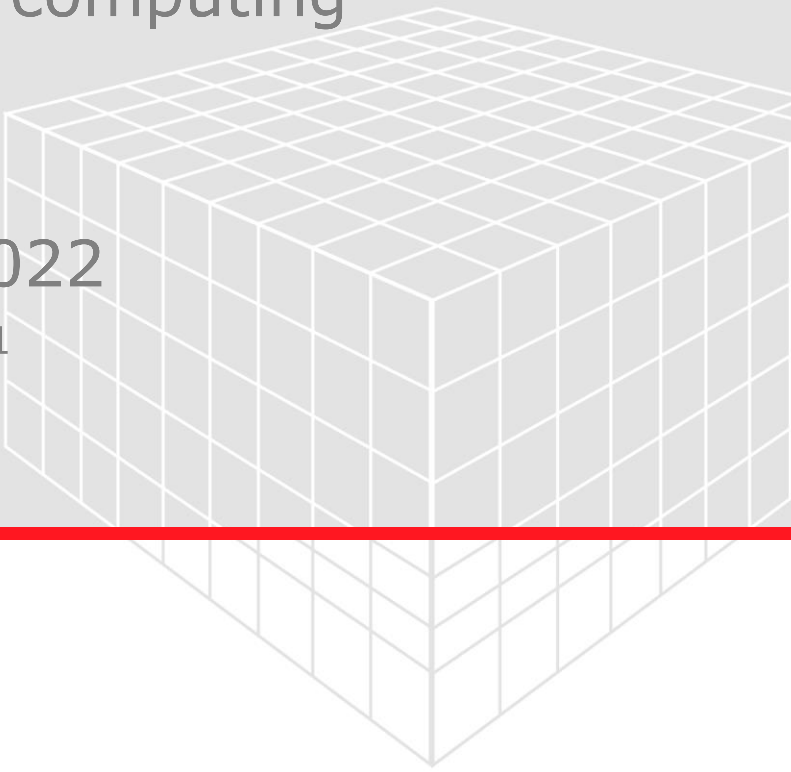
GeoDict®

High performance computing

User Guide

GeoDict release 2022

Published: September 24, 2021



GeoDict

https://doi.org/10.30423/userguide.geodict2022-highperformancecomputing

Introduction to High Performance Computing

Simulation of flow, thermal, electrical or mechanical properties on very large structures (e.g. 2048^3 voxels) need 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:

1. large shared memory machines or
2. computer clusters.

The two possibilities are now explained in more detail.

Shared memory Machines

Large shared memory machines are single computers with a lot 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 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/Support/System/DeltaConfig.php.

The operating system on these machines is often Windows or Linux. Typically one or more users can login to this computer at the same time with a remote desktop connection. The most used combinations are:

1. Local Windows to remote Windows can be done with Microsoft's built-in *Remote Desktop Connection* tool.
2. Local Linux to remote Linux can be done with the built-in *SSH* command line tool or TurboVNC which is a graphical remote desktop connection tool.
3. Local Windows to remote Linux can also be done with TurboVNC.

The graphical remote desktop connection with TurboVNC is a very convenient way to perform simulations on shared memory machines but TurboVNC is not a built-in tool. On page 8, we describe the installation and usage of this tool.

Computer Clusters

Sometimes structures are very large (e.g. 4096^3) and even large shared memory machines do not have enough memory to perform a flow or mechanical simulation. In this situation a computer cluster with many compute nodes is required.

A computer cluster typically consists of compute nodes with same hardware configuration and they are communicating with each other over a very fast interconnection (e.g. InfiniBand). It is possible to use GeoDict with its GUI on such clusters but typically a simulation script is submitted into 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)
2. Slurm Workload Manager (SLURM)

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

PBS is the name of 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.

The Slurm Workload Manager is a free and open-source job scheduler for Linux and Unix-like kernels, used by many of the world's supercomputers and computer clusters.

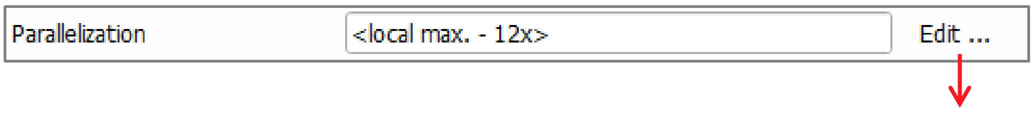
On page 12, we describe how to submit jobs on both systems.

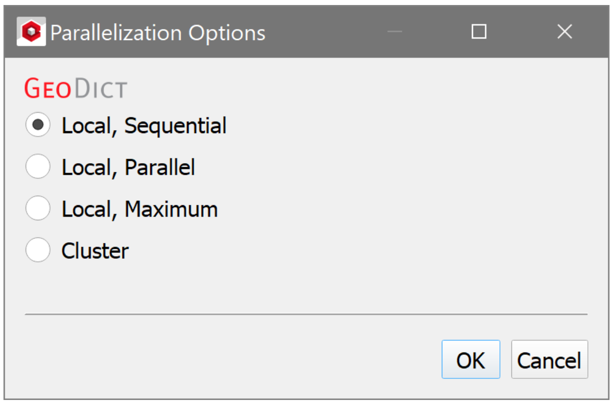
PARALLELIZATION AND CONFIGURATION OF MPI

The solvers in GeoDict can make use of powerful hardware by parallelization of simulations to decrease the runtime.

PARALLELIZATION OPTIONS

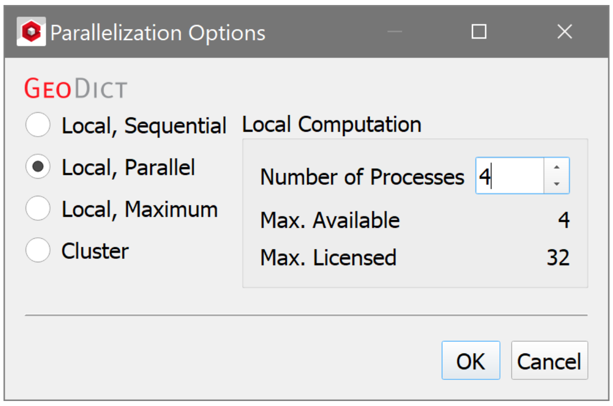
The parallelization can be configured in the parallelization dialog. This dialog is available in the solver tabs of all **Dict** modules:


Parallelization <local max. - 12x>



[● Local, Sequential]
Local, Parallel
Local, Maximum
Cluster

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

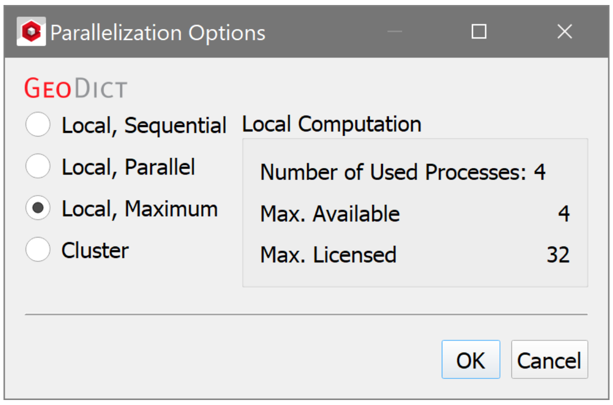


Local, Sequential
[● Local, Parallel]
Local, Maximum
Cluster

Local Computation

Number of Processes	4
Max. Available	4
Max. Licensed	32

When **Local, Parallel** 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.



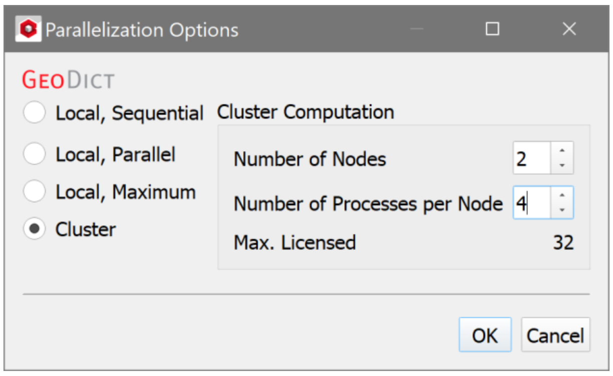
Local, Sequential
Local, Parallel
[● Local, Maximum]
Cluster

Local Computation

Number of Used Processes: 4

Max. Available	4
Max. Licensed	32

When **Local, Maximum** is selected, the maximum number of available and licensed processors / threads is used for the simulation. This is the default option and in most cases the best choice.



Local, Sequential
Local, Parallel
Local, Maximum
[✓ Cluster]

Cluster Computation

Number of Nodes: 2

Number of Processes per Node: 4

Max. Licensed: 32

The choice of **Cluster** is for users of Linux clusters. Then the number of compute nodes and the number of processes per node can be entered. The solver runs parallel and distributes the simulation over different compute nodes.

Not all solvers support all parallelization options, as shown in the following table:

Solver	Parallelization Option			
	Local, Sequential	Local, Parallel	Local, Maximum	Cluster
EJ solver	✓	✓	✓	✓
SimpleFFT solver	✓	✓	✓	✓
LIR solver	✓	✓	✓	✗
FeelMath solver	✓	✓	✓	✓
Particle tracker	✓	✓	✓	✓
BEST solver	✓	✓	✓	✗

The parallelization of the solvers is done with three 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 (e.g. EJ or SimpleFFT) 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 then the sending of data can be done very efficiently because its running on the same machine while Distributed-MPI has to use Ethernet or fast interconnection like InfiniBand to send data between different computers.

The LIR solver and BEST solver use Local-Thread parallelization. These solvers are started only once per simulation and data does not have to be sent between instances because there is only one. But these solvers cannot use multiple compute nodes of a cluster to distribute the simulation data.

The following table shows the support of both parallelization methods:

The **Cluster** parallelization requires that the solver supports the MPI parallelization method. The **Local, Parallel** parallelization can be done with MPI parallelization or Thread parallelization.

Solver	Parallelization method	
	MPI Parallel	Thread Parallel
EJ solver	✓	✗
SimpleFFT solver	✓	✗
LIR solver	✗	✓
FeelMath solver	✓	✓
Particle tracker	✓	✓
BEST solver	✗	✓

An important difference between thread and MPI parallelization is that thread parallelization does not need any special installation procedure. It is available on all operating systems and hardware architectures. MPI parallelization requires the installation of an MPI software package.

Installation of MPI

GeoDict MPI parallel solvers support three MPI software packages:

1. MPICH-3	for Linux (https://www.mpich.org/),
2. OpenMPI-1.10.7	for Linux (https://www.open-mpi.org/), and
3. Microsoft MPI	for Windows.

The Microsoft MPI is automatically installed during the GeoDict installation. MPICH3 or OpenMPI are often available on Linux system but sometimes it has to be manually installed by the user.

Installation of MPI under Linux is possible using the command line or you can trigger the installation from the GeoDict GUI.

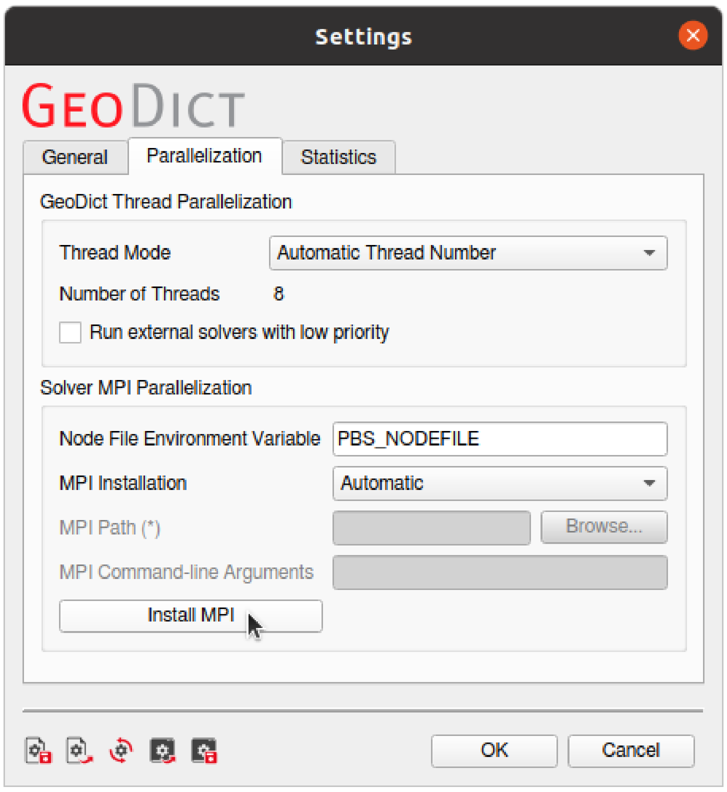
To install MPI from the command line open a terminal and change the current directory into the GeoDict installation folder. The installation folder contains a shell script with the name *setupMPI.sh*. Execute this script with the command:

```
./setupMPI.sh
```

The script compiles MPICH-3.2 and OpenMPI-1.10.7 and installs them in the GeoDict installation folder in the sub-folder *MPI*. Root permission is not required to perform this installation procedure but the compiled MPI packages are available for the current GeoDict installation only. It is also possible to make the MPI packages available for the whole system. This can be done by using the command line argument *root*, i.e.

```
sudo ./setupMPI.sh root
```

Here, root permission is required and the MPI packages are installed in */usr/local/*. GeoDict then automatically detects und uses MPI for parallelization after installation.


Settings

GeoDict

General | Parallelization | Statistics

GeoDict Thread Parallelization

Thread Mode: Automatic Thread Number

Number of Threads: 8

☐ Run external solvers with low priority

Solver MPI Parallelization

Node File Environment Variable: PBS_NODEFILE

MPI Installation: Automatic

MPI Path (*)

MPI Command-line Arguments

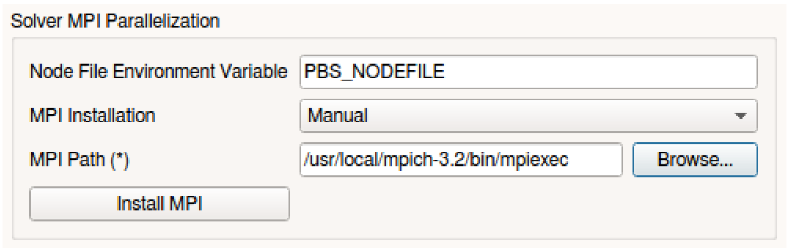
It is also possible to trigger the script ./setupMPI.sh from GeoDict's user interface, and in this case MPI will be installed locally (the root version is not available from the GUI). For this, choose **Settings->Settings** from the main menu of GeoDict. To install, click on **Install MPI** in the **Solver MPI Parallelization** panel.

The compilation and installation of MPI will take approximately 15 to 30 minutes. After the installation, check if the OpenMPI and Mpich3.2. executables have been created, as the script will not return an error message if one of the installations fails.

Switch between different MPI packages

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 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.

The choice of the used MPI package can be changed by setting **MPI Installation** to **Manual** and browsing to the corresponding **mpiexec** executable of the selected MPI version.

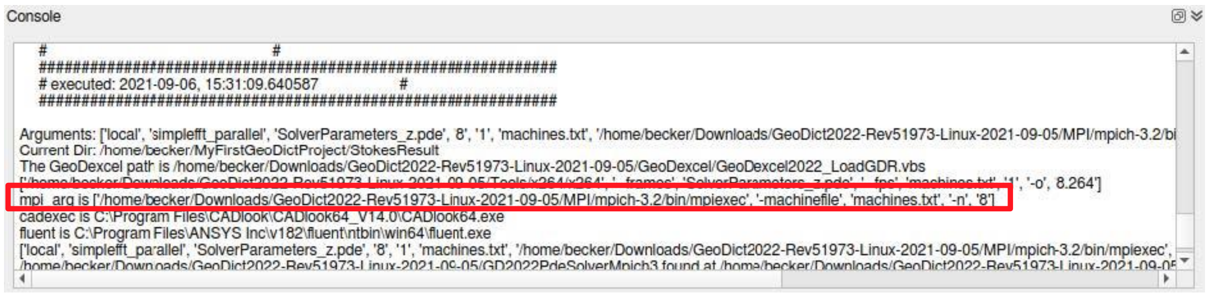

Solver MPI Parallelization

Node File Environment Variable: PBS_NODEFILE

MPI Installation: Manual

MPI Path (*): /usr/local/mpich-3.2/bin/mpiexec

If set as above, GeoDict solvers will use MPICH 3.2 installed in /usr/local for parallelization. It is possible to check if the desired MPI package is used by inspection of the command line output of GeoDict when an MPI parallel solver is started. For Mpich 3.2. the command line output should look similar to:



```
    #                                       #
    ################################################################
    # executed: 2021-09-06, 15:31:09.640587             #
    ################################################################

Arguments: ['local', 'simplefft_parallel', 'SolverParameters_z.pde', '8', '1', 'machines.txt', '/home/becker/Downloads/GeoDict2022-Rev51973-Linux-2021-09-05/MPI/mpich-3.2/bi
Current Dir: /home/becker/MyFirstGeoDictProject/StokesResult
The GeoDexcel path is /home/becker/Downloads/GeoDict2022-Rev51973-Linux-2021-09-05/GeoDexcel/GeoDexcel2022_LoadGDR.vbs
['/home/becker/Downloads/GeoDict2022-Rev51973-Linux-2021-09-05/Tools/x264/x264', '--frames', 'SolverParameters_z.pde', '--fps', 'machines.txt', '1', '-o', 8.264']
mpi_arg is ['/home/becker/Downloads/GeoDict2022-Rev51973-Linux-2021-09-05/MPI/mpich-3.2/bin/mpiexec', '-machinefile', 'machines.txt', '-n', '8']
cadexec is C:\Program Files\CADlook\CADlook64_V14.0\CADlook64.exe
fluent is C:\Program Files\ANSYS Inc\v182\fluent\ntbin\win64\fluent.exe
['local', 'simplefft_parallel', 'SolverParameters_z.pde', '8', '1', 'machines.txt', '/home/becker/Downloads/GeoDict2022-Rev51973-Linux-2021-09-05/MPI/mpich-3.2/bin/mpiexec',
/home/becker/Downloads/GeoDict2022-Rev51973-Linux-2021-09-05/GD2022PdeSolverMpich3 found at /home/becker/Downloads/GeoDict2022-Rev51973-Linux-2021-09-05
```

It is possible to add additional command line arguments to the MPI call by entering them in the MPI Command-line Arguments box in the Settings dialog. 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”.

Graphical remote desktop connection to Linux computer

A graphical remote desktop connection to a remote Linux computer can be used to run GeoDict on it. A GeoDict installation has to be available on the remote Linux computer, but not necessarily on the local Windows computer. Since GeoDict completely runs on the Linux computer all benefits from the Linux computer capabilities in each step, e.g., structure generation, property prediction, visualization and so on are available.

Follow these steps to use the remote desktop for GeoDict. Most of them are to set it up, and carried out only once. The last two (4 and 5) are done every time remote computations are performed:

1. Install TurboVNC on the remote Linux computer. TurboVNC is used as X proxy and video server. Further information of TurboVNC can be found here:

   http://www.turbovnc.org/

2. Install and set up VirtualGL on the remote Linux computer. VirtualGL is used for hardware OpenGL support. Further information of VirtualGL can be found here:

   http://www.virtualgl.org/

3. Install and set up TurboVNC on the local Windows computer.
4. Connect the Windows computer to the remote Linux computer.
5. Start and run GeoDict on the remote Linux computer

Install TurboVNC on the Linux computer

The following description is valid for Ubuntu, and it might be slightly different for other Linux operating systems. For more information see:

https://cdn.rawgit.com/TurboVNC/turbovnc/master/doc/index.html#hd005001

1. Download the appropriate TurboVNC binary package for your system from

   http://sourceforge.net/projects/turbovnc/files/2.2.6/

   e.g.

   https://sourceforge.net/projects/turbovnc/files/2.2.6/turbovnc_2.2.6_amd64.deb

2. Install TurboVNC by switching (cd) to the directory where you downloaded TurboVNC and issuing the following command:

   ```
   sudo dpkg -i turbovnc*.deb
   ```

Install VirtualGL on the Linux computer

The following description is valid for Ubuntu running LightDM, and might be slightly different for other Linux operating systems. Further information can be found at:

https://cdn.rawgit.com/VirtualGL/virtualgl/2.6.5/doc/index.html#hd005001

1. Download the appropriate VirtualGL binary package for your system from:

   https://sourceforge.net/projects/virtualgl/files/2.6.5/

   e.g.

   https://sourceforge.net/projects/virtualgl/files/2.6.5/virtualgl_2.6.5_amd64.deb
2. Install VirtualGL by switching (cd) to the directory where you downloaded VirtualGL and issuing the following command:

   ```
   sudo dpkg -i virtualgl*.deb
   ```
3. Configure the server:
   a. Shut down the x server as follows (or restart computer later):

      ```
      sudo /etc/init.d/lightdm stop
      ```
   b. Run the following:

      ```
      sudo /opt/VirtualGL/bin/vglserver_config
      ```
   c. Select option 1 to configure the server
   d. Choose ***no*** for restriction
   e. Choose ***yes*** for disabling XTEST
   f. Start the x server again as follows (or restart computer):

      ```
      sudo /etc/init.d/lightdm start
      ```
4. In order to get the Nvidia driver loading for remote computers, without having a monitor plugged in, open /etc/X11/xorg.conf with a text editor and add the option "ConnectedMonitor" "CRT" to the section "Device" just below the Driver "nvidia" line. In some linux distributions, the file /etc/X11/xorg.conf does not exist by default anymore.

   In such cases, please check where such an options should be added in your linux distribution.

```
Section "Device"
    Identifier         "Device0"
    Driver             "nvidia"
    ConnectedMonitor   "CRT"
    VendorName         "NVIDIA Corporation"
EndSection
```

Install TurboVNC on the local Windows computer

Download and install TurboVNC64_1.2.exe from:

https://sourceforge.net/projects/turbovnc/files/2.2.6/

Further information can be found here:

https://cdn.rawgit.com/TurboVNC/turbovnc/master/doc/index.html#hd005003

Follow the instructions that are shown in the installation wizard (click "next" button).

START TURBOVNC SERVER ON THE LINUX COMPUTER

To set up the VNC server:

1. Open a ssh connection to the remote Linux computer, for example with **PuTTY** (cf. 2.1.6).
2. Enter: */opt/TurboVNC/bin/vncserver* or for non-default windows size use the parameter *–geometry* <width>x<height>,

For example, enter:

/opt/TurboVNC/bin/vncserver –geometry 1900x950

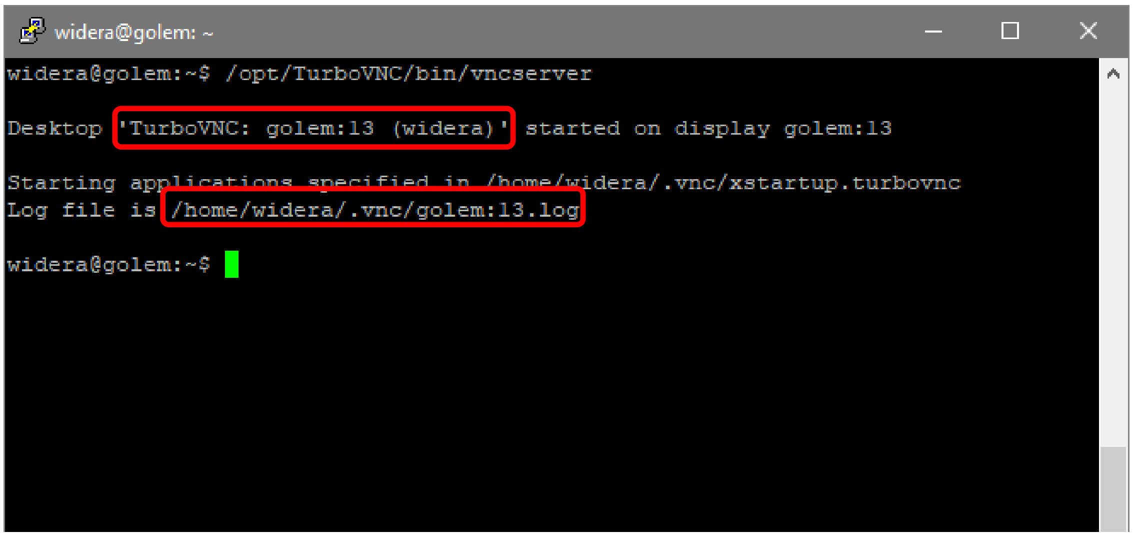
a. A VNC password is requested when started for the first time.
   - Enter a VNC password
   - Verify the password

b. Passwords can be changed later through

   /opt/Turbo*VNC*/bin/vncpasswd.

c. The connection information is returned, e.g.

Desktop 'TurboVNC: golem:13 (widera)'. The session number (here: *13*) is needed later. The IP address of this host is 192.168.1.39.



```
widera@golem:~$ /opt/TurboVNC/bin/vncserver

Desktop 'TurboVNC: golem:13 (widera)' started on display golem:13

Starting applications specified in /home/widera/.vnc/xstartup.turbovnc
Log file is /home/widera/.vnc/golem:13.log

widera@golem:~$
```

3. 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 by entering:

/opt/TurboVNC/bin/vncserver -kill :"your session number".

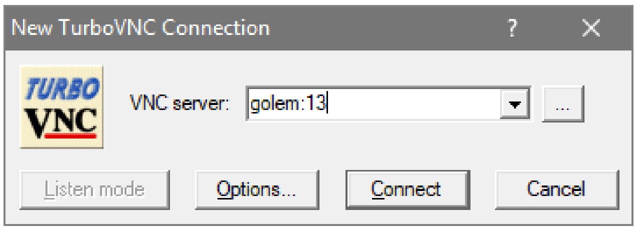
The active VNC servers can be listed by entering:

/opt/TurboVNC/bin/vncserver –list.

Connect Windows to a remote Linux computer

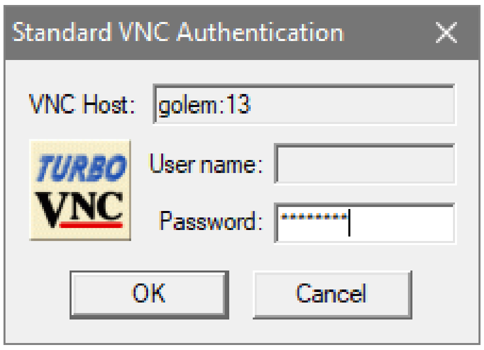
As long as a virtual session exists (i.e., you have followed these steps and have not logged-out or killed the session), the connection to the remote Linux computer is established as follows:

1. Start the TurboVNC Viewer.
2. Enter the session server, either by its hostname or IP address followed by the session number, e.g., 192.168.1.39:13, and click **Connect**


New TurboVNC Connection

VNC server: golem:13

3. Enter your VNC password and click **OK**


Standard VNC Authentication

VNC Host: golem:13

User name:

Password: ********

A virtual desktop of the remote Linux computer opens.

Start and run GeoDict on the remote Linux computer

VirtualGL is used to redirect rendering to the remote graphics hardware, meaning that GeoDict's graphical user interface is available in the Linux computer.

To start GeoDict, open a terminal and enter:

```
vglrun ./YourGeoDictPath/geodict2022
```

The started GeoDict session on the remote Linux computer is maintained until the user logs out or kills the VNC session. Otherwise, the user can close the remote desktop dialog box (by clicking on the x in the upper right corner), turn-off the local Windows computer, go home, and restart the remote session later, finding that the started GeoDict jobs are still running.

Cluster computing on Linux clusters

Many prediction modules (e.g. FlowDict) allow to perform computations on Linux clusters. This allows to utilize many compute nodes at the same time to massively parallelize large computations.

In this case, GeoDict needs to be installed on each compute node or it has to be installed on a shared file systems such that each compute node can access it. A floating license installed in 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
2. Prepare a cluster simulation script
3. Submit a simulation on the cluster

Enable password-less login

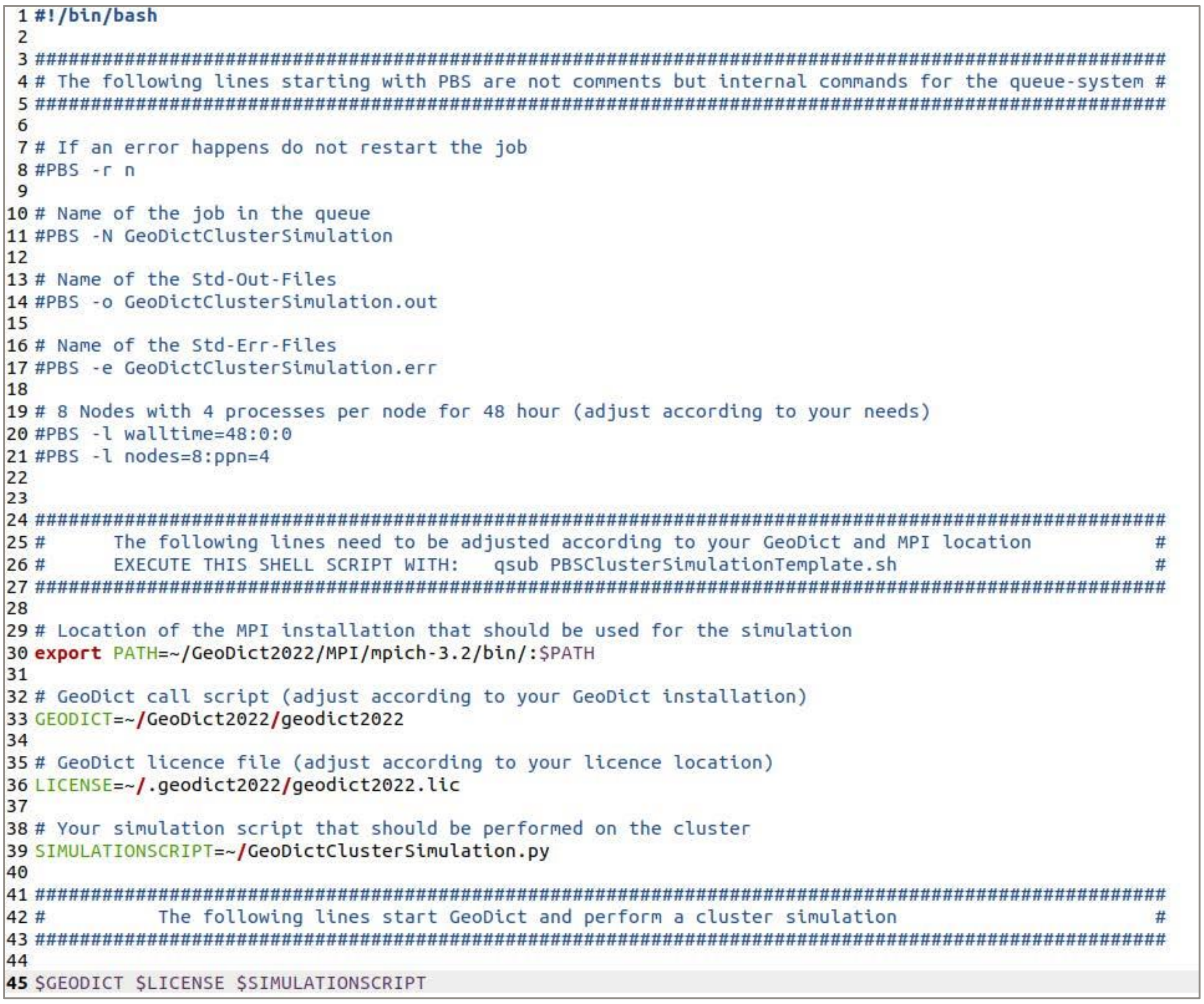
When using SSH to login to (other) cluster nodes, the system typically asks for a password. This is bothersome for multi-process job startup procedures. However, the SSH configuration can be changed to allow password-less login to cluster compute nodes. The script *enablePasswordLessLogin.sh* configures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

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=~/GeoDict2022/MPI/mpich-3.2/bin/:$PATH
31 
32 # GeoDict call script (adjust according to your GeoDict installation)
33 GEODICT=~/GeoDict2022/geodict2022
34 
35 # GeoDict licence file (adjust according to your licence location)
36 LICENSE=~/.geodict2022/geodict2022.lic
37 
38 # Your simulation script that should be performed on the cluster
39 SIMULATIONSCRIPT=~/GeoDictClusterSimulation.py
40 
41 ##########################################################################################
42 #          The following lines start GeoDict and perform a cluster simulation                    #
43 ##########################################################################################
44 
45 $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 cancelled.
   - 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 39: 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',
2       'NumberOfNodes'     : 8,
3       'ProcessorsPerNode' : 4,
      },
```

   - Line 2: Number of nodes. This number has to be smaller or equal to the number of nodes in the submission script.
   - Line 3: Processes per node. This number has to be smaller or equal to the number of processes per node in the submission script.

START A SIMULATION ON THE CLUSTER

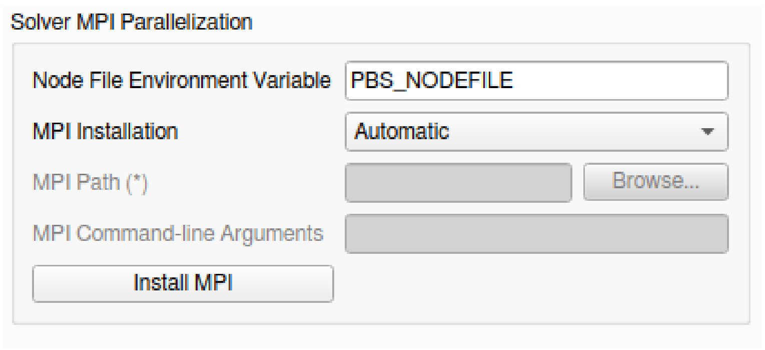
1. Login to the master node of the cluster (e.g. with putty or SSH)
2. Change the working directory to the submission shell script.
3. Make sure that enough disk space is available for temporary saving of flow fields
4. For PBS: Start the simulation with the command:

   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**

CHOICE OF COMPUTE NODES

When using a job submission system, it is not a priori known which compute notes 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, GeoDict allows to change the name of the environment variable in the **Settings->Settings** dialog.


Solver MPI Parallelization

Node File Environment Variable: PBS_NODEFILE

MPI Installation: Automatic

MPI Path (*)

MPI Command-line Arguments

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:

1. 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.
2. 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.
3. Then follow the instructions described above to start a distributed simulation.

Cloud computing

Cloud computing is the on-demand availability of computer system resources, especially data storage and computing power, without direct active management by the user. The term is generally used to describe data centers available to many users over the Internet. Large clouds, predominant today, often have functions distributed over multiple locations from central servers.

Usually, it is possible to lease large shared memory machines or cluster-like environments. These shared memory machines can also be used as virtual machines that allow remote desktop connections. The prices depend on the hardware configuration and range from a few cent per hour to a few dozens of Euro per hour. Cloud computing can be a good alternative to purchasing an own computer system depending on the total usage per year. Especially for peak times it can make sense to supplement the computer resources by cloud computing.

Cloud services

There exist many different cloud solution services. Four examples where GeoDict can be used are:

■ Microsoft Azure	(https://azure.microsoft.com)
■ Google Cloud Platform	(https://cloud.google.com)
■ Amazon Web Services	(https://aws.amazon.com)
■ Rescale	(https://www.rescale.com)
■ KaleidoSim	(https://cloud.math2market.de)

Microsoft Azure is a cloud computing service created by Microsoft for building, deploying, and managing applications and services through a global network of Microsoft-managed data centers. It provides software as a service, platform as a service and infrastructure as a service and supports many different programming languages, tools and frameworks, including both Microsoft-specific and third-party software and systems.

Google Cloud Platform (GCP), offered by Google, is a suite of cloud computing services that runs on the same infrastructure that Google uses internally for its end-user products, such as Google Search and YouTube. Alongside a set of management tools, it provides a series of modular cloud services including computing, data storage, data analytics and machine learning.

Amazon Web Services (AWS) is a subsidiary of Amazon that provides on-demand cloud computing platforms to individuals, companies, and governments, on a metered pay-as-you-go basis. In aggregate, these cloud computing web services provide a set of primitive abstract technical infrastructure and distributed computing building blocks and tools. One of these services is Amazon Elastic Compute Cloud, which allows users to have at their disposal a virtual cluster of computers, available all the time, through the Internet.

Rescale is a startup that develops a cloud computing simulation platform. The Rescale platform combines engineering and science software tools with high performance computing (HPC) to create a cloud simulation environment.

KaleidoSim is a specialist in scientific computing in the cloud. In 2020, Math2Market started to offer GeoDict in the cloud with the Swiss company KaleidoSim. This cloud solution frees the user from the constraints of limited local resources. KaleidoSim offers a simple web framework that allows users to create parameter studies, run the simulations there, and retrieve the data again for detailed analysis locally.

REQUIREMENTS FOR CLOUD COMPUTING

Usage of GeoDict on a cloud service comes with few requirements:

1. A GeoDict floating license is required
2. The floating license server must be accessible from within the cloud computer
3. A valid subscription to one of the cloud services
4. A fast internet connection to the cloud service

The floating license can be obtained by contacting our support.

The second and third requirement depend on the cloud service.

The fourth requirement depend on the location of the cloud server and many cloud service provider have multiple locations that can be chosen when creating a computer resource.

GEODICT ON MICROSOFT AZURE

For usage of Microsoft Azure cloud service we already have scripts and templates available that make the setup and usage of virtual machines much easier.

Please contact our support at: support@math2market.de for more information.

GEODICT ON AMAZON WEB SERVICES

For use cases which require interactive pre-/post-processing or visualization, we offer another solution based on Amazon Web Services (AWS). Here, users can start powerful virtual machine instances and connect to a remote desktop session with GeoDict pre-installed, using only a web browser.

Please contact our support at: support@math2market.de for more information.

GEODICT ON KALEIDOSIM

For usage of the KaleidoSim solution together with GeoDict, we have scripts, templates, and slides available that make it easy to start your first simulation in the cloud. Using a simle web interface,users can upload input structures and GeoPy macros to perform single simulations or parameter studies via batch processing in the cloud. GPU compute instances are offered as well, so that the AI capabilities of GeoDict (e.g., GeoDict-AI, FiberFind-AI, ImportGeo-VOL) ca also be fully utilized.

Please contact our support at: support@math2market.de for more information.

https://doi.org/10.30423/userguide.geodict2022-highperformancecomputing

Technical documentation:

Jürgen Becker

Christian Wagner

Barbara Planas

MATH
2 MARKET

Math2Market GmbH

Richard-Wagner-Str. 1, 67655 Kaiserslautern, Germany
www.geodict.com