A

(QUICK STARTER)

GUIDE TO

HipGISAXS

version 0.01-alpha

Lawrence Berkeley National Laboratory October 2012

Contents

Contents **1**

1 Introduction **3**
1.1 What is HipGISAXS? 3
1.2 Where's What? 4
1.3 Developers of HipGISAXS 4
1.4 Licensing Information 5

2 Supported Platforms **7**
2.1 Operating System Platforms 7
2.2 System Hardware (Compute Environment) 7
2.3 System Hardware (GPU/CPU) 8

3 Software Pre-requisites **9**
3.1 Required Software List 9

4 Directory Layout **13**

5 Building and Installing **15**
5.1 On Carver/Dirac at NERSC, LBNL 15
5.2 On Titan at OLCF, ORNL 16
5.3 On a generic Linux system 16
5.4 On a generic Mac OS X system (single or multiple-node) 17
5.5 On `saxs-waxs-gpu` (for certain local users only) 17

5.6 On `saxs-waxs-gpu2` (for certain local users only) 18

6 Running on a System 19

6.1 Interactively on Dirac 19

6.2 Batch script on Dirac to use multiple GPU nodes 20

6.3 Interactively on Titan 21

6.4 Batch script on Titan to use multiple GPU nodes 22

6.5 Interactively on a generic single-node Linux system equipped with a GPU
(including `saxs-waxs-gpu` and `saxs-waxs-gpu2`) 22

6.6 Interactively on a generic multiple-node Linux cluster equipped with a GPU on each node 22

6.7 Interactively on a generic single-node Mac OS X system equipped with a GPU 22

6.8 Interactively on a generic multiple-node Mac OS X cluster equipped with a GPU on each node 22

7 Inputs 23

8 Tutorial and Examples 25

Chapter

1 Introduction

1.1 What is HipGISAXS?

HipGISAXS, which stands for High-Performance GISAXS (Grazing Incidence Small-Angle X-ray Scattering), is a GISAXS pattern simulation software, based on the Distorted Wave Born Approximation (DWBA) theory. HipGISAXS is a massively parallel code and delivers high-performance by utilizing parallel processing on clusters of graphics processors (GPUs) and multi-core CPUs. It can be used on any system ranging from a standalone desktop computer to large state-of-the-art clusters and supercomputers.

HipGISAXS is a handy tool for synchrotron light-source beamline experimentalists facing a massive flux of data, allowing them to accurately simulate the GISAXS processes and analyze the generated data. It computes scattering intensity patterns for any given sample comprising of a superposition of custom shapes or morphologies, which may be obtained graphically via a discretization scheme, in a user-defined region of the *k*-space, for all possible grazing incidence angles and in-plane sample rotations. This flexibility allows an easy tackling of a wide range of sample geometries, such as macromolecules and nanostructures on top of or embedded in a substrate, or in a multilayered media.

1.2 Where's What?

This document is a starting point in the HipGISAXS documentation. It covers information on the usage of the software, and gives pointers to other documents for further details. Specifically, this document describes the supported systems and platforms, pre-requisites, installation and execution of HipGISAXS. It can also be treated as a quick start guide for those who want to right-away dig into using the software.

Documentation included with the HipGISAXS software package comprises of the following parts.

- This document, which serves as a starting point to HipGISAXS.
- Details on the theory behind HipGISAXS and its working can be found in the *"The theory behind HipGISAXS"* document.
- Implementation details of the software can be found in the *"Implementation details of HipGISAXS"* document.
- Description of all the inputs to this software are given the *"A guide to the inputs of HipGISAXS"* document.
- Frequently asked questions and troubleshooting on the usage of the software are detailed in the *"FAQ's and troubleshooting HipGISAXS"* document.
- A *man page* giving an overview of the usage of HipGISAXS is also included in the package.

1.3 Developers of HipGISAXS

HipGISAXS has been realized through the combined efforts of a team of researchers. The HipGISAXS development team comprises of the following members:

- Slim T. Chourou (*developer and author of initial Matlab implementation,*)
- Abhinav Sarje (*author of high-performance GPU/multi-core C++ implementation,*)
- Elaine R. Chan,
- Alexander Hexemer, and
- Xiaoye S. Li.

1.4 Licensing Information

HipGISAXS is available freely for non-commercial use under the ABCXYZ license. <put licensing information for HipGISAXS here.>

Intentional blank page.

CHAPTER

2 Supported Platforms

HipGISAXS has been tested on, and technically supports, the following major platforms.

2.1 Operating System Platforms

1. GNU/Linux x86_64: Ubuntu, Red Hat Linux, SUSE Linux, Cray Linux Environment (CLE) (tested on Cray XT5, XE5 and XK6 systems).
2. Darwin x86_64: Mac OS X (Lion, Mountain Lion).
3. You could try building and running HipGISAXS on any 64-bit system with a UNIX based operating system, and if you are lucky, it may support it.

Please let us know about your platform so that we can include it in our list of platforms to support.

2.2 System Hardware (Compute Environment)

1. HPC Clusters/Supercomputers equipped with at least one Nvidia graphics processor (GPU) as accelerator on each node.
2. Generic desktop equipped with a compute GPU (see system hardware support below).

3. *Coming soon:* A cluster/supercomputer or desktop equipped with generic Intel/AMD 64-bit CPU (single or multi-core) without any GPU accelerator available.

2.3 System Hardware (GPU/CPU)

1. GPU accelerators: Fermi or Kepler architecture based Nvidia GPUs. Compute capability of the GPUs should be 2.0 or higher. Current HipGISAXS version mandates availability of a GPU accelerator to run.

2. CPU host nodes: HipGISAXS should run on any 64-bit Intel/AMD CPU, including multi-cores. Note that we have not extensively tested on the various types of processors yet. If HipGISAXS does not successfully on your system, let us know about its configuration/processor so that we can support it.

Chapter

3 Software Pre-requisites

The current version of HipGISAXS supports only systems equipped with compute GPUs as accelerators (see Chapter 2). This software uses third-party libraries, and these need to be installed and available on your system in order to compile and run HipGISAXS.

You may have some of these software/libraries already available on your system, and you may use them. Alternatively, you may download and install it yourself.

NOTE: The supplied compiled binary with the HipGISAXS tarball is dynamically linked with these libraries on Carver/Dirac, some of them installed at `/global/homes/a/asarje/local/`. It is recommended to use either the installations already available on the system, or you own local installations, and not rely on this local user installation.

3.1 Required Software List

The following are the dependencies of HipGISAXS:

1. GNU C/C++ compilers, version 4.3 or greater, but no greater than 4.5.x.

   If these are not available on the system, they will need to be installed.

There are different ways to do this on various Linux flavors, for example on a Ubuntu system, you can use the `apt-get` tool:

```
$ sudo apt-get install gcc
```

On Mac OS X, these compilers can be installed either by installing Xcode, or independently without Xcode using sources or through package managers like Homebrew.

2. Nvidia CUDA version 4.x.

   CUDA may be obtained from: `http://developer.nvidia.com/cuda/cuda-downloads`.

   NOTE: CUDA version 5.0rc is also supported.

3. GNU compiled OpenMPI, version 1.4.4 or higher.

   OpenMPI can be obtained from: `http://www.open-mpi.org/software`.

4. Boost, and the 'numeric' extension to Boost GIL (Generic Image Library). This extension is NOT distributed with Boost, so you need to install it explicitly.

   Boost can be obtained from: `http://www.boost.org`.

   The 'numeric' extension to Boost GIL can be obtained from one of the following locations:

   `http://sourceforge.net/adobe/genimglib/wiki/Downloads`, or

   `http://gil-contributions.googlecode.com/svn/trunk`.

   NOTE: Boost and the 'numeric' extension to GIL are also installed at `/global/homes/a/asarje/local/bo` on Carver/Dirac.

5. Parallel HDF5 library.

   HDF5 can be obtained from: `http://www.hdfgroup.org/downloads`.

   NOTE 1: On Carver/Dirac, parallel HDF5 is also installed at `/global/homes/a/asarje/local/hdf5-1.8.8-gnu/parallel`.

   NOTE 2: An `hdf5-parallel` module is also available on Carver/Dirac.

   NOTE 3: HDF5 depends on zlib and szip.

   Zlib (libz) can be obtained from: `http://www.zlib.net`.

   Szip (libsz) can be obtained from: `http://www.hdfgroup.org/doc_resource/SZIP`.

6. Tiff image library (libtiff).

   Tiff library can be obtained from: `http://www.libtiff.org`.

   NOTE: It is also installed at `/global/homes/a/asarje/local/tiff-4.0.2` on Carver/Dirac.

Download and install the above softwares/libraries, if needed, before going on to building or using HipGISAXS.

Intentional blank page.

CHAPTER

4 Directory Layout

The HipGISAXS software package is organized into various directories as shown in the diagram below.

`hipgisaxs/`	:	The root of the HipGISAXS package.
`Makefile`	:	The main makefile to build HipGISAXS.
`README`	:	A quick start information (read first).
`bin/`	:	Contains HipGISAXS binaries generated by compilation.
`build/`	:	Contains a few makefiles for various predefined systems.
`data/`	:	Provides some sample input custom shape definition files in HDF5 format.
`doc/`	:	Contains detailed documentations of HipGISAXS.
`inputs/`	:	Provides some sample input files to the HipGISAXS program in HiG format.
`man/`	:	Contains the man pages about HipGISAXS usage.

`obj/`	:	All object files generated during build are stored here.
`src/`	:	The main source directory containing all source files.

Chapter 5

Building and Installing

If you want to re-build the HipGISAXS binary, use the `make` utility as in the following cases:

5.1 On Carver/Dirac at NERSC, LBNL

1. Some makefiles for different systems are included in the directory `build`. Replace the current `Makefile` with the one provided for Dirac (renaming it as `Makefile`):

```
$ cp build/Makefile.dirac Makefile
```

2. Unload the default PGI modules:

```
$ module unload pgi openmpi
```

3. Load the required modules:

```
$ module load openmpi-gnu gcc/4.5.2 cuda/4.2
$ module load szip zlib
$ module load hdf5-parallel/1.8.3-gnu
```

4. The Boost module available on Carver does NOT include the 'numeric' extension to GIL (Generic Image Library). Please download, install and use your own copy of Boost and this extension, or use the one installed at `/global/homes/a/asarje/local/boost_1`

5. Edit `Makefile` to specify the correct paths in the "`base directories`" section, present towards the beginning of the file, to the actual locations of the various libraries.

   An example of the `base directories` section in the `Makefile`:

   ```
   $ cat Makefile
   ...
   ## base directories
   BOOST_DIR = /global/homes/a/asarje/local/boost_1_49_0
   MPI_DIR =
   CUDA_DIR = /usr/common/usg/cuda/4.2
   HDF5_DIR = /global/homes/a/asarje/local/hdf5-1.8.8-gnu/parallel
   TIFF_DIR = /global/homes/a/asarje/local/tiff-4.0.2
   Z_DIR = $(ZLIB_DIR)      # an environment variable set by
                            # loading zlib module
   SZ_DIR = $(SZIP_DIR)     # an environment variable set by
                            # loading szip module
   ...
   ```

6. Build the code from within the main directory:

   ```
   $ make clean
   $ make
   ```

   This will generate the binary `hipgisaxs` in the directory `bin`. All the generated object files of the source code are stored in the directory `obj`.

5.2 On Titan at OLCF, ORNL

Currently Titan is being built and upgraded, so this section can wait :-).

5.3 On a generic Linux system

1. The Linux system can be a single-node or multiple-node system.

2. Make sure all the software prerequisites are available.

3. Make sure all system environment variables are set accordingly to include the prerequisites. Generally, you may want to check the variables `PATH` and `LD_LIBRARY_PATH` to include the required software paths.

4. Either `Makefile.saxs1` or `Makefile.saxs2` can be used as a template. Edit the sample `Makefile` and set all the paths correctly (see the example in Section 5.1 above).

5. Build the code from within the main directory:

```
$ make clean
$ make
```

5.4 On a generic Mac OS X system (single or multiple-node)

1. The Mac OS X system can be either a single-node or multiple-node system.

2. Make sure all the software prerequisites are available.

3. Make sure all system environment variables are set accordingly to include the prerequisites. Generally, you may want to check the variables `PATH` and `DYLD_LIBRARY_PATH`.

4. Edit the sample `Makefile` and set all the paths correctly (see example in Section 5.1 above).

5. Build the code from within the main directory:

```
$ make clean
$ make
```

5.5 On `saxs-waxs-gpu` (for certain local users only)

1. Replace the current `Makefile` with the one provided for `saxs-waxs-gpu` (renaming it as `Makefile`):

```
$ cp build/Makefile.saxs1 Makefile
```

2. All the "`base directories`" in the specified in this `Makefile` are set to the correct locations of the respective software installations on this system. If you want to use your own installation of any of these, please edit its corresponding entry in the `Makefile`. An example:

   ```
   $ cat Makefile
   ...
   ## base directories
   BOOST_DIR = /usr/local/boost_1_45_0
   MPI_DIR = /usr/local
   CUDA_DIR = /usr/local/cuda
   HDF5_DIR = /home/asarje/local/hdf5-1.8.8-gnu/parallel
   Z_DIR = /root/zlib-1.2.7
   SZ_DIR = /root/szip-2.1
   TIFF_LIB_DIR = /usr/local
   ...
   ```

3. Build the code from within the main HipGISAXS directory:

   ```
   $ make clean
   $ make
   ```

   This will generate the binary `hipgisaxs` in the directory `bin`. All the generated object files are stored in the directory `obj`.

5.6 On `saxs-waxs-gpu2` (for certain local users only)

Follow the same steps as above in Section 5.5, replacing the provided `Makefile` for this system with `build/Makefile.saxs2`.

Chapter

6 Running on a System

Once the HipGISAXS binary is available for the system being used, follow the following steps to run GISAXS simulations depending on your particular system.

6.1 Interactively on Dirac

1. Unload the default loaded PGI modules:

   ```
   $ module unload pgi openmpi
   ```

2. Load the required modules:

   ```
   $ module load openmpi-gnu gcc/4.5.2 cuda/4.2
   $ module load szip zlib
   $ module load hdf5-parallel/1.8.3-gnu
   ```

3. Change to the main HipGISAXS directory. For example:

   ```
   $ cd hipgisaxs
   ```

4. Request an interactive GPU node:

   ```
   $ qsub -I -V -q dirac_int -l nodes=1:ppn=8:fermi
   ```

5. Wait until you are provided with a command prompt on a GPU node.

6. Make sure the loaded modules are correct as above.

   NOTE: By default, PGI version of openMPI might be loaded. So you may need to unloaded it again:

   ```
   $ module unload openmpi
   ```

7. Move to the main HipGISAXS directory:

   ```
   $ cd $PBS_O_WORKDIR
   ```

8. Execute the binary on the assigned node with an input file as the argument.

   Usage: `./bin/hipgisaxs <input-file-in-HiG>`.

   For example:

   ```
   $ ./bin/hipgisaxs inputs/test.27.hig
   ```

9. For more details on running interactive jobs on Dirac, please refer to the documentation available at: `http://www.nersc.gov/users/computational-systems/dirac/running-jobs/interactive`

6.2 Batch script on Dirac to use multiple GPU nodes

In order to use multiple GPU nodes on Dirac, you need to submit a PBS script for your job. Follow the following steps.

1. Note that HipGISAXS uses MPI for inter-node communication. Hence, you need to use `mpirun` or `mpiexec` commands.

2. Create a PBS job script, or modify the provided samples as per your needs. For example:

   ```
   $ cat myscript.pbs
   #PBS -q dirac_reg                  ## name of the queue
   #PBS -l nodes=4:ppn=1:fermi        ## number, type of nodes
   #PBS -l walltime=03:00:00          ## wall time limit
   #PBS -A gpgpu                      ## project identifier
   #PBS -N jobname.4                  ## job name
   #PBS -e jobname.4.$PBS_JOBID.err   ## error file name
   ```

```
#PBS -o jobname.4.$PBS_JOBID.out    ## output file name
#PBS -V                             ## inherit env variables

cd $PBS_O_WORKDIR
module unload pgi openmpi
module load openmpi-gnu/1.4.5 gcc/4.5.4 cuda/4.2
module load szip zlib
module load hdf5-parallel/1.8.3-gnu

mpirun -np 4 ./bin/hipgisaxs inputs/test.27.hig
```

3. In the submission script, make sure that the modules are loaded correctly before specifying the HipGISAXS execution command. The run command for HipGISAXS in the script may be as follows:

   ```
   mpirun -np <nodes> ./bin/hipgisaxs <input-file-in-HiG>
   ```

   where, `nodes` is the number of GPU nodes required (see the example script in previous step), and `input-file-in-HiG` is an input file in HiG format.

4. Submit the script:

   ```
   $ qsub myscript.pbs
   ```

5. Now the job is in the queue, and once it is finished, the standard error and output are saved into the corresponding filenames specified in the job script.

6. The output generated by HipGISAXS are all stored in a directory at the location and name specified in the input file (see inputs in Section 7).

7. For a detailed information on writing and submitting job scripts on Dirac, please refer to the documentation available at `http://www.nersc.gov/users/computational-systems/dirac/running-jobs/batch`.

6.3 Interactively on Titan

Again, this can wait.

6.4 Batch script on Titan to use multiple GPU nodes

This can also wait.

6.5 Interactively on a generic single-node Linux system equipped with a GPU (including `saxs-waxs-gpu` and `saxs-waxs-gpu2`)

1. Make sure all the pre-requisites are available on the system.
2. Execute the HipGISAXS binary as follows.

   Usage: `./bin/hipgisaxs <input-file-in-HiG>`

   For example,

   ```
   $ ./bin/hipgisaxs inputs/test.27.hig
   ```

6.6 Interactively on a generic multiple-node Linux cluster equipped with a GPU on each node

1. Make sure all the pre-requisites are available on all the nodes of the cluster.
2. Execute the binary as follows.

   Usage: `mpirun -np <nodes> ./bin/hipgisaxs <input-file-in-HiG>`

   For example,

   ```
   $ mpirun -np 4 ./bin/hipgisaxs inputs/test.27.hig
   ```

6.7 Interactively on a generic single-node Mac OS X system equipped with a GPU

On a Mac OS X system, follow the exact same steps as on a Linux system given in Section 6.5.

6.8 Interactively on a generic multiple-node Mac OS X cluster equipped with a GPU on each node

On a Mac OS X cluster, follow the exact same steps as on a Linux cluster given in Section 6.6.

CHAPTER

7 Inputs

The HipGISAXS binary takes as input a file in HiG format. Please refer to the detailed HipGISAXS documentation for details on the HiG format and how to write an input file. A few sample input files are located in the directory `inputs`, with extensions `.hig`. You may edit a sample input file as per your needs.

Some of the major components of the input file to update are the following.

1. The `name` attribute of `shape` object defines the input filename, as a string, containing triangulated shape surface data. It should point to the correct location of the shape file. This is a relative path. For example,

   ```
   ...
   shape = {
       ...
       name = "data/Shape_27.hd5", ...
   } ...
   ```

   This file should be in HDF5 format. Some sample shape definition files are provided in the directory `data`, with extensions `.hd5`.

2. The location of output to be generated needs to be defined though the `pathprefix` and `runname` attributes in the `computation` object. For example,

```
...
computation = {
    ...
    pathprefix = "myoutputs",
    runname = "shape27", ...
} ...
```

The `pathprefix` is a relative path to an existing directory.

The value of `runname` is appended with a timestamp, and a directory by this resulting name is created within the directory specified by `pathprefix`. All the generated output files by HipGISAXS simulation are stored in this directory.

In the above example, the location of generated output will be something like `myoutputs/shape27_20120927_142400/`.

3. The `resolution` attribute of `computation` object alters the resolution of the generated GISAXS simulation images in two-dimensions. This also affects the run time of HipGISAXS. Lower resolutions take lesser time, and higher resolutions take more time. It may be modified it as needed. For example,

```
...
computation = {
    ...
    resolution = [ 0.5 0.5 ], ...
} ...
```

4. You may further edit rest of the entries in the input file according to your desired experimental settings. For further details, refer to the Input Guide for HipGISAXS.

Chapter

8

Tutorial and Examples