



## *Installing and Running of Tesseral LC 2D and Tesseral LC 2.5D-3C Parallel Computations Engine for Linux Cluster*

---

### PRODUCT INFORMATION AND INSTALLATION NOTES

For more info about Tesseral and related products, and to download the latest updates, please visit the [Tesseral Technologies](http://www.tesseral-geo.com) website

### *Contents*

I. REQUIREMENTS .....	1
II. INSTALLING TESSERAL LC .....	2
Installing precompiled binaries .....	2
Building from library .....	2
III. PREPARING DATA TO BE PROCESSED .....	3
IV. RUNNING THE COMPUTATIONS .....	3
V. LICENSING THE PACKAGE .....	4
Licensing Tesseral LC on fixed machines .....	4
Licensing Tesseral LC on floating machines .....	4
APPENDIX A. PREPARING THE MODELING JOBS .....	5
APPENDIX B. MODELING JOB SPECIFICATION FORMAT .....	6
APPENDIX C. HOW TO INSTALL THE EXECUTION ENVIRONMENT .....	7
Configure the execution environment .....	7
How to install GNU C++ version 3.4.6 if needed .....	8
How to setup rsh service .....	9
How to setup NFS service .....	9

## **I. Requirements**

This module is designed to run on a 32-bit or 64-bit cluster architecture under Linux. Supported RedHat Enterprise Linux 3,4,5 and derivatives (e.g. CentOS).

Hardware requirements:

- CPU of Pentium III or higher
- (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 3 or higher (either 32 bit or 64 bit)
- Shared disk storage (NFS, SAMBA, etc.) accessible via the same path under every node
- (optional) Compilers: GNU C++ (g++) of version 3.2.3 or higher or Intel C++ Compiler (icpc) of version 9.1 or higher (to build from library)
- (optional) An MPI implementation (to build from library). We tested the software mostly on LAM MPI 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, OpenMPI, 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. Installing Tesseral LC

\* For general installation instructions, please see "[1\\_1\\_ReadMe for Installation of Tesseral products.pdf](#)".

Tesseral LC can be installed using precompiled binary packages (RPM) or by building a program from a library. First, untar the archive package:

```
tar -xvzf ./tesseral-lc-x.x.x.zip
```

### Installing precompiled binaries

To install the precompiled binaries select an RPM package that best suits your machine's configuration and run:

```
rpm -ivh <RPM-file>
```

For example,

```
rpm -ivh tesseral-lc-rh3-em64t.rpm
```

The program will be installed into `/opt/tesseral/rhXxXX/` directory (XxXX is your architecture's abbreviation, e.g. rh3x64).

### Building from library

If you've installed Tesseral LC via RPMs you may skip this chapter. The scripts to build from library are located in `builds/tesseral2d` or `builds/tesseral25d` directory of the archive.

1. Execute the appropriate make file in this directory:

```
./make*.sh
```

These make files assume that your MPI C++ compiler's mnemonic is mpiCC.

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 can.

Next make files are included together with Tesseral libraries:

**make32g323.sh** for 32-bit x86 CPU, GNU C++ compiler v.3.2.3

**make32g346.sh** for 32-bit x86 CPU, GNU C++ compiler v.3.4.6

**make32g412.sh** for 32-bit x86 CPU, GNU C++ compiler v.4.1.2

**make64g323.sh** for 64-bit EM64T CPU, GNU C++ compiler v.3.2.3

**make64g346.sh** for 64-bit EM64T CPU, GNU C++ compiler v.3.4.6

**make64g412.sh** for 64-bit EM64T CPU, GNU C++ compiler v.4.1.2

Take into account that all the Tesseract 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 [tesseract.geo@gmail.com](mailto:tesseract.geo@gmail.com). See more installation notes in Appendix C.

Upon compilation, the executable **tesseract2d-32** or **tesseract2d-64** will be created in the **bin** directory.

2. Create a directory for installing the **tesseract2d-32** or **tesseract2d-64** executable. You can do this in two different ways:
  - a. Create a directory, e.g. **tesseract**, 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. **tesseract**, 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 **tesseract2d-32** or **tesseract2d-64** 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 Tesseract 2D or Tesseract 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 **tesseract** directory). Apart to the **runtask.ini**, the data files are as follows:
  - for modeling: a Tesseract’s TAM format **<modelname>.tam** file with the polygon model and optionally a grid model file in Tesseract’s TGR or SEG-Y; optionally up to three SEG-Y files of the TTI anisotropy parameters ‘epsilon’, ‘delta’ and axis angle ‘phi’ ( $\epsilon, \delta, \varphi$ )
  - for migration: a grid model file in TGR or SEG-Y format, a seismogram file in TGR or SEG-Y format and a Tesseract’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. Load Tesseract LC environment variables and initialize LAM MPI:

```
source /opt/tesseract/rhXxxx/tessvars.sh
```

This command should be run under common (not root) user account.

3. Run the **Tesseract.exe** (for 2D) or **Tesseract25D.exe** (for 2.5D) (-xx bit extension may be present in the name) 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> **Tesseral<var>.exe**

(see your MPI documentation for details).

To run modelling on all processors of the cluster run:

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

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

4. 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).
5. After the program has finished its parallel calculations the resulting files are stored in the current directory.
6. (2D and 2.5D) After the program has finished its parallel calculations the resulting files are stored in the current directory. Three partial velocity components (Vx, Vy (2.5D) and Vz) as well as normal stress Uc are stored as seismograms in SEG-Y format. Snapshots for each shot point are stored as Tesseral TGR files to be viewed and played in Tesseral<var> Windows Workplace.
7. (2.5D) After calculation of a model a big volume of temporary files may be stored. They are K2 sections (K2 is a space frequency along Y) of both shotgathers and snapshots stored in a new directory called like the model file but without the extension **‘.tgr’**. The program re-uses these files on small changes of the observation system like other Y spacing of the receiver lines or another snapshot offset. If the initial model observation calculated snapshots, you can change the snapshot parameter ‘Every’ not re-computing the K2 sections. Else remove them.

## 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

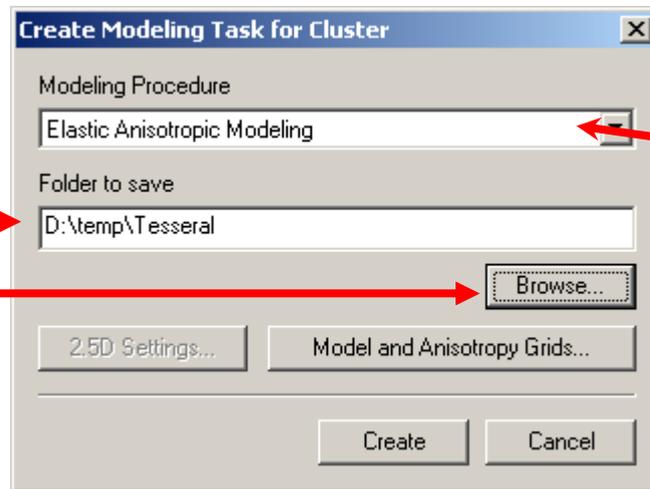
Tesseral LC uses Sentinel™ RMS license manager to manage floating machine licenses.

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 (usually /opt/tesseral/rhXxXX).
2. In this file type a server name or IP-address.

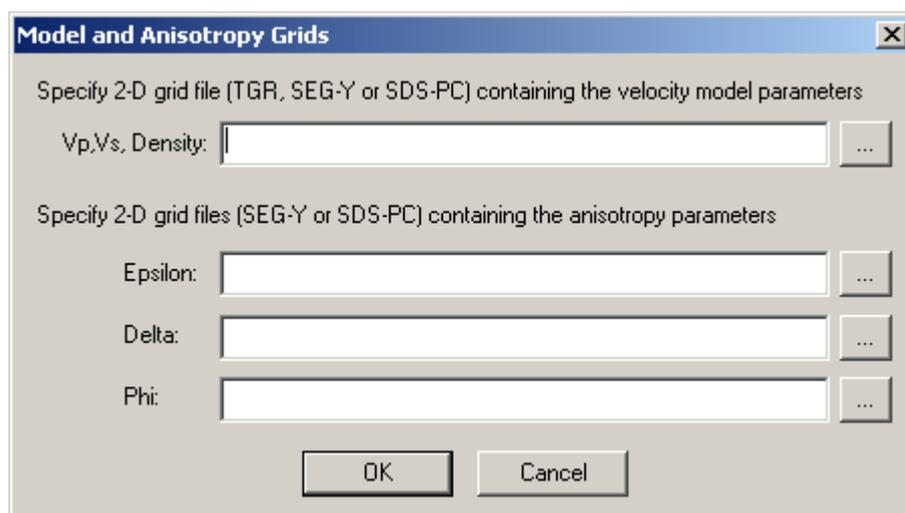
In the case of any problems with licensing please contact Tesseral Technologies Inc. via e-mails [tesseral@shaw.ca](mailto:tesseral@shaw.ca) or [tesseral.geo@gmail.com](mailto:tesseral.geo@gmail.com).

## Appendix A. Preparing the Modeling Jobs

1. Prepare the data by invoking the command `Run>CLUSTER: Create task...(or Run>Migration (Depth Domain)>CLUSTER: Create task...)` in Tesserall Workplace.
2. In the dialog that opens, adjust all necessary options (see Master User Manual for details) and choose or create the folder for storing the data that you'll have to copy to the cluster. The next dialog appears:



3. Select a method of the wave propagation simulation in the upper list.  
**\* Note:** 3D TTI anisotropy and fracturing parameters can be specified in the model polygon properties for 2.5D modeling.  
**\*\*Note:** You need to have Tesserall 2.5D-3C Linux Cluster modeling software program to do such type of simulation for practical applications.
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:



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. The command prepares the `runtask.ini` file and several more files. The task file has a simple structure and can be edited manually if needed in any text editor and saved as plain ASCII text file.
8. Put the files created in *step 1* onto every cluster node in the directory where the executable resides (e.g. `/Tesseral` directory). In addition to the `runtask.ini`, the data files are as follows:
  - a. 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 2D TTI anisotropy parameters ‘epsilon’, ‘delta’ and axis angle ‘phi’ ( $\epsilon$ ,  $\delta$ ,  $\phi$ ). More complex 3D TTI anisotropy (using both axis angle and azimuth ) as well as 3D fracturing can be specified only as polygon properties in the TAM model
  - b. 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.

## 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 Tesseract’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 Tesseract’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 Tesseract Pro and is not recommended for manual editing.

## Appendix C. How to Install the Execution Environment

Tesseract-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 whether 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/tesseract/rh3x32/etc/lam-bhost.def<sup>1</sup>**. It is your default hostfile used when you run the program. You need to configure the file only on the

<sup>1</sup> Note. Depending on your Linux version and hardware your lam directory name may vary (e.g. rh3x64, rh4x32, etc.)

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
```

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 if you have version older than 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 than 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 Tesseract 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 Tesseract 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 Tesseract modeling you need a shared storage available. You can use Lustre, 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
```