se

About this wiki

- **The following best practices document aims to provide some hints and examples on how to install and configure OpenFOAM-2.0.1 released 04/08/11 on a computing grid cluster based on EMI I middleware .**
- For the installation and the configuration of OpenFOAM only third-party software have been used.
- This document is a resource for system administrators responsible for installing and configure this open source CDF Software in a distributed computing infrastructure.
- The software included in this wiki are licensed under the Apache License, Version 2.0, is distributed on an "AS IS" BASIS, WITHOUT WARRANTIES OR CONDITIONS OF ANY KIND, either express or implied.

Application Description

OpenFOAM (Open Field Operation and Manipulation) CFD Toolbox is a free, open source CFD software package produced by a commercial company, OpenCFD Ltd.
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 solid dynamics and electro-magnetics. This wiki provides information on how to build OpenFOAM in a grid-based infrastructure. Here are provided two examples of configuration: the first using an mpi embedded version, the second a system version of mpi library.

System requirements

The instructions from this best practice have been tested on a grid computing cluster having the following configuration:

- Processor: Dual Core AMD Opteron(tm) Processor 275;
- Memory: 4147300k;
- OS: Scientific Linux SL release 5.8 (Boron);
- Compiler: GNU 4.4.0;
- MPI: Open MPI 1.4.2;
- Batch System: LSF (ver. 7.0);
- Middleware: EMI I;
- Arch: x86_64.

Downloading source

Source Pack **Source Pack**

Source Pack

The following tar-zipped gtgz source packs are available for download.

Pack	File	md5sum
OpenFOAM	OpenFOAM-2.0.1.gtgz	0a9dfa42282a5e629523b70e2544c773
Third-Party	ThirdParty-2.0.1.gtgz	4b91af77bdbd3a87d91eeccb0f596f59

Unpacking the source and installation

In a grid-based infrastructure we would recommend to install the software in the experiment software directory for the VO using sgm privileges and share this software
packages with other VO members. In this wiki we chosen to unpack this software under the **$VO_GRIDIT_SW_DIR** directory of the grid CE.

```
$ cd $VO_GRIDIT_SW_DIR
$ tar xzf OpenFOAM-2.0.1.gtgz
$ tar xzf ThirdParty-2.0.1.gtgz
$ chown -R root.root OpenFOAM-2.0.1/ ThirdParty-2.0.1/
```

The files unpack to produce directories OpenFOAM-2.0.1 and ThirdParty-2.0.1

```
$ ll OpenFOAM
total 12
drwxr-xr-x  11 sgmgridit001 sgmgridit   440 Nov 28 16:17 OpenFOAM-2.0.1
drwxr-xr-x  12 sgmgridit001 sgmgridit   768 Nov 28 15:57 ThirdParty-2.0.1
```

Before to start

System Requirements **System Requirements**

System Requirements

OpenFOAM-2.0.1 builds on many Linux distributions but the ParaView-3.10.1 version supplied in ThirdParty requires:

- cmake-2.8.6 or higher
- Qt-4.6.4

Both software packages can be freely downloaded for many Linux distributions.

Before to start building software, please check if the following packages have been installed on the server.

```
$ yum install bison
$ yum install ncurses-devel
$ yum install tix.x86_64
$ yum install glibc-devel
$ yum install flex
$ yum install flex-devel
$ yum install zlib-devel
$ yum install libXt-devel
$ yum install binutils-devel
$ yum install gmp.x86_64
$ yum install gmp-devel.x86_64
$ yum install libstdc++44-devel.x86_64
```

Make sure you have GCC 4.4 installed on your server.

```
$ wget ftp://ftp.ntua.gr/pub/linux/scientificlinux/55/x86_64/SL/gcc44-4.4.0-6.el5.x86_64.rpm
$ rpm -ivh gcc44-4.4.0-6.el5.x86_64.rpm

$ wget ftp://ftp.ntua.gr/pub/linux/scientificlinux/55/x86_64/SL/gcc44-c++-4.4.0-6.el5.x86_64.rpm
$ rpm -ivh gcc44-c++-4.4.0-6.el5.x86_64.rpm
```

Qt-4.6.4 installation

Building Qt-4.6.4 **Building Qt-4.6.4**

Building Qt-4.6.4

First uncompress the archive in the preferred location, then unpack it:

```
$ cd /opt/exp_soft/gridit/OpenFOAM/ThirdParty-2.0.1
$ wget ftp://ftp.trolltech.com/qt/source/qt-everywhere-opensource-src-4.6.4.tar.gz
$ tar zxf qt-everywhere-opensource-src-4.6.4.tar.gz
$ chown -R root.root qt-everywhere-opensource-src-4.6.4
$ cd qt-everywhere-opensource-src-4.6.4
$ ./configure
$ gmake && gmake install
```

cmake installation

Building cmake-2.8.6 **Building cmake-2.8.6**

Building cmake-2.8.6

```
$ cd /opt/exp_soft/gridit/OpenFOAM/ThirdParty-2.0.1
$ wget  http://www.cmake.org/files/v2.8/cmake-2.8.6.tar.gz
$ tar zxvf cmake-2.8.6.tar.gz
$ cd cmake-2.8.6
$ ./configure
$ gmake
```

Configuring the Makefile

Makefile changes for GNU compilers **Makefile changes for GNU compilers**

Makefile changes for GNU compilers

Once you have successfully installed GCC 4.4.0, then changes these files:

```
$ cd /opt/exp_soft/gridit/OpenFOAM/OpenFOAM-2.0.1/wmake/rules/linux64Gcc44/

$ cat wmake/rules/linux64Gcc44/c
.SUFFIXES: .c .h
cWARN        = -Wall
cc          = gcc44 -m64   <== Change Here!
include $(RULES)/c$(WM_COMPILE_OPTION)
cFLAGS      = $(GFLAGS) $(cWARN) $(cOPT) $(cDBUG) $(LIB_HEADER_DIRS) -fPIC
ctoo        = $(WM_SCHEDULER) $(cc) $(cFLAGS) -c $$SOURCE -o $@
LINK_LIBS   = $(cDBUG)
LINKLIBSO   = $(cc) -shared
LINKEXE     = $(cc) -Xlinker --add-needed -Xlinker -z -Xlinker nodefs


$ cat wmake/rules/linux64Gcc44/c++
.SUFFIXES: .C .cxx .cc .cpp
c++WARN     = -Wall -Wextra -Wno-unused-parameter -Wold-style-cast
CC          = g++44 -m64   <== Change Here!
include $(RULES)/c++$(WM_COMPILE_OPTION)
ptFLAGS     = -DNoRepository -ftemplate-depth-100
c++FLAGS    = $(GFLAGS) $(c++WARN) $(c++OPT) $(c++DBUG) $(ptFLAGS) $(LIB_HEADER_DIRS) -fPIC
Ctoo        = $(WM_SCHEDULER) $(CC) $(c++FLAGS) -c $$SOURCE -o $@
```

```
cxxtoo      = $(Ctoo)
cctoo       = $(Ctoo)
cpptoo      = $(Ctoo)
LINK_LIBS   = $(c++DBUG)
LINKLIBSO   = $(CC) $(c++FLAGS) -shared
LINKEXE     = $(CC) $(c++FLAGS) -Xlinker --add-needed
```

Pre-setup

Pre-setup requirements **Pre-setup requirements**

Pre-setup requirements

```
$ cd /opt/exp_soft/gridit/OpenFOAM/OpenFOAM-2.0.1/etc
```

Configure the OpenFOAM **bashrc** file as follows:

- # Location of the OpenFOAM installation

  ```
  foamInstall=/opt/exp_soft/gridit/$WM_PROJECT
  ```

- #- Compiler: # WM_COMPILER = Gcc | Gcc43 | Gcc44 | Gcc45 | Gcc46 | Clang | Icc (Intel icc)

  ```
  export WM_COMPILER=Gcc44
  ```

- # Location of installation

  ```
  export WM_PROJECT_INST_DIR=$FOAM_INST_DIR
  ```

  ```
  export WM_PROJECT_DIR=$WM_PROJECT_INST_DIR/$WM_PROJECT-$WM_PROJECT_VERSION
  ```

- # Location of third-party software

  ```
  export WM_THIRD_PARTY_DIR=$WM_PROJECT_INST_DIR/ThirdParty-$WM_PROJECT_VERSION
  ```

- # Location of user file

  ```
  export WM_PROJECT_USER_DIR=$WM_PROJECT_DIR
  ```

Before building...

Checking the System **Checking the System**

Checking the System

To check your system is ready to build the sources, execute the **foamSystemCheck** script (in the OpenFOAM-2.0.1/bin directory).
If any critical software is missing, or needs updating to a newer version, please contact the system administrator to install the required
software before proceeding to the build.

```
$ cd /opt/exp_soft/gridit/OpenFOAM/OpenFOAM-2.0.1/bin
$ ./foamSystemCheck

Checking basic system...
------------------------------------------------------------------------
```

```
Shell:           /bin/bash
Host:            grid012.ct.infn.it
OS:              Linux version 2.6.9-89.0.11.ELsmp
User:            root

System check: PASS
==================
Continue OpenFOAM installation.
```

Start building...

Building Sources **Building Sources**

Building Sources

Go to the top-level source directory $WM_PROJECT_DIR and execute the top-level build script **./Allwmake**

In principle this will build everything, but if problems occur with the build order it may be necessary to update the environment variables and re-execute.

If you experience difficulties with building the source-pack, or your platform is not currently supported, please contact software support to arrange a
support contract and we will do the port and maintain it for future releases.

```
$ cd /opt/exp_soft/gridit/OpenFOAM/OpenFOAM-2.0.1/
$ source  etc/bashrc
$ ./Allwmake >log 2>log &
```

Compiling Paraview 3.10.1 and the PV3FoamReader Module

Paraview is the third-party software that we provide for graphical post-processing in OpenFOAM. It s compilation is automated using a script called **makeParaView** in the ThirdParty-2.0.1 directory. Installation of Paraview 3.10.1 requires a version of QT that is 3.6.2 or newer and cmake which is 2.8.2 or newer, so again make
sure that this is on your system.

```
$ cd $WM_THIRD_PARTY_DIR
```

Configure the *QMAKE_PATH* and *CMAKE_PATH* variable settings in the **makeParaView** file

- # Set the path to the Qt-4.5 (or later) qmake if the system Qt is older

  ```
  QMAKE_PATH="/opt/exp_soft/gridit/OpenFOAM/ThirdParty-2.0.1/qt-everywhere-opensource-src-4.
  ```

- # Set the path to cmake

  ```
  CMAKE_PATH="/opt/exp_soft/gridit/OpenFOAM/ThirdParty-2.0.1/cmake-2.8.6/bin/"
  ```

To install Paraview, execute the following:

```
$ ./makeParaView
$ ./makeParaView >log_paraview 2>log_paraview &
```

The PV3blockMeshReader and the PV3FoamReader ParaView plugins are compiled as usual for OpenFOAM

utilities:

```
$ cd $FOAM_UTILITIES/postProcessing/graphics/PV3Readers
$ wmSET
$ ./Allwclean
$. /Allwmake
```

Testing OpenFOAM

Testing the Installation **Testing the Installation**

Testing the Installation

To check your installation setup, execute the **foamInstallationTest** script (in the OpenFOAM-2.0.1/bin directory).
If no problems are reported, proceed to getting started with OpenFOAM; otherwise, go back and check you have installed the software correctly
and/or contact your system administrator.

Getting Started

Create a project directory within the $HOME/OpenFOAM directory named -2.0.1 and create a directory named run within it, e.g. by typing:

```
$ mkdir -p $HOME/OpenFOAM/OpenFOAM-2.0.1/$FOAM_RUN
```

Copy the tutorial examples directory in the OpenFOAM distribution to the run directory. If the OpenFOAM environment variables are set correctly,
then the following command will be correct:

```
$ cp -r $FOAM_TUTORIALS $HOME/OpenFOAM/OpenFOAM-2.0.1/$FOAM_RUN
```

Run the first example case of incompressible laminar flow in a cavity:

```
$ cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavity
$ blockMesh
$ icoFoam
$ paraFoam
```

Running OpenFOAM in Parallel

Create a new use case

The results from the previous example are generated using a fairly coarse mesh. In this new case we will demonstrate the parallel processing capability of OpenFOAM access to multiple processors.

```
$ cd $FOAM_RUN/tutorials/incompressible/icoFoam/cavity/system
```

```
$ cat decomposeParDict
/*--------------------------------*- C++ -*----------------------------------*\
| =========                 |                                                 |
| \\      /  F ield         | OpenFOAM: The Open Source CFD Toolbox           |
|  \\    /   O peration     | Version:  1.6                                   |
|   \\  /    A nd           | Web:      www.OpenFOAM.org                      |
|    \\/     M anipulation  |                                                 |
\*---------------------------------------------------------------------------*/
```

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      decomposeParDict;
}
// * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * * //

numberOfSubdomains 4; <== Change Here!
method          scotch;
simpleCoeffs
{
    n               ( 2 2 1 );
    delta           0.001;
}
hierarchicalCoeffs
{
    n               ( 1 1 1 );
    delta           0.001;
    order           xyz;
}
scotchCoeffs
{
    processorWeights (
                        1 <== Add Here!
                        1 <== Add Here!
                        1 <== Add Here!
                        1 <== Add Here!
                     );
}
manualCoeffs
{
    dataFile        "";
}
distributed     no;
roots           ( );
// ************************************************************************* //
```

Create a new tar of the test case:

```
$ tar zcvf cavity.tar.gz cavity
cavity/
cavity/system/
cavity/system/fvSchemes
cavity/system/fvSolution
cavity/system/controlDict
cavity/system/decomposeParDict
cavity/constant/
cavity/constant/transportProperties
cavity/constant/polyMesh/
cavity/constant/polyMesh/boundary
cavity/constant/polyMesh/blockMeshDict
cavity/0/
cavity/0/U
cavity/0/p
```

Testing OpenFOAM in a Grid Infrastructure

This section provides some hints for testing OpenFOAM job on the GRIDIT Infrastructure.

Post-configuration on the LSF master node and client nodes

Add in the */etc/bashrc* file the following settings for sourcing the OpenFoam bashrc profile

```
. /opt/exp_soft/gridit/OpenFOAM/OpenFOAM-2.0.1/etc/bashrc
```

Creation of a new MPI wrapper

Add in */etc/mpi-start/* the definition of a new MPI wrapper for OpenFoam-2.0.1.
This wrapper has to be replicated in all the LSF client nodes.

```
$ cat /etc/mpi-start/openmpi_openfoam.mpi
#!/bin/sh
#
# Copyright (c) 2006-2007 High Performance Computing Center Stuttgart,
#                         University of Stuttgart.  All rights reserved.
#           (c) 2009      Instituto de Fisica de Cantabria - CSIC.
#

# specifies where Open MPI is installed

export MPI_OPENMPI_PATH=/opt/exp_soft/gridit/OpenFOAM/ThirdParty-2.0.1/platforms/linux64Gcc44/ope
export MPI_OPENMPI_MPIEXEC=/opt/exp_soft/gridit/OpenFOAM/ThirdParty-2.0.1/platforms/linux64Gcc44/

if test "x$I2G_OPENMPI_PREFIX" = "x" ; then
    if test "x$MPI_OPENMPI_PATH" = "x" ; then
        if test "x$MPI_START_MPI_PREFIX" != "x"  ; then
            export I2G_OPENMPI_PREFIX=$MPI_START_MPI_PREFIX
        else
            export I2G_OPENMPI_PREFIX=/opt/i2g/openmpi
            debug_msg "use default installtion : $I2G_OPENMPI_PREFIX"
        fi
    else
        export I2G_OPENMPI_PREFIX=$MPI_OPENMPI_PATH
        debug_msg "use user provided prefix : $MPI_OPENMPI_PATH"
    fi
else
    debug_msg "use user provided prefix : $I2G_OPENMPI_PREFIX"
fi

# activate MPI
mpi_start_activate_mpi $I2G_OPENMPI_PREFIX "$MPI_START_MPI_MODULE"

# add necessary PATH to the environment variables
#debug_msg "prepend Open MPI to PATH and LD_LIBRARY_PATH"
export PATH=$I2G_OPENMPI_PREFIX/bin:$PATH
export LD_LIBRARY_PATH=$I2G_OPENMPI_PREFIX/lib:$LD_LIBRARY_PATH

if test "x$I2G_MPI_TYPE" != "xopenmpi" ; then
    # we are not the primary MPI
    # fall back to save settings that should work always
    debug_msg ""
    debug_msg "disable PBS, SGE"
    OPENMPI_PARAMS="-mca pls ^tm,gridengine -mca ras ^tm,gridengine  "
    #OPENMPI_PARAMS="$OPENMPI_PARAMS -x PACX_DEBUG_NODE=$PACX_DEBUG_NODE"
    debug_msg "export GLOBUS_TCP_PORT_RANGE : $GLOBUS_TCP_PORT_RANGE"
    OPENMPI_PARAMS="$OPENMPI_PARAMS -x GLOBUS_TCP_PORT_RANGE "
fi

#
# start an mpi job
```

```
#
mpi_exec () {

    #handle Open MPI 1.2.2 + PBS bug
    if test "x$PBS_NODEFILE" = "x" ; then
        debug_msg "found openmpi and a non-PBS batch system, set machinefile and np parameters"
        export I2G_MACHINEFILE_AND_NP="-machinefile $MPI_START_MACHINEFILE -np $I2G_MPI_NP"
    else
        debug_msg "found openmpi and PBS, don't set machinefile"
        export I2G_MACHINEFILE_AND_NP="-np $I2G_MPI_NP"
    fi

    #set the parameters to be always used with Open MPI:
    MPI_SPECIFIC_PARAMS="-wdir $PWD "

    #check for Marmot
    if test "x$I2G_USE_MARMOT" = "x1" ; then
        debug_msg "export LD_PRELOAD for Open MPI"
        MPI_SPECIFIC_PARAMS="$MPI_SPECIFIC_PARAMS -x LD_PRELOAD -x MARMOT_MAX_TIMEOUT_DEADLOCK -x
    fi

    #if test "x$I2G_USE_MPITRACE" ="x1" ; then
        #MPI_SPECIFIC_PARAMS="-x MPITRACE_ON -x MPTRACE_DIR"
    #fi

    # check for user supplied mpiexec
    MPIEXEC=`which mpiexec`
    if test "x$MPI_OPENMPI_MPIEXEC" != "x"  ; then
        MPIEXEC="$MPI_OPENMPI_MPIEXEC $I2G_MPI_MPIEXEC_PARAMS"
        debug_msg "using user supplied startup : '$MPIEXEC'"
        MPI_SPECIFIC_PARAMS="$MPI_SPECIFIC_PARAMS -x X509_USER_PROXY --prefix $I2G_OPENMPI_PREFIX
        . $MPI_START_PREFIX/../etc/mpi-start/generic_mpiexec.sh
        generic_mpiexec
    elif test "x$MPI_OPENMPI_MPIRUN" != "x" ; then
        MPIEXEC="$MPI_OPENMPI_MPIRUN $MPI_OPENMPI_MPIRUN_PARAMS"
        debug_msg "using user supplied startup : '$MPIEXEC'"
        MPI_SPECIFIC_PARAMS="$MPI_SPECIFIC_PARAMS -x X509_USER_PROXY --prefix $I2G_OPENMPI_PREFIX
        . $MPI_START_PREFIX/../etc/mpi-start/generic_mpiexec.sh
        generic_mpiexec
    else
        MPI_SPECIFIC_PARAMS="$MPI_SPECIFIC_PARAMS -x X509_USER_PROXY --prefix $I2G_OPENMPI_PREFIX
        . $MPI_START_PREFIX/../etc/mpi-start/generic_mpiexec.sh
        generic_mpiexec
    fi
    return $?
}

mpi_start () {
    . $MPI_START_PREFIX/../etc/mpi-start/generic_mpi_start.sh
    generic_mpi_start
    return $?
}
```

JDL & script files (sequential mode)

This is an example of JDL file that can be used for testing OpenFoam in **sequential mode**:

```
[
     Type = "Job";
     JobType = "Normal";

     Executable = "/bin/bash";
     Arguments = "start_openfoam.sh";

     StdOutput = "log.out";
```

```
     StdError = "log.err";

     InputSandbox = {"start_openfoam.sh"};
     OutputSandbox = {"log.err","log.out","openfoam.log","openfoam.err"};
     Requirements = Member("VO-gridit-OpenFoam-2.0.1",other.GlueHostApplicationSoftwareRunTimeEnv
     Rank = (other.GlueCEStateWaitingJobs == 0 ? other.GlueCEStateFreeCPUs : -other.GlueCEStateWa
]
```

This is the bash script sent in InputSandbox with the JDL file:

```
#!/bin/sh
echo "+ Running OpenFoam-2.0.1 on "`hostname -f` as `whoami`
echo;echo "+ Copying the OpenFoam example..."
cp -R $FOAM_TUTORIALS/incompressible/icoFoam/cavity $PWD
cd $PWD/cavity
echo; echo "+ Starting at "`date`

blockMesh
icoFoam >./openfoam.log 2>./openfoam.err
cp openfoam.* ../

# Testing the scratch area
cp openfoam.* $VO_GRIDIT_SW_DIR/scratch
chmod a+w $VO_GRIDIT_SW_DIR/scratch/openfoam.*
echo "+ Done at "`date`
```

JDL & script files (parallel mode)

This is an example of JDL file that can be used for testing OpenFoam in **parallel mode**:

```
[
JobType        = "NORMAL";
CPUNumber      = 4;
Executable     = "mpi-start-wrapper.sh";
Arguments      = "icoFoam OPENMPI_OPENFOAM";
StdOutput      = "mpi-start.out";
StdError       = "mpi-start.err";
InputSandbox   = {"mpi-start-wrapper.sh","bei-hooks.sh"};
OutputSandbox  = {"mpi-start.err","mpi-start.out","results.tgz"};
Environment    = {"I2G_MPI_PRE_RUN_HOOK=./bei-hooks.sh","I2G_MPI_POST_RUN_HOOK=./bei-hooks.sh"};
Requirements   = Member("VO-gridit-OpenFoam-2.0.1",other.GlueHostApplicationSoftwareRunTimeEnviro
]
```

MPI wrapper and hooks are shown below:

```
$ cat mpi-start-wrapper.sh
#!/bin/bash

# Pull in the arguments.
MY_EXECUTABLE=$1
MPI_FLAVOR=$2

# Convert flavor to lowercase for passing to mpi-start.
MPI_FLAVOR_LOWER=`echo $MPI_FLAVOR | tr '[:upper:]' '[:lower:]'`

# Pull out the correct paths for the requested flavor.
eval MPI_PATH=`printenv MPI_${MPI_FLAVOR}_PATH`

# Ensure the prefix is correctly set.  Don't rely on the defaults.
#eval I2G_${MPI_FLAVOR}_PREFIX=$MPI_PATH
#export I2G_${MPI_FLAVOR}_PREFIX

# Touch the executable.  It exist must for the shared file system check.
# If it does not, then mpi-start may try to distribute the executable
```

```
# when it shouldn't.
touch $MY_EXECUTABLE

# Setup for mpi-start.
export I2G_MPI_APPLICATION=$MY_EXECUTABLE
export I2G_MPI_APPLICATION_ARGS="-parallel"
export I2G_MPI_TYPE=$MPI_FLAVOR_LOWER
export I2G_MPI_PRE_RUN_HOOK=$PWD/bei-hooks.sh
export I2G_MPI_POST_RUN_HOOK=$PWD/bei-hooks.sh
export I2G_MPI_START_HOOKS_LOCAL=bei-hooks.sh
export I2G_MPI_FILE_DIST="ssh"

# If these are set then you will get more debugging information.
export I2G_MPI_START_VERBOSE=1
export I2G_MPI_START_TRACE=1
export I2G_MPI_START_DEBUG=1

# Invoke mpi-start.
$I2G_MPI_START

source $I2G_MPI_START_HOOKS_LOCAL

$ cat bei-hooks.sh
#!/bin/sh
export OUTPUT_DIR=cavity
export OUTPUT_ARCHIVE=results.tgz
export OUTPUT_SE=lfn:/grid/gridit/ebogdan
export OUTPUT_VO=gridit
export STORAGE_HOST=atlasse.lnf.infn.it
export JOBID=$(echo ${GRID_JOBID} | awk '{ l=length($1); s=substr($1,l-21); print s }')
export GLOBUS_TCP_PORT_RANGE=20000,25000
export MPI_START_SHARED_FS=0

pre_run_hook () {

   echo;echo "============================================================="
   echo "- Starting OpenFoam (ver.2.0.1) ~ the Open Source CFD toolbox "
   echo
   echo "OpenFOAM (Open Source Field Operation and Manipulation) is a C++"
   echo "toolbox for the development of customized numerical solvers and"
   echo "pre-/post-processing utilities for the solution of continuum"
   echo "mechanics problems, including computational fluid dynamics (CFD)."
   echo "Official OpenFOAM web site http://www.openfoam.com/"
   echo
   echo "- [Job settings]"
   echo "- JOBID="$JOBID
   echo "- MPI_START_SHARED_FS="$MPI_START_SHARED_FS
   echo "- MPI_SHARED_HOME="$MPI_SHARED_HOME
   echo "- MPI_SHARED_HOME_PATH="$MPI_SHARED_HOME_PATH
   echo "- OUTPUT_DIR="$OUTPUT_DIR
   echo "- OUTPUT_ARCHIVE="$OUTPUT_ARCHIVE
   echo "- OUTPUT_VO="$OUTPUT_VO
   echo "- STORAGE_HOST="$STORAGE_HOST
   echo
   echo "=[PRE_RUN_HOOK] Started"
   echo "- Downloading tar archive from SE..."
   echo lcg-cp --vo $OUTPUT_VO $OUTPUT_SE/cavity.tar.gz file:$PWD/cavity.tar.gz
   lcg-cp --vo $OUTPUT_VO $OUTPUT_SE/cavity.tar.gz file:$PWD/cavity.tar.gz
   tar zxf cavity.tar.gz
   echo;echo "- Summarizing disk usage of each FILE, recursively for directories"
   du -h cavity
   cd $OUTPUT_DIR
   echo;echo "=[blockMesh] ============================================"
   blockMesh
   echo;echo "=[decomposePar] ========================================="
   decomposePar
   echo "=[PRE_RUN_HOOK] Finished"
```

```
   return 0
}

# the first parameter is the name of a host in the
copy_from_remote_node() {

     if [[ $1 == `hostname -a` || $1 == `hostname -f`  || $1 == "localhost" ]]; then
         echo "I skip the local host: " $1
         return 1
     fi

     # copy the results data on master node
     CMD="scp -r $1:\"$PWD\" .."
     echo "- Copying file from remote host [" $1 "]"
     echo $CMD
     $CMD
}

post_run_hook () {
     echo "=[POST_RUN_HOOK] Started"

     if [ "x$MPI_START_SHARED_FS" == "x0" ] ; then
         echo "- Gathering output results from remote hosts [FS=0]"
         mpi_start_foreach_host copy_from_remote_node
     fi

     echo; echo "- Working directory => "$PWD
     echo "- Listing data from the master [ "`hostname -f`" ] node..."
     ls -al $PWD
     echo; echo "- Summarizing disk usage of each FILE, recursively for directories"
     du -h $PWD
     echo; echo "- Listing the contents of directories in a tree-like format"
     tree -L 3
     echo "- Packing the final results data"
     cd ..
     tar czf $OUTPUT_ARCHIVE $OUTPUT_DIR
     if [ $? -eq 0 ] ; then
             echo;echo "- Uploading the data to the grid SE"
             echo lcg-cr --vo $OUTPUT_VO -d $STORAGE_HOST -l $OUTPUT_SE/results_$JOBID.tgz file:$
             lcg-cr --vo $OUTPUT_VO -d $STORAGE_HOST -l $OUTPUT_SE/results_$JOBID.tgz file:$PWD/$
     else echo "=[ERROR ] Some problems occurred during packing of data. Please, check log file f
     fi
     echo "=[POST_RUN_HOOK] Finished"
     return 0
}
```

An alternative installation with external MPI.

References

OpenFOAM: The open source CFD toolbox

Running OpenFOAM in parallel on the grid: CFD in the study of cardiovascular disease

---

This topic: UserSupport > OpenFoam
Topic revision: r12 - 2012-10-22 - VaniaBoccia