AMD **Accelerated** Parallel Processing TECHNOLOGY

SAMPLE

Bolt_Introduction

1 Overview

1.1 Location

`$<APPSDKSamplesInstallPath>\samples\bolt\`

1.2 How to Run

See the *Getting Started* guide for how to build samples. You must first compile the sample. Use the command line to change to the directory where the executable is located. The pre-compiled sample executable is at `$<APPSDKSamplesInstallPath>\samples\bolt\bin\x86\` for 32-bit builds, and `$<APPSDKSamplesInstallPath>\samples\bolt\bin\x86_64\` for 64-bit builds.

At the command prompt, type `BoltIntro` to run the program.

2 Introduction

Bolt is a C++ template library optimized for GPUs. Bolt provides high-performance library implementations for common algorithms such as scan, reduce, transform, and sort. The Bolt interface resembles the C++ Standard Template Library (STL); developers familiar with STL will recognize many of the Bolt APIs and customization techniques. In some cases, developers can benefit from GPU acceleration simply by changing the namespace for STL algorithm calls from `std` to `bolt::cl`.

Bolt can directly interface with host memory structures such as `std::vector` or host arrays (`float*`). On current GPU systems the host memory is automatically mapped or copied to the GPU. On future systems that support the Heterogeneous System Architecture (HSA), the GPU directly accesses the host data structures. Bolt also provides a `bolt::cl::device_vector` container that can be used to allocate and manage device-local memory for higher performance on discrete GPU systems. Bolt APIs can accept both STL container or device_vector iterators.

C++ templates can be used to customize the algorithms with new types (for example, the Bolt sort can operate on ints, float, or any custom type defined by the user). Additionally, Bolt lets users customize the template routines using function objects (functors) written in OpenCL; for example, to provide a custom comparison operation for sort, or a custom reduction operation.

3 Implementation Details

The example is very similar to that of the familiar C++ Standard Template Library; the only difference is the include file (`bolt/cl/sort.h`) and the `bolt::cl` namespace before the sort call. If the include is removed and the `bolt::cl` namespace changed to `std`, the code uses the STL version of sort. Bolt developers do not need to learn a new device-specific programming model to leverage the power and performance advantages of GPU computing.

The example demonstrates three important Bolt features.

- The host-allocated vector `randInput` is directly passed to the Bolt sort routine, without a need to explicitly allocate and manage GPU device memory.
- The Bolt `bolt::cl::sort` call submits to the platform, rather than a specific device. The Bolt runtime selects the best device to run the sort, potentially running it on the CPU in the event a GPU is not available or the sort size is too small to benefit from GPU acceleration.
- The Bolt `bolt::cl::sort` call also can work with arrays, such asd `arrlnp`, created on the host.

User-defined datatypes can also be sorted using `bolt::cl::sort` by declaring the type under `BOLT_FUNCTOR()`, overriding the necessary operator functions and using `BOLT_CREATE_TYPENAME()` to register the type with Bolt (for example: `BOLT_CREATE_TYPENAME(bolt::cl::greater< MyType<int> >)`). The implementation of this is left as an exercise to the user.

4 Requirements

Bolt requires an OpenCL™ implementation that supports the static C++ kernel features; specifically, C++ template support for OpenCL™ kernels. Currently, the AMD OpenCL™ SDK 2.8 provides this support; other vendors may adopt this feature in the future.

5 References

1. Bolt documentation, available under `bolt\BoltLib\docs\`

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/**
Forum: **developer.amd.com/openclforum**