

# SHOC: The Scalable HeterOgeneous Computing Benchmark Suite

Dakar Team  
Future Technologies Group  
Oak Ridge National Laboratory

Version 0.9, June 2010

## 1 Introduction

The Scalable HeterOgeneous Computing benchmark suite (SHOC) is a collection of benchmark programs that tests the performance and stability of systems using computing devices with non-traditional architectures for general purpose computing, and the software used to program them. Its initial focus is on systems containing Graphics Processing Units (GPUs) and multi-core processors, and on the OpenCL [3] programming standard. It can be used on clusters as well as individual hosts.

OpenCL is an open standard for programming a variety of types of computing devices. The OpenCL specification describes a language for programming *kernels* to run on an OpenCL-capable device, and an Application Programming Interface (API) for transferring data to such devices and executing kernels on them.

In addition to OpenCL-based benchmark programs, SHOC also includes a C for Compute Unified Device Architecture (CUDA) [4] version of its benchmarks for comparison.

## 2 Supported Platforms

The Dakar team intends SHOC to be useful on any platform with an OpenCL implementation. However, the Dakar team develops and tests SHOC primarily on UNIX-like platforms, specifically Linux and Mac OS X.

### 2.1 Linux

- A recent RedHat-family OS distribution (Fedora or RHEL).<sup>1</sup>
- A working OpenCL implementation. The Dakar team has used the following implementations:
  - NVIDIA GPU Computing SDK version 2.3a

---

<sup>1</sup> Some recent Linux distributions include pre-packaged gcc 4.4 toolchains. At the time of this writing (June 2010), NVIDIA CUDA does not officially support gcc 4.4, and builds of SHOC that include CUDA on such platforms will fail. This is a CUDA issue, not a problem with the SHOC benchmark software. On such platforms, an earlier version of gcc (we recommended gcc 4.3.x) must be used to compile SHOC, and the SHOC configuration files must be modified so that the `-compiler-bindir` switch is passed to `nvcc` to indicate the location of the gcc compiler binaries to use.

- NVIDIA GPU Computing SDK version 3.0
- ATI Stream SDK version 2.0 beta2
- ATI Stream SDK version 2.1 beta2
- (Optional) CUDA 3.0 or later.

SHOC may work on other platforms with other OpenCL implementations than those listed here, but will most likely require modifications for differing OpenCL header and library paths, differing system library versions, and differing compiler versions/vendors.

## 2.2 Mac OS X

- Mac OS X 10.6 ("Snow Leopard") or later.
- Xcode 3.2 or later.
- (Optional) CUDA 3.0 or later

## 2.3 Clusters

In addition to individual systems, SHOC can also build parallel benchmark programs for clusters. Each cluster node must meet the requirements described earlier in this section for the OS distribution used on that node. Also, the cluster must have a working implementation of the Message Passing Interface (MPI) [1, 2] library such as OpenMPI [www.open-mpi.org](http://www.open-mpi.org) or mpich2 [www.mcs.anl.gov/mpi/mpich](http://www.mcs.anl.gov/mpi/mpich).

## 3 Configuring

If you are interested in simply getting SHOC up and running quickly, you can edit the `default.mk` file. This file contains settings for OpenCL, CUDA, and MPI, as well as general compilation options. Each field is commented with a brief description.

If it exists, the SHOC build process will read a file named `$hostname.mk` where `$hostname` is the output of running the `hostname` command. Several configurations from machines used by the development team are included as examples.

## 4 Building

If you have CUDA, OpenCL, and MPI, build the entire SHOC suite by:

```
$ cd $SHOC_ROOT
$ make
```

To build only the CUDA versions of the benchmarks, use

```
$ cd $SHOC_ROOT
$ make cuda
```

Or similarly, only the OpenCL versions with

```
$ cd $SHOC_ROOT
$ make opencil
```

## 5 Running

SHOC includes a driver script for running either the CUDA or OpenCL versions of the benchmarks. The driver script assumes MPI is in your current path, so be sure to set the appropriate environment variables.

```
$ export PATH=$PATH:/path/to/mpi/bin/dir
$ export LD_LIBRARY_PATH=$LD_LIBRARY_PATH:/path/to/mpi/lib/dir
```

To run the benchmarks on a single node, execute the following. Be sure to specify “-cuda” or “-opencil”.

```
$ cd $SHOC_ROOT/tools
$ perl driver.pl -cuda
```

To run on more than one node, supply the script with the number of nodes and the number of available devices per node. For example, if running on 4 nodes that each have 2 devices, execute the following:

```
$ cd $SHOC_ROOT/tools
$ perl driver.pl -cuda -n 4 -d 2
```

These scripts output benchmark results to a file in comma separated value (CSV) format.

After building SHOC, individual benchmark programs will be left in the directory tree rooted at \$SHOC\_ROOT/bin. Run single-process benchmark programs with commands like:

```
$ cd $SHOC_ROOT/bin
$ ./Serial/OpenCL/Scan
```

and parallel benchmark programs with commands like:

```
$ cd $SHOC_ROOT/bin
$ mpirun -np 8 ./EP/Scan
```

Use 1 MPI rank per GPU.

## 6 Benchmark Programs

The SHOC benchmark suite currently contains benchmark programs, categorized based on complexity. Some measure low-level “feeds and speeds” behavior (Level 0), some measure the performance of a higher-level operation such as a Fast Fourier Transform (FFT) (Level 1), and the others measure real application kernels (Level 2).

- Level 0
  - **BusSpeedDownload**: measures bandwidth of transferring data across the PCIe bus to a device.
  - **BusSpeedReadback**: measures bandwidth of reading data back from a device.
  - **DeviceMemory**: measures bandwidth of memory accesses to various types of device memory including global, local, and image memories.
  - **KernelCompile**: measures compile time for several OpenCL kernels, which range in complexity
  - **PeakFlops**: measures maximum achievable floating point performance using a combination of auto-generated and hand coded kernels.
  - **QueueDelay**: measures the overhead of using the OpenCL command queue.
- Level 1
  - **FFT**: forward and reverse 1D FFT.
  - **MD**: computation of the Lennard-Jones potential from molecular dynamics, a specific case of the nbody problem.
  - **Reduction**: reduction operation on an array of single precision floating point values.
  - **SGEMM**: single-precision matrix-matrix multiply.
  - **Scan**: scan (also known as parallel prefix sum) on an array of single precision floating point values.
  - **Sort**: sorts an array of key-value pairs using a radix sort algorithm
  - **Stencil2D**: a 9-point stencil operation applied to a 2D data set. In the MPI version, data is distributed across MPI processes organized in a 2D Cartesian topology, with periodic halo exchanges.
  - **Triad**: STREAM Triad operations, implemented in OpenCL.

To see the options each program supports and their default values, run *program -help* for serial versions and *mpirun -np 1 program -help* for parallel versions.

Benchmarks are built not only as serial programs (S) but also as embarrassingly parallel (EP) or true parallel (TP) programs. The following table indicates which versions of each program that SHOC builds.

## 7 Source Tree

SHOC is distributed as a compressed tar archive. Let `SSHOC_ROOT` represent the directory that will hold the SHOC source tree. The SHOC archive can be uncompressed and extracted using

```
$ cd $SSHOC_ROOT
$ tar xvzf shoc-x.y.tar.gz
```

<i>Program</i>	<i>OpenCL</i>			<i>CUDA</i>		
	<i>S</i>	<i>EP</i>	<i>TP</i>	<i>S</i>	<i>EP</i>	<i>TP</i>
BusSpeedDownload	x	x		x	x	
BusSpeedReadback	x	x		x	x	
DeviceMemory	x	x		x	x	
KernelCompile	x	x				
PeakFlops	x	x		x	x	
QueueDelay	x	x				
FFT	x	x		x	x	
MD	x	x		x	x	
Reduction	x	x		x	x	
S3D	x	x		x	x	
SGEMM	x	x		x	x	
Scan	x	x		x	x	
Sort	x	x		x	x	
Stencil2D	x		x	x		x
Triad	x	x		x	x	

Table 1: Programming APIs and parallelism models of SHOC programs

The SHOC source tree directory structure is as follows:

```

$SHOC_ROOT
  bin      # benchmark executables are built here
    EP    # "embarrassingly parallel" benchmarks
      CUDA
      OpenCL
    TP    # true parallel benchmarks
      CUDA
      OpenCL
  Serial  # single-node benchmarks
    CUDA
    OpenCL
  config  # SHOC configuration files
  doc     # SHOC documentation files
  lib     # SHOC auxiliary libraries are built here
  src     # SHOC source files
  common  # programming-model independent helper code
  cuda   # CUDA-based benchmarks
    level0 # low-level CUDA benchmarks
    level1 # higher-level CUDA benchmarks
    level2 # application level CUDA benchmarks
  mpi     # MPI-specific benchmarks
    common # code needed by programs using MPI
    contention # a contention benchmark
  opencl  # OpenCL benchmarks

```

```
common # code needed for all OpenCL benchmarks
level0 # low-level OpenCL benchmarks
level1 # higher-level OpenCL benchmarks
level2 # application-level OpenCL benchmarks
stability # a CUDA stability test
```

## 8 Support

Support for SHOC is provided on a best-effort basis by the Dakar team members and eventually by its user community via several mailing lists.

- `shoc-announce@email.ornl.gov`: mailing list for announcements regarding new versions or important updates.
- `shoc-help@email.ornl.gov`: email address for requesting help in building or using SHOC, or for providing feedback about the benchmark suite.
- `shoc-dev@email.ornl.gov`: mailing list for internal development discussions by the SHOC development team.

## Revision History

- 0.1 September 2009
- 0.2 December 2009
- 0.3 June 2010

## References

- [1] William Gropp, Ewing Lusk, and Anthony Skjellum. *Using MPI: Portable Parallel Programming with the Message Passing Interface*, 2nd edition. MIT Press, Cambridge, MA, 1999.
- [2] William Gropp, Ewing Lusk, and Rajeev Thakur. *Using MPI-2: Advanced Features of the Message-Passing Interface*. MIT Press, Cambridge, MA, 1999.
- [3] The Khronos Group. The opencl specification, version 1.0, document revision 43. specification, The Khronos Group, 2009.
- [4] NVIDIA. Nvidia cuda reference manual, version 2.3. manual, 2009.