AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

Fine-Grain SVM

1 Overview

1.1 Location

`$<AMDAPPSDKSamplesInstallPath>\samples\opencl\cl\2.0`

1.2 How to Run

See the *Getting Started* guide for how to build samples. You first must compile the sample.

Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$<AMDAPPSDKSamplesInstallPath>\samples\opencl\bin\x86\` for 32-bit builds, and `$<AMDAPPSDKSamplesInstallPath>\samples\opencl\bin\x86_64\` for 64-bit builds. Ensure that the OpenCL 2.0 environment is installed.

Type the following command(s).

1. `FineGrainSVM`
   This command runs the program with the default options.
2. `FineGrainSVM -h`
   This command prints the help file.

1.3 Command Line Options

Table 1 lists, and briefly describes, the command line options.

Table 1 Command Line Options

Short Form	Long Form	Description
`-h`	`--help`	Shows all command options and their respective meanings.
	`--device [cpu|gpu]`	Devices on which the OpenCL kernel is to be run. Acceptable values are `cpu` or `gpu`.
`-q`	`--quiet`	Quiet mode. Suppresses all text output.
`-e`	`--verify`	Verify results against reference implementation.
`-t`	`--timing`	Print timing-related statistics.
`-v`	`--version`	AMD APP SDK version string.
	`--dump [filename]`	Dump the binary image for all devices.
	`--load [filename]`	Load the binary image and execute on the device.
	`--flags [filename]`	Specify the filename containing the compiler flags for building the kernel.
`-i`	`--iterations`	Number of iterations.
`-p`	`--platformId`	Select the platformId to be used[0 to N-1 where N is number platform s available].
`-d`	`--deviceId`	Select deviceId to be used[0 to N-1 where N is number devices available].

Short Form	Long Form	Description
-x	--length	Length of the input array.

2 Introduction

This sample demonstrates the memory model of loads and stores in new C++11 standard, which is adopted by OpenCL 2.0. This sample creates two fine-grain buffers by the host. The kernel updates one of the buffers using atomics call (with relevant memory order and scope); the updates are available on the other fine grain SVM buffer too. This demonstrates an important concept of how updates are visible across devices in C11 memory model with atomics.

3 Implementation

This sample demonstrates the usage of 2.0 atomics and fine grain SVM. This sample creates two fine grain buffers on the host: one with atomics support, and the other with just SVM fine grain. The host updates the atomics buffer first and kernel waits for this update using atomics. In the second stage, the kernel updates the atomics buffer and host waits for this update and exits.

The sample workflow is as follows:

1. The host creates two fine grain buffers, `atomicsBuffer` and `buffer`. The `atomicsBuffer` is with atomics support.
2. The kernel waits for CPU to update the `atomicsBuffer` with 99.
3. The kernel updates the "buffer" with some value (+id) and then updates `atomicsBuffer` with (100+i) with release semantics.
4. The CPU waits for the updates and checks for the final values in the buffers.

This sample must be run in the OpenCL 2.0 environment.

Contact

Advanced Micro Devices, Inc.
One AMD Place
P.O. Box 3453
Sunnyvale, CA, 94088-3453
Phone: +1.408.749.4000

For AMD Accelerated Parallel Processing:
URL: developer.amd.com/appsdk
Developing: developer.amd.com/
Support: developer.amd.com/appsdksupport