



Future Technologies Group
Oak Ridge National Laboratory
Oak Ridge, Tennessee
USA

<http://ft.ornl.gov>

Scalable Heterogeneous Computing (SHOC) Benchmark Suite, Version 0.8

Authors J.S. Vetter, A. Danalis, G. Marin, C. McCurdy, J. Meredith, P.C. Roth, K. Spafford, V. Tipparaju

Abstract The Scalable Heterogeneous Computing Benchmark Suite (SHOC) is a collection of benchmark programs to test the performance and stability of heterogeneous systems, which use computing devices with non-traditional architectures, and their respective software stacks. Its initial focus is on systems containing Graphics Processing Units (GPUs) and multi-core processors, and on the [OpenCL](#) programming standard. A distributed version of SHOC can be used on scalable heterogeneous clusters.

Future Technologies Group Technical Report FTGTR-2009-11
Total Pages 8
Publication Date 12/4/2009
Also published as -

SHOC: The Scalable Heterogeneous Computing Benchmark Suite

Dakar Team
Future Technologies Group
Oak Ridge National Laboratory

Version 0.8, December 2009

1 Introduction

The Scalable Heterogeneous Computing benchmark suite (SHOC) is a collection of benchmark programs testing 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 running kernels on them. The OpenCL specification was ratified by The Khronos Group in late 2008. At the time of this writing, OpenCL implementations are just becoming publicly available. These early OpenCL implementations support running OpenCL kernels on GPUs and commodity multi-core processors, though not all implementations support both device types.

In addition to OpenCL-based benchmark programs, SHOC also includes a Compute Unified Device Architecture (CUDA) [4] version of many of its benchmarks for comparison with the OpenCL version. CUDA, developed by NVIDIA, is an approach for programming NVIDIA GPUs for general purpose computing that predates OpenCL. Like OpenCL, CUDA-based programs use a host program running on the system's CPU to run kernels on an accelerator device (in this case, a GPU).

This document describes how to build and use SHOC. We first detail the supported platforms for using SHOC (Section 3), followed by an overview of the SHOC source code (Section 4), how to configure it (Section 5), build it (Section 6), and run it (Section 7).

2 Benchmark Programs

The SHOC benchmark suite currently contains benchmark programs, categorized as to whether they measure low-level "feeds and speeds" behavior (Level 0) or the performance of a higher-level operation such as an FFT (Level 1).

- Level 0
 - **BusSpeedDownload**: measures bandwidth of transferring data to a device.

- **BusSpeedReadback**: measures bandwidth of reading data back from a device.
 - **DeviceMemory**: measures latency and bandwidth of memory accesses to various types of device memory.
 - **KernelCompile**: compile a simple and more complex OpenCL kernel.
 - **PeakFlops**: dynamically auto-tuned multiply-add and multiply-add with multiplication, for studying maximum achievable floating point performance.
 - **PeakFlopsMADD**: auto-generated kernels using fused multiply-add operations, for studying maximum achievable floating point performance.
 - **PeakFlopsMADDMUL**: auto-generated kernels using fused multiply-add with an extra multiplication, for measuring maximum achievable floating point performance.
 - **QueueDelay**: measures overhead of OpenCL command queue.
- Level 1
 - **FFT**: forward and reverse 1D FFT.
 - **MD**: a molecular dynamics benchmark.
 - **Reduction**: reduction operation on a vector of values.
 - **SGEMM**: single-precision matrix-matrix multiply.
 - **Scan**: scan (also known as parallel prefix) on a vector of values.
 - **Sort**: sorts a vector of values.
 - **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.

In addition to the OpenCL versions, SHOC includes CUDA versions of many of these benchmark programs. Also, many programs 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.

3 Supported Platforms

The Dakar team intends SHOC to be useful on any platform with an OpenCL implementation. However, due to limited resources the Dakar team develops and tests SHOC primarily on UNIX-like platforms. In particular, the Dakar team uses Linux and Mac OS X systems for development and testing.

<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				
PeakFlopsMADD	x	x		x	x	
PeakFlopsMADDMUL	x	x		x	x	
QueueDelay	x	x				
FFT	x	x		x	x	
MD	x	x		x	x	
Reduction	x	x		x	x	
SGEMM	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

3.1 Linux

- A recent RedHat-family OS distribution (Fedora or RHEL).¹
- A working OpenCL implementation. The Dakar team has used the following implementations:
 - NVIDIA GPU Computing SDK version 2.3a
 - NVIDIA GPU Computing SDK version 3.0 beta
 - ATI Stream SDK version 2.0 beta2
- (Optional) CUDA 2.3 or later.

This list describes the platforms to which the Dakar team has access for development and testing. SHOC may work on other Linux distributions with other OpenCL implementations than those listed here. Modifications may be needed for differing OpenCL header and library paths, differing system library versions, and differing compiler versions/vendors.

3.2 Mac OS X

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

¹Some recent Linux distributions include pre-packaged gcc 4.4 toolchains. At the time of this writing (December 2009), NVIDIA CUDA does not 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.

- (Optional) CUDA 2.3 or later, preferring 2.3.1a or later for better support of the Xcode default gcc/g++ version.

3.3 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.

3.4 Documentation

The Dakar team maintains SHOC documentation as \LaTeX and \BibTeX files. Although the SHOC distribution includes PDF files for all documentation, the documentation can be rebuilt on a system with \LaTeX and \BibTeX installed.

4 Source Tree

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

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

The SHOC source tree directory structure is as follows:

```
$SHOC_ROOT
  bin      # benchmark executables are built here
    Parallelep # "embarrassingly parallel" benchmarks
      CUDA
      OpenCL
    ParallelGlobal # 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
    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
    stability # a CUDA stability test (FFT)
```

5 Configuring

For configuration, SHOC uses a collection of files in the `$SHOC_ROOT/config` directory. There are several types of files in this directory.

- Default configuration
- OS-specific configuration
- Programming model-specific configuration
- System-specific configuration
- Simplified configuration

These types of files are described in the rest of this section.

5.1 Default Configuration

The first type of SHOC configuration file includes default settings that are independent of the OS, programming model, and specific system on which SHOC is being built and run. The `config.mk` file controls the order of including configuration files. The `base.mk` file defines the default compiler and adds SHOC directories to the include and linker paths. The `targets.mk` file defines the make rules used for building SHOC.

5.2 OS-Specific Configuration

If it exists, the SHOC build process will read configuration settings from a file called `OS.mk` where `OS` is the output of running `uname -s`. This is the file to use for any settings that are known not to vary from system to system among those using that OS distribution.

5.3 Programming Interface Configuration

The next type of SHOC configuration file includes configuration for a particular programming model.

- `openc1.mk` contains OpenCL settings. If they exist, this file will also read `openc1-OS.mk` and `openc1-OS-OCL_VENDOR.mk`.
- `cuda.mk` contains CUDA-specific settings. If it exists, this file will also read `cuda-OS.mk`.
- `mpi.mk` contains MPI-specific settings.

5.4 System Specific Configuration

If it exists, the SHOC build process will read a file named `hostname.mk` where `hostname` is the output of running the `hostname` command. This is the place to indicate where OpenCL and/or CUDA is installed on a particular system, the vendor of the OpenCL implementation installed on that system (because the include and library paths vary from implementation to implementation), and any other configuration overrides. The SHOC distribution includes several examples of system-specific configuration files.

5.5 Simplified Configuration

If you are interested in simply getting SHOC up and running quickly, you can edit the `default.mk` file. It can be used in place of a host-specific configuration file, and contains example values which after slight modification should be sufficient for getting a full build of SHOC.

6 Building

After editing the configuration files, build the entire SHOC suite by:

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

Some branches of the SHOC directory tree may give expected build failures. For instance, on systems without a CUDA installation, the SHOC makefiles will attempt to build the CUDA benchmark programs but will fail. By default, these failures are not fatal errors for the SHOC build process.

7 Running

SHOC includes scripts for running either the CUDA or OpenCL versions of the benchmarks. To run the scripts on a single node, execute the following:

```
$ cd $SHOC_ROOT/tools
$ perl ocl_driver.pl
```

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 a 4 nodes that each have 2 devices, execute the following:

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

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

After building SHOC, 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 -s 3
```

and MPI benchmark programs with commands like:

```
$ cd $SHOC_ROOT/bin
$ mpirun -np 128 $PWD/ParallelEP/Scan -s 3
```

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

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.