Reynolds cluster user manual

Table of Contents

1 Cluster hardware and OS

The Reynolds cluster consists of:

Machine	**Memory**	**CPU**	**Local disk**	**Purpose**	**GPU?**
headnode	64GB	1x Xeon E5-2620 v4, 8 core	/homereynolds, 15 TB RAID6 /tmp, 19 GB SSD	Login, submit jobs	no
node[01-19]	64GB	2x Xeon E5-2680 v4, 28 core	/tmp, 125 GB SSD	Slurm compute	no
node20	256GB	2x Xeon E5-2680 v4, 28 core	/tmp, 125 GB SSD	Interactive	no
node21,22	64GB	2x Xeon E5-2680 v4, 28 core	/tmp, 125 GB SSD	Slurm compute	Tesla K40m

The OS is Ubuntu 16.04. If your home is on another partition than /homereynolds, it should be automatically mounted.

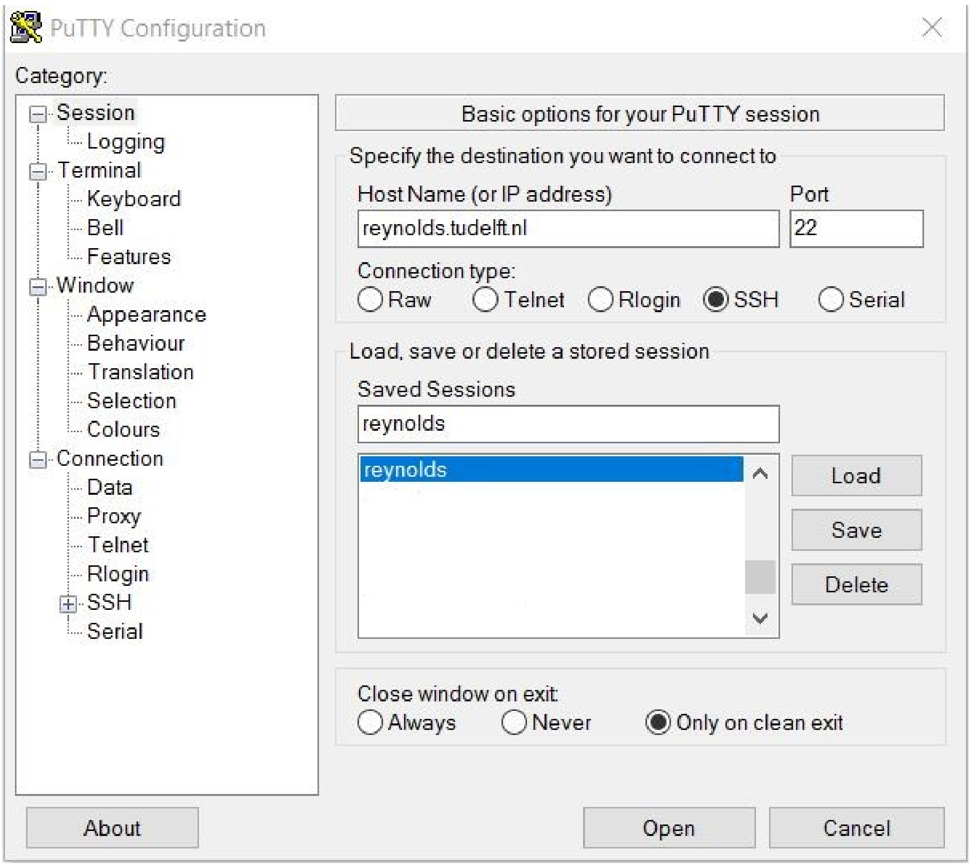
2 Connecting to the Reynolds cluster

2.1 Windows

To connect to the Reynolds cluster, use PuTTY. You can download PuTTY from:

https://the.earth.li/~sgtatham/putty/latest/x86/putty.exe

Save it somewhere (desktop) and run it. Fill in Host Name and Saved Sessions like below.



Specify the destination you want to connect to

Host Name (or IP address): reynolds.tudelft.nl Port: 22

Connection type: Raw Telnet Rlogin [✓ SSH] Serial

Load, save or delete a stored session

Saved Sessions: reynolds

reynolds

Close window on exit: Always Never [✓ Only on clean exit]

Also, to be able to use graphical programs on the cluster, click on +SSH, then on X11 and then "Enable X11 forwarding". Go back to "Session", Click "Save" and if you then click on "Open" you might get a warning about a host key like this:

```
The server's host key is not cached in the registry. You
have no guarantee that the server is the computer you
think it is.
The server's rsa2 key fingerprint is:
ssh-rsa 1024 7b:e5:6f:a7:f4:f9:81:62:5c:e3:1f:bf:8b:57:6c:5a
If you trust this host, hit Yes to add the key to
PuTTY's cache and carry on connecting.
```

Click yes.

Type your username and password, and you're logged in to the machine called "**headnode**". This is the machine that handles the scheduling and logins. You can do simple pre- and postprocessing here, and submit jobs. You can also login to node20, which is a node for interactive (test) work, you can also submit jobs there.

To enable your workstation for receiving graphical output from programs started by a command in PuTTY, in Windows start up either Exceed, or install VcXsrv and run it. VcXsrv is available from https://sourceforge.net/projects/vcxsrv .

2.2 Other OS

If you use OS X or Linux, open a terminal and type (-Y only if you need X forwarding):

```
ssh -Y username@reynolds.tudelft.nl
```

Replace "username" with your username on the reynolds system. You might get a question about the host key, type "yes". Fill in your password when asked, and you're logged in. IF you want to be able to receive the graphical output of GUI-style programs (like mousepad) you have to use a local X server. With OS X you can use XQuartz for that: http://www.hoffman2.idre.ucla.edu/access/x11_forwarding/#Mac_OS_X

3 Job scheduling

The Reynolds cluster uses SLURM as a job scheduler. In this document only a brief usage explanation will be given, there is plenty of documentation available about SLURM on the web, for starters:

http://slurm.schedmd.com/quickstart.html

On the Reynolds cluster there are two partitions available: "normal" and "high". They only differ in priority. User rights regarding the amount of nodes or runtimes, are managed on account- or userlevel. There are two levels at this moment: "standard" and "expert".

Standard user: max 1 node per job, max 4 jobs running, max wall clock time: 8 hours

Expert user: max 10 nodes per job, unlimited jobs running, max wall clock time: 24 hours

The scheduling is done by a "Fairshare" algorithm. This means jobs are scheduled on a first-come-first-served basis, unless someone has already submitted a lot of jobs. Then any queued jobs of others get a higher priority and may be scheduled first.

3.1 Job scripts

Basically, a job script is a text file in which you type the commands that you would on a system without a scheduler, prepended by some hints for the scheduler about your job. Like this example:

```
#!/bin/bash
#SBATCH -D ./                # "." is working folder on the reynolds where you submit from
#SBATCH -J my_job                                  # Name of job
#SBATCH ––mem=2000                                 # Job needs 2 GB memory
#SBATCH ––time=0-8:0:0              # Job needs 0 days, 8 hours. After that SLURM will kill it.
#SBATCH -n 8                                       # number of cores requested
#SBATCH -N 1                                       # number of nodes requested
#SBATCH -o slurm-%N-%j.out                         # output file
#SBATCH -e slurm-%N-%j.err                         # file with error messages

mpirun myjob   # your job. Notice that mpirun via SLURM doesn't need nr of cpus or machinefile.
#or
srun myjob     # like mpirun, and if this works for your job you can get more info via sstat during
#              # the job
#or
# a fluent invocation, see chapter 4.4.1 or something like:
/apps/ansys/ansys-16.2/v162/fluent/bin/fluent 2ddp -t8 -ssh -g -i inputfile &> outputfile
# reminder: the number after "-t" should match the "#SBATCH -n" number.
```

If you have created this script in Windows and want to put it on the cluster, you have two possibilities:

1. use Filezilla to sftp it to the reynolds.tudelft.nl (use ssh protocol). Make sure to create a "unix" type file, if you use Notepad++ set: Edit → EOL Conversion → Unix (LF). Otherwise, in PuTTY on the reynolds, type "dos2unix blah.txt" to convert it to a Unix file.
2. copy it in windows, paste it in your Linux editor. In PuTTY: selecting text is also copy, right-mouse is paste.

If you want to use a "GUI" editor, type "mousepad" (provided X11 forwarding works). Otherwise use vim or nano, of which vim is the most powerful but also a bit harder to learn.
Then, when this job script is in your working directory you can submit it with:

```
sbatch jobscript
```

You will be notified of the Job ID that you can use to see how your job goes, like with "squeue".

3.2 Interactive jobs

There is no queue for interactive jobs, instead login to node20 to do interactive work that can't be done on your own workstation. Don't do interactive high-cpu-usage work on the headnode! To login to the node20, type in putty:

ssh -X node20

and then you're logged in to node20. You can start your programs there.

3.3 Two queues (partitions)

On the Reynolds cluster there are TWO queues, in SLURM also known as partitions: *normal* and *high*. The high partition has a higher priority and is available on request. If you don't specify a partition in your job script with "-p" then your job goes to the normal partition.

This is different than before, there is no expert partition anymore. So remove "-p expert" from your job scripts.

3.4 Time limits

On the cluster the default time limit is 8 hours if you don't set it with

#SBATCH -t days-hours:minutes:seconds

The maximum time for expert users is one day (24 hours), for standard users it's 8 hours. So if you don't specify your time limit, **your job will be killed after 8 hours.** Because SLURM tries to place jobs as efficiently as possible, it is important to set your time limit as low as possible. If your job is short enough, SLURM will schedule it before other big jobs if there is time and cores are available.

3.5 Efficient use of SLURM (and the cluster's resources)

All SLURM nodes (01 - 19) have 28 cores and 64 GB memory. If your job runs more efficient when less than 28 cores per node but still want to equally divide the job, you can set "--ntasks-per-node=24" if you only want 24 processes per node, for example:

```
#SBATCH -n 96  # use 96 cores total
#SBATCH -N 4   # Use 4 nodes
#SBATCH --ntasks-per-node=24  # without this option, the placement could be 28,28,28,12
```

If you have a parallel job that doesn't mind being spread unequally over more than one node, check the availability of the cores with "sinf" (it's an alias from /etc/profile) or sinfo:

```
# sinf
 NODES  STATE    CPUS(A/I/O/T)  NODELIST
     2    mix       8/48/0/56  node[11,19]
    15  alloc      420/0/0/420  node[01,03-10,12-17]
     2   idle       0/56/0/56  node[02,18]
```

Here you see that there are 48 + 56 cores are idle (the 48 in the first and 56 in the third row, labeled "`I`"). So this means if you launch a job with "-n 104" it will run. It is then important NOT to use "--exclusive" or "--ntasks-per-node" when you try to fill in the gaps.

4 Linux usage

On the cluster, the OS is Linux. This means you have to use the command-line. If you don't know Linux please take a look at these sites first.

http://linuxcommand.org/

http://lifehacker.com/5633909/who-needs-a-mouse-learn-to-use-the-command-line-for-almost-anything

http://ryanstutorials.net/linuxtutorial/

4.1 Local folders

On the reynolds cluster, the following two folders are local to the cluster and are the same on all nodes and headnode:

```
/homereynolds
/opt
```

For users, this means anything you store in /homereynolds/<your username> is available on all nodes, which is a requirement for multi-node jobs. In /opt there is software that you can use, but also in the folder /apps (which comes from another server in the TU Delft network).

4.2 Local software

On the cluster the following software is installed in /opt:

What	**where**
`ReFRESCO`	`/opt/refresco/`
`OpenFOAM-v1606+`	`/opt/OpenFOAM/OpenFOAM-v1606+`
`OpenFOAM 3.0.1 with swak4foam`	`/opt/OpenFOAM/OpenFOAM-3.0.1`
`OpenFOAM-extend 3.2`	`/opt/OpenFOAM/foam/foam-exttend-3.2`
`OpenFOAM-4.1`	`/opt/openfoam4`
`OpenMPI 2.0.1`	`/opt/openmpi (system openmpi is in /usr/bin)`
`Intel compiler`	`/opt/intel/`

Please ask your local expert how to use the software.

4.2.1 Intel compiler / MPI

If you want to use the Intel compiler type this in a terminal, or make an alias for it:

```
source /opt/intel/bin/compilervars.sh intel64
```

If you also want to use Intel MPI, source this path too:

```
source /opt/intel2016/compilers_and_libraries/linux/mpi/bin64/mpivars.sh
```

You probably will know if you need Intel MPI. This can be because your software has an option for it or if you have compiled your programs with Intel MPI.

4.2.2 OpenFOAM

OpenFOAM is compiled with OpenMPI and you should not source the Intel mpivars.sh for it. You can use /opt/openmpi (version 2.0.1) or the default /usr/bin/mpirun (version 1.10.2) to run it.

4.3 Useful commands

If you don't have the patience to take a look at the sites mentioned above at chapter 3, here are some of the most used commands.

4.3.1 Files and folders (directories)

Work with directories:

```
cd dir                # change to directory. Only “cd” transfers you to your home.
mkdir, rmdir          # make / remove empty directory. "mkdir -p dir/with/one/two" for whole tree
du -sh ./             # How much diskspace is in your current directory and its subdirs
```

Work with files:

```
cat blah.txt          # view short text files on the command line. Often used with “| grep”
more, less            # view text files. Less is more convient than more. Use q to exit.
rm                    # remove file. Use -f for no questions, use -r for recursive
cp source dest        # copy file. cp -av /dir /tmp/ copies directory /dir to /tmp/dir.
mv yes.txt no.txt     # renames or moves files and dirs
```

Edit files:

```
nano, vi, emacs, ed   # editors, from easy and simple to hard and powerful (ed is only hard)
mousepad, gvim        # GUI editor. Needs X forwarding and local X server
```

Find a file, dir, link, other thing in your current directory, case insentive

```
find . -iname "*hello*"
or
find . |grep -i hello    # this uses 2 commands to do the same
```

Find dir with exact name

```
find . -type d -name Models
```

Find all files less than 1 day old

```
find . -mtime -1
```

Follow a file that holds output during a calculation, to see how the computation goes

```
tail -f output.log     # exit with ctrl-c
```

View the output of a command via "more" or "less":

```
find . -mtime -1 | less       #search with / or ?, exit with q, go to top with g, bottom with G
```

Search for "text" in files:

```
grep text file.*
```

search for "text 123" in all files in current and subdirectories:

```
grep -r "text 123" .
```

search for "text" in output of a command:

```
find . -mtime -1 | grep text
```

4.3.2 Processes

Useful commands on the interactive node node20

```
top    # shows the processes that use the most CPU time
```

in "top" you can kill your own job if you can't stop it otherwise. Just type "k" in top, and type the number of the PID (process ID) (this PID is shown in "top" in the first column) that you want to kill. If you can't kill it, try signal 9 instead of the defaul 15 (you are asked that) You can exit "top" with the letter q.

4.3.3 SLURM interaction

Useful commands on the headnode, to check out the status of your job:

```
scontrol show job <jobid>
squeue
sinfo
sacct
scancel jobID # to remove your job from the queue
```

When you login you have three aliases for SLURM things: sque, sinf, sact. You can see what they do if you type:

```
type sinf
```

to see what it actually is. You can change this alias to something you like better and put it in your $HOME/.bashrc file as:

```
alias sinf='some other variant of sinfo with options'
```

It will then be active the next time you login, or direct if you type this alias command on the command-line.

As with most commands, use "man command" to see the manual of the command. Exit man with "q". Search in man with "/". Search next occurance: "n". Search backwards: "?". Go to the top: "g". Go to the bottom: "G". These keys are also used by the command-line file viewer "less" (because man uses less).

4.4 Fluent

It is possible to run a multi-node Fluent job via a slurm batch script or interactively (but not with a GUI).

4.4.1 Fluent via sbatch script

You need to find out if you Fluent job will scale efficiently over the number of cores you intend to use. Otherwise it might even run slower than on less cores.

A possible Fluent sbatch script, for running Fluent version 18.2 over 2 nodes, 4 cores each (this way you are able to use 2x 60GB of memory for one job, even if the Fluent job doesn't scale well over 8 cores):

```
#!/bin/bash -l
# use bash as shell
# the -l option is necessary to make the local enviroment equal to the login environment

# slurm commands start with #SBATCH
# all other stuff behind a # on the same line is comment
# type "man sbatch" for more info about the options
# This example runs a 2d fluent job on 2 nodes, 4 cores per node.
# It creates a hostsfile via srun.

#SBATCH -D ./                       # working directory (./ means "the present directory")
#SBATCH -J testfluent               # name of the job (can be changed to whichever name you like)
#SBATCH --get-user-env              # to set environment variables
#SBATCH -o output.log               # output file
#SBATCH -e output.log               # error file
#SBATCH --mem=2000                  # memory per node in Megabytes
#SBATCH --time=10:00                # simulation time in d-hh:mm:ss
#SBATCH --mail-type=BEGIN,END,FAIL  # can be ALL, NONE, BEGIN, END, FAIL, REQUEUE and others
#SBATCH -n 8                        # number of CPUS
#SBATCH -N 2                        # number of nodes
#SBATCH --ntasks-per-node=4         # number of processes per node
```

```
HOSTSFILE=.hostlist-job$SLURM_JOB_ID
if [ "$SLURM_PROCID" == "0" ]; then
  srun hostname -f |sort > $HOSTSFILE
  export PATH=/apps/ansys/ansys-18.2/ansys_inc/v182/fluent/bin:$PATH
  fluent 2ddp -ssh -g -t $SLURM_NTASKS -i inputfile.dat -cnf=$HOSTSFILE
  rm -f $HOSTSFILE
fi
exit 0
```

4.4.2 Fluent interactively

To get 2 nodes via Slurm for your cause so you can use them interactively, you can type:

```
srun -n 8 -N 2 --ntasks-per-node=4 --mem=50000 --time=1-0 --pty bash
```

This will get you 2 nodes, space for 8 cores, divided over 4 cores per node. It reserves 50GB per node, and after 1 day your session will end. If you get awarded the job slot immediately, it will look like you ssh'ed to a random node. There you can start a fluent (or any other session) on the command line. For instance:

```
$ srun hostname -f | sort > fluent-hostfile.txt
$ time /apps/ansys/ansys-18.2/ansys_inc/v182/fluent/bin/fluent 2ddp -ssh -g -t 8 -i inputfile.dat
-cnf=fluent-hostfile.txt
```

The first line creates a hostfile with the nodes allocated to your session. The second line starts a fluent job, using this hostfile and at the end shows you the runtime.

This way is fine to play around with fluent for > 1 node on the command line, but the disadvantage is that you might have to wait until you are awared with a job slot.