Abaqus

Abaqus is a comprehensive Finite Element program system for solving complex linear and non-linear tasks in structural analysis, dynamics, heat conduction and acoustics with large geometry non-linearities and the possibilities of substructure technology. Abaqus is commercial software suite developed by Dassault Systèmes Simulia Corp..

There are various Abaqus software products accessible on the cluster: *Abaqus/Standard*, *Abaqus/Explicit*, *Abaqus/CAE*, *Abaqus/Viewer*.

For a complete list of Abaqus products, after loading the appropriate module, type `abaqus help`.

Abaqus licensing

The use of Abaqus on the cluster system is strictly limited to teaching and academic research for non-industrially funded projects only.

Abaqus analysis or interactive application running on the cluster must contact the license server (provided by LUIS) at the beginning of the execution and periodically during the execution, i.e. Abaqus must have uninterrupted communication with the license server. A single CPU job from `Abaqus/Standard` or `Abaqus/Explicit` requires 5 so-called Analysis Tokens. For each additional CPU per job, an additional analysis token is required. Currently 720 Abaqus license-tokens can be totally utilized by cluster jobs. To display the actual status of the license usage, after loading the Abaqus module(see below), type:

```
abaqus licensing lmstat -a
```

Usage on the cluster

You can list all available Abqaus versions by calling `module avail abaqus`. To load a particular software version, use `module load ABAQUS/<version>`. For example, to activate Abaqus version 2019, type

```
module load ABAQUS/2019
```

Abaqus contains a large number of example problems which can be used to become familiar with Abaqus on the system. These example problems are described in the Abaqus documentation and can be obtained using the Abaqus `fetch` command. For example, the following will extract the input file `s4d.inp` for the test problem **s4d**:

```
abaqus fetch job=s4d
```

Abaqus GUI

The pre- and post-processor `Abaqus/CAE` or the post-processor `Abaqus/Viewer` can be used with a

graphical user interface. The `Abaqus/CAE` GUI can be launched by the command:

```
abaqus cae -mesa
```

Whereas the `Abaqus/Viewer` GUI can be started using:

```
abaqus viewer -mesa
```

Abaqus batch usage

if your model makes use of a **user defined subroutine**, in order to run Abaqus the option `user=<your_subroutine>` has to be provided when calling Abaqus in all batch scripts.

Below is the example batch script `abaqus-serial.sh` for a ***serial*** (single CPU core) run:

abaqus-serial.sh

```
#!/bin/bash -l
#SBATCH --job-name=abaqus_smp
#SBATCH --nodes=1
#SBATCH --ntasks-per-node=1
#SBATCH --mem=4G
#SBATCH --time=00:30:00
#SBATCH --mail-user=user@uni-hannover.de
#SBATCH --mail-type=END

# Load modules
module load ABAQUS/2020
unset SLURM_GTIDS

# Change to work dir:
cd $SLURM_SUBMIT_DIR

# Run Abaqus
abaqus job=my_job input=<my_input_file.inp> scratch=$TMPDIR interactive
```

Submit the file `abaqus-serial.sh` to SLURM using the command: `sbatch abaqus-serial.sh`.

The keyword `interactive` in the script is required to tell Abaqus not to return until the simulation has completed. It is assumed that the input file `<my_input_file.inp>` is located in the job submit directory.

For very large Finite Element models (over 100,000 degrees of freedom), computing in parallel mode using several processors at the same time is often better suited to obtain the result of the analysis more quickly. In the ideal case, the wall-clock time is shortened proportionally to the number of processors involved.

The following is a sample batch script to run Abaqus using **thread**-based parallelization (multiple CPU cores on a single compute node):

abaqus-parallel-smp.sh

```
#!/bin/bash -l
#SBATCH --job-name=abaqus_parallel_smp
#SBATCH --nodes=1
#SBATCH --cpus-per-task=20
#SBATCH --mem=60G
#SBATCH --time=00:30:00
#SBATCH --mail-user=user@uni-hannover.de
#SBATCH --mail-type=END

# Load modules
module load ABAQUS/2020
unset SLURM_GTIDS

# Change to work dir:
cd $SLURM_SUBMIT_DIR

# Set Abaqus environment variables based on SLURM job
expand-slurm-nodelist --abaqus

# Run Abaqus
abaqus job=my_job input=<my_input_file.inp> scratch=$TMPDIR interactive
```

In this mode, the script `expand-slurm-nodelist --abaqus` sets the Abaqus variable `mp_mode=threads` and defines the variable `cpus` within the `abaqus_v6.env` file in the working directory. The file `abaqus_v6.env` is created or overwritten based on the SLURM job parameters, such as node allocation and CPU cores. This ensures Abaqus uses the correct settings for SMP (threading) execution on a single node.

Submit this script using:

```
sbatch abaqus-parallel-smp.sh
```

The following script runs Abaqus using **MPI** parallelization (multiple CPU cores across multiple compute nodes):

abaqus-parallel-mpi.sh

```
#!/bin/bash -l
#SBATCH --job-name=abaqus_parallel_mpi
#SBATCH --nodes=2
#SBATCH --cpus-per-task=20
#SBATCH --mem=120G
#SBATCH --time=00:30:00
#SBATCH --mail-user=user@uni-hannover.de
#SBATCH --mail-type=END

# Load modules
module load ABAQUS/2020
```

```
unset SLURM_GTIDS

# Change to work dir:
cd $SLURM_SUBMIT_DIR

# Set Abaqus environment variables based on SLURM job
expand-slurm-nodelist --abaqus

# Run Abaqus
abaqus job=my_job input=<my_input_file.inp> scratch=$TMPDIR interactive
```

For MPI runs, the script `expand-slurm-nodelist --abaqus` sets `mp_mode=mpi` and defines the `mp_host_list` and `cpus` variables in the file `abaqus_v6.env`. This file is created or overwritten in the working directory based on your SLURM job settings (compute nodes and CPU cores allocated).

Submit this script using:

```
sbatch abaqus-parallel-mpi.sh
```

If you want to override the Abaqus environment parameters `mp_mode`, `mp_host_list` or `cpus` for a particular run, you must provide them explicitly on the Abaqus command line. Command-line arguments take precedence over values in `abaqus_v6.env`. Other Abaqus parameters can be set either in the `abaqus_v6.env` file or on the Abaqus command line, depending on your needs.

Performance tipp: try to fill one node before requesting more than one - inter-node communication (between nodes) is almost always slower than intra-node communication (within a node). So try to stay on one node and only grow if you need to. If your job can use all cores on one node, request them, and remember to also request all memory by setting `--mem=0` in your job script when you fill a node. Have a look in the table of Hardware specifications of cluster compute nodes to find a partition that suits your needs, and test which setup gives you the best performance.

In case you get a `LookupError: unknown encoding: ISO-8859-1`, use `export LANG=en_US.utf8`. Abaqus does not seem to like system other languages.