

1 Overview

1.1 Location `$<APPSDKSamplesInstallPath>\samples\bolt\`

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 `$<APPSDKSamplesInstallPath>\samples\bolt\bin\x86\` for 32-bit builds and at `$<APPSDKSamplesInstallPath>\samples\bolt\bin\x86_64\` for 64-bit builds.

Type the following command(s).

1. `BoltSort.exe`
This command runs the program with the default options.
2. `BoltSort.exe -h`
This command prints the help file.
3. `BoltSort_TBB -h`
This command generates a build with the multiCoreCpu path (the Thread Building Block library), enabled.

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	Explicit device selection for Bolt [auto/openCL/multiCoreCpu/SerialCpu].
-q	--quiet	Quiet mode. Suppress most text output.
-e	--verify	Verify results against reference implementation.
-t	--timing	Print timing-related statistics.
-v	--version	BOLT and run-time version string.
-x	--samples	Number of sample input values.
-i	--iterations	Number of iterations.

Note: The `--device multiCoreCpu` option becomes available when the sample is compiled with `ENABLE_TBB` defined. Microsoft Visual Studio build configurations `Debug_TBB` and `Release_TBB` are created for this purpose. These configurations have `ENABLE_TBB` defined to enable the TBB path (multiCoreCpu) for all the AMD BOLT functions used in the sample.

2 Introduction

This sample demonstrates the use of the different sorting routines in the BOLT library.

The following types of sorting provided in the BOLT library:

- `Sort()`
This type sorts a vector in ascending or descending order.
- `Sort_by_key`
This type sorts key-value pairs based on keys.

For example, before sorting:

Keys	2	9	3	7	5	6	3	8
Values	20	90	31	70	50	60	32	80

After sorting:

Keys	2	3	3	5	6	7	8	9
Values	20	31	32	50	60	70	80	90

- `Stable_sort()`

The `stable_sort()` operation is analogous to the `std::stable_sort` function. It is a stable operation with respect to the input data, in that, if two elements are equivalent in the input range, and element X appears before element Y, then element X must maintain that relationship and appear before element Y after the sorting operation. `Stable_sort()` preserves this ordering. In general, stable sorts are usually preferred over unstable sorting algorithms, but may sacrifice a little performance to maintain this relationship.

3 Implementation details

This sample shows the performance of the three different types of sort present in the BOLT library. It also demonstrates how `sort()` is better than `stable_sort()` in terms of performance, but `stable_sort()` preserves the original ordering of elements that are equal in value. The recommended command-line options to show the difference between `sort()` and `stable_sort()` routines are: `-e -t -i 10 -x 65536`.

4 References

1. http://en.wikipedia.org/wiki/Sorting_algorithm

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
Forum: developer.amd.com/openglforum



The contents of this document are provided in connection with Advanced Micro Devices, Inc. ("AMD") products. AMD makes no representations or warranties with respect to the accuracy or completeness of the contents of this publication and reserves the right to make changes to specifications and product descriptions at any time without notice. The information contained herein may be of a preliminary or advance nature and is subject to change without notice. No license, whether express, implied, arising by estoppel or otherwise, to any intellectual property rights is granted by this publication. Except as set forth in AMD's Standard Terms and Conditions of Sale, AMD assumes no liability whatsoever, and disclaims any express or implied warranty, relating to its products including, but not limited to, the implied warranty of merchantability, fitness for a particular purpose, or infringement of any intellectual property right.

AMD's products are not designed, intended, authorized or warranted for use as components in systems intended for surgical implant into the body, or in other applications intended to support or sustain life, or in any other application in which the failure of AMD's product could create a situation where personal injury, death, or severe property or environmental damage may occur. AMD reserves the right to discontinue or make changes to its products at any time without notice.

Copyright and Trademarks

© 2013 Advanced Micro Devices, Inc. All rights reserved. AMD, the AMD Arrow logo, ATI, the ATI logo, Radeon, FireStream, and combinations thereof are trademarks of Advanced Micro Devices, Inc. OpenCL and the OpenCL logo are trademarks of Apple Inc. used by permission by Khronos. Other names are for informational purposes only and may be trademarks of their respective owners.