Project OpenFOAM: How to Deploy OpenFOAM-v2012 applications on Pythagoras Cluster

Panagiotis Nastou, PhD
Department of Mathematics
School of Sciences
University of the Aegean

Petroudis Georgios, McS
Department of Mathematics
School of Sciences
University of the Aegean

Intoduction

OpenFOAM is the free, open source CFD software developed primarily by OpenCFD Ltd since 2004. It has a large user base across most areas of engineering and science, from both commercial and academic organizations. OpenFOAM has an extensive range of features to solve anything from complex fluid flows involving chemical reactions, turbulence and heat transfer, to acoustics, solid mechanics and electromagnetics.

User .bashrc modifications

The user should modify his/her .bashrc properly so as to use OpenFOAM libraries. In home user directory edit .bashrc,

```
# vi .bachrc
```

and add at the end of the file the following lines,

```
# source /share/apps/OpenFOAM/OpenFOAM-v2012/etc/bashrc

#ThirdParty Libs and Compilers

ThirdParty=/share/apps/OpenFOAM/ThirdParty-v2012/platforms

export PATH=${ThirdParty}/linux64/gcc-
6.4.0/bin:${ThirdParty}/linux64Gcc64/openmpi-1.10.4/bin:$PATH

export LD_LIBRARY_PATH=${ThirdParty}/linux64/mpfr-
4.1.0/lib:${ThirdParty}/linux64/mpc-1.2.1/lib:${ThirdParty}/linux64/gcc-
6.4.0/lib:${ThirdParty}/linux64/gcc-6.4.0/lib64:${ThirdParty}/linux64/gmp-6.2.1/ lib:
${ThirdParty}/linux64Gcc64/openmpi-1.10.4/lib64:$LD_LIBRARY_PATH

export LIBRARY_PATH=${ThirdParty}/linux64/gcc-6.4.0/lib:${ThirdParty}/linux64/gcc-
6.4.0/lib64:${ThirdParty}/linux64/gmp-6.2.1/lib:
${ThirdParty}/linux64Gcc64/openmpi-1.10.4/lib64:$LIBRARY_PATH
```

```
export C_INCLUDE_PATH=${ThirdParty}/linux64/gcc-6.4.0/include:${ThirdParty}/linux64/gmp-6.2.1/include:${ThirdParty}/linux64Gcc64/openmpi-1.10.4/include:$C_INCLUDE_PATH
```

```
export CPLUS_INCLUDE_PATH=${ThirdParty}/linux64/gcc-6.4.0/include:${ThirdParty}/linux64/gmp-6.2.1/include:${ThirdParty}/linux64Gcc64/openmpi-1.10.4/include:$CPLUS_INCLUDE_PATH
```

Modifying the Source Code of an Existing OpenFOAM Application

If a user decides to modify and existing OpenFoam application source code for his/her needs then he/she has to do the followings:

1. Under user home directory create a directory, e.g. myMesh, and move under this directory.
2. Copy the whole corresponding OpenFoam application directory or part of it (it depends on user needs) under user's home directory (e.g. myMesh).
   ```
   $ cp -r /share/apps/OpenFOAM/OpenFOAM-v2012/application $HOME/myMesh/application
   ```
3. We move into the source code directory of the application under our application directory i.e. at the directory where the .c and .cpp files of the application are located.
4. In the same directory there is a Make directory where there are two scripts. The user should modify the script with name **files** and change the rename the variable FOAM_APPBIN to **FOAM_USER_APPBIN.** Then save the changes.
5. Then return to the directory where .c and .cpp code exists and apply the command,
   ```
   $ wmake
   ```
6. The user can clear the dependencies by applying the command
   ```
   $ wclean
   ```

MRunning an Application

There are two ways to run an application. The first is on a single processor and the second with multiple processors in parallel. We give both ways by example. In both situations the user should copy the tutorial directory under OpenFOAM-v2012 in his/her home directory

```
$ cp -r /share/apps/OpenFOAM/OpenFOAM-v2012/tutorial $HOME/myMesh
```

```
$cd $HOME/myMesh/Tutorial
```

Single Processor

We will use the example of the incompressible/icoFoam/elbow. We copy this directory and work on it.

```
$cd ./incompressible/icoFoam

$mkdir elbow_test

$ cp -r ./elbow ./elbow_test

$ cd elbow_test
```

In general we run a system script like fluentMeshToFoam with argument the application's setup script in order to setup the environmental variables of the application or blockMesh without an argument.

Now under the directory named elbow_test the directory with the name constant and its subdirectory with the name polyMesh that contains the products of this invocation have been created.

Now we are ready to start the simulation. We have to call the script that is named with the top directory name. For example, the elbow_test is under the application icoFoam. So, we have to run the following script,

myOpenFOAMSingleProcessorSimulationJob.sh – math8Queue

```
#!/bin/bash

#$ -S /bin/bash

#$ -N MyOpenFoamProject

#$ -q math8Queue.q

#$ -pe orte 1

#$ -cwd

#$ -V

#$ -o res.o.$JOB_ID

#$ -e err.e.$JOB_ID

#$ -j y

fluentMeshToFoam elbow.msh

icoFoam
```

myOpenFOAMSingleProcessorSimulationJob.sh – math12Queue

```
#!/bin/bash

#$ -S /bin/bash

#$ -N MyOpenFoamProject
#$ -q math12Queue.q

#$ -pe orte 1

#$ -cwd

#$ -V

#$ -o res.o.$JOB_ID

#$ -e err.e.$JOB_ID

#$ -j y

fluentMeshToFoam elbow.msh

icoFoam
```

```
$qusb myOpenFOAMSingleProcessorSimulationJob.sh
```

At the end of the simulation, the results are saved in the elbow_test directory. The user can add any option he/she wants after or before the icoFoam invocation according to his/her needs.

Parallel

Now we examine an application that exploits an inherent parallelism of a problem. We will use the example of depthCharge2D. It is very important to make sure that under the system directory there is the decomposeParDir. In case it does not exist, we copy the decomposeParDir of another application and we modify it appropriately. Also, it is important in the root directory of the application, i.e. under depthCharge2D, the following directories:

- 0
- Constant
- System

should appear. If any of these directories has a different name we should copy it to a new directory with the above right names. Then, we create the polyMesh directory by polyMesh.

Then we set the field with the setFields.

Now, we are ready to decompose the problem into pieces one per processor. We have to check the decomposeParDir script with application parameters and to modify it appropriately. For that we need the following command, decomposePar.

We can see that there are 4 new directories, one for each processor. The number of processors has been determined in decomposeParDir. Inside those directories there are two directories with names 0 and constant.

The next step it is optional. For our example is to invoke the foamToVTK under any of the 4 directories previously created so as the VTK directory to be created.

```
$ cd processor0

$ foamToVTK

$ cd ../processor1

$ foamToVTK

$ cd ../processor2

$ foamToVTK

$ cd ../processor3

$ foamToVTK
```

Now it is the time to invoke the mpirun to start the simulation in a parallel environment. We are going to export the results in a text. We must not to forget to add the parallel flag otherwise the process will not be parallel.

We can our simulation by using the SGE scheduler. The submission script can be as follows,

myOpenFOAMSimulationJob.sh – math8Queue

```
#!/bin/bash

#$ -S /bin/bash

#$ -N MyOpenFoamProject

#$ -q math8Queue.q

#$ -pe orte 4

#$ -cwd

#$ -V

#$ -o res.o.$JOB_ID

#$ -e err.e.$JOB_ID

#$ -j y

polyMesh

setFields

decomposePar

mpirun --mca btl vader,self -np 4 compressibleInterFoam -parallel
```

myOpenFOAMSimulationJob.sh – math12Queue

```
#!/bin/bash

#$ -S /bin/bash

#$ -N MyOpenFoamProject

#$ -q math12Queue.q

#$ -pe orte 4

#$ -cwd

#$ -V

#$ -o res.o.$JOB_ID

#$ -e err.e.$JOB_ID

#$ -j y

polyMesh

setFields

decomposePar

mpirun --mca btl vader,self -np 4 compressibleInterFoam -parallel
```

Assuming that we are under the directory $HOME/myMesh/…/Tutorial/…/compressibleInterFoam /laminar/depthCharge2D we run

```
$qsub myOpenFOAMSimulationJob.sh
```

The user can add any option he/she wants after or before the mpirun invocation according to his/her needs.