces register presssure.

Supported SM Architecture	SM 8.0, SM 8.6
CUDA API	cudaMallocManaged, cudaDeviceSynchronize, cudaFuncSetAttribute, cudaEventCreate, cudaEventRecord, cudaEventSynchronize, cudaEventElapsedTime, cudaFree
Key Concepts	Matrix Multiply, WMMA, Tensor Cores
Supported OSes	Linux, Windows

fp16ScalarProduct - FP16 Scalar Product

Calculates scalar product of two vectors of FP16 numbers.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FP16
Supported SM Architecture	SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaMallocHost, cudaMemcpy, cudaFree, cudaFreeHost
Key Concepts	CUDA Runtime API
Supported OSes	Linux, Windows

globalToShmemAsyncCopy - Global Memory to Shared Memory Async Copy

This sample implements matrix multiplication which uses asynchronous copy of data from global to shared memory when on compute capability 8.0 or higher. Also demonstrates arrive-wait barrier for synchronization.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CPP11
Supported SM Architecture	SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaEventSynchronize, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Runtime API, Linear Algebra, CPP11 CUDA
Supported OSes	Linux, Windows

immaTensorCoreGemm - Tensor Core GEMM Integer MMA

CUDA sample demonstrating a integer GEMM computation using the Warp Matrix Multiply and Accumulate (WMMA) API for integer introduced in CUDA 10. This sample demonstrates the use of the CUDA WMMA API employing the Tensor Cores introduced in the Volta chip family for faster matrix operations. In addition to that, it demonstrates the use of the new CUDA function attribute cudaFuncAttributeMaxDynamicSharedMemorySize that allows the application to reserve an extended amount of shared memory than it is available by default.

Supported SM Architecture	SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallocManaged, cudaDeviceSynchronize, cudaFuncSetAttribute, cudaEventCreate, cudaEventRecord, cudaEventSynchronize, cudaEventElapsedTime, cudaFree
Key Concepts	Matrix Multiply, WMMA, Tensor Cores
Supported OSes	Linux, Windows

inlinePTX - Using Inline PTX

A simple test application that demonstrates a new CUDA 4.0 ability to embed PTX in a CUDA kernel.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaMallocHost, cudaFree, cudaFreeHost, cudaMemcpy
Key Concepts	Performance Strategies, PTX Assembly, CUDA Driver API
Supported OSes	Linux, Windows

inlinePTX_nvrtc - Using Inline PTX with libNVRTC

A simple test application that demonstrates a new CUDA 4.0 ability to embed PTX in a CUDA kernel.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuMemAlloc, cuLaunchKernel, cuMemcpyDtoH
Key Concepts	Performance Strategies, PTX Assembly, CUDA Driver API, Runtime Compilation
Supported OSes	Linux, Windows

matrixMul - Matrix Multiplication (CUDA Runtime API Version)

This sample implements matrix multiplication which makes use of shared memory to ensure data reuse, the matrix multiplication is done using tiling approach. It has been written for clarity of exposition to illustrate various CUDA programming principles, not with the goal of providing the most performant generic kernel for matrix multiplication.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaEventSynchronize, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Runtime API, Linear Algebra
Supported OSes	Linux, Windows

matrixMul_nvrtc - Matrix Multiplication with libNVRTC

This sample implements matrix multiplication and is exactly the same as Chapter 6 of the programming guide. It has been written for clarity of exposition to illustrate various CUDA programming principles, not with the goal of providing the most performant generic kernel for matrix multiplication. To illustrate GPU performance for matrix multiply, this sample also shows how to use the new CUDA 4.0 interface for CUBLAS to demonstrate high-performance performance for matrix multiplication.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Runtime API, Linear Algebra, Runtime Compilation
Supported OSes	Linux, Windows

matrixMulCUBLAS - Matrix Multiplication (CUBLAS)

This sample implements matrix multiplication from Chapter 3 of the programming guide. To illustrate GPU performance for matrix multiply, this sample also shows how to use the new CUDA 4.0 interface for CUBLAS to demonstrate high-performance performance for matrix multiplication.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMalloc, cudaFree, cudaMemcpy, cublasCreate, cublasSgemm
Key Concepts	CUDA Runtime API, Performance Strategies, Linear Algebra, CUBLAS
Supported OSes	Linux, Windows

matrixMulDrv - Matrix Multiplication (CUDA Driver API Version)

This sample implements matrix multiplication and uses the new CUDA 4.0 kernel launch Driver API. It has been written for clarity of exposition to illustrate various CUDA programming principles, not with the goal of providing the most performant generic kernel for matrix multiplication. CUBLAS provides high-performance matrix multiplication.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Driver API, Matrix Multiply
Supported OSes	Linux, Windows

memMapIPCDrv - Memmap IPC Driver API

This CUDA Driver API sample is a very basic sample that demonstrates Inter Process Communication using cuMemMap APIs with one process per GPU for computation. Requires Compute Capability 3.0 or higher and a Linux Operating System, or a Windows Operating System

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	IPC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuLaunchKernel, cuMemcpyDtoHAsync, cuDeviceGetAttribute, cuDeviceCanAccessPeer, cuStreamCreate, cuStreamSynchronize, cuCtxCreate, cuCtxDestroy, cuStreamDestroy, cuOccupancyMaxActiveBlocksPerMultiprocessor, cuMemGetAllocationGranularity, cuMemAddressReserve, cuMemCreate, cuMemRelease, cuCtxSetCurrent, cuMemExportToShareableHandle, cuMemImportFromShareableHandle, cuMemMap, cuMemSetAccess, cuMemUnmap, cuMemAddressFree
Key Concepts	CUDA Driver API, cuMemMap IPC, MMAP
Supported OSes	Linux, Windows

simpleAssert

This CUDA Runtime API sample is a very basic sample that implements how to use the assert function in the device code. Requires Compute Capability 2.0 .

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaMallocHost, cudaFree, cudaFreeHost, cudaMemcpy
Key Concepts	Assert
Supported OSes	Linux, Windows

simpleAssert_nvrtc - simpleAssert with libNVRTC

This CUDA Runtime API sample is a very basic sample that implements how to use the assert function in the device code. Requires Compute Capability 2.0 .

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuLaunchKernel
Key Concepts	Assert, Runtime Compilation
Supported OSes	Linux, Windows

simpleAtomicIntrinsics - Simple Atomic Intrinsics

A simple demonstration of global memory atomic instructions.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaFree, cudaMemcpy, cudaFreeHost
Key Concepts	Atomic Intrinsics
Supported OSes	Linux, Windows

simpleAtomicIntrinsics_nvrtc - Simple Atomic Intrinsics with libNVRTC

A simple demonstration of global memory atomic instructions.This sample makes use of NVRTC for Runtime Compilation.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuLaunchKernel
Key Concepts	Atomic Intrinsics, Runtime Compilation
Supported OSes	Linux, Windows

simpleAttributes

This CUDA Runtime API sample is a very basic example that implements how to use the stream attributes that affect L2 locality. Performance improvement due to use of L2 access policy window can only be noticed on Compute capability 8.0 or higher.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaCtxResetPersistingL2Cache, cudaDeviceSetLimit, cudaFree, cudaGetDeviceProperties, cudaMalloc, cudaMemcpy, cudaStreamCreate, cudaStreamSetAttribute
Key Concepts	Attributes usage on stream
Supported OSes	Linux, Windows

simpleAWBarrier - Simple Arrive Wait Barrier

A simple demonstration of arrive wait barriers.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CPP11, MBCG
Supported SM Architecture	SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaFree, cudaMemcpyAsync
Key Concepts	Arrive Wait Barrier
Supported OSes	Linux, Windows

simpleCallback - Simple CUDA Callbacks

This sample implements multi-threaded heterogeneous computing workloads with the new CPU callbacks for CUDA streams and events introduced with CUDA 5.0.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaStreamCreate, cudaMemcpyAsync, cudaStreamAddCallback, cudaStreamDestroy
Key Concepts	CUDA Streams, Callback Functions, Multithreading
Supported OSes	Linux, Windows

simpleCooperativeGroups - Simple Cooperative Groups

This sample is a simple code that illustrates basic usage of cooperative groups within the thread block.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Cooperative Groups
Supported OSes	Linux, Windows

simpleCubemapTexture - Simple Cubemap Texture

Simple example that demonstrates how to use a new CUDA 4.1 feature to support cubemap Textures in CUDA C.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaMalloc3DArray, cudaMemcpy3D, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	Texture, Volume Processing
Supported OSes	Linux, Windows

simpleCudaGraphs - Simple CUDA Graphs

A demonstration of CUDA Graphs creation, instantiation and launch using Graphs APIs and Stream Capture APIs.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaStreamBeginCapture, cudaStreamEndCapture, cudaLaunchHostFunc, cudaGraphCreate, cudaGraphLaunch, cudaGraphInstantiate, cudaGraphAddHostNode, cudaGraphAddMemcpyNode, cudaGraphAddKernelNode, cudaGraphAddMemsetNode, cudaGraphExecDestroy, cudaGraphDestroy
Key Concepts	CUDA Graphs, Stream Capture
Supported OSes	Linux, Windows

simpleDrvRuntime - Simple Driver-Runtime Interaction

A simple example which demonstrates how CUDA Driver and Runtime APIs can work together to load cuda fatbinary of vector add kernel and performing vector addition.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuModuleLoadData, cuModuleGetFunction, cuLaunchKernel cudaMemcpy, cudaMalloc, cudaStreamCreateWithFlags
Key Concepts	CUDA Driver API, CUDA Runtime API, Vector Addition
Supported OSes	Linux, Windows

simpleIPC

This CUDA Runtime API sample is a very basic sample that demonstrates Inter Process Communication with one process per GPU for computation. Requires Compute Capability 3.0 or higher and a Linux Operating System, or a Windows Operating System with TCC enabled GPUs

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	IPC

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaIpcGetEventHandle, cudaIpcOpenMemHandle, cudaIpcCloseMemHandle, cudaMemcpyAsync
Key Concepts	CUDA Systems Integration, Peer to Peer, InterProcess Communication
Supported OSes	Linux, Windows

simpleLayeredTexture - Simple Layered Texture

Simple example that demonstrates how to use a new CUDA 4.0 feature to support layered Textures in CUDA C.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaMalloc3DArray, cudaMemcpy3D, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	Texture, Volume Processing
Supported OSes	Linux, Windows

simpleMPI

Simple example demonstrating how to use MPI in combination with CUDA.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	MPI
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallco, cudaFree, cudaMemcpy
Key Concepts	CUDA Systems Integration, MPI, Multithreading
Supported OSes	Linux, Windows

simpleMultiCopy - Simple Multi Copy and Compute

Supported in GPUs with Compute Capability 1.1, overlapping compute with one memcopy is possible from the host system. For Quadro and Tesla GPUs with Compute Capability 2.0, a second overlapped copy operation in either direction at full speed is possible (PCI-e is symmetric). This sample illustrates the usage of CUDA streams to achieve overlapping of kernel execution with data copies to and from the device.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	CUDA Streams and Events, Asynchronous Data Transfers, Overlap Compute and Copy, GPU Performance
Supported OSes	Linux, Windows

simpleMultiGPU - Simple Multi-GPU

This application demonstrates how to use the new CUDA 4.0 API for CUDA context management and multi-threaded access to run CUDA kernels on multiple-GPUs.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events, Multithreading, Multi-GPU
Supported OSes	Linux, Windows

simpleOccupancy

This sample demonstrates the basic usage of the CUDA occupancy calculator and occupancy-based launch configurator APIs by launching a kernel with the launch configurator, and measures the utilization difference against a manually configured launch.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Occupancy Calculator
Supported OSes	Linux, Windows

simpleP2P - Simple Peer-to-Peer Transfers with Multi-GPU

This application demonstrates CUDA APIs that support Peer-To-Peer (P2P) copies, Peer-To-Peer (P2P) addressing, and Unified Virtual Memory Addressing (UVA) between multiple GPUs. In general, P2P is supported between two same GPUs with some exceptions, such as some Tesla and Quadro GPUs.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	only-64-bit
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaDeviceCanAccessPeer, cudaDeviceEnablePeerAccess, cudaDeviceDisablePeerAccess, cudaEventCreateWithFlags, cudaEventElapsedTime, cudaMemcpy
Key Concepts	Performance Strategies, Asynchronous Data Transfers, Unified Virtual Address Space, Peer to Peer Data Transfers, Multi-GPU
Supported OSes	Linux, Windows

simplePitchLinearTexture - Pitch Linear Texture

Use of Pitch Linear Textures

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallocPitch, cudaMallocArray, cudaMemcpy2D, cudaMemcpyToArray, cudaBindTexture2D, cudaBindTextureToArray, cudaCreateChannelDesc, cudaMalloc, cudaFree, cudaFreeArray, cudaUnbindTexture, cudaMemset2D, cudaMemcpy2D
Key Concepts	Texture, Image Processing
Supported OSes	Linux, Windows

simplePrintf

This basic CUDA Runtime API sample demonstrates how to use the printf function in the device code.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Debugging
Supported OSes	Linux, Windows

simpleSeparateCompilation - Simple Static GPU Device Library

This sample demonstrates a CUDA 5.0 feature, the ability to create a GPU device static library and use it within another CUDA kernel. This example demonstrates how to pass in a GPU device function (from the GPU device static library) as a function pointer to be called. This sample requires devices with compute capability 2.0 or higher.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Separate Compilation
Supported OSes	Linux, Windows

simpleStreams

This sample uses CUDA streams to overlap kernel executions with memory copies between the host and a GPU device. This sample uses a new CUDA 4.0 feature that supports pinning of generic host memory. Requires Compute Capability 2.0 or higher.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaMemcpyAsync
Key Concepts	Asynchronous Data Transfers, CUDA Streams and Events
Supported OSes	Linux, Windows

simpleSurfaceWrite - Simple Surface Write

Simple example that demonstrates the use of 2D surface references (Write-to-Texture)

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaMallocArray, cudaBindSurfaceToArray, cudaBindTextureToArray, cudaCreateChannelDesc, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	Texture, Surface Writes, Image Processing
Supported OSes	Linux, Windows

simpleTemplates - Simple Templates

This sample is a templatized version of the template project. It also shows how to correctly templatize dynamically allocated shared memory arrays.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	C++ Templates
Supported OSes	Linux, Windows

simpleTemplates_nvrtc - Simple Templates with libNVRTC

This sample is a templatized version of the template project. It also shows how to correctly templatize dynamically allocated shared memory arrays.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	C++ Templates, Runtime Compilation
Supported OSes	Linux, Windows

simpleTexture - Simple Texture

Simple example that demonstrates use of Textures in CUDA.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

CUDA API	cudaMalloc, cudaMallocArray, cudaMemcpyToArray, cudaCreateChannelDesc, cudaBindTextureToArray, cudaMalloc, cudaFree, cudaFreeArray, cudaMemcpy
Key Concepts	CUDA Runtime API, Texture, Image Processing
Supported OSes	Linux, Windows

simpleTextureDrv - Simple Texture (Driver Version)

Simple example that demonstrates use of Textures in CUDA. This sample uses the new CUDA 4.0 kernel launch Driver API.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuLaunchKernel, cuCtxSynchronize, cuMemcpyDtoH, cuMemAlloc, cuMemFree, cuArrayCreate, cuArrayDestroy, cuCtxDetach, cuMemcpy2D, cuModuleGetTexRef, cuTexRefSetArray, cuTexRefSetAddressMode, cuTexRefSetFilterMode, cuTexRefSetFlags, cuTexRefSetFormat, cuParamSetTexRef
Key Concepts	CUDA Driver API, Texture, Image Processing
Supported OSes	Linux, Windows

simpleVoteIntrinsics - Simple Vote Intrinsics

Simple program which demonstrates how to use the Vote (any, all) intrinsic instruction in a CUDA kernel. Requires Compute Capability 2.0 or higher.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallco, cudaFree, cudaMemcpy, cudaFreeHost
Key Concepts	Vote Intrinsics
Supported OSes	Linux, Windows

simpleVoteIntrinsics_nvrtc - Simple Vote Intrinsics with libNVRTC

Simple program which demonstrates how to use the Vote (any, all) intrinsic instruction in a CUDA kernel with runtime compilation using NVRTC APIs. Requires Compute Capability 2.0 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemFree
Key Concepts	Vote Intrinsics, CUDA Driver API, Runtime Compilation
Supported OSes	Linux, Windows

simpleZeroCopy

This sample illustrates how to use Zero MemCopy, kernels can read and write directly to pinned system memory.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaHostAlloc, cudaHostGetDevicePointer, cudaHostRegister, cudaHostUnregister, cudaFreeHost
Key Concepts	Performance Strategies, Pinned System Paged Memory, Vector Addition
Supported OSes	Linux, Windows
Whitepaper	CUDA2.2PinnedMemoryAPIs.pdf

streamOrderedAllocation - stream Ordered Allocation

This sample demonstrates stream ordered memory allocation on a GPU using cudaMallocAsync and cudaMemPool family of APIs.

Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallocAsync, cudaFreeAsync, cudaMemPoolSetAttribute, cudaDeviceGetDefaultMemPool
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows

streamOrderedAllocationIPC - stream Ordered Allocation IPC Pools

This sample demonstrates IPC pools of stream ordered memory allocated using cudaMallocAsync and cudaMemPool family of APIs.

Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallocAsync, cudaFreeAsync, cudaMemPoolCreate, cudaMemPoolImportPointer, cudaMemPoolSetAccess, cudaMemPoolGetAccess, cudaMemPoolExportToShareableHandle, cudaMemPoolExportPointer, cudaMemPoolDestroy
Key Concepts	Performance Strategies
Supported OSes	Linux

streamOrderedAllocationP2P - stream Ordered Allocation Peer-to-Peer access

This sample demonstrates peer-to-peer access of stream ordered memory allocated using cudaMallocAsync and cudaMemPool family of APIs.

Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallocAsync, cudaFreeAsync, cudaMemPoolSetAccess, cudaDeviceGetDefaultMemPool
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows

systemWideAtomics - System wide Atomics

A simple demonstration of system wide atomic instructions.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	UVM
Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaFree, cudaMemcpy, cudaFreeHost
Key Concepts	Atomic Intrinsics, Unified Memory
Supported OSes	Linux

template - Template

A trivial template project that can be used as a starting point to create new CUDA projects.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaFree, cudaDeviceSynchronize, cudaMemcpy
Key Concepts	Device Memory Allocation
Supported OSes	Linux, Windows

tf32TensorCoreGemm - tf32 Tensor Core GEMM

A CUDA sample demonstrating tf32 (e8m10) GEMM computation using the Warp Matrix Multiply and Accumulate (WMMA) API introduced with CUDA 11 in Ampere chip family tensor cores for faster matrix operations. This sample also uses async copy provided by cuda pipeline interface for gmem to shmem async loads which improves kernel performance and reduces register presssure.

Supported SM Architecture	SM 8.0, SM 8.6
CUDA API	cudaMalloc, cudaDeviceSynchronize, cudaFuncSetAttribute, cudaEventCreate, cudaEventRecord, cudaEventSynchronize, cudaEventElapsedTime, cudaFree
Key Concepts	Matrix Multiply, WMMA, Tensor Cores
Supported OSes	Linux, Windows

UnifiedMemoryStreams - Unified Memory Streams

This sample demonstrates the use of OpenMP and streams with Unified Memory on a single GPU.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	OpenMP, UVM, CUBLAS
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallocManaged, cudaStreamAttachManagedMem
Key Concepts	CUDA Systems Integration, OpenMP, CUBLAS, Multithreading, Unified Memory, CUDA Streams and Events
Supported OSes	Linux, Windows

vectorAdd - Vector Addition

This CUDA Runtime API sample is a very basic sample that implements element by element vector addition. It is the same as the sample illustrating Chapter 3 of the programming guide with some additions like error checking.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaEventCreate, cudaEventRecord, cudaEventQuery, cudaEventDestroy, cudaEventElapsedTime, cudaEventSynchronize, cudaMalloc, cudaFree, cudaMemcpy
Key Concepts	CUDA Runtime API, Vector Addition
Supported OSes	Linux, Windows

vectorAdd_nvrtc - Vector Addition with libNVRTC

This CUDA Driver API sample uses NVRTC for runtime compilation of vector addition kernel. Vector addition kernel demonstrated is the same as the sample illustrating Chapter 3 of the programming guide.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH
Key Concepts	CUDA Driver API, Vector Addition, Runtime Compilation
Supported OSes	Linux, Windows

vectorAddDrv - Vector Addition Driver API

This Vector Addition sample is a basic sample that is implemented element by element. It is the same as the sample illustrating Chapter 3 of the programming guide with some additions like error checking. This sample also uses the new CUDA 4.0 kernel launch Driver API.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Driver API, Vector Addition
Supported OSes	Linux, Windows

vectorAddMMAP - Vector Addition cuMemMap

This sample replaces the device allocation in the vectorAddDrv sample with cuMemMap-ed allocations. This sample demonstrates that the cuMemMap api allows the user to specify the physical properties of their memory while retaining the contiguos nature of their access, thus not requiring a change in their program structure.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel, cuLaunchKernel, cuDeviceGetAttribute, cuDeviceCanAccessPeer, cuMemGetAllocationGranularity, cuMemAddressReserve, cuMemCreate, cuMemMap, cuMemSetAccess, cuMemUnmap, cuMemAddressFree
Key Concepts	CUDA Driver API, Vector Addition, MMAP

Supported OSes	Linux, Windows

3.2. Utilities Reference

bandwidthTest - Bandwidth Test

This is a simple test program to measure the memcopy bandwidth of the GPU and memcpy bandwidth across PCI-e. This test application is capable of measuring device to device copy bandwidth, host to device copy bandwidth for pageable and page-locked memory, and device to host copy bandwidth for pageable and page-locked memory.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaSetDevice, cudaHostAlloc, cudaFree, cudaMallocHost, cudaFreeHost, cudaMemcpy, cudaMemcpyAsync, cudaEventCreate, cudaEventRecord, cudaEventDestroy, cudaDeviceSynchronize, cudaEventElapsedTime
Key Concepts	CUDA Streams and Events, Performance Strategies
Supported OSes	Linux, Windows

deviceQuery - Device Query

This sample enumerates the properties of the CUDA devices present in the system.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaSetDevice, cudaGetDeviceCount, cudaGetDeviceProperties, cudaDriverGetVersion, cudaRuntimeGetVersion
Key Concepts	CUDA Runtime API, Device Query
Supported OSes	Linux, Windows

deviceQueryDrv - Device Query Driver API

This sample enumerates the properties of the CUDA devices present using CUDA Driver API calls

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuInit, cuDeviceGetCount, cuDeviceComputeCapability, cuDriverGetVersion, cuDeviceTotalMem, cuDeviceGetAttribute

Key Concepts	CUDA Driver API, Device Query
Supported OSes	Linux, Windows

p2pBandwidthLatencyTest - Peer-to-Peer Bandwidth Latency Test with Multi-GPUs

This application demonstrates the CUDA Peer-To-Peer (P2P) data transfers between pairs of GPUs and computes latency and bandwidth. Tests on GPU pairs using P2P and without P2P are tested.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaDeviceCanAccessPeer, cudaDeviceEnablePeerAccess, cudaDeviceDisablePeerAccess, cudaEventCreateWithFlags, cudaEventElapsedTime, cudaMemcpy
Key Concepts	Performance Strategies, Asynchronous Data Transfers, Unified Virtual Address Space, Peer to Peer Data Transfers, Multi-GPU
Supported OSes	Linux, Windows

topologyQuery - Topology Query

A simple exemple on how to query the topology of a system with multiple GPU

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaDeviceGetP2PAttribute, cudaGetDeviceAttribute, cudaGetDeviceCount
Key Concepts	Performance Strategies, Multi-GPU
Supported OSes	Linux, Windows

UnifiedMemoryPerf - Unified and other CUDA Memories Performance

This sample demonstrates the performance comparision using matrix multiplication kernel of Unified Memory with/without hints and other types of memory like zero copy buffers, pageable, pagelocked memory performing synchronous and Asynchronous transfers on a single GPU.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	UVM
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMallocManaged, cudaStreamAttachMemAsync, cudaMemcpyAsync, cudaMallocHost, cudaMalloc
Key Concepts	CUDA Systems Integration, Unified Memory, CUDA Streams and Events, Pinned System Paged Memory
Supported OSes	Linux, Windows

3.3. Graphics Reference

bindlessTexture - Bindless Texture

This example demonstrates use of cudaSurfaceObject, cudaTextureObject, and MipMap support in CUDA. A GPU with Compute Capability SM 3.0 is required to run the sample.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Linux, Windows

Mandelbrot

This sample uses CUDA to compute and display the Mandelbrot or Julia sets interactively. It also illustrates the use of "double single" arithmetic to improve precision when zooming a long way into the pattern. This sample uses double precision. Thanks to Mark Granger of NewTek who submitted this code sample.!

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be

installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms
Supported OSes	Linux, Windows

marchingCubes - Marching Cubes Isosurfaces

This sample extracts a geometric isosurface from a volume dataset using the marching cubes algorithm. It uses the scan (prefix sum) function from the Thrust library to perform stream compaction.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	OpenGL Graphics Interop, Vertex Buffers, 3D Graphics, Physically Based Simulation
Supported OSes	Linux, Windows

simpleD3D10 - Simple Direct3D10 (Vertex Array)

Simple program which demonstrates interoperability between CUDA and Direct3D10. The program generates a vertex array with CUDA and uses Direct3D10 to render the geometry. A Direct3D Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, 3D Graphics
Supported OSes	Windows

simpleD3D10RenderTarget - Simple Direct3D10 Render Target

Simple program which demonstrates interop of rendertargets between Direct3D10 and CUDA. The program uses RenderTarget positions with CUDA and generates a histogram with visualization. A Direct3D10 Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Windows

simpleD3D10Texture - Simple D3D10 Texture

Simple program which demonstrates how to interoperate CUDA with Direct3D10 Texture. The program creates a number of D3D10 Textures (2D, 3D, and CubeMap) which are generated from CUDA kernels. Direct3D then renders the results on the screen. A Direct3D10 Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Windows

simpleD3D11 - Simple D3D11

Simple program which demonstrates how to use the CUDA D3D11 External Resource Interoperability APIs to update D3D11 buffers from CUDA and synchronize between D3D11 and CUDA with Keyed Mutexes.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D11GetDevice, cudaImportExternalSemaphore, cudaImportExternalMemory, cudaExternalMemoryGetMappedBuffer
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Windows

simpleD3D11Texture - Simple D3D11 Texture

Simple program which demonstrates Direct3D11 Texture interoperability with CUDA. The program creates a number of D3D11 Textures (2D, 3D, and CubeMap) which are written to from CUDA kernels. Direct3D then renders the results on the screen. A Direct3D Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D11GetDevice, cudaD3D11SetDirect3DDevice, cudaGraphicsD3D11RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Windows

simpleD3D12 - Simple D3D12 CUDA Interop

A program which demonstrates Direct3D12 interoperability with CUDA. The program creates a sinewave in DX12 vertex buffer which is created using CUDA kernels. DX12 and CUDA synchronizes using DirectX12 Fences. Direct3D then renders the results on the screen. A DirectX12 Capable NVIDIA GPU is required on Windows10 or higher OS.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX12
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaWaitExternalSemaphoresAsync, cudaSignalExternalSemaphoresAsync, cudaImportExternalSemaphore, cudaExternalMemoryGetMappedBuffer, cudaImportExternalMemory, cudaDestroyExternalSemaphore, cudaDestroyExternalMemory
Key Concepts	Graphics Interop, CUDA DX12 Interop, Image Processing
Supported OSes	Windows

simpleD3D9 - Simple Direct3D9 (Vertex Arrays)

Simple program which demonstrates interoperability between CUDA and Direct3D9. The program generates a vertex array with CUDA and uses Direct3D9 to render the geometry. A Direct3D capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D9GetDevice, cudaD3D9SetDirect3DDevice, cudaGraphicsD3D9RegisterResource, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop
Supported OSes	Windows

simpleD3D9Texture - Simple D3D9 Texture

Simple program which demonstrates Direct3D9 Texture interoperability with CUDA. The program creates a number of D3D9 Textures (2D, 3D, and CubeMap) which are written to from CUDA kernels. Direct3D then renders the results on the screen. A Direct3D capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D9GetDevice, cudaD3D9SetDirect3DDevice, cudaGraphicsD3D9RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaMemcpy3D, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Texture
Supported OSes	Windows

simpleGL - Simple OpenGL

Simple program which demonstrates interoperability between CUDA and OpenGL. The program modifies vertex positions with CUDA and uses OpenGL to render the geometry.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Vertex Buffers, 3D Graphics
Supported OSes	Linux, Windows

simpleGLES - Simple OpenGLES

Demonstrates data exchange between CUDA and OpenGL ES (aka Graphics interop). The program modifies vertex positions with CUDA and uses OpenGL ES to render the geometry.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GLES
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Vertex Buffers, 3D Graphics
Supported OSes	Linux

simpleGLES_EGLOutput - Simple OpenGLES EGLOutput

Demonstrates data exchange between CUDA and OpenGL ES (aka Graphics interop). The program modifies vertex positions with CUDA and uses OpenGL ES to render the geometry, and shows how to render directly to the display using the EGLOutput mechanism and the DRM library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	EGLOutput, GLES
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Vertex Buffers, 3D Graphics
Supported OSes	Linux

simpleGLES_screen - Simple OpenGLES on Screen

Demonstrates data exchange between CUDA and OpenGL ES (aka Graphics interop). The program modifies vertex positions with CUDA and uses OpenGL ES to render the geometry.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	screen, GLES
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Vertex Buffers, 3D Graphics
Supported OSes	Linux

simpleTexture3D - Simple Texture 3D

Simple example that demonstrates use of 3D Textures in CUDA.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures, Surface Writes
Supported OSes	Linux, Windows

simpleVulkan - Vulkan CUDA Interop Sinewave

This sample demonstrates Vulkan CUDA Interop. CUDA imports the Vulkan vertex buffer and operates on it to create sinewave, and synchronizes with Vulkan through vulkan semaphores imported by CUDA. This sample depends on Vulkan SDK, GLFW3 libraries, for building this sample please refer to "Build_instructions.txt" provided in this sample's directory

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, VULKAN
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaImportExternalMemory, cudaExternalMemoryGetMappedBuffer, cudaImportExternalSemaphore, cudaImportExternalSemaphore, cudaSignalExternalSemaphoresAsync, cudaWaitExternalSemaphoresAsync, cudaDestroyExternalSemaphore, cudaDestroyExternalMemory
Key Concepts	Graphics Interop, CUDA Vulkan Interop, Data Parallel Algorithms
Supported OSes	Linux, Windows

simpleVulkanMMAP - Vulkan CUDA Interop PI Approximation

This sample demonstrates Vulkan CUDA Interop via cuMemMap APIs. CUDA exports buffers that Vulkan imports as vertex buffer. CUDA invokes kernels to operate on vertices and synchronizes with Vulkan through vulkan semaphores imported by CUDA. This sample depends on Vulkan SDK, GLFW3 libraries, for building this sample please refer to "Build_instructions.txt" provided in this sample's directory

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, VULKAN
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuDeviceGetAttribute, cuMemAddressReserve, cuMemCreate, cuMemRelease, cuCtxSetCurrent, cuMemExportToShareableHandle, cuMemImportFromShareableHandle, cuMemMap, cuMemSetAccess, cuMemUnmap, cuMemAddressFree cudaGetDeviceProperties, cudaImportExternalMemory, cudaExternalMemoryGetMappedBuffer, cudaImportExternalSemaphore, cudaImportExternalSemaphore, cudaSignalExternalSemaphoresAsync, cudaWaitExternalSemaphoresAsync, cudaDestroyExternalSemaphore, cudaDestroyExternalMemory
Key Concepts	cuMemMap IPC, MMAP, Graphics Interop, CUDA Vulkan Interop, Data Parallel Algorithms
Supported OSes	Linux, Windows

SLID3D10Texture - SLI D3D10 Texture

Simple program which demonstrates SLI with Direct3D10 Texture interoperability with CUDA. The program creates a D3D10 Texture which is written to from a CUDA kernel. Direct3D then renders the results on the screen. A Direct3D Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D10GetDevice, cudaD3D10SetDirect3DDevice, cudaGraphicsD3D10RegisterResource, cudaGraphicsResourceSetMapFlags, cudaGraphicsSubResourceGetMappedArray, cudaMemcpy2DToArray, cudaGraphicsUnregisterResource
Key Concepts	Performance Strategies, Graphics Interop, Image Processing, 2D Textures
Supported OSes	Windows

volumeFiltering - Volumetric Filtering with 3D Textures and Surface Writes

This sample demonstrates 3D Volumetric Filtering using 3D Textures and 3D Surface Writes.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures, Surface Writes
Supported OSes	Linux, Windows

volumeRender - Volume Rendering with 3D Textures

This sample demonstrates basic volume rendering using 3D Textures.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, 3D Textures
Supported OSes	Linux, Windows

3.4. Imaging Reference

bicubicTexture - Bicubic B-spline Interoplation

This sample demonstrates how to efficiently implement a Bicubic B-spline interpolation filter with CUDA texture.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows

bilateralFilter - Bilateral Filter

Bilateral filter is an edge-preserving non-linear smoothing filter that is implemented with CUDA with OpenGL rendering. It can be used in image recovery and denoising. Each pixel is weight by considering both the spatial distance and color distance between its neighbors. Reference:"C. Tomasi, R. Manduchi, Bilateral Filtering for Gray and Color Images, proceeding of the ICCV, 1998, http://users.soe.ucsc.edu/~manduchi/Papers/ICCV98.pdf"

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows

boxFilter - Box Filter

Fast image box filter using CUDA with OpenGL rendering.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows

convolutionFFT2D - FFT-Based 2D Convolution

This sample demonstrates how 2D convolutions with very large kernel sizes can be efficiently implemented using FFT transformations.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUFFT
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

CUDA API	cufftPlan2d, cufftExecR2C, cufftExecC2R, cufftDestroy
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux, Windows

convolutionSeparable - CUDA Separable Convolution

This sample implements a separable convolution filter of a 2D signal with a gaussian kernel.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, Data Parallel Algorithms
Supported OSes	Linux, Windows
Whitepaper	convolutionSeparable.pdf

convolutionTexture - Texture-based Separable Convolution

Texture-based implementation of a separable 2D convolution with a gaussian kernel. Used for performance comparison against convolutionSeparable.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, Texture, Data Parallel Algorithms
Supported OSes	Linux, Windows

cudaNvSciNvMedia - NvMedia CUDA Interop

This sample demonstrates CUDA-NvMedia interop via NvSciBuf/NvSciSync APIs. Note that this sample only supports cross build from x86_64 to aarch64, aarch64 native build is not supported. For detailed workflow of the sample please check cudaNvSciNvMedia_Readme.pdf in the sample directory.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVSCI, NvMedia

Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaImportExternalMemory, cudaExternalMemoryGetMappedBuffer, cudaExternalMemoryGetMappedMipmappedArray, cudaImportExternalSemaphore, cudaSignalExternalSemaphoresAsync, cudaWaitExternalSemaphoresAsync, cudaDestroyExternalSemaphore, cudaDestroyExternalMemory
Key Concepts	CUDA NvSci Interop, Data Parallel Algorithms, Image Processing
Supported OSes	Linux

dct8x8 - DCT8x8

This sample demonstrates how Discrete Cosine Transform (DCT) for blocks of 8 by 8 pixels can be performed using CUDA: a naive implementation by definition and a more traditional approach used in many libraries. As opposed to implementing DCT in a fragment shader, CUDA allows for an easier and more efficient implementation.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, Video Compression
Supported OSes	Linux, Windows
Whitepaper	dct8x8.pdf

dwtHaar1D - 1D Discrete Haar Wavelet Decomposition

Discrete Haar wavelet decomposition for 1D signals with a length which is a power of 2.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, Video Compression
Supported OSes	Linux, Windows

dxtc - DirectX Texture Compressor (DXTC)

High Quality DXT Compression using CUDA. This example shows how to implement an existing computationally-intensive CPU compression algorithm in parallel on the GPU, and obtain an order of magnitude performance improvement.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Cooperative Groups, Image Processing, Image Compression
Supported OSes	Linux, Windows
Whitepaper	cuda_dxtc.pdf

EGLStream_CUDA_CrossGPU

Demonstrates CUDA and EGL Streams interop, where consumer's EGL Stream is on one GPU and producer's on other and both consumer-producer are different processes.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	EGL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuDeviceGet, cuDeviceGetAttribute, cuDeviceComputeCapability, cuDeviceGetCount, cuDeviceGetName, cuGraphicsResourceGetMappedEglFrame, cuEGLStreamConsumerAcquireFrame, cuEGLStreamConsumerReleaseFrame, cuEGLStreamProducerReturnFrame, cuEGLStreamProducerPresentFrame, cuCtxCreate, cuMemAlloc, cuMemFree, cuMemcpy3D, cuStreamCreate, cuCtxPushCurrent, cuCtxPopCurrent, cuCtxDestroy
Key Concepts	EGLStreams Interop
Supported OSes	Linux

EGLStreams_CUDA_Interop - EGLStreams CUDA Interop

Demonstrates data exchange between CUDA and EGL Streams.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	EGL

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuDeviceGet, cuDeviceGetAttribute, cuDeviceComputeCapability, cuDeviceGetCount, cuDeviceGetName, cuGraphicsResourceGetMappedEglFrame, cuEGLStreamConsumerAcquireFrame, cuEGLStreamConsumerReleaseFrame, cuEGLStreamProducerPresentFrame, cuCtxCreate, cuMemAlloc, cuMemFree, cuMemcpy3D, cuStreamCreate, cuCtxPushCurrent, cuCtxPopCurrent, cuCtxDestroy
Key Concepts	EGLStreams Interop
Supported OSes	Linux

EGLSync_CUDA_Interop - EGLSync CUDA Event Interop

Demonstrates interoperability between CUDA Event and EGL Sync/EGL Image using which one can achieve synchronization on GPU itself for GL-EGL-CUDA operations instead of blocking CPU for synchronization.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	EGL, EGLSync, X11, GLES
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuDeviceGet, cuDeviceGetAttribute, cuDeviceComputeCapability, cuDeviceGetCount, cuDeviceGetName, cuCtxCreate, cuMemAlloc, cuMemFree, cuMemcpy3D, cuStreamCreate, cuCtxPushCurrent, cuCtxPopCurrent, cuCtxDestroy
Key Concepts	EGLSync-CUDAEvent Interop, EGLImage-CUDA Interop
Supported OSes	Linux

histogram - CUDA Histogram

This sample demonstrates efficient implementation of 64-bin and 256-bin histogram.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

Key Concepts	Image Processing, Data Parallel Algorithms
Supported OSes	Linux, Windows
Whitepaper	histogram.pdf

HSOpticalFlow - Optical Flow

Variational optical flow estimation example. Uses textures for image operations. Shows how simple PDE solver can be accelerated with CUDA.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, Data Parallel Algorithms
Supported OSes	Linux, Windows
Whitepaper	OpticalFlow.pdf

imageDenoising - Image denoising

This sample demonstrates two adaptive image denoising techniques: KNN and NLM, based on computation of both geometric and color distance between texels. While both techniques are implemented in the DirectX SDK using shaders, massively speeded up variation of the latter technique, taking advantage of shared memory, is implemented in addition to DirectX counterparts.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing
Supported OSes	Linux, Windows
Whitepaper	imageDenoising.pdf

NV12toBGRandResize

This code shows two ways to convert and resize NV12 frames to BGR 3 planars frames using CUDA in batch. Way-1, Convert NV12 Input to BGR @ Input Resolution-1, then Resize to Resolution#2. Way-2, resize NV12 Input to Resolution#2 then convert it to BGR Output. NVIDIA HW Decoder, both dGPU and Tegra, normally outputs NV12 pitch format frames. For the inference using TensorRT, the input frame needs to be BGR planar format with possibly different size. So, conversion and resizing from NV12 to BGR planar is usually required for the inference following decoding. This CUDA code provides a reference implementation for conversion and resizing.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMemcpy2D, cudaMallocManaged
Key Concepts	Graphics Interop, Image Processing, Video Processing
Supported OSes	Linux, Windows

postProcessGL - Post-Process in OpenGL

This sample shows how to post-process an image rendered in OpenGL using CUDA.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows

recursiveGaussian - Recursive Gaussian Filter

This sample implements a Gaussian blur using Deriche's recursive method. The advantage of this method is that the execution time is independent of the filter width.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows

simpleCUDA2GL - CUDA and OpenGL Interop of Images

This sample shows how to copy CUDA image back to OpenGL using the most efficient methods.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing, Performance Strategies
Supported OSes	Linux, Windows

SobelFilter - Sobel Filter

This sample implements the Sobel edge detection filter for 8-bit monochrome images.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Image Processing
Supported OSes	Linux, Windows

stereoDisparity - Stereo Disparity Computation (SAD SIMD Intrinsics)

A CUDA program that demonstrates how to compute a stereo disparity map using SIMD SAD (Sum of Absolute Difference) intrinsics. Requires Compute Capability 2.0 or higher.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, Video Intrinsics
Supported OSes	Linux, Windows

vulkanImageCUDA - Vulkan Image - CUDA Interop

This sample demonstrates Vulkan Image - CUDA Interop. CUDA imports the Vulkan image buffer, performs box filtering over it, and synchronizes with Vulkan through vulkan semaphores imported by CUDA. This sample depends on Vulkan SDK, GLFW3 libraries, for building this sample please refer to "Build_instructions.txt" provided in this sample's directory

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, VULKAN
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaImportExternalMemory, cudaExternalMemoryGetMappedMipmappedArray, cudaImportExternalSemaphore, cudaSignalExternalSemaphoresAsync, cudaWaitExternalSemaphoresAsync, cudaDestroyExternalSemaphore, cudaDestroyExternalMemory
Key Concepts	Graphics Interop, CUDA Vulkan Interop, Data Parallel Algorithms
Supported OSes	Linux, Windows

3.5. Finance Reference

binomialOptions - Binomial Option Pricing

This sample evaluates fair call price for a given set of European options under binomial model.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Computational Finance
Supported OSes	Linux, Windows
Whitepaper	binomialOptions.pdf

binomialOptions_nvrtc - Binomial Option Pricing with libNVRTC

This sample evaluates fair call price for a given set of European options under binomial model. This sample makes use of NVRTC for Runtime Compilation.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Computational Finance, Runtime Compilation
Supported OSes	Linux, Windows

BlackScholes - Black-Scholes Option Pricing

This sample evaluates fair call and put prices for a given set of European options by Black-Scholes formula.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Computational Finance
Supported OSes	Linux, Windows
Whitepaper	BlackScholes.pdf

BlackScholes_nvrtc - Black-Scholes Option Pricing with libNVRTC

This sample evaluates fair call and put prices for a given set of European options by Black-Scholes formula, compiling the CUDA kernels involved at runtime using NVRTC.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Computational Finance, Runtime Compilation
Supported OSes	Linux, Windows

MonteCarloMultiGPU - Monte Carlo Option Pricing with Multi-GPU support

This sample evaluates fair call price for a given set of European options using the Monte Carlo approach, taking advantage of all CUDA-capable GPUs installed in the system. This sample use double precision hardware if a GTX 200 class GPU is present. The sample also takes advantage of CUDA 4.0 capability to supporting using a single CPU thread to control multiple GPUs

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Supported OSes	Linux, Windows
Whitepaper	MonteCarlo.pdf

quasirandomGenerator - Niederreiter Quasirandom Sequence Generator

This sample implements Niederreiter Quasirandom Sequence Generator and Inverse Cumulative Normal Distribution functions for the generation of Standard Normal Distributions.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Computational Finance
Supported OSes	Linux, Windows

quasirandomGenerator_nvrtc - Niederreiter Quasirandom Sequence Generator with libNVRTC

This sample implements Niederreiter Quasirandom Sequence Generator and Inverse Cumulative Normal Distribution functions for the generation of Standard Normal Distributions, compiling the CUDA kernels involved at runtime using NVRTC.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVRTC
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Computational Finance, Runtime Compilation
Supported OSes	Linux, Windows

SobolQRNG - Sobol Quasirandom Number Generator

This sample implements Sobol Quasirandom Sequence Generator.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Computational Finance

Supported OSes	Linux, Windows

3.6. Simulations Reference

fluidsD3D9 - Fluids (Direct3D Version)

An example of fluid simulation using CUDA and CUFFT, with Direct3D 9 rendering. A Direct3D Capable device is required.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D9SetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, CUFFT Library, Physically-Based Simulation
Supported OSes	Windows

fluidsGL - Fluids (OpenGL Version)

An example of fluid simulation using CUDA and CUFFT, with OpenGL rendering.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL, CUFFT
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, CUFFT Library, Physically-Based Simulation
Supported OSes	Linux, Windows

Whitepaper	fluidsGL.pdf

fluidsGLES - Fluids (OpenGLES Version)

An example of fluid simulation using CUDA and CUFFT, with OpenGLES rendering.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GLES, CUFFT
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, CUFFT Library, Physically-Based Simulation
Supported OSes	Linux

nbody - CUDA N-Body Simulation

This sample demonstrates efficient all-pairs simulation of a gravitational n-body simulation in CUDA. This sample accompanies the GPU Gems 3 chapter "Fast N-Body Simulation with CUDA". With CUDA 5.5, performance on Tesla K20c has increased to over 1.8TFLOP/s single precision. Double Performance has also improved on all Kepler and Fermi GPU architectures as well. Starting in CUDA 4.0, the nBody sample has been updated to take advantage of new features to easily scale the n-body simulation across multiple GPUs in a single PC. Adding "-numbodies=<bodies>" to the command line will allow users to set # of bodies for simulation. Adding “-numdevices=<N>” to the command line option will cause the sample to use N devices (if available) for simulation. In this mode, the position and velocity data for all bodies are read from system memory using “zero copy” rather than from device memory. For a small number of devices (4 or fewer) and a large enough number of bodies, bandwidth is not a bottleneck so we can achieve strong scaling across these devices.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation
Supported OSes	Linux, Windows
Whitepaper	nbody_gems3_ch31.pdf

nbody_opengles - CUDA N-Body Simulation with GLES

This sample demonstrates efficient all-pairs simulation of a gravitational n-body simulation in CUDA. Unlike the OpenGL nbody sample, there is no user interaction.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GLES
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation
Supported OSes	Linux

nbody_screen - CUDA N-Body Simulation on Screen

This sample demonstrates efficient all-pairs simulation of a gravitational n-body simulation in CUDA. Unlike the OpenGL nbody sample, there is no user interaction.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	screen, GLES
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation
Supported OSes	Linux

oceanFFT - CUDA FFT Ocean Simulation

This sample simulates an Ocean height field using CUFFT Library and renders the result using OpenGL.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL, CUFFT
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource, cufftPlan2d, cufftExecR2C, cufftExecC2R, cufftDestroy
Key Concepts	Graphics Interop, Image Processing, CUFFT Library
Supported OSes	Linux, Windows

particles - Particles

This sample uses CUDA to simulate and visualize a large set of particles and their physical interaction. Adding "-particles=<N>" to the command line will allow users to set # of particles for simulation. This example implements a uniform grid data structure using either atomic operations or a fast radix sort from the Thrust library

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6

CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation, Performance Strategies
Supported OSes	Linux, Windows
Whitepaper	particles.pdf

smokeParticles - Smoke Particles

Smoke simulation with volumetric shadows using half-angle slicing technique. Uses CUDA for procedural simulation, Thrust Library for sorting algorithms, and OpenGL for graphics rendering.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation
Supported OSes	Linux, Windows
Whitepaper	smokeParticles.pdf

VFlockingD3D10

The sample models formation of V-shaped flocks by big birds, such as geese and cranes. The algorithms of such flocking are borrowed from the paper "V-like formations in flocks of artificial birds" from Artificial Life, Vol. 14, No. 2, 2008. The sample has CPU- and GPU-based implementations. Press 'g' to toggle between them. The GPU-based simulation works many times faster than the CPU-based one. The printout in the console window reports the simulation time per step. Press 'r' to reset the initial distribution of birds.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	DirectX
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaD3D10SetGLDevice, cudaGraphicsMapResources, cudaGraphicsUnmapResources, cudaGraphicsResourceGetMappedPointer, cudaGraphicsRegisterResource, cudaGraphicsGLRegisterBuffer, cudaGraphicsUnregisterResource
Key Concepts	Graphics Interop, Data Parallel Algorithms, Physically-Based Simulation, Performance Strategies
Supported OSes	Windows

3.7. Advanced Reference

alignedTypes - Aligned Types

A simple test, showing huge access speed gap between aligned and misaligned structures.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows

c++11_cuda - C++11 CUDA

This sample demonstrates C++11 feature support in CUDA. It scans a input text file and prints no. of occurrences of x, y, z, w characters.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CPP11
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CPP11 CUDA
Supported OSes	Linux, Windows

cdpAdvancedQuicksort - Advanced Quicksort (CUDA Dynamic Parallelism)

This sample demonstrates an advanced quicksort implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Cooperative Groups, CUDA Dynamic Parallelism
Supported OSes	Linux, Windows

cdpBezierTessellation - Bezier Line Tessellation (CUDA Dynamic Parallelism)

This sample demonstrates bezier tessellation of lines implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CUDA Dynamic Parallelism
Supported OSes	Linux, Windows

cdpQuadtree - Quad Tree (CUDA Dynamic Parallelism)

This sample demonstrates Quad Trees implemented using CUDA Dynamic Parallelism. This sample requires devices with compute capability 3.5 or higher.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CDP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Cooperative Groups, CUDA Dynamic Parallelism
Supported OSes	Linux, Windows

concurrentKernels - Concurrent Kernels

This sample demonstrates the use of CUDA streams for concurrent execution of several kernels on devices of compute capability 2.0 or higher. Devices of compute capability 1.x will run the kernels sequentially. It also illustrates how to introduce dependencies between CUDA streams with the new cudaStreamWaitEvent function introduced in CUDA 3.2

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows

conjugateGradientMultiBlockCG - conjugateGradient using MultiBlock Cooperative Groups

This sample implements a conjugate gradient solver on GPU using Multi Block Cooperative Groups, also uses Unified Memory.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	UVM, MBCG
Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Unified Memory, Linear Algebra, Cooperative Groups, MultiBlock Cooperative Groups
Supported OSes	Linux, Windows

conjugateGradientMultiDeviceCG - conjugateGradient using MultiDevice Cooperative Groups

This sample implements a conjugate gradient solver on multiple GPUs using Multi Device Cooperative Groups, also uses Unified Memory optimized using prefetching and usage hints.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	UVM, MDCG, CPP11
Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaMemAdvise, cudaMemPrefetchAsync, cudaLaunchCooperativeKernelMultiDevice, cudaStreamSynchronize, cudaOccupancyMaxActiveBlocksPerMultiprocessor
Key Concepts	Unified Memory, Linear Algebra, Cooperative Groups, MultiDevice Cooperative Groups
Supported OSes	Linux, Windows

cudaCompressibleMemory - CUDA Compressible Memory

This sample demonstrates the compressible memory allocation using cuMemMap API.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuMemAlloc, cuMemFree, cuDeviceGetAttribute, cuMemGetAllocationGranularity, cuMemAddressReserve, cuMemCreate, cuMemMap, cuMemSetAccess, cuMemUnmap, cuMemAddressFree cudaMalloc, cudaFree
Key Concepts	CUDA Driver API, Compressible Memory, MMAP
Supported OSes	Linux, Windows

eigenvalues - Eigenvalues

The computation of all or a subset of all eigenvalues is an important problem in Linear Algebra, statistics, physics, and many other fields. This sample demonstrates a parallel implementation of a bisection algorithm for the computation of all eigenvalues of a tridiagonal symmetric matrix of arbitrary size with CUDA.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra
Supported OSes	Linux, Windows
Whitepaper	eigenvalues.pdf

fastWalshTransform - Fast Walsh Transform

Naturally(Hadamard)-ordered Fast Walsh Transform for batching vectors of arbitrary eligible lengths that are power of two in size.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, Data-Parallel Algorithms, Video Compression
Supported OSes	Linux, Windows

FDTD3d - CUDA C 3D FDTD

This sample applies a finite differences time domain progression stencil on a 3D surface.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies
Supported OSes	Linux, Windows

FunctionPointers - Function Pointers

This sample illustrates how to use function pointers and implements the Sobel Edge Detection filter for 8-bit monochrome images.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Graphics Interop, Image Processing

Supported OSes	Linux, Windows

interval - Interval Computing

Interval arithmetic operators example. Uses various C++ features (templates and recursion). The recursive mode requires Compute SM 2.0 capabilities.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Recursion, Templates
Supported OSes	Linux, Windows

jacobiCudaGraphs - Jacobi CUDA Graphs

Demonstrates Instantiated CUDA Graph Update with Jacobi Iterative Method using cudaGraphExecKernelNodeSetParams() and cudaGraphExecUpdate() approach.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaStreamBeginCapture, cudaStreamEndCapture, cudaGraphCreate, cudaGraphLaunch, cudaGraphInstantiate, cudaGraphExecUpdate, cudaGraphExecKernelNodeSetParams, cudaGraphGetNodes, cudaGraphNodeGetType, cudaGraphExecDestroy, cudaGraphDestroy
Key Concepts	CUDA Graphs, Stream Capture, Instantiated CUDA Graph Update, Cooperative Groups
Supported OSes	Linux, Windows

lineOfSight - Line of Sight

This sample is an implementation of a simple line-of-sight algorithm: Given a height map and a ray originating at some observation point, it computes all the points along the ray that are visible from the observation point. The implementation is based on the Thrust library.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Supported OSes	Linux, Windows

matrixMulDynlinkJIT - Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version)

This sample revisits matrix multiplication using the CUDA driver API. It demonstrates how to link to CUDA driver at runtime and how to use JIT (just-in-time) compilation from PTX code. It has been written for clarity of exposition to illustrate various CUDA programming principles, not with the goal of providing the most performant generic kernel for matrix multiplication. CUBLAS provides high-performance matrix multiplication.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuMemAlloc, cuMemFree, cuMemcpyHtoD, cuMemcpyDtoH, cuLaunchKernel
Key Concepts	CUDA Driver API, CUDA Dynamically Linked Library
Supported OSes	Linux, Windows

mergeSort - Merge Sort

This sample implements a merge sort (also known as Batcher's sort), algorithms belonging to the class of sorting networks. While generally subefficient on large sequences compared to algorithms with better asymptotic algorithmic complexity (i.e. merge sort or radix sort), may be the algorithms of choice for sorting batches of short- to mid-sized (key, value) array pairs. Refer to the excellent tutorial by H. W. Lang http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/networks/indexen.htm

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Data-Parallel Algorithms
Supported OSes	Linux, Windows

newdelete - NewDelete

This sample demonstrates dynamic global memory allocation through device C++ new and delete operators and virtual function declarations available with CUDA 4.0.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Supported OSes	Linux, Windows

ptxjit - PTX Just-in-Time compilation

This sample uses the Driver API to just-in-time compile (JIT) a Kernel from PTX code. Additionally, this sample demonstrates the seamless interoperability capability of the CUDA Runtime and CUDA Driver API calls. For CUDA 5.5, this sample shows how to use cuLink* functions to link PTX assembly using the CUDA driver at runtime.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CUDA Driver API
Supported OSes	Linux, Windows

radixSortThrust - CUDA Radix Sort (Thrust Library)

This sample demonstrates a very fast and efficient parallel radix sort uses Thrust library. The included RadixSort class can sort either key-value pairs (with float or unsigned integer keys) or keys only.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows
Whitepaper	readme.txt

reduction - CUDA Parallel Reduction

A parallel sum reduction that computes the sum of a large arrays of values. This sample demonstrates several important optimization strategies for Data-Parallel Algorithms like reduction using shared memory, __shfl_down_sync and __reduce_add_sync.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CPP11
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows

reductionMultiBlockCG - Reduction using MultiBlock Cooperative Groups

This sample demonstrates single pass reduction using Multi Block Cooperative Groups. This sample requires devices with compute capability 6.0 or higher having compute preemption.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	MBCG, CPP11
Supported SM Architecture	SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Cooperative Groups, MultiBlock Cooperative Groups
Supported OSes	Linux, Windows

scalarProd - Scalar Product

This sample calculates scalar products of a given set of input vector pairs.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra
Supported OSes	Linux, Windows

scan - CUDA Parallel Prefix Sum (Scan)

This example demonstrates an efficient CUDA implementation of parallel prefix sum, also known as "scan". Given an array of numbers, scan computes a new array in which each element is the sum of all the elements before it in the input array.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows

segmentationTreeThrust - CUDA Segmentation Tree Thrust Library

This sample demonstrates an approach to the image segmentation trees construction. This method is based on Boruvka's MST algorithm.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows

shfl_scan - CUDA Parallel Prefix Sum with Shuffle Intrinsics (SHFL_Scan)

This example demonstrates how to use the shuffle intrinsic __shfl_up to perform a scan operation across a thread block. A GPU with Compute Capability SM 3.0. is required to run the sample

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CPP11
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows

simpleHyperQ

This sample demonstrates the use of CUDA streams for concurrent execution of several kernels on devices which provide HyperQ (SM 3.5). Devices without HyperQ (SM 2.0 and SM 3.0) will run a maximum of two kernels concurrently.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CUDA Systems Integration, Performance Strategies
Supported OSes	Linux, Windows
Whitepaper	HyperQ.pdf

sortingNetworks - CUDA Sorting Networks

This sample implements bitonic sort and odd-even merge sort (also known as Batcher's sort), algorithms belonging to the class of sorting networks. While generally subefficient, for large sequences compared to algorithms with better asymptotic algorithmic complexity (i.e. merge sort or radix sort), this may be the preferred algorithms of choice for sorting batches of short-sized to mid-sized (key, value) array pairs. Refer to an excellent tutorial by H. W. Lang http://www.iti.fh-flensburg.de/lang/algorithmen/sortieren/networks/indexen.htm

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Data-Parallel Algorithms
Supported OSes	Linux, Windows

StreamPriorities - Stream Priorities

This sample demonstrates basic use of stream priorities.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	Stream-Priorities
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CUDA Streams and Events
Supported OSes	Linux

threadFenceReduction

This sample shows how to perform a reduction operation on an array of values using the thread Fence intrinsic to produce a single value in a single kernel (as opposed to two or more kernel calls as shown in the "reduction" CUDA Sample). Single-pass reduction requires global atomic instructions (Compute Capability 2.0 or later) and the _threadfence() intrinsic (CUDA 2.2 or later).

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Cooperative Groups, Data-Parallel Algorithms, Performance Strategies
Supported OSes	Linux, Windows

threadMigration - CUDA Context Thread Management

Simple program illustrating how to the CUDA Context Management API and uses the new CUDA 4.0 parameter passing and CUDA launch API. CUDA contexts can be created separately and attached independently to different threads.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cuCtxCreate, cuCtxDestroy, cuModuleLoad, cuModuleLoadDataEx, cuModuleGetFunction, cuLaunchKernel, cuMemcpyDtoH, cuCtxPushCurrent, cuCtxPopCurrent
Key Concepts	CUDA Driver API
Supported OSes	Linux, Windows

transpose - Matrix Transpose

This sample demonstrates Matrix Transpose. Different performance are shown to achieve high performance.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies, Linear Algebra
Supported OSes	Linux, Windows
Whitepaper	MatrixTranspose.pdf

warpAggregatedAtomicsCG - Warp Aggregated Atomics using Cooperative Groups

This sample demonstrates how using Cooperative Groups (CG) to perform warp aggregated atomics to single and multiple counters, a useful technique to improve performance when many threads atomically add to a single or multiple counters.

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Cooperative Groups, Atomic Intrinsics
Supported OSes	Linux, Windows

3.8. Cudalibraries Reference

batchCUBLAS

A CUDA Sample that demonstrates how using batched CUBLAS API calls to improve overall performance.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, CUBLAS Library
Supported OSes	Linux, Windows

batchedLabelMarkersAndLabelCompressionNPP - Batched Label Markers And Label Compression NPP

An NPP CUDA Sample that demonstrates how to use the NPP label markers generation and label compression functions based on a Union Find (UF) algorithm including both single image and batched image versions.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NPP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies, Image Processing, NPP Library, Using NPP Batch Functions
Supported OSes	Linux, Windows

boundSegmentsNPP - Bound Segments NPP

An NPP CUDA Sample that demonstrates using nppiLabelMarkers to generate connected region segment labels in an 8-bit grayscale image then compressing the sparse list of generated labels into the minimum number of uniquely labeled regions in the image using nppiCompressMarkerLabels. Finally a boundary is added surrounding each segmented region in the image using nppiBoundSegments.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux, Windows

boxFilterNPP - Box Filter with NPP

A NPP CUDA Sample that demonstrates how to use NPP FilterBox function to perform a Box Filter.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux, Windows

cannyEdgeDetectorNPP - Canny Edge Detector NPP

An NPP CUDA Sample that demonstrates the recommended parameters to use with the nppiFilterCannyBorder_8u_C1R Canny Edge Detection image filter function. This function expects a single channel 8-bit grayscale input image. You can generate a grayscale image from a color image by first calling nppiColorToGray() or nppiRGBToGray(). The Canny Edge

Detection function combines and improves on the techniques required to produce an edge detection image using multiple steps.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux, Windows

conjugateGradient - ConjugateGradient

This sample implements a conjugate gradient solver on GPU using CUBLAS and CUSPARSE library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS, CUSPARSE
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux, Windows

conjugateGradientCudaGraphs - Conjugate Gradient using Cuda Graphs

This sample implements a conjugate gradient solver on GPU using CUBLAS and CUSPARSE library calls captured and called using CUDA Graph APIs.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS, CUSPARSE

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
CUDA API	cudaStreamBeginCapture, cudaStreamEndCapture, cudaGraphCreate, cudaGraphLaunch, cudaGraphInstantiate, cudaGraphExecDestroy, cudaGraphDestroy
Key Concepts	Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux, Windows

conjugateGradientPrecond - Preconditioned Conjugate Gradient

This sample implements a preconditioned conjugate gradient solver on GPU using CUBLAS and CUSPARSE library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS, CUSPARSE
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux, Windows

conjugateGradientUM - ConjugateGradientUM

This sample implements a conjugate gradient solver on GPU using CUBLAS and CUSPARSE library, using Unified Memory

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	UVM, CUBLAS, CUSPARSE
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Unified Memory, Linear Algebra, CUBLAS Library, CUSPARSE Library
Supported OSes	Linux, Windows

cuHook - CUDA Interception Library

This sample demonstrates how to build and use an intercept library with CUDA. The library has to be loaded via LD_PRELOAD, e.g. LD_PRELOAD=<full_path>/libcuhook.so.1 ./cuHook

Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Supported OSes	Linux

cuSolverDn_LinearSolver - cuSolverDn Linear Solver

A CUDA Sample that demonstrates cuSolverDN's LU, QR and Cholesky factorization.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUSOLVER, CUBLAS, CUSPARSE
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, CUSOLVER Library
Supported OSes	Linux, Windows

cuSolverRf - cuSolverRf Refactorization

A CUDA Sample that demonstrates cuSolver's refactorization library - CUSOLVERRF.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUSOLVER, CUBLAS, CUSPARSE
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, CUSOLVER Library
Supported OSes	Linux, Windows

cuSolverSp_LinearSolver - cuSolverSp Linear Solver

A CUDA Sample that demonstrates cuSolverSP's LU, QR and Cholesky factorization.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUSOLVER, CUSPARSE
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, CUSOLVER Library
Supported OSes	Linux, Windows

cuSolverSp_LowlevelCholesky - cuSolverSp LowlevelCholesky Solver

A CUDA Sample that demonstrates Cholesky factorization using cuSolverSP's low level APIs.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUSOLVER, CUSPARSE
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, CUSOLVER Library
Supported OSes	Linux, Windows

cuSolverSp_LowlevelQR - cuSolverSp Lowlevel QR Solver

A CUDA Sample that demonstrates QR factorization using cuSolverSP's low level APIs.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUSOLVER, CUSPARSE
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Linear Algebra, CUSOLVER Library
Supported OSes	Linux, Windows

FilterBorderControlNPP - Filter Border Control NPP

This sample demonstrates how any border version of an NPP filtering function can be used in the most common mode, with border control enabled. Mentioned functions can be used to duplicate the results of the equivalent non-border version of the NPP functions. They can be also used for enabling and disabling border control on various source image edges depending on what portion of the source image is being used as input.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux, Windows

freeImageInteropNPP - FreeImage and NPP Interopability

A simple CUDA Sample demonstrate how to use FreeImage library with NPP.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux, Windows

histEqualizationNPP - Histogram Equalization with NPP

This CUDA Sample demonstrates how to use NPP for histogram equalization for image data.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	FreeImage, NPP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, Performance Strategies, NPP Library
Supported OSes	Linux, Windows

MC_EstimatePiInlineP - Monte Carlo Estimation of Pi (inline PRNG)

This sample uses Monte Carlo simulation for Estimation of Pi (using inline PRNG). This sample also uses the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux, Windows

MC_EstimatePiInlineQ - Monte Carlo Estimation of Pi (inline QRNG)

This sample uses Monte Carlo simulation for Estimation of Pi (using inline QRNG). This sample also uses the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be

installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux, Windows

MC_EstimatePiP - Monte Carlo Estimation of Pi (batch PRNG)

This sample uses Monte Carlo simulation for Estimation of Pi (using batch PRNG). This sample also uses the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux, Windows

MC_EstimatePiQ - Monte Carlo Estimation of Pi (batch QRNG)

This sample uses Monte Carlo simulation for Estimation of Pi (using batch QRNG). This sample also uses the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Random Number Generator, Computational Finance, CURAND Library

Supported OSes	Linux, Windows

MC_SingleAsianOptionP - Monte Carlo Single Asian Option

This sample uses Monte Carlo to simulate Single Asian Options using the NVIDIA CURAND library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Random Number Generator, Computational Finance, CURAND Library
Supported OSes	Linux, Windows

MersenneTwisterGP11213

This sample demonstrates the Mersenne Twister random number generator GP11213 in cuRAND.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CURAND
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Computational Finance, CURAND Library
Supported OSes	Linux, Windows

nvJPEG - NVJPEG simple

A CUDA Sample that demonstrates single and batched decoding of jpeg images using NVJPEG Library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be

installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVJPEG
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Decoding, NVJPEG Library
Supported OSes	Linux, Windows

nvJPEG_encoder - NVJPEG Encoder

A CUDA Sample that demonstrates single encoding of jpeg images using NVJPEG Library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NVJPEG
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Encoding, NVJPEG Library
Supported OSes	Linux, Windows

randomFog - Random Fog

This sample illustrates pseudo- and quasi- random numbers produced by CURAND.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	X11, GL, CURAND
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	3D Graphics, CURAND Library
Supported OSes	Linux, Windows

simpleCUBLAS - Simple CUBLAS

Example of using CUBLAS using the new CUBLAS API interface available in CUDA 4.0.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, CUBLAS Library
Supported OSes	Linux, Windows

simpleCUBLAS_LU - Simple CUBLAS LU

CUDA sample demonstrating cuBLAS API cublasDgetrfBatched() for lower-upper (LU) decomposition of a matrix.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CUBLAS Library, LU decomposition
Supported OSes	Linux, Windows

simpleCUBLASXT - Simple CUBLAS XT

Example of using CUBLAS-XT library.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUBLAS
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	CUBLAS-XT Library
Supported OSes	Linux, Windows

simpleCUFFT - Simple CUFFT

Example of using CUFFT. In this example, CUFFT is used to compute the 1D-convolution of some signal with some filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain. cuFFT plans are created using simple and advanced API functions.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUFFT
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux, Windows

simpleCUFFT_2d_MGPU - SimpleCUFFT_2d_MGPU

Example of using CUFFT. In this example, CUFFT is used to compute the 2D-convolution of some signal with some filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain on Multiple GPU.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUFFT
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux, Windows

simpleCUFFT_callback - Simple CUFFT Callbacks

Example of using CUFFT. In this example, CUFFT is used to compute the 1D-convolution of some signal with some filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain. The difference between this example and the Simple CUFFT example is that the multiplication step is done by the CUFFT kernel with a user-supplied CUFFT callback routine, rather than by a separate kernel call.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	callback, CUFFT
Supported SM Architecture	SM 3.5, SM 5.0, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux

simpleCUFFT_MGPU - Simple CUFFT_MGPU

Example of using CUFFT. In this example, CUFFT is used to compute the 1D-convolution of some signal with some filter by transforming both into frequency domain, multiplying them together, and transforming the signal back to time domain on Multiple GPU.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	CUFFT
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Image Processing, CUFFT Library
Supported OSes	Linux, Windows

watershedSegmentationNPP - Watershed Segmentation NPP

An NPP CUDA Sample that demonstrates how to use the NPP watershed segmentation function.

This sample depends on other applications or libraries to be present on the system to either build or run. If these dependencies are not available on the system, the sample will not be installed. If these dependencies are available, but not installed, the sample will waive itself at build time.

Dependencies	NPP
Supported SM Architecture	SM 3.5, SM 3.7, SM 5.0, SM 5.2, SM 5.3, SM 6.0, SM 6.1, SM 6.2, SM 7.0, SM 7.2, SM 7.5, SM 8.0, SM 8.6
Key Concepts	Performance Strategies, Image Processing, NPP Library
Supported OSes	Linux, Windows

Chapter 4. Dependencies

Some CUDA Samples rely on third-party applications and/or libraries, or features provided by the CUDA Toolkit and Driver, to either build or execute. These dependencies are listed below.

If a sample has a dependency that is not available on the system, the sample will not be installed. If a sample has a third-party dependency that is available on the system, but is not installed, the sample will waive itself at build time.

Each sample's dependencies are listed in the Samples Reference section.

Third-Party Dependencies

These third-party dependencies are required by some CUDA samples. If available, these dependencies are either installed on your system automatically, or are installable via your system's package manager (Linux) or a third-party website.

FreeImage

FreeImage is an open source imaging library. FreeImage can usually be installed on Linux using your distribution's package manager system. FreeImage can also be downloaded from the FreeImage website.

To set up FreeImage on a Windows system, extract the FreeImage DLL distribution into the `7_CUDALibraries/common/` folder such that `7_CUDALibraries/common/FreeImage/Dist/x64/` contains the `.h`, `.dll`, and `.lib` files.

Message Passing Interface

MPI (Message Passing Interface) is an API for communicating data between distributed processes. A MPI compiler can be installed using your Linux distribution's package manager system. It is also available on some online resources, such as Open MPI. On Windows, to build and run MPI-CUDA applications one can install MS-MPI SDK.

Only 64-Bit

Some samples can only be run on a 64-bit operating system.

DirectX

DirectX is a collection of APIs designed to allow development of multimedia applications on Microsoft platforms. For Microsoft platforms, NVIDIA's CUDA Driver supports DirectX. Several CUDA Samples for Windows demonstrates CUDA-DirectX Interoperability, for building such samples one needs to install Microsoft Visual Studio 2012 or higher which provides Microsoft Windows SDK for Windows 8.

DirectX 12

DirectX 12 is a collection of advanced low-level programming APIs which can reduce driver overhead, designed to allow development of multimedia applications on Microsoft platforms starting with Windows 10 OS onwards. For Microsoft platforms, NVIDIA's CUDA Driver supports DirectX. Few CUDA Samples for Windows demonstrates CUDA-DirectX12 Interoperability, for building such samples one needs to install Windows 10 SDK or higher , with VS 2015 or VS 2017.

OpenGL

OpenGL is a graphics library used for 2D and 3D rendering. On systems which support OpenGL, NVIDIA's OpenGL implementation is provided with the CUDA Driver.

OpenGL ES

OpenGL ES is an embedded systems graphics library used for 2D and 3D rendering. On systems which support OpenGL ES, NVIDIA's OpenGL ES implementation is provided with the CUDA Driver.

Vulkan

Vulkan is a low-overhead, cross-platform 3D graphics and compute API. Vulkan targets high-performance realtime 3D graphics applications such as video games and interactive media across all platforms. On systems which support Vulkan, NVIDIA's Vulkan implementation is provided with the CUDA Driver. For building and running Vulkan applications one needs to install the Vulkan SDK.

OpenMP

OpenMP is an API for multiprocessing programming. OpenMP can be installed using your Linux distribution's package manager system. It usually comes preinstalled with GCC. It can also be found at the OpenMP website.

Screen

Screen is a windowing system found on the QNX operating system. Screen is usually found as part of the root filesystem.

X11

X11 is a windowing system commonly found on *-nix style operating systems. X11 can be installed using your Linux distribution's package manager, and comes preinstalled on Mac OS X systems.

EGL

EGL is an interface between Khronos rendering APIs (such as OpenGL, OpenGL ES or OpenVG) and the underlying native platform windowing system.

EGLOutput

EGLOutput is a set of EGL extensions which allow EGL to render directly to the display.

EGLSync

EGLSync is a set of EGL extensions which provides sync objects that are synchronization primitive, representing events whose completion can be tested or waited upon.

NVSCI

NvSci is a set of communication interface libraries out of which CUDA interops with NvSciBuf and NvSciSync. NvSciBuf allows applications to allocate and exchange buffers in memory. NvSciSync allows applications to manage synchronization objects which coordinate when sequences of operations begin and end.

NvMedia

NvMedia provides powerful processing of multimedia data for true hardware acceleration across NVIDIA Tegra devices. Applications leverage the NvMedia Application Programming Interface (API) to process the image and video data.

CUDA Features

These CUDA features are needed by some CUDA samples. They are provided by either the CUDA Toolkit or CUDA Driver. Some features may not be available on your system.

CUFFT Callback Routines

CUFFT Callback Routines are user-supplied kernel routines that CUFFT will call when loading or storing data. These callback routines are only available on Linux x86_64 and ppc64le systems.

CUDA Dynamic Paralellism

CDP (CUDA Dynamic Paralellism) allows kernels to be launched from threads running on the GPU. CDP is only available on GPUs with SM architecture of 3.5 or above.

Multi-block Cooperative Groups

Multi Block Cooperative Groups(MBCG) extends Cooperative Groups and the CUDA programming model to express inter-thread-block synchronization. MBCG is available on GPUs with Pascal and higher architecture.

Multi-Device Cooperative Groups

Multi Device Cooperative Groups extends Cooperative Groups and the CUDA programming model enabling thread blocks executing on multiple GPUs to cooperate and synchronize as they execute. This feature is available on GPUs with Pascal and higher architecture.

CUBLAS

CUBLAS (CUDA Basic Linear Algebra Subroutines) is a GPU-accelerated version of the BLAS library.

CUDA Interprocess Communication

IPC (Interprocess Communication) allows processes to share device pointers.

CUFFT

CUFFT (CUDA Fast Fourier Transform) is a GPU-accelerated FFT library.

CURAND

CURAND (CUDA Random Number Generation) is a GPU-accelerated RNG library.

CUSPARSE

CUSPARSE (CUDA Sparse Matrix) provides linear algebra subroutines used for sparse matrix calculations.

CUSOLVER

CUSOLVER library is a high-level package based on the CUBLAS and CUSPARSE libraries. It combines three separate libraries under a single umbrella, each of which can be used independently or in concert with other toolkit libraries. The intent ofCUSOLVER is to provide useful LAPACK-like features, such as common matrix factorization and triangular solve routines for dense matrices, a sparse least-squares solver and an eigenvalue solver. In addition cuSolver provides a new refactorization library useful for solving sequences of matrices with a shared sparsity pattern.

NPP

NPP (NVIDIA Performance Primitives) provides GPU-accelerated image, video, and signal processing functions.

NVJPEG

NVJPEG library provides high-performance, GPU accelerated JPEG decoding functionality for image formats commonly used in deep learning and hyperscale multimedia applications.

NVRTC

NVRTC (CUDA RunTime Compilation) is a runtime compilation library for CUDA C++.

Stream Priorities

Stream Priorities allows the creation of streams with specified priorities. Stream Priorities is only available on GPUs with SM architecture of 3.5 or above.

Unified Virtual Memory

UVM (Unified Virtual Memory) enables memory that can be accessed by both the CPU and GPU without explicit copying between the two. UVM is only available on Linux and Windows systems.

16-bit Floating Point

FP16 is a 16-bit floating-point format. One bit is used for the sign, five bits for the exponent, and ten bits for the mantissa. FP16 is only available on specific mobile platforms.

C++11 CUDA

NVCC Support of C++11 features.

Chapter 5. Key Concepts and Associated Samples

The tables below describe the key concepts of the CUDA Toolkit and lists the samples that illustrate how that concept is used.

Basic Key Concepts

Basic Concepts demonstrates how to make use of CUDA features.

Table 2. Basic Key Concepts and Associated Samples

Basic Key Concept	Description	Samples
3D Graphics	*3D Rendering*	Random Fog, Simple Direct3D10 (Vertex Array), Simple OpenGL, Simple OpenGLES, Simple OpenGLES EGLOutput, Simple OpenGLES on Screen
3D Textures	*Volume Textures*	Simple Texture 3D
Arrive Wait Barrier		Simple Arrive Wait Barrier
Assert	*GPU Assert*	simpleAssert, simpleAssert with libNVRTC
Asynchronous Data Transfers	*Overlapping I/O and Compute*	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Multi Copy and Compute, Simple Multi-GPU, Simple Peer-to-Peer Transfers with Multi-GPU, asyncAPI, simpleStreams
Atomic Intrinsics	*Using atomics with GPU kernels*	Simple Atomic Intrinsics, Simple Atomic Intrinsics with libNVRTC, System wide

Basic Key Concept	Description	Samples
		Atomics, Warp Aggregated Atomics using Cooperative Groups
Attributes usage on stream		simpleAttributes
C++ Function Overloading	*Use C++ overloading with GPU kernels*	cppOverload
C++ Templates	*Using Templates with GPU kernels*	Simple Templates, Simple Templates with libNVRTC
CPP11 CUDA	*Samples demonstrating how to use C++11 feature support in CUDA.*	Global Memory to Shared Memory Async Copy
CUBLAS	*CUDA BLAS samples*	Matrix Multiplication (CUBLAS), Unified Memory Streams
CUBLAS Library	*CUDA BLAS samples*	Simple CUBLAS, Simple CUBLAS LU, batchCUBLAS
CUBLAS-XT Library	*cuBLAS XT is a library which further accelerates Level 3 BLAS calls by spreading work across multiple GPUs connected to the same motherboard.*	Simple CUBLAS XT
CUDA Driver API	*Samples that show the CUDA Driver API*	CUDA Compressible Memory, Device Query Driver API, Matrix Multiplication (CUDA Driver API Version), Memmap IPC Driver API, Simple Driver-Runtime Interaction, Simple Texture (Driver Version), Simple Vote Intrinsics with libNVRTC, Using Inline PTX, Using Inline PTX with libNVRTC, Vector Addition Driver API, Vector Addition cuMemMap, Vector Addition with libNVRTC
CUDA Dynamic Parallelism	*Dynamic Parallelism with GPU Kernels (SM 3.5)*	Simple Print (CUDA Dynamic Parallelism)
CUDA Graphs	*Samples demonstrating how to use CUDA Graphs API to create, instantiate and launch CUDA Operations.*	Jacobi CUDA Graphs, Simple CUDA Graphs

Basic Key Concept	Description	Samples
CUDA NvSci Interop	*Samples showing CUDA and NvSciBuf/ NvSciSync Interop.*	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop
CUDA Runtime API	*Samples that use the Runtime API*	Device Query, FP16 Scalar Product, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Matrix Multiplication with libNVRTC, Simple Driver-Runtime Interaction, Simple Texture, Vector Addition
CUDA Streams	*Stream API definies a sequence of operations that can be overlapped with I/O*	Simple CUDA Callbacks
CUDA Streams and Events	*Synchronizing Kernels with Event Timers and Streams*	Bandwidth Test, Simple Multi Copy and Compute, Simple Multi-GPU, Unified Memory Streams, Unified and other CUDA Memories Performance, asyncAPI, cppOverload, simpleStreams
CUDA Systems Integration	*Samples that integrate with Multi Process (OpenMP, IPC, and MPI)*	Unified Memory Streams, Unified and other CUDA Memories Performance, cudaOpenMP, simpleIPC, simpleMPI
CUFFT Library	*Samples that use the CUDA FFT accelerated library*	Simple CUFFT, Simple CUFFT Callbacks, Simple CUFFT_MGPU, SimpleCUFFT_2d_MGPU
CURAND Library	*Samples that use the CUDA random number generator*	MersenneTwisterGP11213, Random Fog
CUSOLVER Library	*Samples that use the cuSOLVER accelerated library*	cuSolverDn Linear Solver , cuSolverRf Refactorization, cuSolverSp Linear Solver , cuSolverSp Lowlevel QR Solver, cuSolverSp LowlevelCholesky Solver
Callback Functions	*Creating Callback functions with GPU kernels*	Simple CUDA Callbacks
Compressible Memory		CUDA Compressible Memory
Computational Finance	*Finance Algorithms*	Black-Scholes Option Pricing, Black-Scholes Option Pricing with libNVRTC, MersenneTwisterGP11213

Basic Key Concept	Description	Samples
Cooperative Groups	*Cooperative Groups is an extension to the CUDA programming model that allows the CUDA program to express the granularity at which different-sized groups of threads are communicating.*	Advanced Quicksort (CUDA Dynamic Parallelism), Binary Partition Cooperative Groups, DirectX Texture Compressor (DXTC), Jacobi CUDA Graphs, Quad Tree (CUDA Dynamic Parallelism), Reduction using MultiBlock Cooperative Groups, Simple Cooperative Groups, Warp Aggregated Atomics using Cooperative Groups, conjugateGradient using MultiBlock Cooperative Groups, conjugateGradient using MultiDevice Cooperative Groups, threadFenceReduction
Data Parallel Algorithms	*Samples that show good usage of Data Parallel Algorithms*	CUDA Separable Convolution, Texture-based Separable Convolution
Debugging	*Samples useful for debugging*	simplePrintf
Device Memory Allocation	*Samples that show GPU Device side memory allocation*	Template
Device Query	*Sample showing simple device query of information*	Device Query, Device Query Driver API
EGLImage-CUDA Interop	*Samples demonstrating how to use EGL Image and CUDA Interop.*	EGLSync CUDA Event Interop
EGLStreams Interop	*Samples demonstrating how to use EGL Streams and CUDA Interop.*	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop
EGLSync-CUDAEvent Interop	*Samples demonstrating interoperability between CUDA Event and EGL Sync for achieving GPU-side synchronization between EGL and CUDA operations without blocking CPU for synchronization.*	EGLSync CUDA Event Interop
GPU Performance	*Samples demonstrating high performance and data I/O*	Simple Multi Copy and Compute
Graphics Interop	*Samples that demonstrate interop between graphics APIs and CUDA*	Bicubic B-spline Interoplation, Bilateral Filter, Box Filter, CUDA and OpenGL

Basic Key Concept	Description	Samples
		Interop of Images, NV12toBGRandResize, Simple D3D10 Texture, Simple D3D11, Simple D3D11 Texture, Simple D3D12 CUDA Interop, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target, Simple Direct3D9 (Vertex Arrays), Simple OpenGL, Simple OpenGLES, Simple OpenGLES EGLOutput, Simple OpenGLES on Screen, Simple Texture 3D
Image Decoding	*Samples demonstrating image decoding on GPU.*	NVJPEG simple
Image Encoding		NVJPEG Encoder
Image Processing	*Samples that demonstrate image processing algorithms in CUDA*	Batched Label Markers And Label Compression NPP, Bicubic B-spline Interoplation, Bilateral Filter, Bound Segments NPP, Box Filter, Box Filter with NPP, CUDA Separable Convolution, CUDA and OpenGL Interop of Images, Canny Edge Detector NPP, Filter Border Control NPP, FreeImage and NPP Interopability, Histogram Equalization with NPP, NV12toBGRandResize, Pitch Linear Texture, Simple CUBLAS, Simple CUFFT, Simple CUFFT Callbacks, Simple CUFFT_MGPU, Simple D3D11, Simple D3D11 Texture, Simple D3D12 CUDA Interop, Simple Surface Write, Simple Texture, Simple Texture (Driver Version), Simple Texture 3D, SimpleCUFFT_2d_MGPU, Texture-based Separable Convolution, Watershed Segmentation NPP
Instantiated CUDA Graph Update		Jacobi CUDA Graphs

Basic Key Concept	Description	Samples
InterProcess Communication	*Samples that demonstrate Inter Process Communication between processes*	simpleIPC
LU decomposition		Simple CUBLAS LU
Linear Algebra	*Samples demonstrating linear algebra with CUDA*	Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Matrix Multiplication with libNVRTC, batchCUBLAS, cuSolverDn Linear Solver , cuSolverRf Refactorization, cuSolverSp Linear Solver , cuSolverSp Lowlevel QR Solver, cuSolverSp LowlevelCholesky Solver
MMAP		CUDA Compressible Memory, Memmap IPC Driver API, Vector Addition cuMemMap, Vulkan CUDA Interop PI Approximation
MPI	*Samples demonstrating how to use CUDA with MPI programs*	simpleMPI
Matrix Multiply	*Samples demonstrating matrix multiply CUDA*	CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Matrix Multiplication (CUDA Driver API Version), Tensor Core GEMM Integer MMA, bfloat16 Tensor Core GEMM, tf32 Tensor Core GEMM
Multi-GPU	*Samples demonstrating how to take advantage of multiple GPUs and CUDA*	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Multi-GPU, Simple Peer-to-Peer Transfers with Multi-GPU, Topology Query
Multithreading	*Samples demonstrating how to use multithreading with CUDA*	Simple CUDA Callbacks, Simple Multi-GPU, Unified Memory Streams, cudaOpenMP, simpleMPI
NPP Library	*Samples demonstrating how to use NPP (NVIDIA Performance Primitives) for image processing*	Batched Label Markers And Label Compression NPP, Bound Segments NPP, Box Filter with NPP, Canny Edge Detector NPP, Filter Border Control NPP, FreeImage and NPP Interopability,

Basic Key Concept	Description	Samples
		Histogram Equalization with NPP, Watershed Segmentation NPP
NVJPEG Library	*NVJPEG library samples*	NVJPEG Encoder, NVJPEG simple
Occupancy Calculator	*Samples demonstrating how to use the CUDA Occupancy Calculator*	simpleOccupancy
OpenMP	*Samples demonstrating how to use OpenMP*	Unified Memory Streams, cudaOpenMP
Overlap Compute and Copy	*Samples demonstrating how to overlap Compute and Data I/O*	Simple Multi Copy and Compute
PTX Assembly	*Samples demonstrating how to use PTX code with CUDA*	Using Inline PTX, Using Inline PTX with libNVRTC
Peer to Peer	*Samples demonstrating how to handle P2P data transfers between multiple GPUs*	simpleIPC
Peer to Peer Data Transfers	*Samples demonstrating how to handle P2P data transfers between multiple GPUs*	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
Performance Strategies	*Samples demonstrating high performance with CUDA*	Bandwidth Test, Batched Label Markers And Label Compression NPP, Bound Segments NPP, Box Filter with NPP, CUDA and OpenGL Interop of Images, Canny Edge Detector NPP, Clock, Clock libNVRTC, Filter Border Control NPP, FreeImage and NPP Interopability, Histogram Equalization with NPP, Matrix Multiplication (CUBLAS), Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU, Topology Query, Using Inline PTX, Using Inline PTX with libNVRTC, Watershed Segmentation NPP, simpleZeroCopy, stream Ordered Allocation, stream Ordered Allocation IPC Pools, stream Ordered Allocation Peer-to-Peer access

Basic Key Concept	Description	Samples
Pinned System Paged Memory	*Samples demonstrating how to properly handle data I/O efficiently between the CPU host and GPU video memory*	Unified and other CUDA Memories Performance, simpleZeroCopy
Separate Compilation	*Samples demonstrating how to use CUDA library linking*	Simple Static GPU Device Library
Stream Capture	*Samples demonstrating how to use Stream Capture API to create CUDA Graphs.*	Jacobi CUDA Graphs, Simple CUDA Graphs
Surface Writes	*Samples demonstrating how to use Surface Writes with GPU kernels*	Simple Surface Write, Simple Texture 3D
Texture	*Samples demonstrating how to use textures GPU kernels*	Pitch Linear Texture, Simple Cubemap Texture, Simple D3D10 Texture, Simple D3D9 Texture, Simple Direct3D10 Render Target, Simple Layered Texture, Simple Surface Write, Simple Texture, Simple Texture (Driver Version), Texture-based Separable Convolution
Unified Memory	*Samples demonstrating how to use Unified Memory*	ConjugateGradientUM, System wide Atomics, Unified Memory Streams, Unified and other CUDA Memories Performance, conjugateGradient using MultiBlock Cooperative Groups, conjugateGradient using MultiDevice Cooperative Groups
Unified Virtual Address Space	*Samples demonstrating how to use UVA with CUDA programs*	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
Using NPP Batch Functions		Batched Label Markers And Label Compression NPP
Vector Addition	*Samples demonstrating how to use Vector Addition with CUDA programs*	Simple Driver-Runtime Interaction, Vector Addition, Vector Addition Driver API, Vector Addition cuMemMap, Vector Addition with libNVRTC, simpleZeroCopy

Basic Key Concept	Description	Samples
Vertex Buffers	*Samples demonstrating how to use Vertex Buffers with CUDA kernels*	Simple OpenGL, Simple OpenGLES, Simple OpenGLES EGLOutput, Simple OpenGLES on Screen
Video Processing		NV12toBGRandResize
Volume Processing	*Samples demonstrating how to use 3D Textures for volume rendering*	Simple Cubemap Texture, Simple Layered Texture
Vote Intrinsics	*Samples demonstrating how to use vote intrinsics with CUDA*	Simple Vote Intrinsics, Simple Vote Intrinsics with libNVRTC
cuMemMap IPC		Memmap IPC Driver API, Vulkan CUDA Interop PI Approximation

Advanced Key Concepts

Advanced Concepts demonstrate advanced techniques and algorithms implemented with CUDA.

Table 3. Advanced Key Concepts and Associated Samples

Advanced Key Concept	Description	Samples
2D Textures	*Texture Mapping*	SLI D3D10 Texture
3D Graphics	*3D Rendering*	Marching Cubes Isosurfaces
3D Textures	*Volume Textures*	Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
CPP11 CUDA	*Samples demonstrating how to use C++11 feature support in CUDA.*	C++11 CUDA
CUBLAS Library	*CUDA BLAS samples*	Conjugate Gradient using Cuda Graphs, ConjugateGradient, ConjugateGradientUM, Preconditioned Conjugate Gradient
CUDA DX12 Interop	*Samples showing CUDA and D3D12 interop.*	Simple D3D12 CUDA Interop

Advanced Key Concept	Description	Samples
CUDA Driver API	*Samples that show the CUDA Driver API*	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), PTX Just-in-Time compilation
CUDA Dynamic Parallelism	*Dynamic Parallelism with GPU Kernels (SM 3.5)*	Advanced Quicksort (CUDA Dynamic Parallelism), Bezier Line Tessellation (CUDA Dynamic Parallelism), Quad Tree (CUDA Dynamic Parallelism), Simple Quicksort (CUDA Dynamic Parallelism)
CUDA Dynamically Linked Library	*Dynamic loading of the CUDA DLL using CUDA Driver API*	Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version)
CUDA Streams and Events	*Synchronizing Kernels with Event Timers and Streams*	Stream Priorities
CUDA Systems Integration	*Samples that integrate with Multi Process (OpenMP, IPC, and MPI)*	simpleHyperQ
CUDA Vulkan Interop	*Samples demonstrating how to use Vulkan - CUDA Interop.*	Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
CUFFT Library	*Samples that use the CUDA FFT accelerated library*	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version)
CURAND Library	*Samples that use the CUDA random number generator*	Monte Carlo Estimation of Pi (batch PRNG), Monte Carlo Estimation of Pi (batch QRNG), Monte Carlo Estimation of Pi (inline PRNG), Monte Carlo Estimation of Pi (inline QRNG) , Monte Carlo Single Asian Option
CUSPARSE Library	*Samples that use the cuSPARSE (Sparse Vector Matrix Multiply) functions*	Conjugate Gradient using Cuda Graphs, ConjugateGradient, ConjugateGradientUM, Preconditioned Conjugate Gradient

Advanced Key Concept	Description	Samples
Computational Finance	*Finance Algorithms*	Binomial Option Pricing, Binomial Option Pricing with libNVRTC, Monte Carlo Estimation of Pi (batch PRNG), Monte Carlo Estimation of Pi (batch QRNG), Monte Carlo Estimation of Pi (inline PRNG), Monte Carlo Estimation of Pi (inline QRNG) , Monte Carlo Single Asian Option, Niederreiter Quasirandom Sequence Generator, Niederreiter Quasirandom Sequence Generator with libNVRTC, Sobol Quasirandom Number Generator
Data Parallel Algorithms	*Samples that show good usage of Data Parallel Algorithms*	CUDA Histogram, CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, CUDA NvSciBuf/NvSciSync Interop, Mandelbrot, NvMedia CUDA Interop, Optical Flow, Particles, Smoke Particles, VFlockingD3D10, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
Data-Parallel Algorithms	*Samples that show good usage of Data Parallel Algorithms*	CUDA Parallel Prefix Sum (Scan), CUDA Parallel Prefix Sum with Shuffle Intrinsics (SHFL_Scan), CUDA Parallel Reduction, CUDA Radix Sort (Thrust Library), CUDA Segmentation Tree Thrust Library, CUDA Sorting Networks, Fast Walsh Transform, Merge Sort, threadFenceReduction
Graphics Interop	*Samples that demonstrate interop between graphics APIs and CUDA*	Bindless Texture, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version), Function Pointers, Mandelbrot, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, SLI D3D10 Texture, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with

Advanced Key Concept	Description	Samples
		3D Textures, Volumetric Filtering with 3D Textures and Surface Writes, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
Image Compression	*Samples that demonstrate image and video compression*	DirectX Texture Compressor (DXTC)
Image Processing	*Samples that demonstrate image processing algorithms in CUDA*	1D Discrete Haar Wavelet Decomposition, CUDA FFT Ocean Simulation, CUDA Histogram, CUDA NvSciBuf/NvSciSync Interop, DCT8x8, DirectX Texture Compressor (DXTC), FFT-Based 2D Convolution, Function Pointers, Image denoising, NvMedia CUDA Interop, Optical Flow, Post-Process in OpenGL, Recursive Gaussian Filter, SLI D3D10 Texture, Sobel Filter, Stereo Disparity Computation (SAD SIMD Intrinsics), Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
Linear Algebra	*Samples demonstrating linear algebra with CUDA*	Conjugate Gradient using Cuda Graphs, ConjugateGradient, ConjugateGradientUM, Eigenvalues, Fast Walsh Transform, Matrix Transpose, Preconditioned Conjugate Gradient, Scalar Product, conjugateGradient using MultiBlock Cooperative Groups, conjugateGradient using MultiDevice Cooperative Groups
MultiBlock Cooperative Groups	*Multi Block Cooperative Groups enables to express inter-thread-block synchronization.*	Reduction using MultiBlock Cooperative Groups, conjugateGradient using MultiBlock Cooperative Groups
MultiDevice Cooperative Groups	*Multi Device Cooperative Groups enables thread blocks executing on multiple GPUs to cooperate and synchronize as they execute.*	conjugateGradient using MultiDevice Cooperative Groups
OpenGL Graphics Interop	*Samples demonstrating how to use interoperability CUDA with OpenGL*	Marching Cubes Isosurfaces

Advanced Key Concept	Description	Samples
Performance Strategies	*Samples demonstrating high performance with CUDA*	Aligned Types, CUDA C 3D FDTD, CUDA Parallel Prefix Sum (Scan), CUDA Parallel Prefix Sum with Shuffle Intrinsics (SHFL_Scan), CUDA Parallel Reduction, CUDA Radix Sort (Thrust Library), CUDA Segmentation Tree Thrust Library, Concurrent Kernels, Matrix Transpose, Particles, SLI D3D10 Texture, VFlockingD3D10, simpleHyperQ, threadFenceReduction
Physically Based Simulation	*Samples demonstrating high performance collisions and/or physocal interactions*	Marching Cubes Isosurfaces
Physically-Based Simulation	*Samples demonstrating high performance collisions and/or physocal interactions*	CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version), Particles, Smoke Particles, VFlockingD3D10
Random Number Generator	*Samples demonstrating how to use random number generation with CUDA*	Monte Carlo Estimation of Pi (batch PRNG), Monte Carlo Estimation of Pi (batch QRNG), Monte Carlo Estimation of Pi (inline PRNG), Monte Carlo Estimation of Pi (inline QRNG) , Monte Carlo Single Asian Option
Recursion	*Samples demonstrating recursion on CUDA*	Interval Computing
Runtime Compilation	*Samples demonstrating how to use NVRTC APIs for runtime compilation of CUDA Kernels*	Binomial Option Pricing with libNVRTC, Black-Scholes Option Pricing with libNVRTC, Clock libNVRTC, Matrix Multiplication with libNVRTC, Niederreiter Quasirandom Sequence Generator with libNVRTC, Simple Atomic Intrinsics with libNVRTC, Simple Templates with libNVRTC, Simple Vote Intrinsics with libNVRTC, Using Inline PTX with libNVRTC, Vector Addition with libNVRTC, simpleAssert with libNVRTC

Advanced Key Concept	Description	Samples
Surface Writes	*Samples demonstrating how to use Surface Writes with GPU kernels*	Volumetric Filtering with 3D Textures and Surface Writes
Templates	*Samples demonstrating how to use templates GPU kernels*	Interval Computing
Tensor Cores	*Samples demonstrating use of Tensor Cores, introduced in the Volta chip family. Useful for faster matrix operations.*	CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Tensor Core GEMM Integer MMA, bfloat16 Tensor Core GEMM, tf32 Tensor Core GEMM
Texture	*Samples demonstrating how to use textures GPU kernels*	Bindless Texture
Vertex Buffers	*Samples demonstrating how to use Vertex Buffers with CUDA kernels*	Marching Cubes Isosurfaces
Video Compression	*Samples demonstrating how to use video compression with CUDA*	1D Discrete Haar Wavelet Decomposition, DCT8x8, Fast Walsh Transform
Video Intrinsics	*Samples demonstrating how to use video intrinsics with CUDA*	Stereo Disparity Computation (SAD SIMD Intrinsics)
WMMA	*Samples demonstrating how to use Warp Matrix Multiply and Accumulate (WMMA) CUDA APIs.*	CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Tensor Core GEMM Integer MMA, bfloat16 Tensor Core GEMM, tf32 Tensor Core GEMM

Chapter 6. CUDA API and Associated Samples

The tables below list the samples associated with each CUDA API.

CUDA Driver API Samples

The table below lists the samples associated with each CUDA Driver API.

Table 4. CUDA Driver API and Associated Samples

CUDA Driver API	Samples
cuArrayCreate	Simple Texture (Driver Version)
cuArrayDestroy	Simple Texture (Driver Version)
cuCtxCreate	CUDA Context Thread Management, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop, Memmap IPC Driver API
cuCtxDestroy	CUDA Context Thread Management, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop, Memmap IPC Driver API
cuCtxDetach	Simple Texture (Driver Version)
cuCtxPopCurrent	CUDA Context Thread Management, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop
cuCtxPushCurrent	CUDA Context Thread Management, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop
cuCtxSetCurrent	Memmap IPC Driver API, Vulkan CUDA Interop PI Approximation

CUDA Driver API	Samples
cuCtxSynchronize	Simple Texture (Driver Version)
cuDeviceCanAccessPeer	Memmap IPC Driver API, Vector Addition cuMemMap
cuDeviceComputeCapability	Device Query Driver API, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop
cuDeviceGet	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop
cuDeviceGetAttribute	CUDA Compressible Memory, Device Query Driver API, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop, Memmap IPC Driver API, Vector Addition cuMemMap, Vulkan CUDA Interop PI Approximation
cuDeviceGetCount	Device Query Driver API, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop
cuDeviceGetName	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop
cuDeviceTotalMem	Device Query Driver API
cuDriverGetVersion	Device Query Driver API
cuEGLStreamConsumerAcquireFrame	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop
cuEGLStreamConsumerReleaseFrame	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop
cuEGLStreamProducerPresentFrame	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop
cuEGLStreamProducerReturnFrame	EGLStream_CUDA_CrossGPU
cuGraphicsResourceGetMappedEglFrame	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop
cuInit	Device Query Driver API
cuLaunchKernel	CUDA Context Thread Management, Clock libNVRTC, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Matrix Multiplication with libNVRTC, Memmap IPC Driver API, Simple Atomic Intrinsics with libNVRTC, Simple Driver-Runtime Interaction, Simple Texture (Driver Version), Using Inline PTX with libNVRTC, Vector Addition Driver API, Vector Addition cuMemMap, simpleAssert with libNVRTC

CUDA Driver API	Samples
cuMemAddressFree	CUDA Compressible Memory, Memmap IPC Driver API, Vector Addition cuMemMap, Vulkan CUDA Interop PI Approximation
cuMemAddressReserve	CUDA Compressible Memory, Memmap IPC Driver API, Vector Addition cuMemMap, Vulkan CUDA Interop PI Approximation
cuMemAlloc	CUDA Compressible Memory, Clock libNVRTC, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Matrix Multiplication with libNVRTC, Simple Atomic Intrinsics with libNVRTC, Simple Texture (Driver Version), Simple Vote Intrinsics with libNVRTC, Using Inline PTX with libNVRTC, Vector Addition Driver API, Vector Addition cuMemMap, Vector Addition with libNVRTC
cuMemCreate	CUDA Compressible Memory, Memmap IPC Driver API, Vector Addition cuMemMap, Vulkan CUDA Interop PI Approximation
cuMemExportToShareableHandle	Memmap IPC Driver API, Vulkan CUDA Interop PI Approximation
cuMemFree	CUDA Compressible Memory, Clock libNVRTC, EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Matrix Multiplication with libNVRTC, Simple Atomic Intrinsics with libNVRTC, Simple Texture (Driver Version), Simple Vote Intrinsics with libNVRTC, Vector Addition Driver API, Vector Addition cuMemMap, Vector Addition with libNVRTC
cuMemGetAllocationGranularity	CUDA Compressible Memory, Memmap IPC Driver API, Vector Addition cuMemMap
cuMemImportFromShareableHandle	Memmap IPC Driver API, Vulkan CUDA Interop PI Approximation
cuMemMap	CUDA Compressible Memory, Memmap IPC Driver API, Vector Addition cuMemMap, Vulkan CUDA Interop PI Approximation
cuMemRelease	Memmap IPC Driver API, Vulkan CUDA Interop PI Approximation

CUDA Driver API	Samples
cuMemSetAccess	CUDA Compressible Memory, Memmap IPC Driver API, Vector Addition cuMemMap, Vulkan CUDA Interop PI Approximation
cuMemUnmap	CUDA Compressible Memory, Memmap IPC Driver API, Vector Addition cuMemMap, Vulkan CUDA Interop PI Approximation
cuMemcpy2D	Simple Texture (Driver Version)
cuMemcpy3D	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop
cuMemcpyDtoH	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Matrix Multiplication with libNVRTC, Simple Texture (Driver Version), Using Inline PTX with libNVRTC, Vector Addition Driver API, Vector Addition cuMemMap, Vector Addition with libNVRTC
cuMemcpyDtoHAsync	Memmap IPC Driver API
cuMemcpyHtoD	Clock libNVRTC, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Matrix Multiplication with libNVRTC, Simple Atomic Intrinsics with libNVRTC, Simple Vote Intrinsics with libNVRTC, Vector Addition Driver API, Vector Addition cuMemMap, Vector Addition with libNVRTC
cuModuleGetFunction	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Matrix Multiplication with libNVRTC, Memmap IPC Driver API, Simple Driver-Runtime Interaction, Simple Texture (Driver Version), Vector Addition Driver API, Vector Addition cuMemMap
cuModuleGetTexRef	Simple Texture (Driver Version)
cuModuleLoad	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Matrix Multiplication with libNVRTC, Memmap IPC Driver API, Simple Texture (Driver Version), Vector Addition Driver API, Vector Addition cuMemMap

CUDA Driver API	Samples
cuModuleLoadData	Simple Driver-Runtime Interaction
cuModuleLoadDataEx	CUDA Context Thread Management, Matrix Multiplication (CUDA Driver API Version), Matrix Multiplication (CUDA Driver API version with Dynamic Linking Version), Matrix Multiplication with libNVRTC, Memmap IPC Driver API, Simple Texture (Driver Version), Vector Addition Driver API, Vector Addition cuMemMap
cuOccupancyMaxActiveBlocksPerMultiprocessor	Memmap IPC Driver API
cuParamSetTexRef	Simple Texture (Driver Version)
cuStreamCreate	EGLStream_CUDA_CrossGPU, EGLStreams CUDA Interop, EGLSync CUDA Event Interop, Memmap IPC Driver API
cuStreamDestroy	Memmap IPC Driver API
cuStreamSynchronize	Memmap IPC Driver API
cuTexRefSetAddressMode	Simple Texture (Driver Version)
cuTexRefSetArray	Simple Texture (Driver Version)
cuTexRefSetFilterMode	Simple Texture (Driver Version)
cuTexRefSetFlags	Simple Texture (Driver Version)
cuTexRefSetFormat	Simple Texture (Driver Version)

CUDA Runtime API Samples

The table below lists the samples associated with each CUDA Runtime API.

Table 5. CUDA Runtime API and Associated Samples

CUDA Runtime API	Samples
cublasCreate	Matrix Multiplication (CUBLAS)
cublasSgemm	Matrix Multiplication (CUBLAS)
cudaBindSurfaceToArray	Simple Surface Write
cudaBindTexture2D	Pitch Linear Texture

CUDA Runtime API	Samples
cudaBindTextureToArray	Pitch Linear Texture, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture
cudaCreateChannelDesc	Pitch Linear Texture, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture
cudaCtxResetPersistingL2Cache	simpleAttributes
cudaD3D10GetDevice	SLI D3D10 Texture, Simple D3D10 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaD3D10SetDirect3DDevice	SLI D3D10 Texture, Simple D3D10 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaD3D10SetGLDevice	VFlockingD3D10
cudaD3D11GetDevice	Simple D3D11, Simple D3D11 Texture
cudaD3D11SetDirect3DDevice	Simple D3D11 Texture
cudaD3D9GetDevice	Simple D3D9 Texture, Simple Direct3D9 (Vertex Arrays)
cudaD3D9SetDirect3DDevice	Simple D3D9 Texture, Simple Direct3D9 (Vertex Arrays)
cudaD3D9SetGLDevice	Fluids (Direct3D Version)
cudaDestroyExternalMemory	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop, Simple D3D12 CUDA Interop, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
cudaDestroyExternalSemaphore	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop, Simple D3D12 CUDA Interop, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
cudaDeviceCanAccessPeer	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceDisablePeerAccess	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceEnablePeerAccess	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
cudaDeviceGetDefaultMemPool	stream Ordered Allocation, stream Ordered Allocation Peer-to-Peer access
cudaDeviceGetP2PAttribute	Topology Query
cudaDeviceSetLimit	simpleAttributes

CUDA Runtime API	Samples
cudaDeviceSynchronize	Bandwidth Test, CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Template, Tensor Core GEMM Integer MMA, bfloat16 Tensor Core GEMM, tf32 Tensor Core GEMM
cudaDriverGetVersion	Device Query
cudaEventCreate	Bandwidth Test, CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Tensor Core GEMM Integer MMA, Vector Addition, asyncAPI, bfloat16 Tensor Core GEMM, simpleStreams, simpleZeroCopy, tf32 Tensor Core GEMM
cudaEventCreateWithFlags	Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Peer-to-Peer Transfers with Multi-GPU
cudaEventDestroy	Bandwidth Test, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Vector Addition, asyncAPI, simpleStreams, simpleZeroCopy
cudaEventElapsedTime	Bandwidth Test, CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Multi Copy and Compute, Simple Multi-GPU, Simple Peer-to-Peer Transfers with Multi-GPU, Tensor Core GEMM Integer MMA, Vector Addition, asyncAPI, bfloat16 Tensor Core GEMM, simpleStreams, simpleZeroCopy, tf32 Tensor Core GEMM
cudaEventQuery	Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Vector Addition, asyncAPI, simpleStreams, simpleZeroCopy
cudaEventRecord	Bandwidth Test, CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Simple Multi Copy and Compute, Simple Multi-GPU, Tensor Core GEMM

CUDA Runtime API	Samples
	Integer MMA, Vector Addition, asyncAPI, bfloat16 Tensor Core GEMM, simpleStreams, simpleZeroCopy, tf32 Tensor Core GEMM
cudaEventSynchronize	CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUDA Runtime API Version), Tensor Core GEMM Integer MMA, Vector Addition, bfloat16 Tensor Core GEMM, tf32 Tensor Core GEMM
cudaExternalMemoryGetMappedBuffer	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop, Simple D3D11, Simple D3D12 CUDA Interop, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave
cudaExternalMemoryGetMappedMipmappedArray	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop, Vulkan Image - CUDA Interop
cudaFree	Bandwidth Test, C++ Integration, CUDA Compressible Memory, CUDA Tensor Core GEMM, Clock, Double Precision Tensor Core GEMM, FP16 Scalar Product, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Pitch Linear Texture, Simple Arrive Wait Barrier, Simple Atomic Intrinsics, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture, Simple Vote Intrinsics, System wide Atomics, Template, Tensor Core GEMM Integer MMA, Using Inline PTX, Vector Addition, bfloat16 Tensor Core GEMM, cudaOpenMP, simpleAssert, simpleAttributes, simpleMPI, tf32 Tensor Core GEMM
cudaFreeArray	Pitch Linear Texture, Simple Cubemap Texture, Simple Layered Texture, Simple Surface Write, Simple Texture
cudaFreeAsync	stream Ordered Allocation, stream Ordered Allocation IPC Pools, stream Ordered Allocation Peer-to-Peer access
cudaFreeHost	Bandwidth Test, FP16 Scalar Product, Simple Atomic Intrinsics, Simple Vote Intrinsics, System wide Atomics, Using Inline PTX, simpleAssert, simpleZeroCopy
cudaFuncGetAttributes	cppOverload
cudaFuncSetAttribute	CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, Tensor Core GEMM Integer MMA, bfloat16 Tensor Core GEMM, tf32 Tensor Core GEMM
cudaFuncSetCacheConfig	cppOverload
cudaGetDeviceAttribute	Topology Query

CUDA Runtime API	Samples
cudaGetDeviceCount	Device Query, Topology Query
cudaGetDeviceProperties	Device Query, Vulkan CUDA Interop PI Approximation, simpleAttributes
cudaGraphAddHostNode	Simple CUDA Graphs
cudaGraphAddKernelNode	Simple CUDA Graphs
cudaGraphAddMemcpyNode	Simple CUDA Graphs
cudaGraphAddMemsetNode	Simple CUDA Graphs
cudaGraphCreate	Conjugate Gradient using Cuda Graphs, Jacobi CUDA Graphs, Simple CUDA Graphs
cudaGraphDestroy	Conjugate Gradient using Cuda Graphs, Jacobi CUDA Graphs, Simple CUDA Graphs
cudaGraphExecDestroy	Conjugate Gradient using Cuda Graphs, Jacobi CUDA Graphs, Simple CUDA Graphs
cudaGraphExecKernelNodeSetParams	Jacobi CUDA Graphs
cudaGraphExecUpdate	Jacobi CUDA Graphs
cudaGraphGetNodes	Jacobi CUDA Graphs
cudaGraphInstantiate	Conjugate Gradient using Cuda Graphs, Jacobi CUDA Graphs, Simple CUDA Graphs
cudaGraphLaunch	Conjugate Gradient using Cuda Graphs, Jacobi CUDA Graphs, Simple CUDA Graphs
cudaGraphNodeGetType	Jacobi CUDA Graphs
cudaGraphicsD3D10RegisterResource	SLI D3D10 Texture, Simple D3D10 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaGraphicsD3D11RegisterResource	Simple D3D11 Texture
cudaGraphicsD3D9RegisterResource	Simple D3D9 Texture, Simple Direct3D9 (Vertex Arrays)
cudaGraphicsGLRegisterBuffer	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, CUDA and OpenGL Interop of Images, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple OpenGLES, Simple

CUDA Runtime API	Samples
	OpenGLES EGLOutput, Simple OpenGLES on Screen, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsMapResources	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, CUDA and OpenGL Interop of Images, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple OpenGLES, Simple OpenGLES EGLOutput, Simple OpenGLES on Screen, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsRegisterResource	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, CUDA and OpenGL Interop of Images, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple OpenGLES, Simple OpenGLES EGLOutput, Simple OpenGLES on Screen, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsResourceGetMappedPointer	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, CUDA and OpenGL Interop of Images, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple OpenGLES, Simple OpenGLES EGLOutput, Simple OpenGLES on Screen, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes

CUDA Runtime API	Samples
cudaGraphicsResourceSetMapFlags	SLI D3D10 Texture, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaGraphicsSubResourceGetMappedArray	SLI D3D10 Texture, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaGraphicsUnmapResources	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, CUDA and OpenGL Interop of Images, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, Simple OpenGL, Simple OpenGLES, Simple OpenGLES EGLOutput, Simple OpenGLES on Screen, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaGraphicsUnregisterResource	Bicubic B-spline Interoplation, Bilateral Filter, Bindless Texture, Box Filter, CUDA FFT Ocean Simulation, CUDA N-Body Simulation, CUDA N-Body Simulation on Screen, CUDA N-Body Simulation with GLES, CUDA and OpenGL Interop of Images, Fluids (Direct3D Version), Fluids (OpenGL Version), Fluids (OpenGLES Version), Mandelbrot, Marching Cubes Isosurfaces, Particles, Post-Process in OpenGL, Recursive Gaussian Filter, SLI D3D10 Texture, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target, Simple Direct3D9 (Vertex Arrays), Simple OpenGL, Simple OpenGLES, Simple OpenGLES EGLOutput, Simple OpenGLES on Screen, Simple Texture 3D, Smoke Particles, Sobel Filter, VFlockingD3D10, Volume Rendering with 3D Textures, Volumetric Filtering with 3D Textures and Surface Writes
cudaHostAlloc	Bandwidth Test, simpleZeroCopy
cudaHostGetDevicePointer	simpleZeroCopy
cudaHostRegister	simpleZeroCopy
cudaHostUnregister	simpleZeroCopy

CUDA Runtime API	Samples
cudaImportExternalMemory	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop, Simple D3D11, Simple D3D12 CUDA Interop, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
cudaImportExternalSemaphore	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop, Simple D3D11, Simple D3D12 CUDA Interop, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
cudaIpcCloseMemHandle	simpleIPC
cudaIpcGetEventHandle	simpleIPC
cudaIpcOpenMemHandle	simpleIPC
cudaLaunchCooperativeKernelMultiDevice	conjugateGradient using MultiDevice Cooperative Groups
cudaLaunchHostFunc	Simple CUDA Graphs
cudaMallco	Simple Vote Intrinsics, simpleMPI
cudaMalloc	C++ Integration, CUDA Compressible Memory, Clock, FP16 Scalar Product, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Pitch Linear Texture, Simple Arrive Wait Barrier, Simple Atomic Intrinsics, Simple Cubemap Texture, Simple Driver-Runtime Interaction, Simple Layered Texture, Simple Surface Write, Simple Texture, System wide Atomics, Template, Unified and other CUDA Memories Performance, Using Inline PTX, Vector Addition, cudaOpenMP, simpleAssert, simpleAttributes, tf32 Tensor Core GEMM
cudaMalloc3DArray	Simple Cubemap Texture, Simple Layered Texture
cudaMallocArray	Pitch Linear Texture, Simple Surface Write, Simple Texture
cudaMallocAsync	stream Ordered Allocation, stream Ordered Allocation IPC Pools, stream Ordered Allocation Peer-to-Peer access
cudaMallocHost	Bandwidth Test, FP16 Scalar Product, Unified and other CUDA Memories Performance, Using Inline PTX, simpleAssert
cudaMallocManaged	CUDA Tensor Core GEMM, Double Precision Tensor Core GEMM, NV12toBGRandResize, Tensor Core GEMM Integer

CUDA Runtime API	Samples
	MMA, Unified Memory Streams, Unified and other CUDA Memories Performance, bfloat16 Tensor Core GEMM
cudaMallocPitch	Pitch Linear Texture
cudaMemAdvise	conjugateGradient using MultiDevice Cooperative Groups
cudaMemPoolCreate	stream Ordered Allocation IPC Pools
cudaMemPoolDestroy	stream Ordered Allocation IPC Pools
cudaMemPoolExportPointer	stream Ordered Allocation IPC Pools
cudaMemPoolExportToShareableHandle	stream Ordered Allocation IPC Pools
cudaMemPoolGetAccess	stream Ordered Allocation IPC Pools
cudaMemPoolImportPointer	stream Ordered Allocation IPC Pools
cudaMemPoolSetAccess	stream Ordered Allocation IPC Pools, stream Ordered Allocation Peer-to-Peer access
cudaMemPoolSetAttribute	stream Ordered Allocation
cudaMemPrefetchAsync	conjugateGradient using MultiDevice Cooperative Groups
cudaMemcpy	Bandwidth Test, C++ Integration, Clock, FP16 Scalar Product, Global Memory to Shared Memory Async Copy, Matrix Multiplication (CUBLAS), Matrix Multiplication (CUDA Runtime API Version), Peer-to-Peer Bandwidth Latency Test with Multi-GPUs, Simple Atomic Intrinsics, Simple Cubemap Texture, Simple Driver-Runtime Interaction, Simple Layered Texture, Simple Peer-to-Peer Transfers with Multi-GPU, Simple Surface Write, Simple Texture, Simple Vote Intrinsics, System wide Atomics, Template, Using Inline PTX, Vector Addition, cudaOpenMP, simpleAssert, simpleAttributes, simpleMPI
cudaMemcpy2D	NV12toBGRandResize, Pitch Linear Texture
cudaMemcpy2DToArray	SLI D3D10 Texture, Simple D3D10 Texture, Simple D3D11 Texture, Simple D3D9 Texture, Simple Direct3D10 (Vertex Array), Simple Direct3D10 Render Target
cudaMemcpy3D	Simple Cubemap Texture, Simple D3D9 Texture, Simple Layered Texture
cudaMemcpyAsync	Bandwidth Test, Simple Arrive Wait Barrier, Simple CUDA Callbacks, Simple Multi Copy and Compute, Simple Multi-

CUDA Runtime API	Samples
	GPU, Unified and other CUDA Memories Performance, asyncAPI, simpleIPC, simpleStreams
cudaMemcpyToArray	Pitch Linear Texture, Simple Texture
cudaMemset2D	Pitch Linear Texture
cudaOccupancyMaxActiveBlocksPerMultiprocessor	conjugateGradient using MultiDevice Cooperative Groups
cudaRuntimeGetVersion	Device Query
cudaSetDevice	Bandwidth Test, Device Query
cudaSignalExternalSemaphoresAsync	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop, Simple D3D12 CUDA Interop, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
cudaStreamAddCallback	Simple CUDA Callbacks
cudaStreamAttachManagedMem	Unified Memory Streams
cudaStreamAttachMemAsync	Unified and other CUDA Memories Performance
cudaStreamBeginCapture	Conjugate Gradient using Cuda Graphs, Jacobi CUDA Graphs, Simple CUDA Graphs
cudaStreamCreate	Simple CUDA Callbacks, simpleAttributes
cudaStreamCreateWithFlags	Simple Driver-Runtime Interaction
cudaStreamDestroy	Simple CUDA Callbacks
cudaStreamEndCapture	Conjugate Gradient using Cuda Graphs, Jacobi CUDA Graphs, Simple CUDA Graphs
cudaStreamSetAttribute	simpleAttributes
cudaStreamSynchronize	conjugateGradient using MultiDevice Cooperative Groups
cudaUnbindTexture	Pitch Linear Texture
cudaWaitExternalSemaphoresAsync	CUDA NvSciBuf/NvSciSync Interop, NvMedia CUDA Interop, Simple D3D12 CUDA Interop, Vulkan CUDA Interop PI Approximation, Vulkan CUDA Interop Sinewave, Vulkan Image - CUDA Interop
cufftDestroy	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution
cufftExecC2R	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution

CUDA Runtime API	Samples
cufftExecR2C	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution
cufftPlan2d	CUDA FFT Ocean Simulation, FFT-Based 2D Convolution

Chapter 7. Frequently Asked Questions

Answers to frequently asked questions about CUDA can be found at http://developer.nvidia.com/cuda-faq and in the CUDA Toolkit Release Notes.

Notice

This document is provided for information purposes only and shall not be regarded as a warranty of a certain functionality, condition, or quality of a product. NVIDIA Corporation ("NVIDIA") makes no representations or warranties, expressed or implied, as to the accuracy or completeness of the information contained in this document and assumes no responsibility for any errors contained herein. NVIDIA shall have no liability for the consequences or use of such information or for any infringement of patents or other rights of third parties that may result from its use. This document is not a commitment to develop, release, or deliver any Material (defined below), code, or functionality.

NVIDIA reserves the right to make corrections, modifications, enhancements, improvements, and any other changes to this document, at any time without notice.

Customer should obtain the latest relevant information before placing orders and should verify that such information is current and complete.

NVIDIA products are sold subject to the NVIDIA standard terms and conditions of sale supplied at the time of order acknowledgement, unless otherwise agreed in an individual sales agreement signed by authorized representatives of NVIDIA and customer ("Terms of Sale"). NVIDIA hereby expressly objects to applying any customer general terms and conditions with regards to the purchase of the NVIDIA product referenced in this document. No contractual obligations are formed either directly or indirectly by this document.

VESA DisplayPort

DisplayPort and DisplayPort Compliance Logo, DisplayPort Compliance Logo for Dual-mode Sources, and DisplayPort Compliance Logo for Active Cables are trademarks owned by the Video Electronics Standards Association in the United States and other countries.

HDMI

HDMI, the HDMI logo, and High-Definition Multimedia Interface are trademarks or registered trademarks of HDMI Licensing LLC.

OpenCL

OpenCL is a trademark of Apple Inc. used under license to the Khronos Group Inc.

Trademarks

NVIDIA and the NVIDIA logo are trademarks or registered trademarks of NVIDIA Corporation in the U.S. and other countries. Other company and product names may be trademarks of the respective companies with which they are associated.

Copyright

© 2007-2021 NVIDIA Corporation. All rights reserved.

NVIDIA Corporation | 2788 San Tomas Expressway, Santa Clara, CA 95051
http://www.nvidia.com