Using Fluent on HPC @ UW's CHTC

General Information

CHTC has their own information here:

https://chtc.cs.wisc.edu/uw-research-computing/hpc/guides.html

However, it can be somewhat confusing and contains lots of information that is not necessary for running Fluent. It also neglects some information shared here.

Getting an Account:

Start by getting an account here:

https://chtc.cs.wisc.edu/uw-research-computing/form.html

Make sure to request usage of the HPC, not the HTC, if following this guide. The HTC uses a different process and will require separate instructions. The HPC allows for up to 128 cores and 512 GB of RAM per node and is probably sufficient for most Fluent applications.

Useful Software:

WinSCP

WinSCP is extremely useful when using the HPC, it allows for moving files between your computer and the HPC, moving files within directories in the HPC, and editing batch files. It provides a visual GUI to see directories which makes troubleshooting and updating batch files significantly easier. You can download it for free here:

https://winscp.net/download/WinSCP-6.3.7-Setup.exe/download

One note with WinSCP is that it does not refresh directories automatically, if you expect to see a file and don't, try CTRL + R to refresh the directory.

PuTTY:

The CHTC recommends the use of PuTTY as a terminal editor, it's not particularly useful but you can download it here:

https://www.putty.org/

Notepad ++:

If you don't already have Notepad++ you may find it useful for editing large and complicated batch files. For most use cases, editing with WinSCP is probably sufficient.

https://notepad-plus-plus.org/downloads/

Getting Access to Fluent

Access to Fluent is not given by default to HPC users, you will need to request access from the CHTC staff. They will put you into the correct Unix group which will allow you to access the Fluent license. Without this access Fluent will crash and won't provide any error codes.

Their email is: **chtc@cs.wisc.edu**

Connecting to HPC

If you are not on UW Net (Campus wi-fi), you will need to connect to the full vpn using GlobalProtect. Then to connect to the HPC you will need to utilize either a terminal (Powershell) or WinSCP to ssh into the server.

ssh netID@spark-login.chtc.wisc.edu

Where 'netID' is your personal netID. It will then prompt you for your netID password, this is not your CAE password if they're different. Finally, it will require DUO authentication. With WinSCP you can save the address and password to make logging in easier. A second DUO authentication is required when moving or editing files (keep your phone nearby when using the HPC).

I typically open both a WinSCP window to edit files and move files between directories or computers and a Powershell window to run commands and submit jobs.

Operating Space

You will probably want to store and run your files out of the scratch directory as by default you are given 100 GB of storage in this directory, whereas the home directory only gives you 30GB. By default, when logging into HPC you are placed into your home/netID directory. To get to your scratch directory first get out of the home directory by running twice:

cd ..

This command moves you back to the previous directory. From here you can change directory into your personal scratch directory (and any subdirectories within that directory).

cd scratch/netID

It's easy to do this with WinSCP's GUI but make sure you're in the right directory when submitting jobs through a terminal.

You can request more storage space with the proper justification.

Creating Submit Files

Requestion Resources:

Submit files (also called “batch” files in this guide) tell HPC what to do, both by reserving resources and running commands. The top half of the file contains the options that reserve nodes, CPUS, RAM etc., and looks like the following:

```
#!/bin/sh
#This file is called submit-script.sh
#SBATCH --partition=shared       # default "shared", if not specified
#SBATCH --time=0-04:30:00       # run time in days-hh:mm:ss
#SBATCH --nodes=1               # require 1 nodes
#SBATCH --ntasks-per-node=64    # cpus per node (by default, "ntasks"="cpus")
#SBATCH --mem=4000             # RAM per node in megabytes
#SBATCH --error=job.%J.err
#SBATCH --output=job.%J.out
```

The partition should always be chosen as ‘shared’ unless running interactive jobs in ‘int’. The maximum time for a job on the shared partition is 7 days, try to set this to a realistic timeline. This guide only shows how to utilize a single node, but jobs can use more than one node if necessary. CHTC asks that CPUs be requested in intervals of 32 up to 128. They also request that each CPU be requested for 4 GB of memory (4000 as the value), and if more memory is needed to increase CPUs requested rather than increase memory per CPU. This helps to manage memory usage between users operating on the same node.

You need to adjust the output and error functions to put the output and error files into either your home or scratch directory. I typically make a directory inside each project to output to which helps keep things organized. Here’s what a line could look like:

#SBATCH --error=/scratch/netID/subdirectory/outputs/job.%J.err

Loading Ansys:

Ansys can be loaded by utilizing the module load command:

module load Ansys/version

At the time of creating this guide, the ‘version’ used is 2024r2. To find the available versions you can utilize this command:

module avail Ansys

You can also just have it load the default version of Ansys if the version doesn’t matter:

module load Ansys

Remember that this command only works once you’ve been given access to the Fluent licenses by CHTC staff.

Navigating to the Correct Directory

You should be in the same directory as your submit file already but I typically include a cd command to change into the desired directory (or verify that I am already there).

Running the Fluent Journal File

Fluent can then be opened and run with a journal file and associated case file with this command:

fluent 3ddp -g -t32 -i "journal_name.jou" > "$output_folder/run_name_fluent.log"

fluent → load fluent

3ddp → run fluent in 3D double precision mode

-g → runs fluent in batch (headless) mode, this prevents it from loading the GUI

-t32 → this specifies the number of CPU threads to use, this needs to match what was requested earlier under 'ntasks-per-node'

-i "journal_name.jou" → specifies the input journal file to use to control fluent

"$output_folder/run_name_fluent.log" → redirects the fluent output to a log file to be saved and viewed later, this stores it in a specific folder with a given name

A complete sample batch file is provided with this guide.

Fluent Journal Files

Journal files can be created using Fluent's built in journal creator, however I've noticed several issues with these. My best guess is that they rely (particularly for parametric calculations) on workbench features that are not loaded in when Fluent is in headless mode. For this reason, I make most of the journal files myself and for parametric studies just create different journals that slightly modify the base case file. Below is an example of a batch script I created to parametrically change the inlet velocity. This code, when pasted below the code which reserves resources and changes into the appropriate directory, will make journals of various inlet velocities (1 – 10 m/s) and execute them.

Fluent also seems to have a tendency of running in interactive mode when changing the case file. For instance, when updating the inlet conditions, it will run through all the options for defining the inlet condition, rather than just updating the one parameter specified. You can see how I deal with this in the below code (the lines of yes/no/values). This makes writing the journal file tedious as you must fully define the inlet condition, not just what needs to be changed.

One very important note with journal files is that you must use the full unshortened path to the case file, this includes the 'mnt/cephfs' directories, otherwise fluent will not find your case file.

```
# Define inlet velocities and result storage location
v_inlet_list=(1 2 3 4 5 6 7 8 9 10)
base_folder="/mnt/cephfs/scratch/netID/subfolder/Results"

for v_inlet in "${v_inlet_list[@]}"; do
    output_folder="${base_folder}/vinlet_${v_inlet}"

    # Ensure both parent and fluent_files directories exist
    mkdir -p "${output_folder}"

    # Create the journal file (with updated boundary condition logic)
    cat > "run_${v_inlet}.jou" <<EOF
; Read base case
file/read-case "/mnt/cephfs/scratch/netID/subfolder/case_file_name.cas"

; Set inlet boundary conditions (fully specified)
define/boundary-conditions/velocity-inlet velinlet
no
no
yes
yes
no
${v_inlet}
no
0
no
no
yes
5
10

; Initialize and run
solve/initialize/initialize-flow
solve/iterate 500

; Exit Fluent
exit yes
EOF

    # Run Fluent and capture all its output to the correct location
    fluent 3ddp -g -t96 -i "run_${v_inlet}.jou" > "${output_folder}/vinlet${v_inlet}_fluent.log"

done
```