Installing and running the Linux Cluster version of Tesseral-2D, 2.5D

I. Requirements

This module is designed to run on a 32-bit or 64-bit cluster architecture under Linux. Although the modules generally should run on any Linux system it is recommended to use RedHat Enterprise Linux 4,5,6,7 and derivatives (e.g. CentOS or Scientific Linux).

Hardware requirements:

- x86 compatible CPU (Intel or AMD)
- (recommended) 64-bit architecture supported
- 200 MB hard disk space for binaries
- 20 GB or more hard disk space for data
- 2 GB memory per CPU core
- Network connection between computers in the cluster

Software requirements:

- RedHat/CentOS Enterprise Linux of version 4 or higher (either 32 bit or 64 bit). Other Linux OSes may require additional support.
- Shared disk storage (NFS, SAMBA, etc.) accessible via the same path under every node
- (optional) Compilers: GNU C++ (g++) of version 3.4.6 or higher to build from library.
- (optional) An MPI implementation (to build from library). We tested the software mostly on LAM MPI 7.1.4 and installation and setup instruction of LAM MPI is described in this manual, but we suggest the software will run on most MPI implementations (MPICH, Open MPI, Scali, etc.)

The data to be processed must be either put in a shared directory accessible to each node or copied to each the node so to have the same pathname.

II. Install Tesseral LC

Use following command to unpack installation package:

```
tar –xzvf ./tesseral-*-x.x.x.tar.gz
```

Use precompiled binaries

Tesseral LC installation package comes with prebuilt binary files in 'precompiled' directory. These files are ready to be executed on a cluster with **LAM MPI v7.1.4.** If for some reason you want to use different MPI implementation please read following chapter to build a binary using arbitrary MPI implementation.

Table 1. Tesseral 2D precompiled binary files

Tesseral2D-32.lexe	Binary for x86 32-bit Linux (i386/i586)
Tesseral2D-64.lexe	Binary for x86 64-bit Linux (AMD64/EM64T)
Tesseral2D-GPU-32.lexe	Binary for 32-bit Linux with GPU/CUDA/OpenCL support
Tesseral2D-GPU-64.lexe	Binary for 64-bit Linux with GPU/CUDA/OpenCL support

Table 2. Tesseral 2.5D precompiled binary files

Tesseral25D-32.lexe	Binary for x86 32-bit Linux (i386/i586)
Tesseral25D-64.lexe	Binary for x86 64-bit Linux (AMD64/EM64T)

For 2.5D simulation for CUDA binary a binary file must be ran with 'lbcudart.so.4' file located in the same directory. It is runtime library for CUDA.

Table 3. Tesseral 2.5D GPU/CUDA precompiled binary files

Tesseral25D-CUDA-32.lexe	Binary for x86 32-bit Linux (i386/i586)
Tesseral25D-CUDA-64.lexe	Binary for x86 64-bit Linux (AMD64/EM64T)

Building from library

If you've installed Tesseral LC via precompiled binaries you may skip this chapter. The script to build from library is located in the root of the unpacked directory.

1. Execute the following file in this directory:

```
      ./mpi_build.sh
```

   The script should automatically identify MPI available MPI implementation by searching for "mpiCC" of "mpic++" in you your $PATH. If it is not the case, edit the file prior to execution. Also consult your compiler manual for a list of available optimization options. You may improve the compiler optimization if you like to.

   Take into account that all the Tesseral libraries included in the installation set can be incompatible with your Linux or MPI libraries. In this case please contact us via the e-mail tesseral.geo@gmail.com. See more installation notes in Appendix C.

   Upon compilation, the executable **Tesseral2D.lexe** will be created in the **bin** directory. For 2.5D similar binary file will be created.

2. Create a directory for installing the **Tesseral2D.lexe**. You can do this in two different ways:

   a. Create a directory, e.g. **tesseral**, on one of the cluster nodes and make it directly accessible to every node of the cluster (e.g. with NFS).

   b. Create directories with the same name, e.g. **tesseral**, on every node of the cluster. You can use the **rsync** Linux utility to do this. You can also use **rsync** to make copies of the data to be processed (see the next section).

3. Copy the **Tesseral2D.lexe** executable file created at the step 2 to the directory (-ies) created at the step 3.

III. Preparing data to be processed

1. Prepare the data by invoking the menu command 'Run/CLUSTER: Create task' in either Tesseral-2D or Tesseral Pro desktop Windows program. In the appeared dialog adjust all necessary options (see Appendix A or your desktop program manual for details) and choose directory for storing the data that you'll have to copy to the cluster.

2. The command prepares a runtask.ini file and several more files. The **runtask.ini** file has a simple structure described in Appendix B and can be edited manually if needed. To edit it use any text editor and save the result as a plain ASCII text file.
3. Put the files created in step 1 onto every cluster node in the directory where the executable resides (e.g. the **tesseral** directory). Apart to the **runtask.ini**, the data files are as follows:
   – for modeling: a Tesseral's TAM format <modelname>**.tam** file with the polygon model and optionally a grid model file in Tesseral's TGR or SEG-Y; optionally up to three SEG-Y files of the TTI anisotropy parameters 'epsilon', 'delta' and axis angle 'phi' (ε, δ, φ)

– for migration: a grid model file in TGR or SEG-Y format, a seismogram file in TGR or SEG-Y format and a Tesseral's BAS file with the migration aperture description; optionally a time file created at the modeling step in TGR format.

IV. Running the computations

1. Change Linux current directory to the one containing the file **`runtask.ini`**.
2. Make sure LAM MPI binaries are in the $PATH:

```
export PATH=$PATH:/<path-to-lam>/bin
```

3. If you use ssh for passwordless remote login between cluster nodes use following command for LAM MPI:

```
export LAMRSH="ssh -x"
```

4. Boot up MPI Environment (required for LAM MPI, may be optional for other MPI implementations)

```
lamboot -v -prefix <path-to-lam> <hostfile.txt>
```

5. Run the **`Tesseral-64.lexe`** or similar executable in the MPI environment. Usually, you must be signed in as an ordinary user, not as a root. The sample command is

```
mpirun -np <a number of processors to involve> <path>/Tesseral-64.lexe
```

   (see your MPI documentation for details).

   To run modelling on all processors of the cluster run (LAM MPI specific):

```
mpirun C <path>/tesseral2d-64
```

   (this works only under LAM MPI environment that is shipped with Tesseral LC)

6. If you run the program for the first time it generates a license request and stops. You have to obtain and install a license file (see the next section).
7. After the program has finished its parallel calculations the resulting files are stored in the current directory.

V. Licensing the package

Licensing Tesseral LC on fixed machines

1. On the program first run a passport file called **`app_passport`** (for 2d) and/or **`m25_passport`** (for 2.5d) is created. It contains the outgoing license codes. Transfer these files to your Tesseral software distributor and request a license file.
2. The software distributor must provide you with the license file called **`app_license`** (for 2d) or **`m25_license`** (for 2.5d).
3. Put the license file to each node in the Tesseral directory (or in the shared directory where the executable resides).
4. The license is given for a certain number of nodes. If you are trying to run the package on more nodes than your license allows, the processes on undue nodes immediately stops and don't participate in the computations.

Licensing Tesseral LC on floating machines

Tesseral LC uses Sentinel™ RMS license manager to manage floating machine licenses.

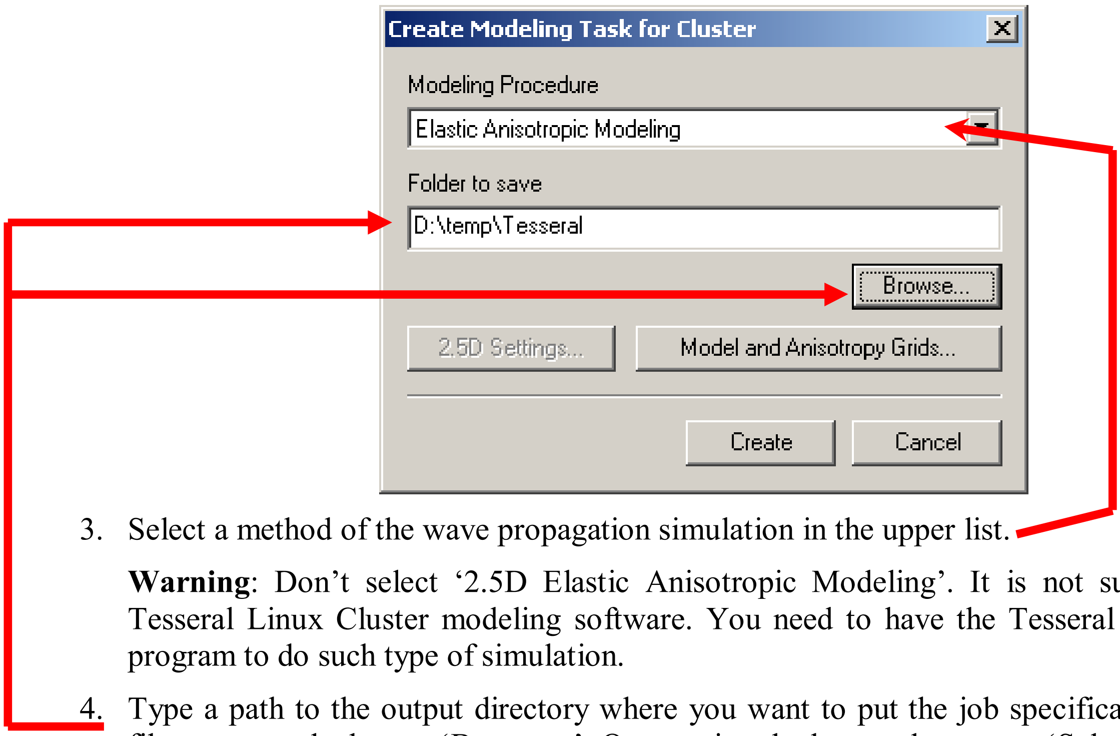
1. To specify a server to request licenses from create a file 'app_network' (for 2d) and/or 'm25_network' (for 2.5d) in your Tesseral LC program directory.

2. In this file type a server name or IP-address.
3. Optionally you can specify a port by using colon (example: “hostname:5093”)

In the case of any problems with licensing please contact Tesseral Technologies Inc. via e-mails tesseral@shaw.ca or tesseral.geo@gmail.com. See more contact information on the web-site www.tesseral-geo.com.

Appendix A. Preparing the Modeling Jobs

1. Load or create a polygonal model (see Tesseral-2D or Tesseral Pro manual for more details).
2. In Tesseral-2D 6.0 or Tesseral Pro select the menu ‘Run’, menu item ‘CLUSTER: Create Task...’. The next dialog appears:

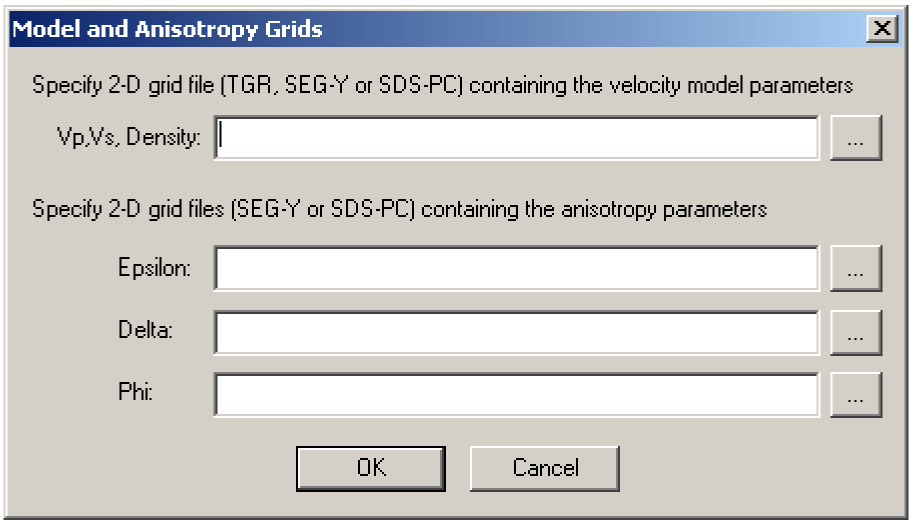


Modeling Procedure: Elastic Anisotropic Modeling

Folder to save: D:\temp\Tesseral

3. Select a method of the wave propagation simulation in the upper list.

   Warning: Don’t select ‘2.5D Elastic Anisotropic Modeling’. It is not supported by the Tesseral Linux Cluster modeling software. You need to have the Tesseral 2.5D Modeling program to do such type of simulation.
4. Type a path to the output directory where you want to put the job specification and related files or press the button ‘Browse...’. On pressing the button the system ‘Select Folder’ dialog appears.
5. If the model you want to process is partially presented in a grid format like SEG-Y or TGR files, press the button ‘Model and Anisotropy Grids...’ for more settings. The next dialog appears:



Specify 2-D grid file (TGR, SEG-Y or SDS-PC) containing the velocity model parameters

Vp,Vs, Density:

Specify 2-D grid files (SEG-Y or SDS-PC) containing the anisotropy parameters

Epsilon:

Delta:

Phi:

Else go to the step 7.

6. Type paths to the TGR or SEG-Y input files or press correspondent '...' buttons to browse directories for the files. Press 'OK' to finish.
7. Press the button 'Create...' to create the job specification in the selected directory. Then you can transmit all the job files on your Linux cluster for calculations.

Appendix B. Modeling Job Specification Format

The Tesseral.exe program execution is controlled by the job passport. The job passport is a text file **`runask.ini`** located in the current work directory.

The job passport consists of two sections. First section is started from the string '`[TASK]`', second one is started after '`[modeling]`'. Section name is put here in square brackets. Each section contains a set of parameters in format 'name=value'. A passport sample is shown below:

```
[TASK]
taskType=modeling
[modeling]
Model Name=Model.tam
First Point=31
Last Point=35
Run Computation=Elastic Anisotropic
Raster Model=model_P-0.tgr
Anisotropy Epsilon File=
```

The passport parameters values are supposed to be changed but the parameters and section names are fixed and case sensitive words. The parameter sequence in the bounds of each section does not meter.

`taskType` is a parameter determining the job type. Its possible values are 'modeling' and 'migration'. For the modeling tasks first variant is required.

`Model Name` is a pathname to the model file in the Tesseral's TAM format. It is an obligatory parameter. You can specify either relative or absolute path to the file. Automatically generated passports always contain local paths of a file name only as all necessary files are copied in the same directory.

`First Point` and **`Last Point`** specify diapason of shots to process. So you can narrow the diapason defined in the model. But you can't expand it as the program calculates shotgathers and snapshots for intersection of these two diapasons. These parameters are optional and can be omitted.

`Run Computation` is an obligatory parameter with a fixed set of possible values: '`Elastic Anisotropic`', '`Elastic`', '`Acoustic`', '`Scalar`', and '`Vertical Incidence`'. The values are case sensitive.

`Raster Model` is an optional path to the main grid (raster) model file in Tesseral's TGR or SEG-Y format. SEG-Y file has to contain compression wave velocities in m/sec. TGR can contain up to 3 useful components: compression wave velocity, shear wave velocity and density. In any case the program interpolate missing components according to built-in dependencies.

`Anisotropy Epsilon File`, **`Anisotropy Delta File`** and **`Anisotropy PhiMedia File`** are optional file paths to the anisotropy components in Tesseral's TGR or SEG-Y format. In the case of TGR each correspondent component should persists in the file. You can use the same TGR file for both primary and anisotropy model parameters, but all the 4 or 6 parameters should be included in it as components. Take into account that anisotropy parameters are ignored in any sort of simulation except for the ‘Elastic Anisotropic’ one.

`MultiTamFiles` is an optional parameter with the default value of 0. If its value is non-zero different models are used for different sources. This parameter is included to support complex jobs of Tesseral Pro and is not recommended for manual editing.

Appendix C. How to Install the Execution Environment

Tesseral-2D Modeling for Linux cluster may be run on most HPC architectures with MPI enabled. Environment installation is required only if the cluster don’t have one of the required software components. The components are:

– C++ Compiler (GNU C++, Intel C++, etc.)

– Shared storage (NFS, SAMBA, GFS, Lustra, etc.)

– MPI (LAM MPI, OpenMPI, MPICH, etc.)

– (optional) Resource Manager (OpenPBS, Torque, Slurm, etc.)

Cluster can have various configurations. To be more specific we’ll use node imaginary names: `Node1 - Node8` and a system user, responsible to run the computations: “`tesuser`”. In the guide we provide raw list of the commands and some of the output. These sections have grey background. The commands that you need to enter by hand are marked by bold font.

Configure the execution environment

You can check wherther LAM MPI is available on the node by running the next command:

```
[root@node1 /]# laminfo
             LAM/MPI: 7.1.4
              Prefix: /usr/local
        Architecture: i686-pc-linux-gnu
...
             SSI rpi: sysv (API v1.0, Module v7.1)
             SSI rpi: tcp (API v1.0, Module v7.1)
             SSI rpi: usysv (API v1.0, Module v7.1)
              SSI cr: self (API v1.0, Module v1.0)
[root@node1 /]#
```

To configure LAM you need to modify the file **/opt/tesseral/rh3x32/etc/lam-bhost.def**[1]. It is your default hostfile used when you run the program. You need to configure the file only on the node from which you want to run computations. An example of **lam-bhost.def** for a 8 node cluster of 2 CPU per node is shown below.

```
node1 cpu=2
node2 cpu=2
node3 cpu=2
node4 cpu=2
node5 cpu=2
node6 cpu=2
node7 cpu=2
node8 cpu=2
```

[1] Note. Depending on your Linux version and hardware your lam directory name may vary (e.g. rh3x64, rh4x32, etc.)

To check the configuration run **lamboot** and make sure the output is ok. You must run this command by non-root user account:

```
[tesuser@node1 data]# lamboot

LAM 7.1.4/MPI 2 C++/ROMIO - Indiana University

[tesuser@node1 data]#
```

If there are error messages make sure you have rsh-service available on all cluster nodes and lamboot executable is located in one of the PATH directory:

```
[tesuser@node1 data]# rsh node2 which lamboot
connect to address 192.168.1.2: Connection refused
Trying krb4 rsh...
connect to address 192.168.1.2: Connection refused
trying normal rsh (/usr/bin/rsh)
/usr/bin/lamboot
[tesuser@node1 data]#
```

How to install GNU C++ version 3.4.6 if needed

Installation of GNU C++ is required only of you have version older then 3.2.3. If you have GNU C++ version 3.2.3 and higher, you can use it and skip this chapter. Installation of GNU C++ is required only on one computer.

Releases of most versions of GNU C++ can be obtained on ftp://ftp.gwdg.de/pub/misc/gcc/. We recommend version of 3.4.6 if you have gcc older then 3.2.3.

We recommend you to install gcc not to default directory to save your current version of gcc. To install the gcc use the following procedure:

```
[root@node1 gcc]# tar -xvzf gcc-3.4.6.tar.gz
...
[root@node1 gcc]# mkdir gcc-bin
[root@node1 gcc]# cd gcc-bin
[root@node1 gcc-bin]# ../gcc-3.4.6/configure \
   --prefix=/usr/local/gcc-3.4.6 --program-suffix=346 \
   --enable-shared --enable-languages=c,c++,f77,objc \
   --enable-version-specific-runtime-libs
...
[root@node1 gcc-bin]# make
...
[root@node1 gcc-bin]# make install
...
[root@node1 gcc-bin]#
```

To compile the Tesseral you may need to add the next line to **/etc/profile** :

```
PATH=$PATH:/usr/local/gcc346/bin
```

and change the compilation script a bit. If you installed gcc with 346 suffix you may need to change **make*.sh** script from Tesseral installation line from

```
LAMMPICXX=g++
```

to

```
LAMMPICXX=g++346
```

How to setup rsh service

Edit **`.rhosts`** file in tesuser's directory (**/home/tesuser/**). Specify the nodes which have rsh access to current node. .rhosts file should look like this:

```
node1     tesuser
node2     tesuser
node3     tesuser
node4     tesuser
node5     tesuser
node6     tesuser
node7     tesuser
node8     tesuser
```

Make sure you have rsh is listed in **`/etc/securetty`** file.

How to setup NFS service

To run Tesseral modeling you need a shared storage available. You can use Lustra, GFS, SAMBA, NFS, etc. In this guide we describe how to setup NFS with `Node1` as server and other nodes as clients.

If you don't have any shared storage installed select a node to be NFS server (e.g. `node1`), create a directory to be shared and specify in **`/etc/exports`** the text parameter value:

```
/data *(rw,sync)
```

Run the next command to apply the settings.

```
[root@node1 /]# chkconfig --levels=35 nfs on
[root@node1 /]# service nfs restart
...
[root@node1 /]#
```

To setup a NFS client enter the next line in **/etc/fstab** file on each client node:

```
node1:/data       /data      nfs       rw,rsize=8192,wsize=8192
```

To establish the shared folder connection use the command:

```
[root@node1 /]# mount /data
```