Running OpenFOAM and LAMMPS coupled by CPLibrary on ARCHER2

Gavin J. Pringle, EPCC, University of Edinburgh
Gabriele Gennari, University of Nottingham
19 July 2023

Table of Contents

Introduction

This document describes how to employ cpl-library, CPL_APP_OPENFOAM and CPL_APP_LAMMPS-DEV onto ARCHER2 using modules and how to install them yourself from scratch.

NB In all the examples in this document and in the example batch scripts provided here and in the git repositories, take care to **replace my username, `gavincpl`, with your own, and replace my project code, `y23`, with your own**.

Logging in to Archer2

```
ssh gavincpl@login.archer2.ac.uk
```

Employing matplotlib

The examples to plot the results can employ matplotlib. If you wish to install matplotlib on Archer2, then the recommended way is via a local virtual environment as follows, replacing y23 with your project code and gavincpl with your username.

.

```
module load cray-python
python -m venv --system-site-packages
/work/y23/y23/gavincpl/myvenv
source /work/y23/y23/gavincpl/myvenv/bin/activate
```

```
python -m pip install -U pip
python -m pip install -U matplotlib pyqt5
```

To deactivate the environment, type:

```
deactivate
```

To use your own installation of matplotlib within a batch job, include the following line

```
source /work/y23/y23/gavincpl/myvenv/bin/activate
```

Further details regarding installing python packages on Archer2 can be found via https://docs.archer2.ac.uk/user-guide/python/.

Reading and Plotting Data with PyDataView

Both LAMMPS and OpenFOAM write all data in ascii format, and because they use a simple uniform grid, pyDataView is a simple tool to view the data in this format. If you'd prefer not to, then you can use paraview or any other of your favourite lammps chunk visualisation tool instead (be sure to point them at the lammps or openfoam directory). The advantage of pyDataView here is it is designed to view the data for the coupled case, putting both domains in the right position using coupler_header in the cpl folder. To use pyDataView, you need python with matplotlib (as well as scipy, numpy and wxpython for the GUI). On ARCHER2, wxpython is not trivial to install, so we use just matplotlib with `pyqt5` backend to generate plots. To get pyDataView, first clone it, ideally to your work directory,

```
git clone git@github.com:edwardsmith999/pyDataView.git
```

Then in scripts which use it, you should change the ppdir

```
ppdir = '/path/where/pyDataView/has/been/cloned/'
sys.path.append(ppdir)
import postproclib as ppl
```

Running using centrally-installed modules

You can either install all the components yourself or simply the centrally-installed version via modules.This section described using the modules.

Dummy CFD coupled with dummy MD application

Setup default environment for a dummy CFD application coupled with a dummy MD application

Upon logging in, please run these three commands, which will make the CPL and OpenFOAM compilers, executables, and libraries available via the command line on the login nodes.

```
module load other-software
module load cpl-openfoam/2106
source $FOAM_CPL_APP/SOURCEME.sh
```

If matplotlib is needed for plotting or within python scripts, remember to employ your own installation via:

```
source /work/y23/y23/gavincpl/myvenv/bin/activate
```

Running the minimal_send_recv_mocks example

This copies the necessary files into your /work directory. This must be done in /work as /home is not visible on the back-end compute nodes.

Note there are C++ and Fortran90 examples compiled using `cplc++` or `cplf90`, respectively.

After setting up the default environment as above:

```
cd /work/y23/y23/gavincpl
cp -r $CPL_PATH/examples/minimal_send_recv_mocks .
cd minimal_send_recv_mocks
cplc++ minimal_MD.cpp -o MD
cplc++ minimal_CFD.cpp -o CFD
```

Below is an example batch job file, but remember to change the account, directories and username accordingly.

```
#!/bin/bash

#SBATCH --job-name=my_cpl_test
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none

# you will need to change the account code
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard

# at least two nodes are required, as each application coupled requires at
least one node each
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
```

```
export OMP_NUM_THREADS=1

module load other-software
module load cpl-openfoam/2106
source $FOAM_CPL_APP/SOURCEME.sh

# set SHARED_ARGS environment variable
SHARED_ARGS="--distribution=block:block --hint=nomultithread"

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1 MD :
--het-group=1 --nodes=1 --tasks-per-node=1 CFD
```

You can now submit your batch job file named, say, example_archer2.bat, using

```
sbatch example_archer2.bat
```

You can monitor your job using

```
squeue -u gavincpl
```

You can cancel your job using

```
scancel <jobid>
```

NB This batch script couples two applications which each have their own MPI_Comm_world communicators. It is possible to run the two applications within a single MPI_Comm_world communicator, and this involves only a small number of changes to the batch script with no need to rebuild. See Section “Shared vs distinct MPI communicators”.

OpenFOAM coupled with dummy MD application

Setup default environment for OpenFOAM coupled with a dummy MD

Upon logging in, please run these three commands, which will make the CPL and OpenFOAM compilers, executables, and libraries available via the command line on the login nodes.

```
module load other-software
module load cpl-openfoam/2106
source $FOAM_CPL_APP/SOURCEME.sh
```

If matplotlib is needed for plotting or within python scripts, remember to employ your own installation via:

```
source /work/y23/y23/gavincpl/myvenv/bin/activate
```

Running the CPLTestFoam example

This copies the necessary files into your /work directory. This must be done in /work as /home is not visible on the back-end compute nodes.

Note there is a Fortran90 example which is compiled using `cplf90`.

After setting up the default environment for OpenFOAM coupled with a dummy MD:

```
cd /work/y23/y23/gavincpl
cp -r $FOAM_CPL_APP/examples/CPLTestFoam .
cd CPLTestFoam
cplc++ minimal_MD.cpp -o MD
```

Below is an example batch job file, but remember to change the account, directories and username accordingly.

```
#!/bin/bash

#SBATCH --job-name=my_cpl_test
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none

# you will need to change the account code
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard

# at least two nodes are required, as each application coupled requires at
least one node each
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load other-software
module load cpl-openfoam/2106
source $FOAM_CPL_APP/SOURCEME.sh

# run blockMesh and decompasePar
# NB these are both serial codes and should only be run within a parallel job
if
# they are short and the parallel jobs does not employ a high core count
```

```
blockMesh
decomposePar -force

# set SHARED_ARGS environment variable
SHARED_ARGS="--distribution=block:block --hint=nomultithread"

# srun the MD executable and the CPLTestFoam executable in a shared
communicator
# MD is built locally
# CPLTestSocketFoam is not copied but its location resides in the users PATH
srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1 MD :
--het-group=1 --nodes=1 --tasks-per-node=1 CPLTestFoam -parallel
```

You can now submit your batch job file named, say, example_archer2.bat, using

```
sbatch example_archer2.bat
```

Running the CPLTestSocketFoam example

After setting up the default environment for OpenFOAM coupled with a dummy MD:

```
cd /work/y23/y23/gavincpl
cp -r $FOAM_CPL_APP/examples/CPLTestSocketFoam .
cd CPLTestSocketFoam
cplc++ minimal_MD.cpp -o MD
```

And an example batch script:

```
#!/bin/bash

#SBATCH --job-name=my_cpl_test
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none

# you will need to change the account code
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard

# at least two nodes are required, as each application coupled requires at
least one node each
#SBATCH --nodes=2
```

```
# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

# once again load the modules
module load other-software
module load cpl-openfoam/2106
source $FOAM_CPL_APP/SOURCEME.sh


# run blockMesh and decompasePar
# NB these are both serial codes and should only be run within a parallel job
if
# they are short and the parallel jobs does not employ a high core count
blockMesh
decomposePar -force

# set SHARED_ARGS environment variable
SHARED_ARGS="--distribution=block:block --hint=nomultithread"

# srun the MD executable and the CPLTestScoketFoam executable in a shared
communicator
# MD is copied from the y23 project, but can be built locally
# CPLTestSocketFoam is not copied but its location resides in the users PATH
srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2 MD :
--het-group=1 --nodes=1 --tasks-per-node=2 CPLTestSocketFoam -parallel
```

Running the interCondensatingEvaporatingFoam example

After setting up the default environment for OpenFOAM coupled with a dummy MD:

```
cd /work/y23/y23/gavincpl
cp -r $FOAM_CPL_APP/examples/interCondensatingEvaporatingFoam .
cd interCondensatingEvaporatingFoam
cplc++ minimal_MD.cpp -o MD
```

And an example batch script:

```
#!/bin/bash

#SBATCH --job-name=my_cpl_test
#SBATCH --time=0:10:0
#SBATCH --exclusive
```

```
#SBATCH --export=none

# you will need to change the account code
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard

# at least two nodes are required, as each application coupled requires at
least one node each
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

# once again load the modules
module load other-software
module load cpl-openfoam/2106
source $FOAM_CPL_APP/SOURCEME.sh

# run restor0Dir tool
. ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions
restore0Dir

# run blockMesh and decompasePar
# NB these are both serial codes and should only be run within a parallel job
if
# they are short and the parallel jobs does not employ a high core count
blockMesh > log.blockMesh 2>&1
decomposePar > log.decomposePar 2>&1

# set SHARED_ARGS environment variable
SHARED_ARGS="--distribution=block:block --hint=nomultithread"

# srun the MD executable and the CPLinterCondensatingEvaporatingFoam
executable in a shared communicator
# MD is copied from the y23 project, but can be built locally
# CPLinterCondensatingEvaporatingFoam is not copied but its location resides
in the users PATH
srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2 MD :
--het-group=1 --nodes=1 --tasks-per-node=2
CPLinterCondensatingEvaporatingFoam -parallel
```

```
#clean up
reconstructPar > log.reconstructPar 2>&1
```

Running the interFoamHardtPhaseChange example

After setting up the default environment for OpenFOAM coupled with a dummy MD:

```
cd /work/y23/y23/gavincpl
cp -r $FOAM_CPL_APP/examples/interFoamHardtPhaseChange .
cd interFoamHardtPhaseChange
cplc++ minimal_MD.cpp -o MD
sbatch slurm_script
```

where `slurm_script` is similar to

```
#!/bin/bash
#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:30:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-python

source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_OPENFOAM/SOURCEME.sh

# load restor0Dir tool
. ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions
restore0Dir

blockMesh > log.blockMesh 2>&1
decomposePar > log.decomposePar 2>&1
```

```
SHARED_ARGS="--distribution=block:block --hint=nomultithread"

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2 MD :
--het-group=1 --nodes=1 --tasks-per-node=2 CPLinterFoamHardtPhaseChange
-parallel

reconstructPar > log.reconstructPar 2>&1
```

LAMMPS coupled with dummy CFD application

Setup default environment for LAMMPS coupled with a dummy CFD

Upon logging in, please run these two commands, which will make the CPL and LAMMPS compilers, libraries and executables available via the command line on the login nodes.

```
module load other-software
module load cpl-lammps
```

Running the Couette_coupled/Partial_overlap test

After setting up the default environment for LAMMPS coupled with a dummy CFD:

```
cd /work/y23/y23/gavincpl
cp -r $LAMMPS_CPL_APP/test/Couette_coupled .
cd Couette_coupled/Partial_overlap
```

And an example batch script is provided:

```
#!/bin/bash

#SBATCH --job-name=my_cpl_test
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none

# you will need to change the account code
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard

# at least two nodes are required, as each application coupled requires at
least one node each
#SBATCH --nodes=2
```

```
# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load other-software
module load cpl-lammps

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1 lmp_cpl -in
./one_wall_imaginary_sliding_coupled.in :  --het-group=1 --nodes=1
--tasks-per-node=1 python ./python_dummy/CFD_test_vs_Couette.py
```

LAMMPS coupled with OpenFOAM

Setup default environment for coupling LAMMPS and OpenFOAM

Upon logging in, please run these four commands, which will make the CPL, OpenFOAM and LAMMPS compilers, libraries and executables available via the command line on the login nodes.

```
module load other-software
module load cpl-openfoam
source $FOAM_CPL_APP/SOURCEME.sh
module load cpl-lammps
```

Running the LAMMPS_OPENFOAM case

After setting up the default environment for coupling LAMMPS and OpenFOAM, set-up your working directory:

```
cd /work/y23/y23/gavincpl
cp -r $CPL_PATH/examples/LAMMPS_OPENFOAM .
cd LAMMPS_OPENFOAM
sbatch example_archer2.bat
```

where example_archer2.bat batch script looks like the following but with your username and your project instead of `gavincpl` and `y23`, respectively.

```
#!/bin/bash
#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:30:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --partition=standard
```

```
#SBATCH --qos=standard
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load other-software
module load cpl-openfoam
source $FOAM_CPL_APP/SOURCEME.sh
module load cpl-lammps

cd openfoam
python clean.py -f
blockMesh
decomposePar
cd ..

SHARED_ARGS="--distribution=block:block --hint=nomultithread"

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2  CPLIcoFoam
-case ./openfoam -parallel : --het-group=1 --nodes=1 --tasks-per-node=2
lmp_cpl -i lammps.in
```

Viewing the Output

Once the run has finished, we need to have logged into ARCHER2 with x forwarding (`ssh -X username@login.archer2.a.uk`) and have matplotlib setup as discussed above. We need to clone a version of pyDataView to read the data, then run the python script

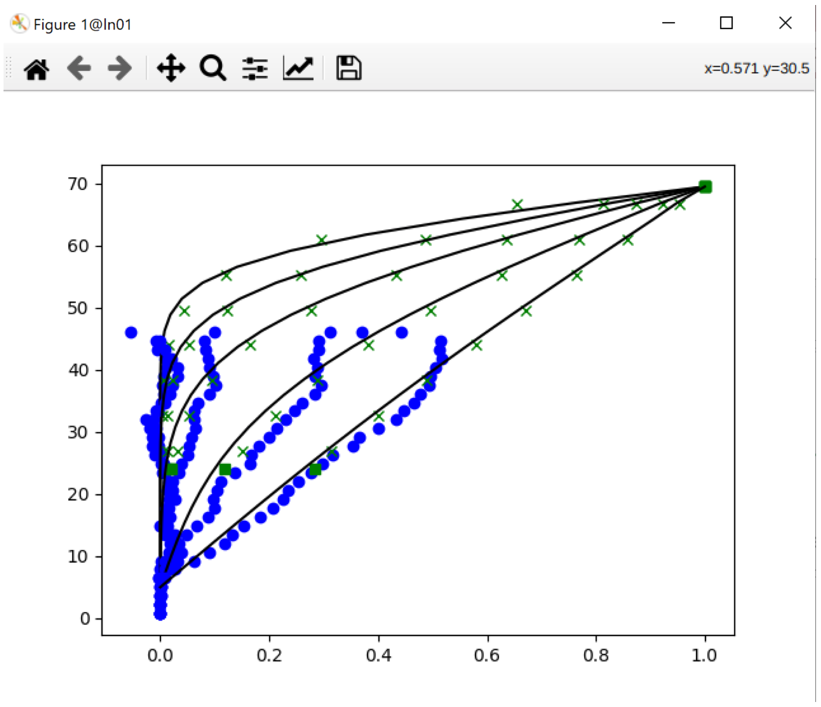
```
python plot_coupled.py
```

If the plot has finished, output should looks something like this,

```
Available outputs in ./ include:

                field                          records
      CPL Momentum           =     799.000000,
      CPL Velocity           =     799.000000,
```

And an image should pop up:



Changing the number of cores

This Section presents how to change this coupled run to a different size, processor topologies or numbers of cells.

Starting from examples/LAMMPS-OPENFOAM folder we change both LAMMPS and OpenFOAM codes and COUPLER.in.

Domain Size

LAMMPS

For LAMMPS, domain size is changed by adjusting number of FCC units when constructing the system in lammps.in

```
# Domain size and walls
variable        x equal 10
variable        y equal 24
variable        z equal 10
variable    wallwidth equal 1.0
```

The domain size is then in multiples of "1.679596191", where this value is based on density 0.8442 as this determines the spacing for the FCC cells. As such, changing variable x is as follows:

```
variable        x equal 10
```

where domain was 16.795961913825074 to
```
variable        x equal 12
```
your domain would become 20.15515429659

In order to make sure the cell size is still consistent (assuming you have 8 cells) we also need to adjust the compute:

compute layers all chunk/atom bin/3d x lower 2.09949524 y lower 1.41715925 z lower 2.09949524 units box ids every

So, previously
dx = domain / cells = 16.795961913825074/8 = 2.09949524
now,
dx = domain / cells = 20.15515429659/8 = 2.51939428707375

which means we update to

compute layers all chunk/atom bin/3d x lower 2.51939428707375 y lower 1.41715925 z lower 2.09949524 units box ids every

OpenFOAM

To change OPENFOAM to ensure the domain is the same size as follows. Edit the file `examples/LAMMPS-OPENFOAM/openfoam\constant\polyMesh\blockMeshDict`, and adjust the following:

```
scale 16.795961913825074;

interp_BC true;

vertices
(
    (0 0 0)
    (1.0 0 0)
    (1.0 2.7 0)
    (0 2.7 0)
    (0 0 1.0)
    (1.0 0 1.0)
    (1.0 2.7 1.0)
    (0 2.7 1.0)
);
```

Where domain is 1.0*scale in x and z with 2.7*scale in y. To update x components from 10 units (1.0) to 12 units (1.2)

```
vertices
(
    (0 0 0)
    (1.2 0 0)
    (1.2 2.7 0)
    (0 2.7 0)
    (0 0 1.0)
    (1.2 0 1.0)
    (1.2 2.7 1.0)
    (0 2.7 1.0)
);
```

Be sure to run

```
blockmesh
decomposePar -force
```

to update the input file for OpenFOAM. The commands appear in the batch script.

This changes the number of molecules solved for in the MD code (so MD will be slower) while keeping the grid in OpenFOAM the same. In this way, you could span up domain sizes to test MD scaling without having to change the CFD code.

Number of Processors

The number of processes used for LAMMPS could be adjusted using

```
processors 4 1 1
```

in lammps.in. For OpenFOAM, we change

```
openfoam/system/decomposeParDict
```

and the lines

```
numberOfSubdomains 2;

method          simple;

simpleCoeffs
{
    n               ( 2 1 1 );
    delta           0.001;
}
```

where both 2 for numberOfSubdomains and n must be changed

```
numberOfSubdomains 4;

method          simple;
```

```
simpleCoeffs
{
    n               ( 4 1 1 );
    delta           0.001;
}
```

and then the processor folders generated

```
decomposePar -force
```

After this the command line call, submission or run.sh scripts must be changed, e.g.

```
cplexec -c 4 "CPLIcoFoam -case ./openfoam -parallel" -m 4 "lmp_cpl
-i lammps.in"
```

Grid

For grid changes to OpenFOAM, edit `openfoam\constant\polyMesh\blockMeshDict` and change the second brackets under blocks (currently 8x8x8 grid)

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (8 8 8) simpleGrading (1 1 1)
);
```

Which we could swap to 12

```
blocks
(
    hex (0 1 2 3 4 5 6 7) (12 8 8) simpleGrading (1 1 1)
);
```

Be sure to run

```
blockmesh
decomposePar -force
```

either now or as part of the job submission.

Next open `COUPLER.in` under cpl folder and change to be consistent

```
OVERLAP_EXTENTS
1
12           #PREVIOUSLY 8
```

```
CONSTRAINT_INFO
2
0
1
12      #PREVIOUSLY 8

BOUNDARY_EXTENTS
1
12      #PREVIOUSLY 8
```

We also no need to update the compute in lammps, to reflect that it is now split into 12 cells,

dx = domain / cells = 20.15515429659/12 = 1.679596191

so the compute line becomes,

```
compute layers all chunk/atom bin/3d x lower 1.679596191 y lower
1.41715925 z lower 2.09949524 units box ids every
```

Installing locally from scratch

The Sections above describe how to use the modules of ARCHER2 to employ the centrally-installed versions of CPL library, OpenFOAM and LAMMPS. In this section, we describe how to install CPL library and LAMMPS in your work directory, and how to employ the centrally-installed version of OpenFOAM. This work is close to the bare metal and is intended for code developers.

The following employs the project code `y23` and the username `gavincpl`. Change all occurrences to your own project code and your own username.

Setup default environment

```
module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-python
```

Enabling the CPL Library

If CPL library is not installed

This must be done in /work as /home is not visible on the back-end compute nodes.

```
cd /work/y23/y23/gavincpl
git clone https://github.com/Crompulence/cpl-library.git
cd cpl-library
```

```
make PLATFORM=ARCHER2
source SOURCEME.sh
```

If you have uploaded your ssh public key to github then you can make changes and submit pull requests if you clone using

```
git clone git@github.com:Crompulence/cpl-library.git
```

If CPL library is already installed and up-to-date

Each time you log in, run the following command.

```
source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
```

Batch scripts to run test to check CPL library

First, visit the test directory

```
cd /work/y23/y23/gavincpl/cpl-library
cd examples/minimal_send_recv_mocks
```

Then create your favourite two of the following four executables:

```
cplf90 minimal_MD.f90  -o ./f_MD
cplf90 minimal_CFD.f90  -o ./f_CFD
cplc++ minimal_MD.cpp  -o ./c_MD
cplc++ minimal_CFD.cpp  -o ./c_CFD
```

Then create a batch script, `slurm.bat` say, which looks like the following, where you have updated the project and username accordingly.

```
#!/bin/bash
#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-python
module load xthi
source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
```

```
SHARED_ARGS="--distribution=block:block --hint=nomultithread"

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1
xthi : --het-group=1 --nodes=1 --tasks-per-node=1 xthi

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1
f_MD : --het-group=1 --nodes=1 --tasks-per-node=1 f_CFD
```

Note 1: this batch script loads the xthi module and launches two instances of xthi. This is not required to run coupled applications; however, the application reports which nodes are being employed and, as such, are a good way to check that you are using ARCHER2 as expected.

Note 2: this example batch script above assumes a shared MPI_Comm_world communicator across both applications. We provide an additional example batch script below where each application has its own MPI_Comm_world communicator. (See below).

You can now submit your job using

```
sbatch slurm.bat
```

You can monitor your job using

```
squeue -u gavincpl
```

You can cancel your job using

```
scancel <jobid>
```

Changing the number of tasks per application

The default setup is to use 1 MPI task for CFD and 1 MPI task for MD.

Let's assume you wish to run using *num_mpi_tasks* MPI tasks for an application then,for C, it is the first element of the `npxyz[3]` declaration or, for Fortran,the first element of the `CFD_COMM` declaration, that holds *num_mpi_tasks*. The batch script is then updated such that `nodes x tasks-per-node` = *num_mpi_tasks*.

NB the number of MPI tasks for the CFD application cannot be larger than the number of MPI tasks for the MD application.

For example, if you wish to use 2 MPI tasks for the MD app and 1 MPI tasks for the CFD app then, change the line in the C++ file `minimal_MD.cpp`:

```
    int npxyz[3] = {1, 1, 1}; int periods[3] = {1, 1, 1};
```

to read

```
    int npxyz[3] = {2, 1, 1}; int periods[3] = {1, 1, 1};
```

Similarly, for the Fortran90 case, edit `minimal_MD.f90` and change

```
   call MPI_Cart_create(CFD_COMM, 3, (/1, 1, 1/), &
```

to read

```
   call MPI_Cart_create(CFD_COMM, 3, (/2, 1, 1/), &
```

Recompile and then update the batch script to employ `–nodes=1 –tasks-per-node=2` for the MD application.

Shared vs distinct MPI communicators

Coupling codes on ARCHER2 is referred to as running multiple heterogeneous jobs, where each job resides inside its own so-called het-group.

There are two ways to run coupled jobs on ARCHER2: the first assumes both codes share the same MPI_Comm_world communicator, whilst the second assumes both codes have their own distinct MPI_Comm_world communicators. For more information on heterogenous jobs on Archer2, please visit the following URL:
https://docs.archer2.ac.uk/user-guide/scheduler/#heterogeneous-jobs

The example batch script above assumes a shared MPI communicator across both applications.

The next two examples show a simple coupling of two MPI applications running on one core each, first with a shared communicator and then with a distinct communicator. The text is highlighted where the differences occur. The CPL repository includes examples with comments.

Shared MPI communicators example

The following example batch script for shared MPI communicators, with comments removed for clarity.

```
#!/bin/bash
#SBATCH --job-name=shared_example
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=2
export OMP_NUM_THREADS=1
module load other-software
module load cpl-openfoam/2106
source $FOAM_CPL_APP/SOURCEME.sh
SHARED_ARGS="--distribution=block:block --hint=nomultithread"
srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1 CFD :
--het-group=1 --nodes=1 --tasks-per-node=1 MD
```

Distinct MPI communicators example

The following example batch script for shared MPI communicators, with comments removed for clarity.

```
#!/bin/bash
#SBATCH --job-name=distinct_example
#SBATCH --time=0:10:00
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --export=none
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1
#SBATCH hetjob
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=1
#SBATCH --tasks-per-node=1
#SBATCH --cpus-per-task=1
export OMP_NUM_THREADS=1
export PMI_UNIVERSE_SIZE=3
export MPICH_SINGLE_HOST_ENABLED=0
module load other-software
module load cpl-openfoam/2106
source $FOAM_CPL_APP/SOURCEME.sh
SHARED_ARGS="--distribution=block:block --hint=nomultithread"
srun ${SHARED_ARGS} --het-group=0 CFD &
srun ${SHARED_ARGS} --het-group=1 MD &
wait
```

Running an OpenFOAM example

This section describes how to run an OpenFOAM example, where the input files have already been acquired and reside in your working directory.

```
cd /work/y23/y23/gavincpl/OpenFoam-2DCouette
module load openfoam/com/v2106
export HOME='/work/y23/y23/gavincpl/OpenFoam-2DCouette'
source ${FOAM_INSTALL_DIR}/etc/bashrc
sbatch of_job.bat
```

where `of_job.bat` contains

```
#!/bin/bash
#SBATCH --job-name=OFtest
#SBATCH --nodes=1
#SBATCH --tasks-per-node=128
```

```
#SBATCH --cpus-per-task=1
#SBATCH --time=01:00:00

#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard

module load openfoam/com/v2106

export HOME='work/y23/y23/gavincpl/OpenFoam-2DCouette'
source ${FOAM_INSTALL_DIR}/etc/bashrc

SHARED_ARGS="--distribution=block:block --hint=nomultithread"

# OpenFOAM run
./Allclean # This wipes the case run previously
blockMesh > log.blockMesh 2>&1
decomposePar > log.decomposePar 2>&1
srun ${SHARED_ARGS} icoFoam -parallel > log.icoFoam 2>&1
reconstructPar > log.reconstructPar 2>&1
```

CPL_APP_OPENFOAM

These cases run a coupled test OpenFOAM application with a dummy MD simulation.

First, setup CPL as above, and the default module environment has been loaded, namely

```
module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-python
```

If matplotlib is needed for plotting or within python scripts, remember to employ your own installation via:

```
source /work/y23/y23/gavincpl/myvenv/bin/activate
```

If CPL_APP_OPENFOAM is not installed

Now acquire and configure CPL_APP_OPENFOAM using the following commands

```
cd /work/y23/y23/gavincpl
git clone https://github.com/Crompulence/CPL_APP_OPENFOAM.git
cd CPL_APP_OPENFOAM
# the following SOURCEME.sh is different from the SOURCEME.sh above
source SOURCEME.sh
# the following updates the CPLPstream directory depending on the
underlying OpenFOAM version: here it is v2106.
cd src;ln -s CPLPstream_v2106 CPLPstream;cd ..
```

If you have uploaded your ssh public key to github then you can make changes and submit pull requests if you clone using

```
git clone git@github.com:Crompulence/CPL_APP_OPENFOAM.git
```

If CPL_APP_OPENFOAM is already installed and up-to-date

Each time you log in, run the following command.

```
source /work/y23/y23/gavincpl/CPL_APP_OPENFOAM/SOURCEME.sh
```

make pstream

The library pstream is the OpenFOAM communications library which is built using MPI. Here, we are building our own version of pstream using our CPL library. The process is only one line:

```
make pstream
```

make cpltestfoam

This case builds and runs a coupled test OpenFOAM application with a dummy MD simulation. To test on 1 MPI task for the MD and 1 MPI task for the CFD test application, CPLTestFoam.

```
make cpltestfoam
cd examples/CPLTestFoam
cplc++ minimal_MD.cpp -o MD
sbatch slurm_script
```

where the file slurm_script contains the following (remember to change your project name and username):

```
#!/bin/bash

#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23

#SBATCH --partition=standard
#SBATCH --qos=standard

#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1
```

```
module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-python
module load xthi

source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_OPENFOAM/SOURCEME.sh

blockMesh
decomposePar -force

SHARED_ARGS="--distribution=block:block --hint=nomultithread"

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1
xthi : --het-group=1 --nodes=1 --tasks-per-node=1 xthi

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1 MD
: --het-group=1 --nodes=1 --tasks-per-node=1 CPLTestFoam -parallel
```

make cpltestsocketfoam

This case runs a small OpenFOAM test case with a dummy MD simulation.

```
cd $FOAM_CPL_APP
make cpltestsocketfoam
cd examples/CPLTestSocketFoam/
cplc++ minimal_MD.cpp -o MD
sbatch slurm_script
```

where slurm_script contains the following:

```
#!/bin/bash

#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23

#SBATCH --partition=standard
#SBATCH --qos=standard

#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-python
module load xthi

source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_OPENFOAM/SOURCEME.sh

blockMesh
decomposePar -force

SHARED_ARGS="--distribution=block:block --hint=nomultithread"

#srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2
xthi : --het-group=1 --nodes=1 --tasks-per-node=2 xthi

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2 MD
: --het-group=1 --nodes=1 --tasks-per-node=2 CPLTestSocketFoam
-parallel
```

make cplinterfoam

This case runs a small OpenFOAM simulation, using `CPLinterCondensatingEvaporatingFoam`

```
cd $FOAM_CPL_APP
make cplinterfoam
cd examples/interCondensatingEvaporatingFoam
cplc++ minimal_MD.cpp -o MD
sbatch slurm_script
```

where slurm_script is similar to

```
#!/bin/bash
#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:30:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-python
```

```
source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_OPENFOAM/SOURCEME.sh

# load restor0Dir tool
. ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions
restore0Dir

blockMesh > log.blockMesh 2>&1
decomposePar > log.decomposePar 2>&1

SHARED_ARGS="--distribution=block:block --hint=nomultithread"

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2 MD
: --het-group=1 --nodes=1 --tasks-per-node=2
CPLinterCondensatingEvaporatingFoam -parallel

reconstructPar > log.reconstructPar 2>&1
```

make cplinterfoamhardtphasechange

This case runs a small OpenFOAM simulation, using `CPLinterFoamHardtPhaseChange`

```
cd $FOAM_CPL_APP
make cplinterfoamhardtphasechange
cd examples/interFoamHardtPhaseChange
cplc++ minimal_MD.cpp -o MD
sbatch slurm_script
```

where slurm_script is similar to

```
#!/bin/bash
#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:30:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-python

source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_OPENFOAM/SOURCEME.sh
```

```
# load restor0Dir tool
. ${WM_PROJECT_DIR:?}/bin/tools/RunFunctions
restore0Dir

blockMesh > log.blockMesh 2>&1
decomposePar > log.decomposePar 2>&1

SHARED_ARGS="--distribution=block:block --hint=nomultithread"

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2 MD
: --het-group=1 --nodes=1 --tasks-per-node=2
CPLinterFoamHardtPhaseChange -parallel

reconstructPar > log.reconstructPar 2>&1
```

CPL_APP_LAMMPS-DEV

As before, log in and setup CPL, then configure CPL_APP_LAMMPS-DEV using the following commands.

NB this case runs LAMMPS with a dummy CFD simulation and does not employ OpenFOAM *per se*, and yet the openfoam module is loaded. This is to ensure that this step towards a fully coupled LAMMPS-OpenFOAM employs the target environment variables.

```
module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-fftw
module load cray-python
cd /work/y23/y23/gavincpl
source cpl-library/SOURCEME.sh
git clone https://github.com/Crompulence/CPL_APP_LAMMPS-DEV.git
git clone -b stable https://github.com/lammps/lammps.git mylammps
cd CPL_APP_LAMMPS-DEV
echo "/work/y23/y23/gavincpl/mylammps" > CODE_INST_DIR
source SOURCEME.sh
cd config; ./enable-packages.sh make; cd ..
make patch-lammps
make CC=CC LINK=CC
```

If you have uploaded your ssh public key to github then you can make changes and submit pull requests if you clone using
`git clone git@github.com:Crompulence/CPL_APP_LAMMPS-DEV.git`

NB if you have made changes, say, to the packages, it is good practice to stash personal files, remove CPL_APP_LAMMPS-DEV entirely, clone again and replace your personal files.

Running the Couette_coupled/Partial_overlap test

```
cd CPL_APP_LAMMPS-DEV/test/Couette_coupled/Partial_overlap
sbatch example_archer2.bat
```

where example_archer2.bat contains

```
#!/bin/bash

#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:10:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-fftw
module load cray-python
source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_LAMMPS-DEV/SOURCEME.sh

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=1 lmp_cpl -in
./one_wall_imaginary_sliding_coupled.in :  --het-group=1 --nodes=1
--tasks-per-node=1 python ./python_dummy/CFD_test_vs_Couette.py
```

LAMMPS_OPENFOAM

As before, log in and set up cpl-library, CPL_APP_OPENFOAM and CPL_APP_LAMMPS-DEV, as above.

If these are all up-to-date, then we can simply configure our environment to run the coupled LAMMPS/OpenFOAM example using the following.

```
module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-fftw
module load cray-python
source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_OPENFOAM/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_LAMMPS-DEV/SOURCEME.sh
```

Running the LAMMPS_OPENFOAM example

```
cd /work/y23/y23/gavincpl/cpl-library/examples/LAMMPS_OPENFOAM
```

```
sbatch example_archer2.bat
```

where `example_archer2.bat` contains

```
#!/bin/bash
#SBATCH --job-name=my_cpl_demo
#SBATCH --time=0:30:0
#SBATCH --exclusive
#SBATCH --export=none
#SBATCH --account=y23
#SBATCH --partition=standard
#SBATCH --qos=standard
#SBATCH --nodes=2

# single thread export overriders any declaration in srun
export OMP_NUM_THREADS=1

module load openfoam/com/v2106
module load gcc/10.3.0
module load cray-fftw
module load cray-python
source /work/y23/y23/gavincpl/cpl-library/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_OPENFOAM/SOURCEME.sh
source /work/y23/y23/gavincpl/CPL_APP_LAMMPS-DEV/SOURCEME.sh

cd openfoam
python clean.py -f
blockMesh
decomposePar
cd ..

SHARED_ARGS="--distribution=block:block --hint=nomultithread"

srun ${SHARED_ARGS} --het-group=0 --nodes=1 --tasks-per-node=2  CPLIcoFoam -case
./openfoam -parallel : --het-group=1 --nodes=1 --tasks-per-node=2 lmp_cpl -i
lammps.in
```