
NVIDIA CUDA
MacOS X Release Notes
Version 2.0

New Features

Hardware Support

- o Additional hardware support:
 - GeForce 9800 GX2
 - GeForce 9800 GTX
 - GeForce 9600 GT
 - GeForce 8800 GS
 - GeForce 8600 GTS
 - Quadro FX 3700
 - Quadro NVS 130M
 - Quadro NVS 135M
 - Quadro NVS 140M
 - Quadro NVS 140M
 - Quadro NVS 135M
 - Quadro NVS 130M
 - Quadro FX 3600M

Platform Support

- o Additional OS support
 - Windows Vista 32-bit
 - Windows Vista 64-bit
 - Red Hat Enterprise Linux 4.6
 - Red Hat Enterprise Linux 5.1
 - SUSE Linux 10.3
 - Fedora 8
 - Ubuntu 7.10

Double Precision Computing

- o Compiler support for double-precision math
 - Datatype 'double' compiles to native FP64 types when using SM 1.4
- Note that math functions in the CUDA math library are overloaded.
In general, there are three prototypes for each math function:
- (1) double <func-name>(double), e.g. double log(double)
 - (2) float <func-name>(float), e.g. float log(float)
 - (3) float <func-name>f(float), e.g. float logf(float)
- In particular, note that passing a float argument always results in a float result [variants (2) and (3) above].

- o CUBLAS Library Support

- Added the BLAS1 functions:
 - * cublasIdamax()
 - * cublasIdamin()
 - * cublasDasum()
 - * cublasDaxpy()
 - * cublasDcopy()
 - * cublasDdot()
 - * cublasDnrm2()
 - * cublasDrot()
 - * cublasDrotg()
 - * cublasDrotm()
 - * cublasDrotmg()
 - * cublasDscal()
 - * cublasDswap()
- Added the BLAS2 functions:
 - * cublasDgemv()
 - * cublasDger()
 - * cublasDsyr()
 - * cublasDtrsv()

- Added the BLAS3 functions:

- * cublasDgemm()
- * cublasDsymbm()
- * cublasDsyrk()
- * cublasDsyr2k()
- * cublasDtrmm()
- * cublasDtrsm()
- * cublasZgemm()

New ISA Support SM 1.4

- o Support for any() and all() intrinsics
- o Support for atomic operations on shared memory
- o Support for 64-bit atomic operations
 - atomicAdd() (unsigned 64-bit int)
 - atomicExch() (unsigned 64-bit int)
 - atomicCAS() (unsigned 64-bit int)

API Features

- o 3D texture API
 - cuArray3DCreate
 - cuArray3DGetDescriptor
 - cuMemcpy3D
 - cuMemcpy3DAsync
 - CUDA_MEMCPY3D and CUDA_ARRAY3D_DESCRIPTOR structures
- o Improved Direct3D interoperability API
 - cuD3D9CtxCreate
 - cuD3D9GetDirect3DDevice
 - cuD3D9RegisterResource
 - cuD3D9UnregisterResource
 - cuD3D9MapResources
 - cuD3D9UnmapResources
 - cuD3D9ResourceSetMapFlags
 - cuD3D9ResourceGetMappedPointer
 - cuD3D9ResourceGetMappedSize
 - cuD3D9ResourceGetMappedPitch
 - cuD3D9ResourceGetMappedPitchSlice
 - cuD3D9ResourceGetSurfaceDimensions
- o Context migration API
 - cuCtxAttach
 - cuCtxDestroy
 - cuCtxDetach
 - cuCtxSynchronize
 - cuCtxPushCurrent
 - cuCtxPopCurrent
- o Async constant memory update
 - cudaMemcpyToSymbolAsync
- o Improved device attribute query
 - cuDeviceGetAttribute

Performance Enhancements

- o Improved device->array memcpy performance

Major Bug Fixes

Known Issues

There are several known issues with this release:

- o OpenGL interop will always use a software path leading to reduced performance when compared to interop on other platforms.

- o CUDA kernels which do not terminate or run without interruption for several tens of seconds may trigger the GPU to reset causing a disruption of any attached displays. This may cause display image to become corrupted, which will disappear upon a reboot.
- o If a GPU is used without a display attached it may not exit a reduced power state, causing CUDA programs to perform poorly when run on that GPU. Cycling the system's power saving state or rebooting should reset the GPU. In general it is best to use a GPU with a display attached.
- o The kernel driver may leak wired (i.e. unpageable memory) if CUDA applications terminate in unexpected ways. Continued leaks will lead to severely degraded system performance and requires a reboot to fix.
- o When compiling GCC, special care must be taken for structs that contain 64-bit integers. This is because GCC aligns long longs to a 4 byte boundary by default, while NVCC aligns long longs to an 8 byte boundary by default. Thus, when using GCC to compile a file that has a struct/union, users must give the `-malign-double` option to GCC. When using NVCC, this option is automatically passed to GCC.
- o On systems with multiple GPUs installed or systems with multiple monitors connected to a single GPU, OpenGL interoperability always copies shared buffers through host memory.
- o Current hardware limits the number of asynchronous memcopies that can be overlapped with kernel execution. Overlap is also limited to kernels executing for less than 1 second. These limitations are expected to improve on future hardware.
- o The following APIs exhibit high CPU utilization if they wait for the hardware for a significant amount of time. As a workaround, apps may use `cu(da)StreamQuery` and/or `cu(da)EventQuery` to check whether the GPU is busy and yield the thread as desired.
 - `cuCtxSynchronize`
 - `cuEventSynchronize`
 - `cuStreamSynchronize`
 - `cudaThreadSynchronize`
 - `cudaEventSynchronize`
 - `cudaStreamSynchronize`
- o When the profiler gathers performance signals on G80-based products, the driver reduces the clock rate on the device. If the CUDA app crashes or otherwise exits uncleanly, the clocks will not be reset to their previous values. The system must be rebooted to restore the original clock rate.

Open64 Sources

The Open64 source files are controlled under terms of the GPL license. Current and previously released versions are located via anonymous ftp at download.nvidia.com in the `CUDAOpen64` directory.

Revision History

01/2008 - Version 1.1 - Initial public Beta

More Information

For more information and help with CUDA, please visit
<http://www.nvidia.com/cuda>