CUDA Demo Suite

Reference Manual

TRM-06704-001_v11.8 | October 2022

Table of Contents

Chapter 1. Introduction

The CUDA Demo Suite contains pre-built applications which use CUDA. These applications demonstrate the capabilities and details of NVIDIA GPUs.

Chapter 2. Demos

Below are the demos within the demo suite.

2.1. deviceQuery

This application enumerates the properties of the CUDA devices present in the system and displays them in a human readable format.

2.2. vectorAdd

This application is a very basic demo that implements element by element vector addition.

2.3. bandwidthTest

This application provides the memcopy bandwidth of the GPU and memcpy bandwidth across PCI#e. This 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.

Arguments:

```
Usage:  bandwidthTest [OPTION]...
Test the bandwidth for device to host, host to device, and device to device
 transfers

Example:  measure the bandwidth of device to host pinned memory copies in the range
 1024 Bytes to 102400 Bytes in 1024 Byte increments
./bandwidthTest --memory=pinned --mode=range --start=1024 --end=102400 --
increment=1024 --dtoh
```

Options	Explanation
--help	Display this help menu
--csv	Print results as a CSV
--device=[deviceno]	Specify the device device to be used

Options	Explanation
all	compute cumulative bandwidth on all the devices
0,1,2,...,n	Specify any particular device to be used
--memory=[MEMMODE]	Specify which memory mode to use
pageable	pageable memory
pinned	non-pageable system memory
--mode=[MODE]	Specify the mode to use
quick	performs a quick measurement
range	measures a user-specified range of values
shmoo	performs an intense shmoo of a large range of values
--htod	Measure host to device transfers
--dtoh	Measure device to host transfers
--dtod	Measure device to device transfers
--wc	Allocate pinned memory as write-combined
--cputiming	Force CPU-based timing always
Range Mode options	
--start=[SIZE]	Starting transfer size in bytes
--end=[SIZE]	Ending transfer size in bytes
--increment=[SIZE]]	Increment size in bytes

2.4. busGrind

Provides detailed statistics about peer-to-peer memory bandwidth amongst GPUs present in the system as well as pinned, unpinned memory bandwidth.

Arguments:

Options	Explanation
-h	print usage
-p [0,1]	enable or disable pinned memory tests (default on)
-u [0,1]	enable or disable unpinned memory tests (default off)
-e [0,1]	enable or disable p2p enabled memory tests (default off)
-d [0,1]	enable or disable p2p disabled memory tests (default off)
-a	enable all tests
-n	disable all tests

```
Order of parameters matters.
Examples:
    ./BusGrind -n -p 1 -e 1   Run all pinned and P2P tests
    ./BusGrind -n -u 1        Runs only unpinned tests
    ./BusGrind -a             Runs all tests (pinned, unpinned, p2p enabled, p2p
 disabled)
```

2.5. nbody

This demo does an efficient all-pairs simulation of a gravitational n-body simulation in CUDA. It scales the n-body simulation across multiple GPUs in a single PC if available. Adding "-numbodies=num_of_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.

Arguments:

Options	Explanation
-fullscreen	run n-body simulation in fullscreen mode
-fp64	use double precision floating point values for simulation
-hostmem	stores simulation data in host memory
-benchmark	run benchmark to measure performance
-numbodies=N	number of bodies (>= 1) to run in simulation
-device=d	where d=0,1,2.... for the CUDA device to use
-numdevices=i	where i=(number of CUDA devices > 0) to use for simulation
-compare	compares simulation results running once on the default GPU and once on the CPU
-cpu	run n-body simulation on the CPU
-tipsy=file.bin	load a tipsy model file for simulation

2.6. oceanFFT

This is a graphical demo which simulates an ocean height field using the CUFFT library, and renders the result using OpenGL.

The following keys can be used to control the output:

Keys	Function
w	Toggle wireframe

2.7. randomFog

This is a graphical demo which does pseudo- and quasi- random numbers visualization produced by CURAND. On creation, randomFog generates 200,000 random coordinates in spherical coordinate space (radius, angle rho, angle theta) with curand's XORWOW algorithm. The coordinates are normalized for a uniform distribution through the sphere. The X axis is drawn with blue in the negative direction and yellow positive. The Y axis is drawn with green in the negative direction and magenta positive. The Z axis is drawn with red in the negative direction and cyan positive.

The following keys can be used to control the output:

Keys	Function
s	Generate new set of random nos and display as spherical coordinates (Sphere)
e	Generate new set of random nos and display on a spherical surface (shEll)
b	Generate new set of random nos and display as cartesian coordinates (cuBe/Box)
p	Generate new set of random nos and display on a cartesian plane (Plane)
i, l, j	Rotate the negative Z-axis up, right, down and left respectively
a	Toggle auto-rotation
t	Toggle 10x zoom
z	Toggle axes display
x	Select XORWOW generator (default)
c	Select Sobol' generator
v	Select scrambled Sobol' generator
r	Reset XORWOW (i.e. reset to initial seed) and regenerate
]	Increment the number of Sobol' dimensions and regenerate
[	Reset the number of Sobol' dimensions to 1 and regenerate
+	Increment the number of displayed points by 8,000 (max. 200,000)
-	Decrement the number of displayed points by 8,000 (down to min. 8000)
q/[ESC]	Quit the application.

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.

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

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