Nutzeranleitung Einsteiger

Nutzeranleitung Einsteiger

English Version see below

Login

Um auf dem Phoenix-Cluster zu arbeiten, sind einige wenige Grundkenntnisse mit Linux-System erforderlich. Zur Arbeit auf dem Cluster stellen Sie eine SSH-Verbindung zu einem Loginknoten unter `username@phoenix.hlr.rz.tu-bs.de` her. Dabei ist `username` Ihr Benutzername, den Sie für alle Dienste des Gauß-IT-Zentrum nutzen; das Passwort ist das dazu gehörige Passwort.

Falls Sie sich noch nicht für die Nutzung des Phoenix registriert haben, können Sie sich an phoenix-support@tu-bs.de wenden.

```
ssh username@phoenix.hlr.rz.tu-bs.de
```

Für die Verwendung graphischer Anwendungen ist eine X-Weiterleitung mittels des Parameters „-Y“ (nicht „-X“) nötig.

```
ssh -Y username@phoenix.hlr.rz.tu-bs.de
```

Um sich abzumelden, geben Sie einfach den Befehl `exit` ein.

Der Fingerprint zum Login Server lautet:

```
phoenix.hlr.rz.tu-bs.de ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDFDm9AGYEiUIZ6wBwXHoO6YmrMD/QKieMz
phoenix.hlr.rz.tu-bs.de ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABE
phoenix.hlr.rz.tu-bs.de ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAII8Vj77eCWfqTLHNyuW/vvvfALIhd8oTDQX
```

Jobs starten

Wenn Sie mit einem Loginknoten verbunden sind, gibt es zwei Möglichkeiten Jobs zu starten: Direkt per Jobfile (dies ist die übliche Vorgehensweise) oder per Allokierung („Reservierung“) von

Ressourcen mithilfe eines Jobfiles, um interaktiv auf einzelnen Knoten zu arbeiten.

Jobfile

Ein Jobfile bzw. Jobscript sieht beispielsweise wie folgt aus:

```
#!/bin/bash -l

#SBATCH --partition=standard
#SBATCH --nodes=2
#SBATCH --time=12:00:00
#SBATCH --job-name=nearest
#SBATCH --ntasks-per-node=20


working_dir=~
cd $working_dir
module load comp/gcc/6.3.0
module load mpi/openmpi/2.1.0/gcc
mpiexec -np 40 ./test/mpinearest
```

`partition` gibt die Partition (auch Queue genannt) an, auf der Sie rechnen wollen.
`time` gibt die maximale Laufzeit Ihres Jobs an, bevor dieser dann abgebrochen wird.
`job-name` gibt einen Namen an, unter dem Sie Ihren Job später in SLURM wieder finden können.

Eine Startzeit kann mit dem Schalter `–begin` vorgegeben werden. Beispielsweise:

```
--begin=16:00
--begin=now+1hour
--begin=now+60 (seconds by default)
--begin=2010-01-20T12:34:00
```

Bei dem Phoenix-Cluster gibt es verschiedene Arten von Knoten, die genutzt werden können. Die in den Knoten verbaute Hardware kann eingesehen werden, wenn auf die jeweilige Knotenart in der Tabelle geklickt wird.
Diese sind zudem in folgende verschiedene Partitionen aufgeteilt:

Partition	Genutze Knoten	RAM	GPU	Maximale Knotenanzahl/Job	Walltime	Shared
standard	node[001-304]	64 GB	-	20	7 d	Ja
shortrun_small	node[001-304]	64 GB	-	50	3 d	Ja
shortrun_large	node[001-304]	64 GB	-	150	3 d	Ja

Partition	Genutze Knoten	RAM	GPU	Maximale Knotenanzahl/Job	Walltime	Shared
longrun	node[001-304]	64 GB	-	4	14 d	Ja
testing	node[001-304]	64 GB	-	150	1 h	Ja
fat	fat[001-008]	256 GB	-	8	7 d	Nein
vis	vis[01-06]	512 GB	-	6	2 d	Nein
gpu01_queue	gpu[01-02]	64 GB	4x NVIDIA Tesla 16GB	2	3 d	Nein
gpu02_queue	gpu[03-08]	64 GB	4x NVIDIA Tesla 16GB	6	7 d	Nein
gpu03_queue	gpu[01-08]	64 GB	4x NVIDIA Tesla 16GB	8	7 d	Nein

Ein Beispiel für ein Jobfile für einen der GPU-Knoten sieht wie folgt aus:

```
#!/bin/bash -l

#SBATCH --partition=gpu02_queue
#SBATCH --nodes=2
#SBATCH --time=2-00:00:00
#SBATCH --job-name=GPUExample
#SBATCH --ntasks-per-node=4
#SBATCH --gres=gpu:4

~/anaconda3/bin/python "/path/to/file/example.py"
```

Ein Jobfile muss dann mittels `sbatch` gestartet werden.

```
username@login01 [~] sbatch jobname.job
```

Weitere Informationen bietet auch `man sbatch`. Sämtliche dort verwendeten Parameter können auch im Jobscript selber mit `#SBATCH` angeben werden.

Interaktive Anwendungen

Wenn Sie sich Knoten reservieren wollen, um interaktiv auf diesen zu arbeiten, ist dies mit einem simplen Jobfile möglich:

```
#!/bin/bash -l

#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --time=7-00:00:00
#SBATCH --job-name=SleepExample
#SBATCH --ntasks-per-node=1

cd ~/data_dir
sleep 168h
```

Bei der Ausführung per `sbatch` wird dann ein Knoten für die angegebene Zeit benutzt. Da ein Job läuft, kann auf diesen vom Besitzer per SSH zugegriffen werden. Um den richtigen Knoten zu finden, verwenden Sie `squeue` und suchen nach dem angegebenen Jobnamen. Danach aktivieren Sie den Sitzungsmanager mit `screen`.

```
username@login01 [~] sbatch jobname.job
[Knoten finden]
username@login01 [~] ssh node265
username@node265 [~] screen
username@node265 [~] ...
```

Nachdem Sie nun eine Anwendung gestartet haben, können Sie die Sitzung mit `Strg + A + D` in den Hintergrund verschieben. Wichtig ist dabei, dass Sie vorher `screen` eingegeben haben, um den Sitzungsmanager zu starten.

Eine in den Hintergrund verschobene Sitzung wird nicht geschlossen, wenn Sie die SSH-Verbindung trennen. Damit laufen Ihre Prozesse in diesen Sitzungen entsprechend weiter, andernfalls würden Sie beendet werden!

Um eine einzelne Sitzung wieder aufzunehmen, verwenden Sie auf dem entsprechenden Knoten `screen -r`. Sollten mehrere Sitzungen im Hintergrund offen sein, wird eine Liste angezeigt, aus der Sie spezifizieren müssen:

```
username@node265 [~] screen -r

    154577.pts-0.node265    (Detached)
    154308.pts-0.node265    (Detached)
    153240.pts-0.node265    (Detached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
username@node265 [~] screen -r 153240.pts-0.node265
```

Diese Sitzung lässt sich danach mit Eingabe von `screen -d` wieder in den Hintergrund verschieben.

Grafische Anwendungen via VNC nutzen

Mit Hilfe eines VNC (Virtual Network Computing) kann der Bildschirminhalt des Phoenix auf dem eigenen lokalen Rechner in einem Fenster angezeigt werden. Das Bildschirmfenster kann dann wie ein lokales Fenster bedient werden und Tastatur- und Mausbewegungen des eigenen lokalen

Rechners werden an den Phoenix gesendet.

Dazu müssen Sie sich wie üblich beim Phoenix anmelden und einen interaktiven Job in der vis Partition (Queue) starten, ein entsprechendes Job-File könnte so aussehen:

```
#!/bin/bash -l

#SBATCH --partition=vis
#SBATCH --nodes=1
#SBATCH --time=1:00:00
#SBATCH --job-name=nearest
#SBATCH --ntasks-per-node=20

sleep 1h
```

Das Job-File blockt einen vis-Knoten für eine Stunde, in dieser Zeit kann dann via VNC auf dem Phoenix gearbeitet werden.
Als nächsten Schritt muss sich auf dem zugewiesenen vis-Knoten eingeloggt werden:

```
ssh vis0X wobei X der zugewiesene vis-Knoten ist, diesen kann man mit "squeue -u $USER" herausfi
```

Falls noch nie vorher VNC genutzt wurde, muss ein VNC Passwort festgelegt werden, dies geht mit Hilfe des Befehls:

```
/cluster/share/vnc/setup-vnc.sh
```

Bei vergessenem Passwort kann mit dem selben Befehl auch ganz einfach ein neues Passwort festgelegt werden.
Daraufhin muss ein VNC-Server mit dem folgenden Befehl gestartet werden:

```
vncserver
```

Oder für eine bessere Auflösung kann folgender Befehl genutzt werden:

```
vncserver -geometry 1280x1024
```

In einem weiteren Terminal muss nun auf dem eigenen lokalen Rechner ein VNC-Client geöffnet werden. Wir empfehlen **remmina**, bei Windows muss zusätzlich **Xming** installiert und gestartet werden. Den VNC-Client remmina wird über den Befehl gestartet:

```
remmina
```

Eventuell muss vorher, falls remmina nicht startet, der folgende Befehl ausgeführt werden:

```
export DISPLAY=:0
```

Bei remmina muss mit Hilfe des Knopfes in der oberen linken Ecke ein neues Verbindungsprofil erstellt werden mit folgenden Angaben:

- [**Protocol]:** Remmina VNC Plugin
- unter dem Reiter [**Basic]**:
  - **[Server]**: vis0X:Y, wobei X der zugewiesene vis-Knoten ist und Y die Displaynummer des VNC-Servers, diese ist Teil der Ausgabe nach dem Befehl „vncserver“
  - **[Color depth]**: High color (16 bpp)
  - **[Quality]**: Medium

- unter den Reitern **[Advanced]** und **[Autostart]**:
  - keine Änderungen

- unter dem Reiter **[SSH Tunnel]**:
  - **[Enable SSH Tunnel]** auswählen
  - **[Custom]** auswählen und eingeben: phoenix.hlr.rz.tu-bs.de
  - **[Username]**: eigenen Username eingeben
  - Anmeldung per **[Passwort]** auswählen

Daraufhin das Verbindungsprofil mit dem **[Save]**-Knopf speichern und mit dem **[Connect]**-Knopf verbinden. Es wird dann nach dem eigenen Passwort für den Phoenix gefragt und nach Eingabe des eingenen Passworts nach dem VNC-Passwort, das in setpup-vnc.sh Schritt festgelegt wurde.

In dem Fenster kann dann ein Terminal geöffnet werden. Zur Nutzung einer Software im visuellen und interaktiven Modus muss das entsprechende Modul geladen werden. Wenn die Grafikkarte, also zur Visualisierung genutzt werden soll, muss vor dem entsprechenden Terminalbefehl zur Öffnung der Software noch **vglrun** hinzugefügt werden. Um ParaView im interaktiven Modus zu nutzen, muss also nach dem Laden des Moduls folgender Terminalbefehl ausgeführt werden:

```
vglrun paraview
```

Nachdem Sie fertig mit der Nutzung sind, muss der VNC-Server auf dem Phoenix beendet werden, dies geht mit dem Befehl:

```
vncserver -kill :Y wobei Y für die Displaynummer steht
```

Wird der VNC-Server nicht geschlossen, sollte es beim nächsten Einloggen getan werden, ansonsten verschiebt sich die Displaynummer (:1 wird beispielsweise zu :2).

Jobs überwachen

Sie können sich mit dem Befehlt `squeue` eine Liste aller Jobs in SLURM anzeigen lassen. Mehr Informationen gibt es mit `squeue -l`.

```
username@login01 [~]squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               333  standard MD_BA_GA user0001  R      58:56      1 node265
               336  standard     bash username  R       1:12      2 node267,node268
               334  standard     bash user0002  R      50:16      1 node266
```

```
               331       vis      vis user0003  R    1:25:22      1 vis01
               329  standard    name1 user0003  PD     00:00      1 (Resources)
```

`JOBID` ist die ID der Jobs, um diese in SLURM ansprechen zu können. `NAME` ist der von Ihnen gewählte Name zur leichteren Identifikation des Jobs. `ST` gibt den Status des Jobs an (`R` = Running, `PD` = Wartend). `TIME` beschreibt die Zeit, die der Job bereits läuft. Sollte der Job nicht laufen (`PD`), wird unter `NODELIST` statt der Knoten eine knappe Begründung angegeben, warum der Job (noch) nicht läuft.

Jobs abbrechen

Mit der ID des Jobs lässt sich ein Job mit `scancel` abbrechen:

```
username@login01 [~]scancel 336
salloc: Job allocation 336 has been revoked.
```

Modulsystem

Auf dem Phoenix ist das Environment Modules System (kurz Modules) installiert.

Module verändern/erweitern Pfade und Variablen, um Programme und Bibliotheken von anderen Orten zu laden. Jede auf dem Phoenix installierte Software hat ein Modulfile, welches die beim Laden die nötigen Variablen setzt, die für den Einsatz der Software erforderlich sind.

Verfügbare Module anzeigen

Eine Übersicht aller verfügbaren Module erhält man mittels

```
module avail
```

Die aktuell geladenen Module sieht man mit

```
module list
```

Auf diese Weise erkennen Sie auch, welche Software auf dem Phoenix verfügbar ist und in welchen Versionen diese vorliegen.

Module laden und unloaden

Module können mittels

```
module load MODULNAME
```

geladen werden. Es ist auch möglich, mehrere Module auf einmal zu laden. Diese werden dann getrennt durch Leerzeichen eingegeben.

Entfernt werden die Module dann durch

```
module remove MODULNAME
```

Hier ist es ebenfalls möglich, mehrere Module gleichzeitig zu entfernen.

Alle Module auf einmal entfernen mittels

```
module purge
```

Befehle / Aliase automatisch beim Anmelden ausführen

Die Datei /home/username/.bash_profile verhält sich wie die .bashrc auf den meisten anderen Linux Distributionen. Befehle die dort eingetragen werden, werden automatisch beim Anmelden ausgeführt. Zum Editieren kann zum Beispiel nano benutzt werden: nano .bash_profile

Abaqus Job starten

Um Abaqus mit Slurm und MPI kompatibel zu machen, müssen einige Befehle im Jobfile abgearbeitet werden. Hier ist ein Beispiel Jobfile.

```
#!/bin/bash -l

#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --job-name=dingALE
#SBATCH --ntasks-per-node=20
#SBATCH --time=48:00:00
#SBATCH -o bo-%j.log


module purge
module load software/abaqus/abaqus_2016


input_file=dingALE.inp

working_dir=~/DingALE
cd $working_dir

### Create ABAQUS environment file for current job, you can set/add your own options (Python syn
env_file=custom_v6.env


##############################################################################
```

```
cat << EOF > ${env_file}
mp_file_system = (DETECT,DETECT)
EOF

node_list=$(scontrol show hostname ${SLURM_NODELIST} | sort -u)

mp_host_list="["
for host in ${node_list}; do
    mp_host_list="${mp_host_list}['$host', ${SLURM_CPUS_ON_NODE}],"
done

mp_host_list=$(echo ${mp_host_list} | sed -e "s/,$/]/")

echo "mp_host_list=${mp_host_list}"  >> ${env_file}


### Set input file and job (file prefix) name here
job_name=${SLURM_JOB_NAME}


### ABAQUS parallel execution
abaqus job=${job_name} input=${input_file} cpus=${SLURM_NTASKS} standard_parallel=all mp_mode=mp
```

Falls mehr RAM benötigt wird, kann alternativ auch die *fat*-Partition genutzt werden.

Ansys Job starten

Auf das vorhandene Script (Beispiel.sh) mit dem SBATCH und den auszuführenden Befehlen das Script /cluster/share/make_ansys_job_check_for_license.sh anwenden und dazu Lizenzenanzahl mit Lizenzart angeben. Das Skript hängt an den auszuführenden Skript die Lizenzcheck und zusätzlich benötigte Variablen mit an. Ausgegeben wird das Skript in der Standardausgabe, kann also mit Hilfe der Ein-/Ausgabe Umleitung von Linux in eine Datei mit dem Operator „>“ geschrieben werden.

Beispiel für eine normale Ansys CFX Datei, die noch keinen Lizenzcheck besitzt:

```
#!/bin/bash -l

#SBATCH --partition=standard
#SBATCH --nodes=2
#SBATCH --time=3:00:00
#SBATCH --job-name=TCTM
#SBATCH --ntasks-per-node=20

module load software/ansys/18.0


######################
#### HIER VARIABEL ####
######################
```

```
NUMPROCS=20
module load software/ansys/18.0

cfx5solve -batch -chdir $working_dir -double -verbose -def $working_dir/Heater_pt_Loss.def -star
```

Beispiel für eine normale Ansys Fluent Datei, die noch keinen Lizenzcheck besitzt:

```
#!/bin/bash

#SBATCH --partition=standard
#SBATCH --nodes=3
#SBATCH --time=3:00:00
#SBATCH --job-name=ansys_flu
#SBATCH --ntasks-per-node=20
#SBATCH --exclusive

# load module fluent v19.2
module load software/ansys/19.2

# The Journal file
JOURNALFILE=journalFile.jou

# Total number of Processors. Nodes * 20
NPROCS=60

# MPI key, dont change!
export MPI_IB_PKEY=0x8001

fluent 3ddp -g -t $NPROCS -slurm -i $JOURNALFILE > fluent.out
```

Syntax:

```
/cluster/share/make_ansys_job_check_for_license.sh Script.sh NumberOfLicences1 LicenceType1 Numb
```

Beispiel:

```
/cluster/share/make_ansys_job_check_for_license.sh Beispiel.sh 60 ansys > BeispielMitLizenzCheck
```

Wenn Sie nicht wissen, welcher Lizenztyp genutzt werden soll, wird empfohlen, den Typ **ansys** zu nutzen.

Der Lizenzcheck funktioniert so, dass vorher die angegebene Anzahl an Lizenzen geprüft werden und je nachdem das Programm weiterlaufen kann oder nach einer Stunde der Check nochmal durchgeführt wird, wenn nicht genug Lizenzen vorhanden sein sollten.

Ansys für mechanische Simulationen

```
#!/bin/bash -l
#
### Grosse Knoten mit 256GB RAM (fat) SBATCH --partition=fat
```

```
### Kleinere Knoten mit 64GB RAM (standard) SBATCH --partition=standard
### Die Anzahl der Kerne ist bei beiden Knotentypen (fat und standard) gleich, naemlich 20
### Es gibt insgesamt 8 Knoten (fat) mit jeweils 256GB RAM
### ####### Nehmt bitte grundsaetzlich immer zuerst die Standardknoten !!!!! #######
###
### Variable NUMPROC = (#SBATCH --nodes=3) x (SBATCH --ntasks-per-node=20) = 60

#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --time=10:00:00
#SBATCH --job-name=Tensiletest
#SBATCH --ntasks-per-node=20


######################
#### HIER VARIABEL ####
######################
export working_dir=/beegfs/work/y0090888/cfx
#################
NUMPROCS=20
#################
cd $working_dir

export TMI_CONFIG=/cluster/tools/ansys_2018/ansys_inc/v182/commonfiles/MPI/Intel/5.1.3.223/linx6
export I_MPI_FABRICS=shm:tmi
export I_MPI_FABRICS_LIST=tmi
export I_MPI_FALLBACK=0
export I_MPI_TMI_PROVIDER=psm2
#export I_MPI_DEBUG=5

module load software/ansys/19.2


# Befehle:
# Ausführung der Rechnung im Arbeitsverzeichnis: cfx5solve -batch -chdir $working_dir -single -v
# Def-File: -def "filename.def"
# kein Plan was die alle machen (diverse Clusterbefehle), war copy-paste: -start-method 'Intel M
# Benennung des res-Files: -fullname "filename"
# andere Ergebnisdatei als Initialisierung verwenden: -cont-from-file "filename.res"
# Vorgabe eines ccl-Files für Änderungen im def-File (z.B. andere Randbedingung, Druck, etc.): -
#
### komplettes Bsp:cfx5solve -batch -chdir $working_dir -single -verbose -def V3_4_closed.def -c

ansys192 -B -batch -chdir $working_dir -single -verbose -i tensiletest.dat \ -start-method 'Inte
```

> Der folgende Legacy Code ist veraltet. Er sollte nicht benutzt werden, außer Sie haben sehr gute Gründe dafür.

Alte Methode:

Es müssen die Dateien CFX.sh und nodes2ansys.py angelegt werden. Dann kann ein Ansys Job via sbatch CFX.sh gestartet werden. StaticMixer.def sollte durch die eigene Datei ersetzt werden.

CFX.sh:

```
#!/bin/bash -l
# export PATH=/cluster/tools/ansys_inc/v180/CFX/tools/multiport/mpi/lnamd64/intel/bin:$PATH
# export PATH=/cluster/tools/ansys_inc/v180/commonfiles/MPI/Intel/5.1.3.223/linx64:$PATH

#SBATCH --partition=standard
#SBATCH --nodes=2
#SBATCH --time=3:00:00
#SBATCH --job-name=Ansys
#SBATCH --ntasks-per-node=20
#

#######################
#### HIER VARIABEL ####
#######################
export working_dir=$HOME/ansys
# export ALLMACHINES=$(scontrol show hostname $SLURM_JOB_NODELIST |paste -d, -s )
/usr/bin/python nodes2ansys.py
export ALLMACHINES=`cat $HOME/ansys/cfx_nodefile.dat`
echo $SLURM_JOB_NODELIST>$HOME/ansys/slurmjoblist
echo $ALLMACHINES>$HOME/ansys/machines
#################
NUMPROCS=40
#################
cd $working_dir

export TMI_CONFIG=/cluster/tools/ansys_inc/v180/commonfiles/MPI/Intel/5.1.3.223/linx64/etc/tmi.c
export I_MPI_FABRICS=shm:tmi
export I_MPI_FABRICS_LIST=tmi
export I_MPI_FALLBACK=0
export I_MPI_TMI_PROVIDER=psm2

module load software/ansys/18.0


cfx5solve -batch -chdir $working_dir -double -verbose -def $working_dir/StaticMixer.def -start-m
```

nodes2ansys.py:

```
#!/usr/bin/python

import os

# READ SLURM JOB LIST

#line = example data: fat[1-3]
line = os.environ['SLURM_JOB_NODELIST']
```

```
#line = 'fat[001-003]\n'

line.strip()

file = open( "cfx_nodefile.dat" , "w" )

#name = fat , numbers = 001-003
name,numbers = line.split('[')
numbers = numbers.replace(']', '')

#001-003 -> 001 003
start,end = numbers.split('-')

#001 003 -> 1 3
start_int = int(start)
end_int = int(end)

#from 1 to 3 do:
for number in range(start_int, end_int+1):
    number_string = str(number)

    #add the leading zeroes to node-number 2 -> 002
    while len(number_string) < len(start):
        number_string = "0" + number_string

    #write down the nodename like fat002 20 times... you could do this most likely with ntasks-p
    for i in range (1,21):
        file.write(name + str(number_string) + ',' )

file.close
```

Lehre

Für die Lehre existiert eine Queue: teaching und in Zukunft teaching_gpu. Diese Knoten verfügen nicht über eine funktionierende Infinity Band Verbindung, sondern nutzen klassisches Ethernet für die MPI Kommunikation. Daher muss in der Jobfile angegeben werden, dass Ethernet genutzt werden soll.

```
NUMBER_OF_CPU_CORES_TO_USE=16
mpirun -n $NUMBER_OF_CPU_CORES_TO_USE --mca btl tcp,sm,self --mca btl_tcp_if_include eth0  /home
```

Beispiel Jobfile:

```
#!/bin/bash -l

#SBATCH --partition=visalt
#SBATCH --nodes=2
#SBATCH --time=1:00:00
#SBATCH --job-name=nearest
```

```
#SBATCH --ntasks-per-node=8
#SBATCH --exclude=visalt01

module load mpi/openmpi/2.1.0/gcc comp/gcc/6.3.0
mpirun -n 16  --mca btl tcp,sm,self --mca btl_tcp_if_include eth0  /home/raskrato/alt/visalt/pte
```

Users Guide Beginners (English Version)

Login

For your work with the Phoenix cluster you will need a little experience with Linux systems. At first you need to connect via SSH to `username@phoenix.hlr.rz.tu-bs.de`. Where `username` is your account name for all services of the Gauß-IT-Zentrum with the same password.

In case you aren't registered yet, you can contact phoenix-support@tu-bs.de.

```
ssh username@phoenix.hlr.rz.tu-bs.de
```

If you need to work with graphical interfaces, you need to use trusted X11-forwarding with the parameter „-Y“ (instead of „-X“).

```
ssh -Y username@phoenix.hlr.rz.tu-bs.de
```

To log out you can use the command `exit`.

The Fingerprint to the login server is:

```
phoenix.hlr.rz.tu-bs.de ssh-rsa AAAAB3NzaC1yc2EAAAADAQABAAABAQDFDm9AGYEiUIZ6wBwXHoO6YmrMD/QKieMZ
phoenix.hlr.rz.tu-bs.de ecdsa-sha2-nistp256 AAAAE2VjZHNhLXNoYTItbmlzdHAyNTYAAAAIbmlzdHAyNTYAAABE
phoenix.hlr.rz.tu-bs.de ssh-ed25519 AAAAC3NzaC1lZDI1NTE5AAAAII8Vj77eCWfqTLHNyuW/vvvfALIhd8oTDQX
```

Starting jobs

When you are connected to one of the login nodes, there are two ways to start jobs: With a jobfile (usual approach) or by allocating resources to interactively work on nodes.

Jobfile

For example a jobfile (or jobscript) can look like this:

```
#!/bin/bash -l

#SBATCH --partition=standard
#SBATCH --nodes=2
#SBATCH --time=12:00:00
#SBATCH --job-name=nearest
#SBATCH --ntasks-per-node=20


working_dir=~
cd $working_dir
module load comp/gcc/6.3.0
module load mpi/openmpi/2.1.0/gcc
mpiexec -np 40 ./test/mpinearest
```

`partition` determines the partition (queue) you want to use. `time` is the maximum time your job will run before being cancelled. `job-name` is the name for your job to identify in the queuing system.

The jobfile then has to be started with `sbatch`.

```
username@login01 [~] sbatch jobname.job
```

A starting time can be given with `–begin`.

```
--begin=16:00
--begin=now+1hour
--begin=now+60 (seconds by default)
--begin=2010-01-20T12:34:00
```

There are in total eight GPU-nodes. For these there are three GPU queues with different workloads:

- gpu01_queue:

- 2 GPU-nodes used for this queue
- high priority
- just for jobs, that allocate all four GPUs of a node
- a maximum of one job per user
- walltime: 3 days

- gpu02_queue:

- 6 GPU-nodes used
- high priority
- also for jobs, that just allocate a single GPU of a node
- a maximum of two job per user

- walltime: 7 days

- gpu03_queue:

- 8 GPU-nodes used
- low priority
- also for jobs, that just allocate a single GPU of a node
- number of jubs per user is unlimited
- walltime: 7 days
- jobs can be paused, therefore use checkpoints in your code

In the following you can find an example for a jobfile for one of the GPU-node:

```
#!/bin/bash -l

#SBATCH --partition=gpu02_queue
#SBATCH --nodes=2
#SBATCH --time=2-00:00:00
#SBATCH --job-name=GPUExample
#SBATCH --ntasks-per-node=4
#SBATCH --gres=gpu:4  

~/anaconda3/bin/python "/path/to/file/example.py"
```

You can find further information with `man sbatch`. Any parameters found there can also instead be included in the jobscript with `#SBATCH`.

Interactive jobs

To allocate a node for interactive usage you need a simple jobfile:

```
#!/bin/bash -l

#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --time=7-00:00:00
#SBATCH --job-name=SleepExample
#SBATCH --ntasks-per-node=1

cd ~/data_dir
sleep 168h
```

When executing this file with `sbatch` a node will be used for the given time. Since a job is running the user can connect to the node via SSH. To find the node allocated for you, use `squeue` and look for your given job name. After connecting to the node you will have to activate the session manager with `screen`.

```
username@login01 [~] sbatch jobname.job
[Knoten finden]
```

```
username@login01 [~] ssh node265
username@node265 [~] screen
username@node265 [~] ...
```

After starting an application or processes on the node you can press `Crtl + A + D` to move the session to the background.

A session that was moved to the background will not be closed when you disconnect from the node. Processes and application will continue to run in this session.

The get back into sessions use `screen -r`. If you had multiple sessions moved to the background this will open a list from which you can specify:

```
username@node265 [~] screen -r

    154577.pts-0.node265    (Detached)
    154308.pts-0.node265    (Detached)
    153240.pts-0.node265    (Detached)
Type "screen [-d] -r [pid.]tty.host" to resume one of them.
username@node265 [~] screen -r 153240.pts-0.node265
```

A session can be moved to the background again with `screen -d`.

Use graphical applications via VNC

With the help of a VNC (Virtual Network Computing), the screen content of the Phoenix can be displayed in a window on your own local computer. The window can then be operated like a local window. Keyboard and mouse movements of the local computer are sent to the Phoenix.

Therefore, you have to log on to the Phoenix as usual and start an interactive job in the vis partition (queue). A corresponding job file could look like this:

```
#!/bin/bash -l

#SBATCH --partition=vis
#SBATCH --nodes=1
#SBATCH --time=12:00:00
#SBATCH --job-name=nearest
#SBATCH --ntasks-per-node=20

sleep 1h
```

The job file blocks a vis node for 1 hour, during this time you can then work on the Phoenix via VNC.

The next step is to log in to the assigned vis node:

```
ssh vis0X whereby X is the assigned vis node, you can find this out with "squeue -u $USER".
```

If you have never used VNC before, you must set a VNC password, this can be done with the command:

```
/cluster/share/vnc/setup-vnc.sh
```

If you forget your password, you can easily set a new password with the same command.
A VNC server must then be started with the following command:

```
vncserver
```

Or for a better resolution the following command can be used:

```
vncserver -geometry 1280x1024
```

In another terminal on your own local computer a VNC client must now be opened. We recommend **remmina**, for Windows **Xming** must also be installed and started for X11-forwarding. The VNC client remmina is started with the command:

```
remmina
```

If remmina does not start, the following command may have to be executed beforehand:

```
export DISPLAY=:0
```

With remmina, a new connection profile must be created with the help of the button in the upper left corner with the following information:
- **Protocol**: Remmina VNC Plugin
- in the **Basic** tab:

- *Server*: vis0X:Y, whereby X is the assigned vis node and Y is the display number of the VNC server, this is part of the output after the „vncserver“ command
- *Color depth*: High color (16 bpp)
- *Quality*: Medium

- in the **Advanced** and **Autostart** tabs:

- no changes

- in the **SSH Tunnel** tab:

- Select *Enable SSH Tunnels*
- Select *Custom* and enter: phoenix.hlr.rz.tu-bs.de
- *Username*: Enter your own username
- Select authentication via *Password*

Then save the connection profile with the **Save** button and connect with the **Connect** button. You will then be asked for your own password for the Phoenix and after entering this for the VNC

password that was created in the setup-vnc.sh step.

In the window you can open a terminal. To use software in visual and interactive mode the corresponding module has to be loaded. If the graphic card should be used (this is the case for visualisation), then in front of the terminalcommand to open the software **vglrun** has to be added. E.g. to start ParaView in interactive mode, the following command has to executed after loading the corresponding module:

```
vglrun paraview
```

After you have finished using it, the VNC server on the Phoenix must be terminated, this can be done with the command

```
vncserver -kill :Y where Y is the display number
```

If the VNC server is not closed, it should be done the next time you log in, otherwise the display number will be shifted (e.g. :1 becomes :2).

Monitoring jobs

Using `squeue` you get a list of all jobs currently running on the system. You can get more information with `squeue -l`.

```
username@login01 [~]squeue
             JOBID PARTITION     NAME     USER ST       TIME  NODES NODELIST(REASON)
               333  standard MD_BA_GA user0001  R      58:56      1 node265
               336  standard     bash username  R       1:12      2 node267,node268
               334  standard     bash user0002  R      50:16      1 node266
               331       vis      vis user0003  R    1:25:22      1 vis01
               329  standard    name1 user0003  PD     00:00      1 (Resources)
```

`JOBID` is your jobs ID to specify your job to SLURM. `NAME` is the job name you chose for better identification. `ST` is the status of your job (`R` = running, `PD` = pending). `TIME` gives the time your job is currently running. If your job is pending, `NODELIST` will give a short explanation, why it is not (yet) running.

Cancelling jobs

You can cancel your jobs with the ID from `squeue` using `scancel <ID>`.

```
username@login01 [~]scancel 336
salloc: Job allocation 336 has been revoked.
```

Module system

We use the Environment Modules System (short Modules) on the Phoenix.

Modules change and extend pathes and variables to load applications and libraries from different places. All installed software used on the Phoenix has a modulefile which loads the necessary variables to run the software.

Show available modules

For an overview of available modules you can use

```
module avail
```

The currently loaded modules are shown with

```
module list
```

This way you can see which software and which versions are installed on the Phoenix.

Module load and unload

You can load modules with

```
module load MODULNAME
```

You can remove modules with

```
module remove MODULNAME
```

To load or remove multiple modules at once you just have to seperate them by a space.

Executing commands and aliases on login

/home/username/.bash_profile acts like .bashrc on most Linux distributions. Commands added there will be automatically executed on login. To edit the file you can for example use nano: `nano .bash_profile`

Starting Abaqus jobs

To make Abaqus compatible with Slurm and MPI, some commands have to be executed in the jobfile. Here is an exemplary jobfile:

```
#!/bin/bash -l

#SBATCH --partition=standard
#SBATCH --nodes=1
#SBATCH --job-name=dingALE
#SBATCH --ntasks-per-node=20
#SBATCH --time=48:00:00
#SBATCH -o bo-%j.log


module purge
module load software/abaqus/abaqus_2016


input_file=dingALE.inp

working_dir=~/DingALE
cd $working_dir

### Create ABAQUS environment file for current job, you can set/add your own options (Python syn
env_file=custom_v6.env


##############################################################################


cat << EOF > ${env_file}
mp_file_system = (DETECT,DETECT)
EOF

node_list=$(scontrol show hostname ${SLURM_NODELIST} | sort -u)

mp_host_list="["
for host in ${node_list}; do
    mp_host_list="${mp_host_list}['$host', ${SLURM_CPUS_ON_NODE}],"
done

mp_host_list=$(echo ${mp_host_list} | sed -e "s/,$/]/")

echo "mp_host_list=${mp_host_list}"  >> ${env_file}


### Set input file and job (file prefix) name here
job_name=${SLURM_JOB_NAME}


### ABAQUS parallel execution
abaqus job=${job_name} input=${input_file} cpus=${SLURM_NTASKS} standard_parallel=all mp_mode=mp
```

If more RAM is needed, the *fat*-Partition can also be used.

Starting Ansys jobs

We recommend use our new script, which converts your jobfile automaticly. Just write a normal jobfile and call this script and use > to save the output to a new file.

Here is an example for normal Ansys CFX file without any license checks:

```
#!/bin/bash -l

#SBATCH --partition=standard
#SBATCH --nodes=2
#SBATCH --time=3:00:00
#SBATCH --job-name=TCTM
#SBATCH --ntasks-per-node=20

module load software/ansys/18.0


########################
#### HIER VARIABEL ####
########################
NUMPROCS=40
module load software/ansys/18.0

cfx5solve -batch -chdir $working_dir -double -verbose -def $working_dir/Heater_pt_Loss.def -star
```

Here is an example for normal Ansys Fluent file without any license checks:

```
#!/bin/bash

#SBATCH --partition=standard
#SBATCH --nodes=3
#SBATCH --time=3:00:00
#SBATCH --job-name=ansys_flu
#SBATCH --ntasks-per-node=20
#SBATCH --exclusive

# load module fluent v19.2
module load software/ansys/19.2

# The Journal file
JOURNALFILE=journalFile.jou

# Total number of Processors. Nodes * 20
NPROCS=60

# MPI key, dont change!
export MPI_IB_PKEY=0x8001

fluent 3ddp -g -t $NPROCS -slurm -i $JOURNALFILE > fluent.out
```

Here is the code to add an license check:

```
/cluster/share/make_ansys_job_check_for_license.sh Script.sh NumberOfLicences1 LicenceType1 Numb
```

An example:

```
/cluster/share/make_ansys_job_check_for_license.sh myjob.sh 50 aa_r_hpc > myjob_now_working_on_p
```

There are different license types. If you are not sure which one to choose, just use aa_r_hpc.

Following is the old, legacy way of starting an ansys job.

Don't use the code under this warning, unless you have a very good reason to do otherwise.

The files CFX.sh and nodes2ansys.py must be created. Then an Ansys job can be started via sbatch CFX.sh. StaticMixer.def should be replaced with your own file.

CFX.sh:

```
#!/bin/bash -l
# export PATH=/cluster/tools/ansys_inc/v180/CFX/tools/multiport/mpi/lnamd64/intel/bin:$PATH
# export PATH=/cluster/tools/ansys_inc/v180/commonfiles/MPI/Intel/5.1.3.223/linx64:$PATH

#SBATCH --partition=standard
#SBATCH --nodes=2
#SBATCH --time=3:00:00
#SBATCH --job-name=Ansys
#SBATCH --ntasks-per-node=20
#

######################
### HERE CHANGEABLE ###
######################
export working_dir=$HOME/ansys
# export ALLMACHINES=$(scontrol show hostname $SLURM_JOB_NODELIST |paste -d, -s )
/usr/bin/python nodes2ansys.py
export ALLMACHINES=`cat $HOME/ansys/cfx_nodefile.dat`
echo $SLURM_JOB_NODELIST>$HOME/ansys/slurmjoblist
echo $ALLMACHINES>$HOME/ansys/machines
#################
NUMPROCS=40
#################
cd $working_dir

export TMI_CONFIG=/cluster/tools/ansys_inc/v180/commonfiles/MPI/Intel/5.1.3.223/linx64/etc/tmi.c
export I_MPI_FABRICS=shm:tmi
export I_MPI_FABRICS_LIST=tmi
export I_MPI_FALLBACK=0
export I_MPI_TMI_PROVIDER=psm2

module load software/ansys/18.0
```

```
cfx5solve -batch -chdir $working_dir -double -verbose -def $working_dir/StaticMixer.def -start-m
```

nodes2ansys.py

```
#!/usr/bin/python

import os

# READ SLURM JOB LIST

#line = example data: fat[1-3]
line = os.environ['SLURM_JOB_NODELIST']
#line = 'fat[001-003]\n'

line.strip()

file = open( "cfx_nodefile.dat" , "w" )

#name = fat , numbers = 001-003
name,numbers = line.split('[')
numbers = numbers.replace(']', '')

#001-003 -> 001 003
start,end = numbers.split('-')

#001 003 -> 1 3
start_int = int(start)
end_int = int(end)

#from 1 to 3 do:
for number in range(start_int, end_int+1):
    number_string = str(number)

    #add the leading zeroes to node-number 2 -> 002
    while len(number_string) < len(start):
        number_string = "0" + number_string

    #write down the nodename like fat002 20 times... you could do this most likely with nt
    for i in range (1,21):
        file.write(name + str(number_string) + ',' )

file.close
```

Anaconda

Anaconda is a distribution of the Python and R programming languages for scientific computing, that aims to simplify package management and deployment. In this part we will explain, how you can download and install Anaconda on Phoenix and how you can use it to set up custom

environments for different purposes.

Installation

- Go to the Anaconda download page using your web browser: Anaconda Download Page.
- Choose the appropriate distribution for Phoenix „Linux 64-Bit (x86) Installer“ and copy the link address.
- Log in to the Phoenix as usual.
- Use the „wget“ command to download the Anaconda installer using the link you copied (replace <paste_link_here> with the actual link you copied):

```
wget <paste_link_here>
```

- After the download is complete, make the Anaconda installer script executable (replace <version> with the version number you downloaded):

```
chmod +x Anaconda3-<version>-Linux-x86_64.sh
```

- Run the Anaconda installer script. Follow the prompts and accept the license agreement:

```
./Anaconda3-<version>-Linux-x86_64.sh
```

- During the installation process, you will be asked to confirm the installation location. Use the default location, this should be /home/<your_username>/anaconda3. The installation process will take several minutes.

Environment Management

Conda is a powerful package manager and environment manager that you can use in a terminal window. Conda allows you to create separate environments containing files, packages, and their dependencies that will not interact with other environments. When you begin using conda, you already have a default environment named `base`. You don't want to put programs into your base environment, though. Create separate environments to keep your programs isolated from each other. To start the `base` environment, navigate to your `/home`-folder, in there should be a folder named `anaconda3` and use the command:

```
source anaconda3/bin/activate
```

If you installed Anaconda to the default location (your `/home`-folder), you can also use the command:

```
source ~/anaconda3/bin/activate
```

In front of your prompt **(base)** should be displayed.

Created 2024-03-12 13:42:16 UTC by Michael Giemsa
Updated 2024-05-22 16:32:53 UTC by Michael Giemsa