Tutorial: BwUniCluster 3.0/HoreKa
Starting Parallel Computations with OpenFOAM

In this tutorial we will learn how to submit a job (using a script-file) for execution of parallel computation on the BwUniCluster 3.0 or HoreKa. Parallel simulations can be used to speed up the process of simulation by a significant margin. However, they have to be executed according to the guidelines below in order to ensure that the simulation runs correctly. Also, special preprocessing is necessary for OpenFOAM in case of parallel simulations.

First of all, it is important to understand what the logical structure of our clusters looks like.

1. Logical structure of the supercomputers

The bwUniCluster 3.0 or HoreKa are parallel computers which are divided into logical units called nodes. Each of those nodes consists of multiple cores with memory attached to them (each core can be thought as a single PC-processor). All components of the whole logical structure can communicate with each other via a communication protocol called MPI, which allows for parallel computing. MPI (Message Passing Interface) is necessary for the data exchange between the processors – both between the cores within a node or between the cores of multiple nodes, see the Figure. This MPI-based communication concept corresponds to the numerical concept of domain decomposition which in turn requires regular exchange of information at the boundaries of the numerical blocks.

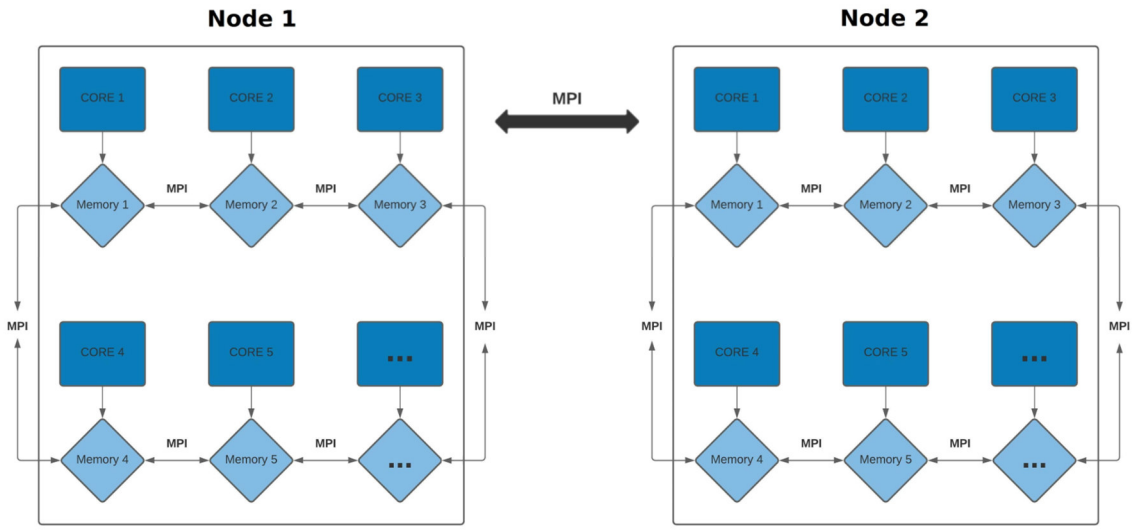

Fig. 1 The logical structure of the cluster

When submitting a job, the user should specify the name of the queue, the number of cores and the number of nodes for the particular simulation. There are different queues for testing, for using only one node, or using two or more nodes. The job submission is carried out by one single command followed by the name of a job-submit-file that describes the desired number of cores and nodes, the memory, the job-queue and the name of the OpenFOAM-solver. But before submitting the job, a special preprocessing of OpenFOAM needs to be done.

2. Preprocessing of OpenFOAM for parallel computations

First, we need to decide on how many parallel cores we would like to start the simulation. For starters, it is always a good idea to take a small number of cores – in our case four. We are going to prepare OpenFOAM for this exact number of cores.

Log onto the server and go to your workspace. Then go to the directory where you wish to start OpenFOAM, i.e. – to the directory with the case you will simulate.

In the 'system' subdirectory you can (usually) find the decomposeParDict file where you can set up the way subdomains are created; if this file is not available, you can take it from an appropriate tutorial case. Notice the number of subdomains which is equal to the product of the coordinates of the partition vector (in this example: for the (5 4 2) partition this number is equal to 5x4x2 = 40). This value will be needed in the creation of the job file later in the tutorial.

A typical decomposeParDict file usually looks like this:

```
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 40;      <-- the mesh will be divided into 40 subdomains
method  simple;             <-- for this method we also specify the number of splits in each direction
coeffs;                     <-- some methods (like "scotch") do not require coefficients
{
    n       (5 4 2);        <-- 40 subdomains (splits) are created in the y-direction
}

//*********************************************************************************//
```

After you have configured the file to your liking execute the following commands in your working directory:

```
$>  blockMesh
$>  setFields
$>  decomposePar
```

3. Creating the job-file

After that, log onto the cluster in your command-line interpreter and go to your working directory in your workspace. Create an empty job file using the following command:

```
$>  touch {name of the the job-file}.sh
```

(For example: *$> touch job1_development_queue_2_cores.sh*)

Open the job-file in order to edit it. A typical job-submit-file takes the following form:

```
////////////////////////////////////// Example of a job-submit-file ////////////////////////////////////////////

#!/bin/bash                          <-- The header of the file

#SBATCH --partition dev_cpu_il       <-- the job will be submitted to the queue dev_cpu_il
#SBATCH –nodes=2                     <-- two nodes will be used
#SBATCH --ntasks-per-node=20         <-- the number of cores to be used on each node
#SBATCH –time=00:30:00               <-- the job will run for the maximum of 30 minutes
#SBATCH –mem=8000mb                  <-- the job may use the max. of 8 Gb
#SBATCH –job-name=coursejob          <-- the name of the job given by the user

module purge
module load cae/openfoam/v2206       } loading OpenFOAM on the execute core
source $FOAM_INIT

mpirun -n 40 compressibleInterFoam -parallel   <-- the name of the solver

////////////////////////////////////////////////////////////////////////////////////////////
```

Note!: On HoreKa use the queue “dev_cpuonly” instead of “dev_cpu_il”.

Note!: The number of cores in the job-submit-file should match the number of subdomains in the decomposeParDict-file (see above).

A table that describes the available queues and their parameters on the bwUniCluster is given here:

https://wiki.bwhpc.de/e/BwUniCluster3.0/Running_Jobs#Batch_Jobs:_sbatch

For HoreKa the information about the options of the queue is given here:

https://www.nhr.kit.edu/userdocs/horeka/batch/

4. Submitting the job (commands of the SLURM job scheduler)

We recommend always to use a job-submit-file (like the above example) that describes the parameters of the queue. This is consistent with the work algorithm in the tutorial “hot radiation” where we provide examples of a job-submit-file. After editing and saving the job-submit-file we can submit the job using the following command from our working directory:

$> sbatch {name of the job-submit-file}.sh

(For example: *$> sbatch job1_development_queue_10_cores.sh*)

which, for example, sends our job on the development queue for single nodes and then causes all of the commands in the job-submit-file to be executed step by step. The extension ".sh" can be omitted. In order to see the list of all your pending jobs you can use the command:

$> squeue --start

Information about all jobs (pending and running) can be obtained by:

$> squeue -l

We can also see a very detailed information about the state of our job using the following command:

$> scontrol show job

After our job has been completed, we can use the following OpenFOAM command to combine results from all subdomains (all processor* - directories) into one directory:

$> reconstructPar

The results of our simulation can be downloaded from our working directory onto our own PC where we can then perform the visualization in ParaView on our own PC. This is the recommended way for visualization of small and medium sized grids (up to 30 or 50 Million grid points, depending on your local RAM capacity).