

GADGET-2 Best Practices for Intel® Cluster Ready



BEST PRACTICES

1. Introduction

The following best practices document is provided as courtesy of the HPC Advisory Council.

2. Application Description:

* GADGET-2 - A code for collisionless and gasdynamical cosmological simulations

(GALaxies with Dark matter and GasintEraCT)

GADGET, including its sources and pointers to the authors can be found at:

<http://www.mpa-garching.mpg.de/gadget/>

3. Version Information:

Version of this date is used: GADGET version 2.0.7

4. Prerequisites:

The instructions from this best practice have been tested with the following configuration:

4.1 Hardware:

- Dell PowerEdge M610 38-node cluster
- Intel Xeon X5670 CPUs @ 2.93 MHz
- Memory: 24GB per node @ 1333MHz
- Mellanox ConnectX-2 QDR InfiniBand Adapters
- Mellanox QDR InfiniBand Switch

4.2 Software:

- Intel® Cluster Ready running RHEL 5.5
- Mellanox OFED 1.5.2 InfiniBand Software Stack
- Application: GADGET
- Compilers: Intel compilers
- MPI: Intel MPI 4, Open MPI 1.5.3, Platform MPI 8.0.1
- Benchmark workload: H2O-128.inp

5. Building GADGET

a. HDF5

You will need HDF5 version 1.6.x to avoid the latest changes in the HDF5 API. If you are using HDF5 version 1.8, you will run into error like this below:

```
io.c(774): error #165: too few arguments in function call
```

```
hdf5_headergrp = H5Gcreate(hdf5_
file, "/Header", 0);
```

^

Get the compiled versions of the previous release (1.6, not the current 1.8) of HDF5 from this location:

<http://www.hdfgroup.org/ftp/HDF5/prev-releases/>

b. ZLIB and SZIP

HDF5 will have a dependency on zlib and szip.

ZLIB is available in RHEL distribution as RPM. (zlib-devel-1.2.3-3). ZLIB is available through RHEL yum repository or RHEL installation DVD.

SZIP is available as source from HDF Group. You can download SZIP 2.1, extract, configure and build SZIP using these instructions:

```
http://www.hdfgroup.org/ftp/lib-external/szip/2.1/src/szip-2.1.tar.gz
```

```
% tar xvfz szip-2.1.tar.gz
```

```
% ./configure --prefix=<PATH> --enable-shared=no --enable-static=yes
```

I build static library only to avoid linking issue when compiling GADGET. Although you could build shared libraries but you may have to set LD_LIBRARY_PATH to specify the location of the SZIP libraries.

c. FFTW

For FFTW, make sure you are using FFTW 2.1.5 and not earlier. Also do not use FFTW version 3 because of the MPI support issue.

Also make sure FFTW is compiled using these flags if you want to compile both single and double precision FFTW library in the same distribution:

```
./configure --prefix=<PATH> --enable-type-prefix --enable-mpi
```

```
make
```

```
make install
```

```
make clean
```

```
./configure --prefix=<PATH> --enable-float --enable-type-prefix --enable-mpi
```

```
make
make install
```

Otherwise you will run into this error below:

```
pm_periodic.c(21): catastrophic error: could not open source file "srfftw_mpi.h"
```

```
#include <srfftw_mpi.h>
      ^
```

```
compilation aborted for pm_periodic.c (code 4)
```

d. GSL

The GSL, GNU Scientific Library, is available from this RPM (gsl-devel). You can either install from RHEL distribution or the YUM repository. Here is what you will run into if you don't have it installed.

```
mpicc -O3 -Wall -DPERIODIC
-DUNEQUALSOFTENINGS
-DPEANOHILBERT -DWALLCLOCK
-DPMGRID=128 -DSYNCHRONIZATION
-DHAVE_HDF5 -I/usr/common/pdsoft/
include -c -o main.o main.c
```

```
allvars.h(20): catastrophic error: could not open source file "gsl/gsl_rng.h"
```

```
#include <gsl/gsl_rng.h>
      ^
```

```
compilation aborted for main.c (code 4)
```

```
make: *** [main.o] Error 4
```

6. Makefile Change

a. Compiler Flags changes

In the Makefile, look for a line called SYSTYPE, around line 89.

```
SYSTYPE="MPA"
```

This is the SYSTYPE that for your configure setting, which you will need to define or modify the path to your library dependencies. You can either create a new SYSTYPE or modify the content for this SYSTYPE. This is around line 101:

```
ifeq ($(SYSTYPE),"MPA")
CC = mpicc
OPTIMIZE = -O3 -Wall
GSL_INCL =
GSL_LIBS =
FFTW_INCL= -I/application/fftw-2.1.5/install-gnu/
include
```

```
FFTW_LIBS= -L/application/fftw-2.1.5/install-gnu/
lib
```

```
MPICHLIB =
```

```
HDF5INCL = -I/application/hdf5-1.6.10-linux-
x86_64-static/include -I/application/szip-2.1/install-
gnu/include
```

```
HDF5LIB = -L/application/hdf5-1.6.10-linux-
x86_64-static/lib -L/application/szip-2.1/install-gnu/
lib -lhdf5 -lz -lsz
```

```
endif
```

For Intel compilers with Intel MPI, you need to use this:

```
CC = mpiicc
```

There's an optional change to allow the executable to link with static library, which allows the depending libraries to be compiled into the executable so it avoids linking the libraries at runtime. Line 226:

```
CFLAGS = $(OPTIONS) $(GSL_INCL) $(FFTW_INCL)
$(HDF5INCL) -static
```

b. Other Makefile options changes

In order to run GADGET-2 successfully, there are changes in both compile-time changes in the Makefile and also the run-time changes in the param file for the dataset. Changes in the Makefile during compile-time allow generation of highly optimized binaries by the compiler.

The following information is captured from the NGS site in the UK for the Makefile changes in GADGET-2.

<http://www.ngs.ac.uk/applications/astrophysics/gadget>

To create the Gadget2 executable for the **cluster** example:

```
#OPT += -DPERIODIC
OPT += -DUNEQUALSOFTENINGS
OPT += -DPMGRID=128
OPT += -DHAVE_HDF5
#OPT += -DDOUBLEPRECISION
#OPT += -DDOUBLEPRECISION_FFTW
```

To create the Gadget2 executable for the **galaxy** example:

```
#OPT += -DPERIODIC
OPT += -DUNEQUALSOFTENINGS
#OPT += -DPMGRID=128
OPT += -DHAVE_HDF5
#OPT += -DDOUBLEPRECISION
#OPT += -DDOUBLEPRECISION_FFTW
```

To create the Gadget2 executable for the **gassphere** example:

```
#OPT += -DPERIODIC
#OPT += -DUNEQUALSOFTENINGS
#OPT += -DPMGRID=128
OPT += -DHAVE_HDF5
#OPT += -DDOUBLEPRECISION
#OPT += -DDOUBLEPRECISION_FFTW
```

To create the Gadget2 executable for the **lcdm_gas** example:

```
OPT += -DPERIODIC
#OPT += -DUNEQUALSOFTENINGS
OPT += -DPMGRID=128
OPT += -DHAVE_HDF5
#OPT += -DDOUBLEPRECISION
#OPT += -DDOUBLEPRECISION_FFTW
```

To create the Gadget2 executable for the **test_large** example:

```
OPT += -DPERIODIC
#OPT += -DUNEQUALSOFTENINGS
OPT += -DPEANOIHILBERT
OPT += -DWALLCLOCK
#OPT += -DCPUSPEEDADJUSTMENT
OPT += -DPMGRID=2048
#OPT += -DHAVE_HDF5
#OPT += -DDOUBLEPRECISION
#OPT += -DDOUBLEPRECISION_FFTW
```

7. Running GADGET:

First, copy the directory "ICs" to the directory where you want to run:

```
% cd /lustre/home/pak/Gadget-2.0.7
% cp -fr ICs Gadget2/
% cd Gadget2/
```

To run GADGET using the sample dataset with Open MPI:

```
% pwd
<PATH>/Gadget-2.0.7/Gadget2
```

Create directory for running the test. For running gassphere as an example:

```
% mkdir parameterfiles/gassphere
```

To run:

```
% mpdboot --parallel-startup -r ssh -f <PATH_
TO_HOSTFILE > -n 32
% mpiexec -np 456 -IB <PATH>/Gadget2
<PATH>/parameterfiles/gassphere.param
%mpdallexit
```

For lcdm_gas.param

For running the gassphere dataset, you need to set this parameter .param file.

```
PeriodicBoundariesOn 1
% mkdir lcdm_gas
```

For cluster.param

If you run into this error below, you need to check your .param file for the boundary (TimeBegin and TimeMax)

```
gsl: qag.c:261: ERROR: could not integrate
function
Default GSL error handler invoked.
```

<http://www.mpa-garching.mpg.de/gadget/gadget-list/0421.html>

If you are seeing this message during runtime with 192 CPU cores, I need to increase the MAXTOPNODES in the allvars.h file and recompile.

```
We are out of Topnodes. Increasing the constant
MAXTOPNODES might help.
```

```
#define MAXTOPNODES 200000 /*!< Maximum
number of nodes in the top-level tree used for domain
decomposition */
```

```
#define MAXTOPNODES 800000 /*!< Maximum
number of nodes in the top-level tree used for domain
decomposition */
```

test_large.param

For the large dataset, you will need to request the benchmark from the developer.

