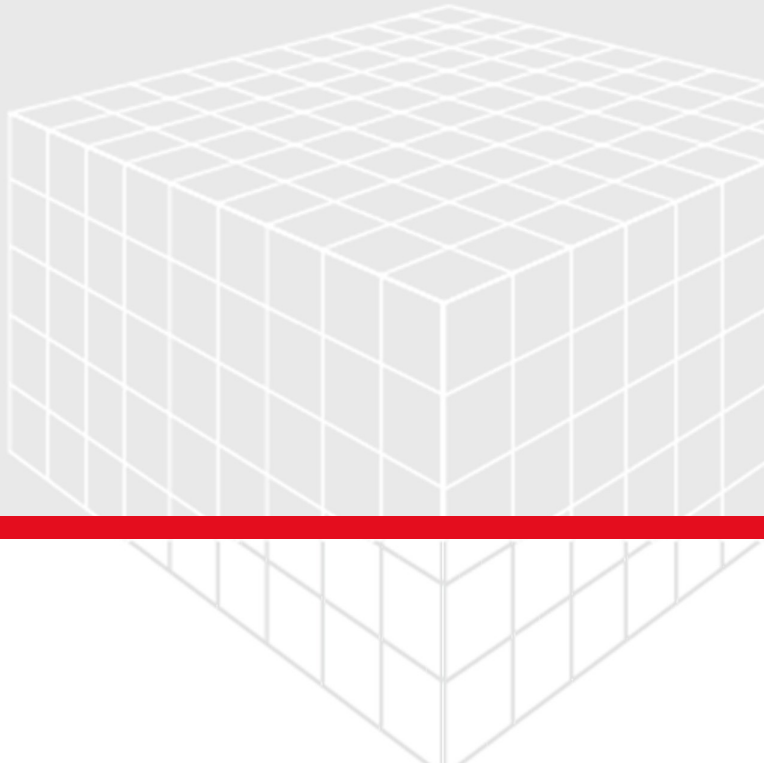
High Performance Computing

User Guide



GEODICT Release 2026

Published: June 2026

GeoDict

https://doi.org/10.30423/userguide.geodict

High Performance Computing

1 High Performance Computing

Three-dimensional computer tomography images can consist of 2000^3 or even more voxels. GeoDict can compute flow, thermal, electrical, and mechanical properties for large datasets. In these computations, multiple floating point values per grid cell must be computed and stored. Since 2000^3 already means 8 billion grid cells, these computations require strong computational resources. These simulations require:

1. a large amount of memory (RAM),
2. a high number of cores to be computed in a short amount of time and
3. a large hard drive to store the simulation results.

These requirements can be met by large shared memory machines, or computer clusters. Often, such computational resources are not available locally, and there is a need for on-demand availability of computer resources in the cloud.

Shared Memory Machines

Large shared memory machines are single computers with a large amount of RAM, many CPU cores, and one or more large hard drives. These machines have the advantage that they are relatively cheap to purchase and easy to maintain. They allow you to perform and visualize simulations on very large structures. Shared memory machines are the best option for most of the application cases. Recommendations about the hardware configurations for different use cases can be found at:

https://www.geodict.com/service-support/technical-support/system-requirements.html

The operating system on these machines may be Windows or Linux. A GeoDict installation on such a shared memory machine can be used by several users. You can use a remote desktop connection 12 to connect to this computer, or you can use GeoDict's built-in job queue system to submit compute jobs from a local desktop to this computer.

Be aware that multiple licenses are required if several users want to use GeoDict at the same time or if a single user wants to use GeoDict on several computers at the same time (local desktop and remote server).

Computer Clusters

A computer cluster typically consists of compute nodes with the same hardware configuration. They communicate with each other using a very fast interconnection (e.g., InfiniBand).

A floating license is required to use GeoDict on such clusters. If several users want to use GeoDict at the same time, or if a single user wants to submit several GeoDict computations to the clusters job system at the same time, they will need multiple licenses.

GeoDict uses MPI (Message Passing Interface) to exchange data between different compute nodes (distributed memory). For this, a supported MPI version 9 has to be installed on the cluster. Not all of GeoDicts solvers are able to use multiple compute nodes, see the Cluster Computing 15 topic for details.

Learn how to use high-performance computing with GeoDict.

- Parallelization Options 5
  Description of the Parallelization settings dialog, that lets you select the number of parallel processes used in many GeoDict commands.
- MPI Configuration 9
  Learn how to install MPI on a Linux computer and how to select the appropriate MPI version.
- Remote Desktop 12
  To enable 3D rendering on a remote Linux computer, install TurboVNC and Virtual GL.
- Cluster Computing 15
  Use the PBS or SLURM job system to submit jobs on a distributed memory Linux cluster.

Join the conversation

Visit the Community on GeoDict Forum to be inspired and get answers to top questions.
Find illustrative application examples on the website.
Our tutorials and seminar videos give you a first hands-on experience. They are collected in the Learning Center.
Send your Support Request to our technical Support.

1.1 Parallelization Options

GeoDict commands can use parallelization as a method to speed-up computational processes. For those commands, you can choose how many parallel processes should be used for the parallelization in the command's **Options** dialog. Using multiple processes usually requires additional licenses for each process.

Know how! The parallelization of a solver can use three different technical methods:

1. Local-MPI parallelization,
2. Distributed-MPI parallelization, and
3. Local-Thread parallelization.

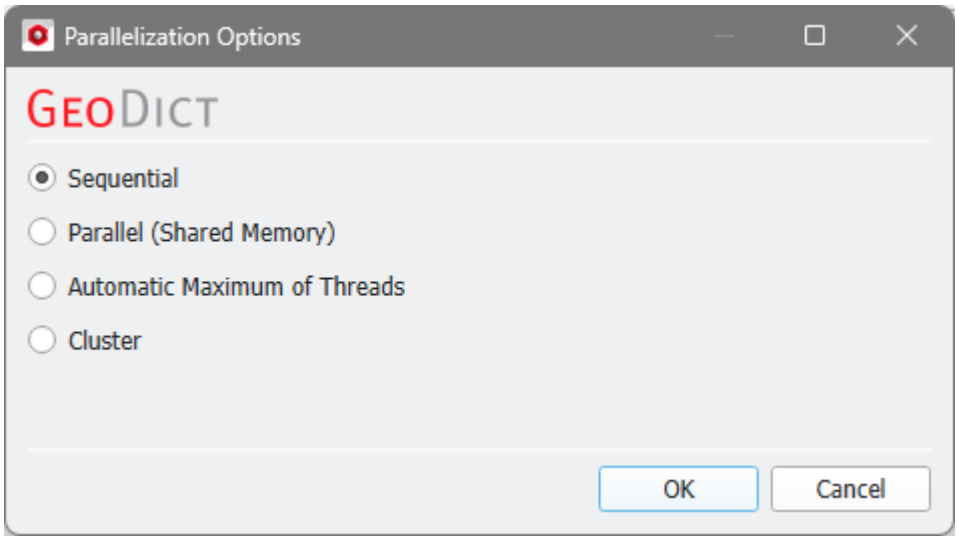
MPI stands for Message Passing Interface and is a standardized and portable message-passing standard designed by a group of researchers from academia and industry to function on a wide variety of parallel computing architectures. Solvers that use MPI are started multiple times as different instances. Each instance performs a simulation on a sub-volume of the whole structure. MPI is used to send data of intermediate results between the running instances. Local-MPI parallelization works within the same computer while Distributed-MPI works across different computers. When Local-MPI is used data can be exchanged very efficiently while Distributed-MPI has to use Ethernet or fast interconnection like InfiniBand to send data between different computers.

Local-thread parallel solvers are started only once per simulation. Thus, no data has to be exchanged between different instances. But these solvers cannot use multiple compute nodes of a cluster to distribute the simulation data.

Click on the **Parallelization**'s **Edit...** button to open the Parallelization Options dialog. In this dialog, you can select between up to four different modes:

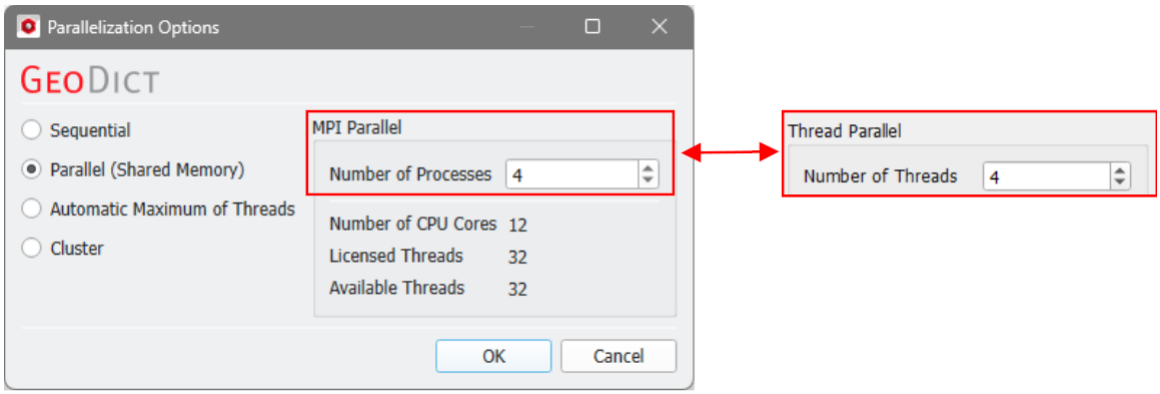
Sequential

When **Sequential** is selected, no further parameters are needed, and the solver runs sequential without parallelization. This option can be useful for very small structures.



Parallel (Shared Memory)

When **Parallel (Shared Memory)** is selected, the **Number of Processes** can be entered. Then the maximum number of available processors and the maximum number of licensed parallel processes is shown in the dialog. The solver runs parallel with the specified number of processes / threads.

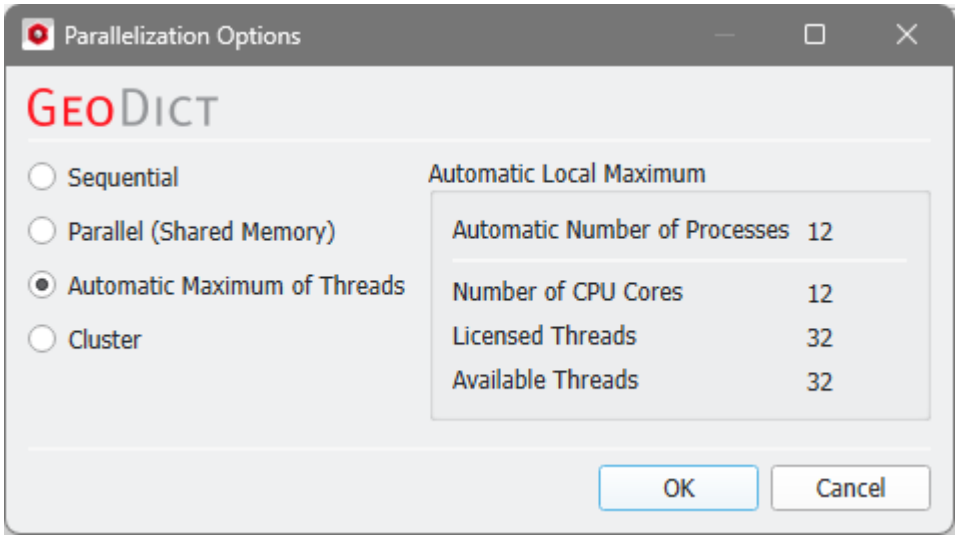


Depending on the implementation of the specific solver, the parallelization might use the MPI Message Passing Interface or use local thread parallelization. The method is denoted on the panel and automatically selected by GeoDict. Make sure that MPI is installed 9 if the solver uses this method for parallelization.

Note! The **Parallel (Shared Memory)** parallelization option of the EJ and SimpleFFT solvers does not work on cluster nodes. Here, you will get the error message "-1815 Invalid parent process ID". Use the **Cluster** 7 parallelization option instead of **Parallel (Shared Memory)** even if just one node is used for the computation.

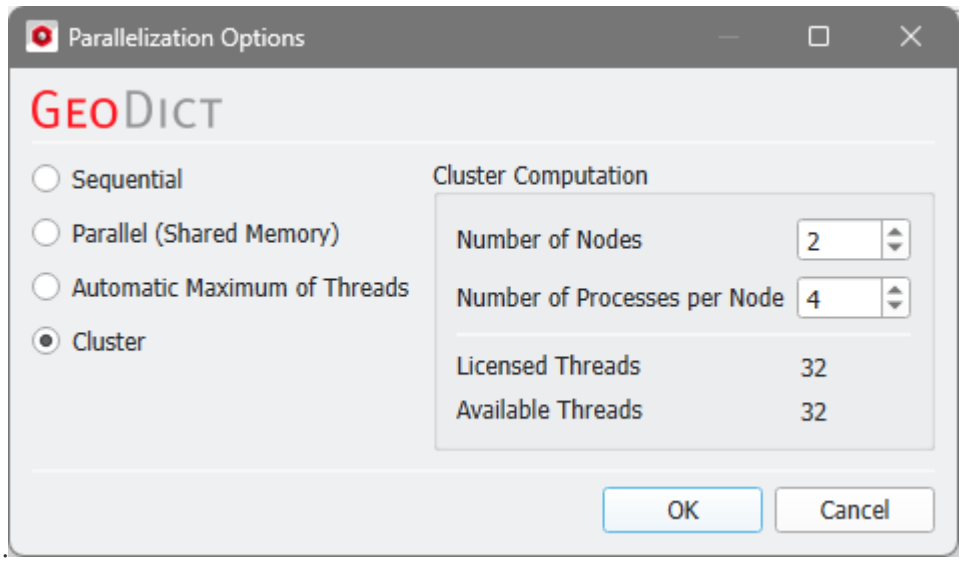
Automatic Maximum of Threads

If **Automatic Maximum of Threads** is selected, the number of parallel processes is automatically selected for optimal speed, based on the CPU cores and licensed parallel processes. This is the default option and in most cases the best choice.



Cluster

The choice of **Cluster** is for users of Linux clusters. Use this option to [set up a cluster simulation script](#)[15]. Enter the number of compute nodes and the number of processes per node here.



Cluster parallelization requires that the solver supports the MPI parallelization method. Otherwise, this option is not selectable in the dialog. The following table shows the supported parallelization methods:

Solver	Parallelization method	
	MPI Parallel	Thread Parallel

EJ solver	✓	✗
SimpleFFT solver	✓	✗
LIR solver	✗	✓
FeelMath solver	✓	✓
Particle tracker	✓	✓
BEST solver	✗	✓
Transport solver	✗	✓

1.2 MPI Configuration

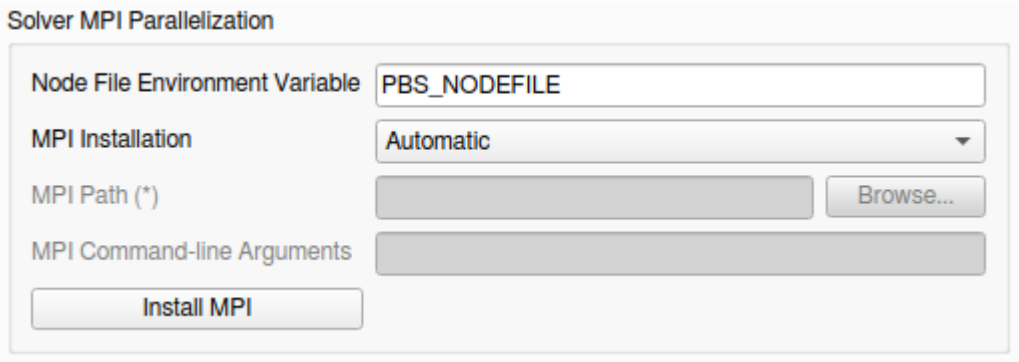
MPI Installation

In Windows, the necessary Microsoft MPI library is installed during the GeoDict Windows installation.

In Linux, the necessary libraries have to be installed separately. Refer to the GeoDict Linux installation.

MPI Configuration

You can access the **Solver MPI Parallelization** configuration options through the main menu of GeoDict. Select **Settings** - **Settings** from the menu bar to open the dialog, and in the dialog open the Parallelization tab. The **GeoDict Thread Parallelization** options are explained here.



Node File Environment Variable (Select compute nodes)

When using a job submission system, it is not a priori known which compute nodes will be used for the computation. The job system will make that decision when starting the job depending on which nodes are available at that moment. The mechanism used by PBS and SLURM to pass the information about the assigned compute nodes to MPI is as follows. The job system will create a file containing a list of the signed compute nodes. The path to this file will be set as an environment variable, typically named PBS_NODEFILE. If another job submission system is used, which may use a different variable name, you can change the name of the environment variable here.

This mechanism can also be exploited to allow for a manual choice of the compute nodes, e.g., when no job submission system is used. In this case, you have to create your own "node" file. Create a text file and fill it with the names of the computers that should be used for the computation, e.g.,

```
node001
node002
node003
```

The names are separated by newlines. Then, set the path to the node file as value for the PBS_NODEFILE variable. This can be done with the Linux command

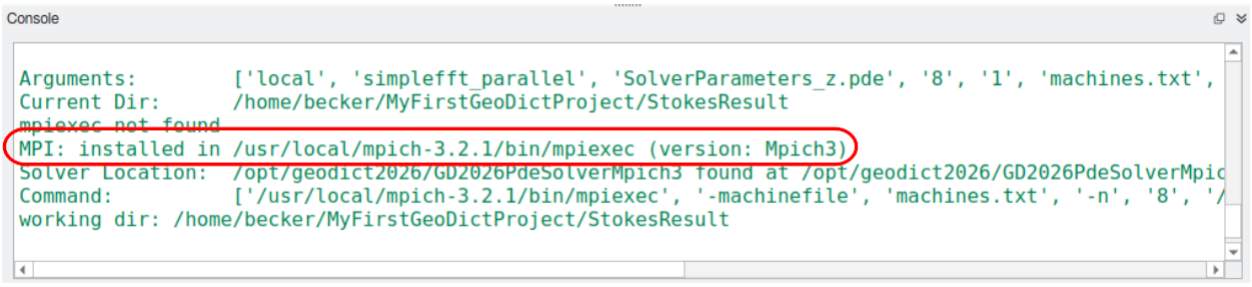
```
export PBS_NODEFILE=<path to node file>
```

This command has to be executed before GeoDict is started. Then follow the Cluster Computing 15 instructions to start a distributed simulation.

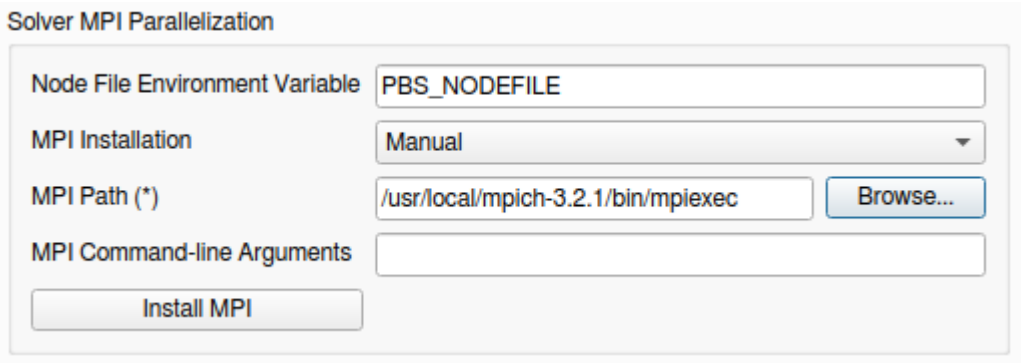
MPI Installation and MPI Path (Switch between different MPI packages)

You can switch between **Automatic** and **Manual** in the **MPI Installation** drop-down menu.

In the default case **Automatic**, GeoDict searches for a usable installed MPI and uses it. Check the Console output (enable **Show GeoPython Messages**) if you are unsure which MPI version is used:



To change the used MPI version, select **Manual** and browse to the corresponding **mpiexec** executable of the selected MPI version.



MPICH-3.2 is used by default in Linux but sometimes OpenMPI-1.10.7 might be a better choice in computer cluster environments. Many clusters have InfiniBand interconnection between the compute nodes. Unfortunately, MPICH-3.2 cannot use this interconnection natively and therefore the scaling behavior might not be ideal for more than four compute nodes. OpenMPI-1.10.7 can use InfiniBand interconnections natively and provides a good scaling behavior beyond four compute nodes.

Important! Currently, OpenMPI 1.10.7 does not work on a cluster in combination with the FeelMath solver (ElastoDict). Use MPICH3.2 instead of OpenMPI 1.10.7.

MPI Command-line Arguments

It is possible to add additional command line arguments to the MPI call by entering them in the box. These additional command line arguments may be used to configure proper usage of the InfiniBand adapter or to show more debug information in the command line output. As an example for MPICH 3.2., the usage of the Internet Protocol over InfiniBand (IPoIB) feature can be configured with the command line argument: `-iface ib0`.

Install MPI

It is also possible to trigger the MPI installation script `setupMPI.sh` from GeoDict's user interface, and in this case MPI will be installed locally (the root version is not available from the GUI).

1.3 Remote Desktop

If GeoDict is installed on a server that is shared by many users, a graphical remote desktop connection from your local computer to the remote server is needed to use GeoDict.

If both the local computer and the remote server use Microsoft Windows, Microsoft's built-in *Remote Desktop Connection* tool can be used for this.

If it is a Linux server, we recommend to use TigerVNC. The graphical remote desktop connection with TigerVNC is a very convenient way to perform simulations on a remote Linux server.

Note! If multiple users use GeoDict at the same time, multiple GeoDict licenses are required. One GeoDict license can only be used in one TigerVNC session. If the same user starts multiple remote sessions (e.g., from different desktops) at the same time, multiple licenses are needed as well.

Follow these steps to use the remote desktop for GeoDict. Steps 1 to 3 are to set it up (administrator privileges required), and carried out only once. Steps 4 to 6 are done every time remote computations are performed:

1. Install TigerVNC server on the remote Linux computer

TigerVNC is used as X proxy and video server.

The source code and the latest .rpm and .deb packages can be downloaded from https://github.com/TigerVNC/tigervnc/releases.

Installing TigerVNC is also possible using `yum` or `apt`.

2. Install VirtualGL on the remote Linux computer

VirtualGL is used for hardware OpenGL support.

The source code and the latest .rpm and .deb packages can be downloaded from https://github.com/VirtualGL/virtualgl/releases and the VirtualGL 3.1.3 User Guide describes the installation process.

Installing VirtualGL is also possible using `yum` or `apt` and the installation process is described here.

3. Install and set up TigerVNC viewer on the local Windows computer

The latest .exe files can be downloaded from https://github.com/TurboVNC/turbovnc/releases.

4. Start TigerVNC server on the remote Linux computer

To start the VNC server, open a ssh connection to the remote Linux computer, for example with **PuTTY**.

Check if there is already a vncserver running with

```
tigervncserer -list
```

If no server is listed, start a new one with

```
tigervncserver
```

The server will ask you to set a password (if you start a TigerVNC server for the first time, otherwise the last password will be taken), then tell you the number you need to connect

```
New Xtigervnc server 'zyklop.kl.math2market.de:6 (becker)' on port 5906 f
```

The session number (here: 6) is needed later.

If you have forgotten your password, you can set a new one with

```
tigervncpasswd
```

The ssh connection can be closed now.

This server stays open as long as you do not log-out the virtual session. A virtual session can be stopped using the `-kill` option and the session number (here `:6`) by entering

```
tigervncserver -kill :6
```

5. Connect the Windows computer to the remote Linux computer

As long as a virtual session exists (i.e., you have not logged-out or killed the session), you can connect to the remote Linux computer by starting a VNC Viewer and entering the hostname and session number:

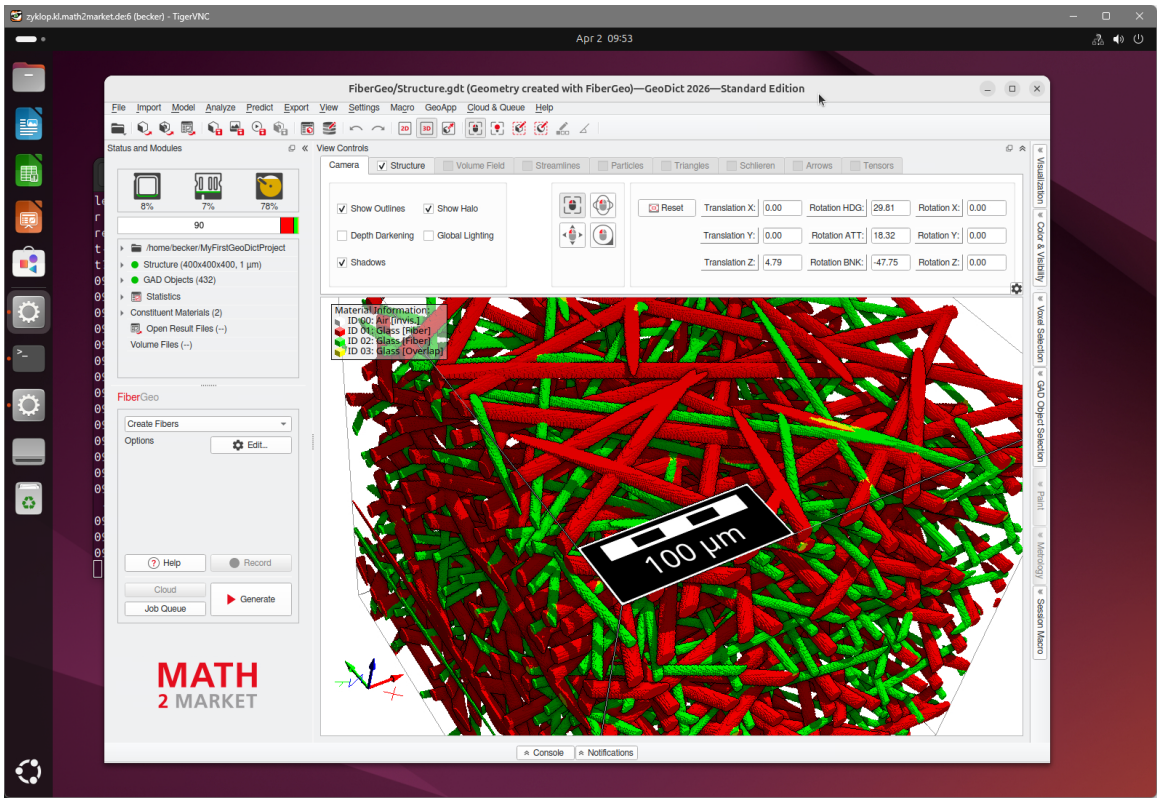
A virtual desktop of the remote Linux computer opens:

6. Start and run GeoDict on the remote Linux computer

To start GeoDict, open a terminal and enter:

```
/opt/geodict2026/geodict2026
```

The started session on the remote Linux computer runs until you kill the VNC server. If you just close the remote desktop dialog box by clicking on the X in the upper right corner, the VNC server will keep running. Also, any GeoDict processes you started will continue to run in this case. You can also log out and shutdown your local desktop and the session will continue to run on the remote desktop. You can connect to the remote VNC server again later.



X-Forwarding in Linux

It is in principle possible to connect from a Linux terminal to a Linux server using `ssh -x`, without the installation of VirtualGL and TigerVNC. In this case, visualization in 3D is only possible when **Indirect GLX** is enabled on the Linux terminal (it is not necessary to enable this on the Linux server that runs GeoDict, but it must be enabled on the terminal that runs the X11 server). By default, this option is disabled in many cases. The status of this option can be checked with

```
grep -i "Indirect" /var/log/Xorg.0.log
```

Output is, for example:

```
[1344382.166] (II) Indirect GLX disabled
```

To enable this option is not straightforward, because it works differently on different Linux systems or different graphic cards, and often requires a system administrator to change the config files of the system. Because of this complexity in the setup, we recommend in general to use TigerVNC for remote logins instead of `ssh -X`.

1.4 Cluster Computing

A computer cluster typically consists of compute nodes with the same hardware configuration, and they are communicating with each other over a very fast interconnection (e.g., InfiniBand). GeoDict can be used with its GUI on such clusters if it has a floating license. Usually, a simulation script is submitted to a job queue management system.

A job queue scheduler is a computer application for controlling unattended background program execution of jobs. This is commonly called batch scheduling, as execution of non-interactive jobs is often called batch processing. Two commonly used job schedulers are:

1. Portable Batch System (PBS): a computer software that performs job scheduling. Its primary task is to allocate computational tasks, i.e., batch jobs, among the available computing resources. It is often used in conjunction with UNIX cluster environments.
2. Slurm Workload Manager (SLURM): a free and open-source job scheduler for Linux and Unix-like kernels, used by many of the world's supercomputers and computer clusters.

Both schedulers can be used to submit and perform GeoDict simulation jobs.

Important! You can only make use of multiple compute nodes on a distributed memory cluster for the following solvers: EJ (flow, conduction, diffusion), SimpleFFT (flow, conduction, diffusion), FeelMath (stiffness) and Tracker (particulate flow).

All structure generation commands and any command using the LIR or BEST solver cannot use distributed memory, and thus can only make use of a single node on a distributed memory cluster!

GeoDict needs to be installed on each compute node or has to be installed on a shared file systems such that each compute node can access it. A floating license installed on a license server is needed and each compute node must have access to the license server. Node-locked licenses do not work for cluster computing.

Three steps are needed to start large simulations on a Linux cluster:

1. Enable password-less login on cluster compute nodes

When using SSH to login to (other) cluster nodes, the system typically asks for a password. This is bothersome for multi-process job start-up procedures. However, the SSH configuration can be changed to allow password-less login to cluster compute nodes. The script `enablePasswordLessLogin.sh` configures that for you:

1. Switch to the GeoDict installation folder.
2. Execute `./enablePasswordLessLogin.sh`

This script has to be executed when logged in to your Linux account on the cluster. It can also be done on your “local” Linux computer, if your local computer and the cluster share the same account. After execution of the script, it is possible to login to cluster node without password input.

2. Prepare a cluster simulation script

A submission shell script is needed to start a simulation on a cluster. This shell script contains control information for the job submission system (e.g., number of nodes) and calls GeoDict with a floating license and a simulation script. A template for such a script is available in the GeoDict installation folder (Linux version only) with the name:

- **`PBSClusterSimulationTemplate.sh`** for PBS, and
- **`SLURMClusterSimulationTemplate.sh`** for SLURM.

The following description considers PBS but is very similar for SLURM.

```
1 #!/bin/bash
2
3 ##############################################################################################################
4 # The following lines starting with PBS are not comments but internal commands for the queue-system #
5 ##############################################################################################################
6
7 # If an error happens do not restart the job
8 #PBS -r n
9
10 # Name of the job in the queue
11 #PBS -N GeoDictClusterSimulation
12
13 # Name of the Std-Out-Files
14 #PBS -o GeoDictClusterSimulation.out
15
16 # Name of the Std-Err-Files
17 #PBS -e GeoDictClusterSimulation.err
18
19 # 8 Nodes with 4 processes per node for 48 hour (adjust according to your needs)
20 #PBS -l walltime=48:0:0
21 #PBS -l nodes=8:ppn=4
22
23
24 ##############################################################################################################
25 #      The following lines need to be adjusted according to your GeoDict and MPI location           #
26 #      EXECUTE THIS SHELL SCRIPT WITH:   qsub PBSClusterSimulationTemplate.sh                       #
27 ##############################################################################################################
28
29 # Location of the MPI installation that should be used for the simulation
30 export PATH=/opt/geodict2026/MPI/mpich-3.2.1/bin/:$PATH
31
32 # GeoDict call script (adjust according to your GeoDict installation)
33 GEODICT=/opt/geodict2026/geodict2026
34
35 # GeoDict licence file (adjust according to your licence location)
36 LICENSE=/opt/geodict2026/License/geodict2026.lic
37 # Your simulation script that should be performed on the cluster
38 SIMULATIONSCRIPT=~/GeoDictClusterSimulation.py
39
40 ##############################################################################################################
41 #          The following lines start GeoDict and perform a cluster simulation                          #
42 ##############################################################################################################
43
44 $GEODICT $LICENSE $SIMULATIONSCRIPT
```

1. Adjust the following lines of the submission script according to your GeoDict installation and cluster settings:

- Line 20: Maximum runtime of the simulation. If the simulation last longer than the specified runtime than the job is canceled.
- Line 21: Number of nodes and processes per nodes (ppn).
- Line 30: Path to your MPI installation.
- Line 33: Path to your GeoDict installation.
- Line 36: Path to your GeoDict floating license.

- Line 38: Path to your simulation script.

2. Create a simulation script that performs the simulation. Make sure that the parallelization settings in the simulation script use the cluster, e.g.:

```
'Parallelization' : {

  'Mode' : 'CLUSTER',

  'NumberOfNodes' : 8,          # This number has to be smaller or
                                equal to the number of nodes in the
                                submission script.

  'ProcessorsPerNode' : 4,      # This number has to be smaller or
  },                            equal to the number of processes per
                                node in the submission script.
```

3. Submit a simulation on the cluster

1. Login to the master node of the cluster (e.g., with putty or SSH).
2. Change the working directory to the submission shell script.
3. Make sure that enough disk space is available for temporary saving of files (e.g., flow fields).
4. For PBS: Start the simulation with the command:
   `qsub <Path to Shell Script Folder>/ PBSClusterSimulationTemplate.sh`
5. For SLURM: Start the simulation with the command:
   `sbatch <Path to Script Folder>/ SLURMClusterSimulationTemplate.sh`
6. A log file for the simulation run is created with the name
   `GeoDictClusterSimulation.out`

Citation: Math2Market GmbH, 2026: GeoDict 2026 User Guide, High Performance Computing, Math2Market GmbH, Germany, doi.org/10.30423/userguide.geodict

MATH 2 MARKET

Math2Market GmbH
Richard-Wagner-Str. 1, 67655 Kaiserslautern / Germany
www.math2market.com