Writing a cluster job script for Fluent

Introduction

The purpose of this document is to explain the steps required in the development of a job script for running Fluent on the CHPC's Lengau cluster.

General comments

By default, in most Linux distributions, the user is given the "bash" interface shell. There are many shells available for Linux, such as ash, csh, ksh, zsh, tcsh, etc. All of these have some advantages and disadvantages. Unless you know exactly what you are doing, simply use the default bash shell. The hash symbol "#" in bash (and other shells) is used to indicate that the rest of the line is a "comment", in other words, it is ignored by the bash interpreter. If you attempt to simply run a PBS job script as a normal Linux script, all lines starting with # will be ignore. However, if you **qsub** the script, the lines starting with **#PBS** will be interpreted by the PBS scheduler as instructions for it.

Unless you are an expert, keep your job scripts as simple as possible. It will make it far easier to debug when things don't work. You can always add complexity later once things are working smoothly.

A line-by-line guide to a simple PBS script for Fluent

```
#!/bin/bash
```

This line instructs the system that the script must be interpreted with bash.

```
#PBS -l select=4:ncpus=24:mpiprocs=24:mem=32GB:nodetype=haswell_reg
```

This line requests resources from the PBS scheduler. The resources that we are requesting are:

- 4 "chunks" or units of processing resources
- Each chunk should contain 24 processor cores
- We are going to assign 24 MPI ranks to the processor cores that we have requested, in other words, one MPI ranks for each core.
- Reserve 32 GB of memory for each chunk
- Request the nodes that have haswell_reg processors

We prefer it if a "chunk" is the same a physical compute node. It does not have to be. You can request 8 chunks with 12 cores and 12 MPI ranks, and chances are that you will get exactly the same compute nodes. PBS will probably assign two chunks to each compute node. When using Fluent, there is no need to mess around. If you need 24 or more parallel processes, make sure that you ask for complete compute nodes, as in the above example.

CFD runs do not normally require very large amounts of memory. Don't request memory at all, unless you require an abnormally large amount, which is very unlikely.

You don't have to request the nodetype. The system default will be fine.

On Lengau, do NOT request more than 24 cores or MPI ranks per chunk. Our default compute nodes have only 24 cores each. If you request more than 24, the job will sit in the queue for ever. We do have a very small number of 56-core nodes, but those are not relevant to this document.

The above example line will request resources for a total of 96 MPI processes. Try to match the number of MPI processes intelligently to the size of your case. In our experience, you should aim for about 10 000 to 20 000 cells per MPI process. If you are using very simple physics, that number can go as low as 3 000 cells per MPI process, but it is safer to rather aim for the higher numbers, which will ensure efficient parallelism.

`#PBS -l cfd_base=1`

`#PBS -l anshpc=92`

These lines inform the scheduler how many licenses your job will need. For a Fluent run, you need 1 **`cfd_base`** license, and (n-4) **`anshpc`** licenses, where n is the number of MPI processes that you will use.

- Requesting the license in the PBS script is not mandatory. However, it avoids the situation of your job attempting to start, and failing, when insufficient licenses are available. In other words, PBS will hold your job back until the requested number of licenses become available.
- Currently (end of April 2021), this works only for R 21.1 of the software. Due to some compatibility issues, users of older versions need to check out licenses from a license server running on chpclic1 rather than login1, and no license resource management is available for the interim license server running on chpclic1.
- If you are using an older version (pre- R 21.1), don't insert these two lines in your job script.

`#PBS -q normal`

Specify the queue that you will be using.

- For less than a full compute node of 24 cores, use the **`serial`** queue
- For one full 24-core compute node, use the **`smp`** queue
- For 2 to 10 compute nodes, use the **`normal`** queue
- For 11 to 100 compute nodes, use the **`large`** queue, but you will need to request access and demonstrate that you can use it efficiently
- Paying commercial users can use the **`express`** queue

`#PBS -P myprojectcode`

The system has to be informed from which project's allocation the resources should be drawn. myprojectcode will be something like **`MECH1234`**

`#PBS -l walltime=1:00:00`

Put some effort into estimating how long your job will run, and ask for a bit more time than that. This example requests one hour. Make sure that you ask for enough to prevent the scheduler from killing your job prematurely, but don't request much more than you need, because this may delay the start of your job as the scheduler has to find a suitable slot for it.

```
#PBS -o /home/username/lustre/FluentTesting/fluent.out
#PBS -e /home/username/lustre/FluentTesting/fluent.err
```

These two lines specify where the stdout and stderr streams will be written to file. It is really important to ensure that these paths are valid. If they are not, you will not be able to find these files in order to check and diagnose any problems with your runs. Use your own username, don't use "username" here.

```
#PBS -m abe
#PBS -M username@email.co.za
```

These two lines are optional. They instruct the scheduler to send you an email on **a**bort, **b**eginning and **e**xit of your job.

```
export LM_LICENSE_FILE=1055@login1
export ANSYSLMD_LICENSE_FILE=1055@login1
export PATH=/apps/chpc/compmech/CFD/ansys_inc/v211/fluent/bin:$PATH
export FLUENT_ARCH=lnamd64
```

These are not instructions for PBS. These environment variables are set in order to inform Fluent where to find the license server, where to find Fluent and to specify the architecture (Linux running on a computer using the AMD-64 instruction set). For now, for older versions of Fluent (such as 195 or 202) use chpclic1 instead of login1 as a license server. If you set these environment variables in your **$HOME/.bashrc** file, it is not necessary to have them in the job script.

```
export PBS_JOBDIR=/home/username/scratch/FluentTesting
cd $PBS_JOBDIR
```

Specify the directory where your job is located, and change to that directory. You must use a valid path here.

```
nproc=`cat $PBS_NODEFILE | wc -l`
```

The purpose of this line is to find the number of MPI processes that will be launched. The scheduler writes a hostname for each MPI process in a file which can be accessed by means of the environment variable **PBS_NODEFILE**. The above bash instruction simply counts the number of lines in that file, and assigns that number to the environment variable **nproc**. If you want to check the contents of that file, you can put the instruction

```
cat $PBS_NODEFILE
```

in your job script, and the list of nodes will be written to the stdout stream.

```
fluent 3d -t$nproc -pinfiniband -cnf=$PBS_NODEFILE -g <
fileContainingTUIcommands > run.out
```

This is the all-important command line to run Fluent.

- You have already set the **$PATH** environment file to the directory containing the Fluent executable, therefore you don't need to give the full path here. **fluent** is all that is needed.
- You need to select the **2d, 2ddp, 3d** or **3ddp** version (two-dimensional or three-dimensional, single or double precision) of the solver
- **-t$nproc** specifies the number of MPI processes. Doing it this way means that you only need to specify the required resources with the **#PBS -l select=….** Directive and the license resource request.
- You need to instruct the solver to use the high-speed Infiniband network with the **pinfiniband** option.
- **-cnf=$PBS_NODEFILE** instructs the system where to place the MPI processes
- The **-g** option suppresses the graphical user interface, which is not by default available when the code is run n batch mode.
- **< fileContainingTUIcommands** instructs Fluent to process batch instructions from a text file of that name containing Fluent Text User Interface commands.
- **> run.out** redirects the output to a file called run.out. Once the run is going, you can monitor progress here with the command tail -f run.out
- Refer to the Fluent documentation on how to put together a set of working TUI commands. You can also experimentally develop these instructions by typing them into the text input box (or from the command line when running with the -g option) and verifying the syntax. A typical file with TUI instructions looks like this:

```
file/confirm-overwrite no

file/read-case-data caseFile

solve/iterate 500

file/write-case-data caseFileEnd

exit
```

- If you want to use a normal Fluent journal file, or wish to create images "on the fly" while the job is running, use the following command line options instead of simply just -g:

```
-gu -driver null -i fluentJournalFile.txt
```

Where fluentJournalFile.txt contains Fluent journal file statements.

Submitting and monitoring the job

Submit the job to the system with the command

```
qsub jobscript.txt
```

where jobscript.txt is the name of the job script file.

You can check progress with the command:

```
qstat -awu auser
```

where auser is YOUR username on Lengau. The status should change from Q to R once the job is running.

Terminating the job

You can terminate a job with the command

```
qdel
```

followed by the job number which you were able to find from the qstat command. There will also be a Fluent cleanup script in your job directory. You can run this with a command like

```
sh ./cleanup-fluent-cnode*.sh
```

which will attempt to kill all remaining Fluent processes.