

BOLT Monte Carlo PI estimator

1 Overview

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

Type the following command(s).

1. `MonteCarloPI`
This command runs the program with the default options.
2. `MonteCarloPI -h`
This command prints the help file.
3. `MonteCarloPI_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 AMD APP SDK version string.
-x	--samples	Number of sample input values.
-i	--iterations	Number of iterations.
-g	--gui	Show the GUI.

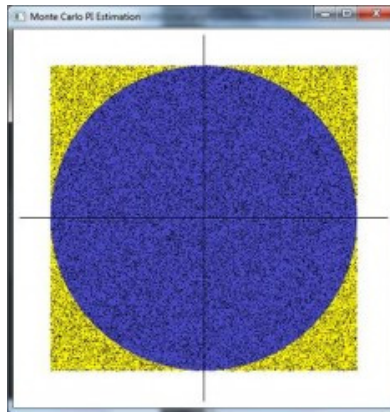
Note: The `--device multiCoreCpu` option becomes available when the sample is compiled with `ENABLE_TBB` defined. Microsoft Visual Studio build configurations `Debug_TBB` and

`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 implements an estimator of π using the Monte Carlo method. Monte Carlo methods solve computation problems by observing the outcome of a large number of random samples in a system. It is common knowledge that the value of $\pi/4$ is equal to the probability of a "dot" falling inside a circle bounded by a circumscribing square. This factor can be arrived at by simply dividing the area of the circle by the area of the square. This sample uses a Monte Carlo simulation to find the probability of a point falling inside the circle and uses that probability to compute the value of π .

The following figure shows an example of the simulation performed by the sample. A circle is shown in its bounding square, with random points uniformly scattered over its area.



3 Implementation details

The implementation is broken down into 2 steps:

1. For all the given points (numbering `totalPoints`), check whether or not the point falls inside the circle. A point is inside the circle, if the distance between the point and center of circle is less than the radius of the circle.
2. Count all the points that fall inside the circle (say `numPointsInCircle`)

The value of π is then computed as $(4 * \text{numPointsInCircle}) / \text{totalPoints}$.

This algorithm can be implemented in three different ways using different Bolt APIs.

1. Using the `transform()` and `reduce()` BOLT APIs.
This implementation uses the two Bolt APIs, Transform and Reduce.

An intermediate `device_vector`, `insideCircle`, of type `int`, is created. This array indicates whether or not the point exists within the circle.

- i. Apply `bolt::cl::transform()` on the input points and store the result in `insideCircle`.
- ii. Count the number of 1s in `insideCircle` using `bolt::cl::reduce()`.

2. Using the `transform_reduce()` BOLT API.

This method uses the `transform_reduce()` API. Instead of creating a temporary array to store intermediate results, using the `bolt::cl::transform_reduce()` API allows fusing the operations used in the previous method and reducing some memory overhead.

3. Using the `count_if()` BOLT API.

The algorithm can also be implemented using the `count_if()` API. `count_if()` counts the number of elements in the specified range for which the specified predicate is `true`. Here the elements are the input points and the predicate is `insideCircle`.

4 References

1. <http://developer.amd.com/community/blog/monte-carlo-sample-in-bolt/>

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



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

© 2015 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.