SPECFEM3D Tutorial for 2013 CIG-QUEST-IRIS Workshop

Carl, Qinya, Elliott, Emanuele

Last compiled: August 28, 2013

Overview

Document Conventions

1. In this document, commands you should run in your shell or code blocks you should edit are indicated by an framed grey box:

```
This is a code block.
```

2. Things you should type at the prompt are prefixed by a dollar sign $, and the output from commands is not. Comments are prefixed by a hash #.

```
$ echo "A␣command␣you␣can␣run" | sed 's/can/have/g' # This is a comment.
A command you have run
```

3. Lines that have been wrapped are indicated by ⇁ at the end of the line and ↪ at the beginning of the continued line. Line breaks do not occur in the middle of words, so remember to include a space between each wrapped line if you are typing it out yourself.

```
This is a very long line that will wrap because it is long and also a ⇁
    ↪ run-on sentence, too.
```

Log-in instructions

1. get account info from ARSC
2. ssh into ARSC (is X11 okay from mac? -X?)
3. Explain text editing (vi, gedit, emacs)

Step-by-step instructions, example 1: homogeneous halfspace synthetic seismograms (better call it forward simulation?)

The purpose of this example is to step through all the key steps of SPECFEM3D: get the code, configure the code for your cluster, generate a mesh using (GEO)CUBIT, partition the mesh, generate databases, run the solver, and check the output. The example is trivial, but it is important to remember that

1. check what modules are loaded by default

```
$ module list
Currently Loaded Modulefiles:
  1) pgi/13.4                 3) netcdf/4.3.0.pgi-13.4
  2) openmpi-pgi-13.4/1.4.3   4) PrgEnv-pgi/13.4
```

2. Check out two copies[1] of SPECFEM3D version 22719 from the code repository at CIG (http://www.geodynamics.org/)

```
$ svn co -r 22719 http://geodynamics.org/svn/cig/seismo/3D/SPECFEM3D/trunk →
    ↪ SPECFEM3D_22719_default
$ svn co -r 22719 http://geodynamics.org/svn/cig/seismo/3D/SPECFEM3D/trunk →
    ↪ SPECFEM3D_22719_socal
```

   (This may take a few minutes.)

3. check version info

```
$ cd SPECFEM3D_22719_default
$ svn info
Path: .
URL: http://geodynamics.org/svn/cig/seismo/3D/SPECFEM3D/trunk
Repository Root: http://geodynamics.org/svn/cig
Repository UUID: 42e91aa9-f6fe-0310-9bd8-dd834c9eb30a
Revision: 22728
Node Kind: directory
Schedule: normal
Last Changed Author: danielpeter
Last Changed Rev: 22719
Last Changed Date: 2013-08-20 06:16:30 -0800 (Tue, 20 Aug 2013)
```

   The *base directory* for the code is `SPECFEM3D_22719_default`, which we will call `SPECFEM3D` in the instructions.

4. Take a quick look at the user manual, mainly as a reminder to come back to it for details.

```
$ evince doc/USER_MANUAL/manual_SPECFEM3D_Cartesian.pdf &
```

5. Check that all software is available (or that modules are loaded):

```
$ which mpirun # OpenMPI
$ which cubit  # Cubit
$ which python # Python
```

6. Configure package, e.g. using the portland compiler:

```
$ ./configure F90=pgf90 MPIFC=mpif90
```

[1] There's no need for two copies, but they will make it a bit easier to keep track of the different exampled.

If successful, this command will generate several `Makefiles` in the `SPECFEM3D` main directory as well as subdirectories of verb+src/+, as well as `shared/constants.h` and `shared/precision.h`, among others.

7. Adapt the run scripts for your cluster and your example run.

   - copy run scripts from `utils/Cluster/` into `SPECFEM3D/`, e.g.,

```
$ cd utils/Cluster/pbs/
$ cp go_decomposer_pbs.bash go_generate_databases_pbs.bash ↵
   ↪ go_solver_pbs.bash ../../../
$ cd ../../../
```

   Set all bash scripts to run in the `standard` queue (in general, the queue will depend on your specific cluster) by editing the line (do we need to warn people to also delete one # from the line of ##PBS -q debug?)

```
#PBS -q standard
```

   (The default simulation time and number of cores is okay for this example.)

8. The rest of the instructions follow from the homogeneous halfspace example here

```
SPECFEM3D/examples/homogeneous_halfspace/README
```

   Copy the input files from examples directory into `SPECFEM3D/DATA/`

```
$ cd examples/homogeneous_halfspace/DATA/
$ cp * ../../../DATA/
```

   Note that only three input files are required: for the source (`CMTSOLUTION`), for the stations (`STATIONS`), and for the simulation parameters (`Par_file`).

9. create mesh:

   From the directory `SPECFEM3D/examples/homogeneous_halfspace`, open the CUBIT GUI, `claro`:

```
$ claro
```

   maybe warn people that the X11 forwarding can be slow? (Close the "Tip of the Day".) To ensure that the path is local and the needed python modules and scripts are accessible, `File --> Set Directory`, then click `Choose` (Choose what? examples/homogenous_halfspace/?), which is equivalent to typing in the CUBIT command window `cd 'examples/homogeneous_halfspace/'` (note the single quotes are required).

   Run the meshing script: from the Menu bar, select `Tools -> Play Journal File`, set `Files of Type` to `All Files`, then select `block_mesh.py`.

   I can't seem to get this working!

   If everything goes fine, this script creates the ten mesh files in subdirectory `MESH/`:

```
$ ls MESH
MESH/absorbing_surface_file_bottom
MESH/absorbing_surface_file_xmax
MESH/absorbing_surface_file_xmin
MESH/absorbing_surface_file_ymax
MESH/absorbing_surface_file_ymin
MESH/free_surface_file
MESH/materials_file
MESH/mesh_file
MESH/nodes_coords_file
MESH/nummaterial_velocity_file
```

You should be able to translate, rotate, and zoom on the mesh using a three-button mouse. (This can be emulated if you set X11 preferences, then (on a Mac) hold the `control`, `alt`, or `command` buttons while clicking and moving the mouse.)

The CUBIT graphics window should show a mesh similar to the file

`picture_of_this_homogeneous_regular_mesh.png`

10. decompose mesh files:

    Compile the decomposer in directory `SPECFEM3D/`:

```
$ make xdecompose_mesh
```

    This will compile the partitioner SCOTCH.

    Then run the decomposer:

```
$ qsub go_decomposer_pbs.bash
```

    You can check the status of the job with the command

```
$ qmap | grep USERNAME
```

    The job should take 20 seconds or so. It creates the four mesh partitions `proc000***_Database` in the directory `OUTPUT_FILES/DATABASES_MPI/`. The output file `OUTPUT_FILES/*.o` contains information on the partitioning.

11. generate databases:

    Compile `generate_databases` in the directory `SPECFEM3D/`:

```
$ make xgenerate_databases
```

    Submit the job:

```
$ qsub go_generate_databases_pbs.bash
```

    The job should take about a minute. It creates binary mesh files, e.g. `proc000***_external_mesh.bin` in the directory `OUTPUT_FILES/DATABASES_MPI/`.

    It is a good idea to look at the partitions of the mesh files. Load some vtk files (e.g., vs) into paraview:

```
$ cd OUTPUT_FILES/DATABASES_MPI/
$ module load paraview
$ paraview
```

Then `File --> Open`, and select all four `vs*vtk` files. When you are done, be sure to unload the `paraview` module, since here it was compiled with `gnu`, which conflicts with the `portland` we are using.

```
$ module unload paraview
$ cd ../../
```

12. run simulation:

    - compile specfem3D (from `SPECFEM3D/`):

```
$ make xspecfem3D
```

    - submit script to run solver:

```
$ qsub go_solver_pbs.bash
```

    The simulation runs on 4 cores and should take about 30 minutes. You can track the progress with the timestamp files generated in `OUTPUT_FILES/` (type `ls -ltr` to see the most recent files). When the job is complete, you should have 3 sets (semd,semv,sema) of 12 (`ls -1 *semd | wc`) seismogram files in the directory `OUTPUT_FILES`, as well as 51 `timestamp******` files.

13. Compare your computed seismograms with the reference seismograms.

    A quick visual comparison can be done from `SPECFEM3D/` using `xmgrace`:

```
$ module load grace
$ xmgrace examples/homogeneous_halfspace/REF_SEIS/*Z.semd &
$ xmgrace OUTPUT_FILES/*Z.semd &
```

Step-by-step instructions, example 2: homogeneous halfspace sensitivity kernel

1. QINYA'S INSTRUCTIONS HERE – note that there are some new scripts added (by Daniel?)

Step-by-step instructions, example 3: southern California

This example is at a scale that will likely not run on a single laptop or desktop computer. In other words, the required memory must be distributed over a number of different machines in order to run the simulation. Here we use 96 cores of the cluster.

1. Copy bash scripts into the base directory:

```
$ cd SPECFEM3D_22719_socal
$ cp ../SPECFEM3D_22719_default/*bash .
```

2. configure

```
$ ./configure F90=pgf90 MPIFC=mpif90
```

3. compile all

```
$ make all
```

4. Link the mesh directory as an example

```
$ cd examples
$ ln -s /import/c/d/ERTHQUAK/GEOCUBIT_MESH/socal_med400km .
```

5. Modify `go_decomposer_pbs.bash` to point to the new directory:

```
MESHDIR=examples/socal_med400km/MESH/
```

6. Link the tomography file and copy input files

```
$ cd DATA
$ ln -s /import/c/d/ERTHQUAK/MODEL/cvm119_1000_1000_0250_741_549.xyz →
   ↪ tomography_model.xyz
$ cp ../examples/socal_med400km/in_data_files/* .
```

7. modify `go_generate_databases_pbs.bash` and `go_solver_pbs.bash` to have the proper time limits and number of cores

```
#PBS -l nodes=6:ppn=16,walltime=1:00:00
#PBS -q standard
```

8. Follow the same steps as in Example 1: decompose, generate databases, solver. Here the programs have all been compiled, so only submitting the run scripts is needed. (But wait for each one to finish, and check the output before proceeding to the next step.)

   - The decomposer takes about 3 minutes.
   - The generate databases takes about 15 minutes.
   - The solver takes about 25 minutes.

   The simulation runs on 96 cores and should take about 25 minutes. You can track the progress with the timestamp files generated in `OUTPUT_FILES/` (type `ls -ltr` to see the most recent files). When the job is complete, you should have 3 sets (semd,semv,sema) of 1107 (`ls -1 *semd | wc`) seismogram files in the directory `OUTPUT_FILES`, as well as 4 `timestamp******` files.

   As expected, the seismograms contain numerical noise, since the source half duration in `CMTSOLUTION` was set to 0 s. This allows for maximal flexibility in post-processing, since the seismograms can be convolved with any source time function (see manual). However, it is important to know what the minimum resolving period of a particular mesh and model is, since these periods provide a guide for how to filter the seismograms in post-processing.

9. Now make a change to one of the input files in `SPECFEM3D/DATA/`, either `Par_file`, `CMTSOLUTION`, or `STATIONS`. Rename the `OUTPUT_FILES` directory if you do not want to over-write your previous output. Then submit the new job. (Note that no recompilation is needed.)

Step-by-step instructions, example 4: GPU

In this example, we show how SPECFEM3D can be used on a GPU cluster. In general, there are only two *required* steps to use GPU computing:

1. Enable CUDA during configuration:

```
$ ./configure --enable-cuda MPI_INC=-I$MPI_DIR/include
```

2. Enable CUDA at runtime in `DATA/Par_file`:

```
GPU_MODE    =   .true.
```

In practice, you may need to do several other steps depending on how your cluster has been set up. When running on your own cluster, you should consult their documentation in case there are additional steps that must be taken. The following example shows the extra steps that are required on the FISH cluster at ARSC.

1. *TODO*: start with login to fish
2. Check out version 22719 from SVN again:

```
$ svn co -r 22719 http://geodynamics.org/svn/cig/seismo/3D/SPECFEM3D/trunk →
   ↪ SPECFEM3D_22719_GPU
$ cd SPECFEM3D_22719_GPU
```

3. Load the CUDA toolkit:

```
$ module load cudatoolkit
```

4. The compilers on fish work a little differently than on pacman. There are wrappers that automatically run the real compilers and apply MPI or CUDA options once the modules are loaded.

   When compiling for CUDA, the CUDA compiler (which is separate from the C or Fortran compilers) needs to know where the MPI headers are installed. This information must be set using the `MPI_INC` variable when running `./configure`. On fish, loading the MPI module sets the `MPICH_DIR` environment variable, which is used in the instructions below.

```
$ ./configure MPIFC=ftn CC=cc FC=ftn --with-cuda MPI_INC=-I$MPICH_DIR/include
```

5. *TODO*: Copy example config stuff...
6. The GPU mode only affects the solver (`xspecfem3D`), but it is a good idea to remember to enable it now. Edit `DATA/Par_file` and set this option:

```
GPU_MODE    =   .true.
```

7. Now, you can compile things as usual:

```
$ make xdecompose_mesh
$ make xgenerate_databases
$ make xspecfem3D
```

8. The scripts to run jobs on fish are similar to the ones on pacman. Fish uses a slightly different method to run MPI programs compared to pacman. Instead of using `mpirun -np <#procs>`, you use `aprun -n <#procs>`, and to use GPUs, you must use the `gpu` queue.

```
$ cd utils/Cluster/pbs/
$ cp go_decomposer_pbs.bash go_generate_databases_pbs.bash ↪
    ↪ go_solver_pbs.bash ../../../
$ cd ../../../
$ vim *.bash
  # Change 'mpirun -np' to 'aprun -n'
  # Add '#PBS -q gpu'
```

   TODO: Add note about one process per node for GPU runs. This probably depends on which example we're using.

9. Now, you can submit your jobs as usual:

```
$ qsub go_decomposer_pbs.bash
# Wait for it to finish

$ qsub go_generate_databases_pbs.bash
# Wait for it to finish

$ qsub go_solver_pbs.bash
# Wait for it to finish
```

10. Depending on how many GPUs you use, you may see up to 20 or 30 times faster execution. If everything ran correctly on the GPU, you should see some additional files in `OUTPUT_FILES`:

```
$ ls OUTPUT_FILES/gpu*.txt
gpu_device_info_proc_000000.txt
gpu_device_mem_usage_proc_000000.txt
gpu_device_info_proc_000001.txt
gpu_device_mem_usage_proc_000001.txt
gpu_device_info_proc_000002.txt
gpu_device_mem_usage_proc_000002.txt
gpu_device_info_proc_000003.txt
gpu_device_mem_usage_proc_000003.txt
```

    These files contain statistics of the run on the GPU devices. If the jobs crash, you should check these files to ensure that you did not use too much memory per GPU.