Parallel scalability with OpenFOAM

Table of Contents

Access to the system & load the environment

- ✓ To access to the system use the username provided at the registration:

```
ssh -X a08tra<N>@login.marconi.cineca.it
<N> = 01, 02, … , 30
Pwd = rAscmlflP
```

- ✓ You are directly logged in into Marconi A1 Partition, (i.e. Broadwell):
- ✓ To switch to A2 partition, (i.e. KNL)

```
[ispisso0@r000u07l02 ~]$ module load env-knl
(KNL) [ispisso0@r000u07l02 ~]$
```

- ✓ To revert to A1 partition:

```
(KNL) [ispisso0@r000u07l02 ~]$ module load env-bdw
(BDW) [ispisso0@r000u07l02 ~]$
```

Access to the system & load the environment

- ✓ Account (for batch jobs)
  - A1 → train_wcfdap17
  - A2 → train_wcfdap17_0

- ✓ Reservation
  - A1 → reservation R969661
  - A2 → reservation R469121

- ✓ Batch script modification

```
#PBS -A <account_number>
#PBS -q <reservation_number>
#PBS -W group_list=<account_number>
```

Access to the system & load the environment

✓ To load the OpenFoam module, use the module environment

```
(BDW) [ispisso0@r000u07l02 ~]$ modmap -m openfoam
…
Profile: archive
                openfoam
                 2.3.0_knl
                 2.4.0
                openfoam+
                 v1606
Profile: knl
                openfoam+
                 v1612_knl
…
Profile: phys
                openfoam
                 5.0
                openfoam+
                 v1612
                 v1706
                openfoam-ext
                 3.2
```

Access to the system & load the environment

✓ To load the last version of OpenFoam+ in A1-BDW partition

```
(BDW) [user@r000u07l02 ~]$ module load profile/phys autoload openfoam+/v1706
(BDW) [user@r000u07l02 ~]$ module li
Currently Loaded Modulefiles:
 1) profile/base                          5) intel/pe-xe-2017--binary
 2) env-bdw/1.0                           6) intelmpi/2017--binary
 3) profile/phys                          7) fftw/3.3.4--intelmpi--2017--binary
 4) autoload                              8) openfoam+/v1706

(BDW) [user@r000u07l02 ~]$ module show openfoam+/v1706

(BDW) [user@r000u07l02 ~]$ module help openfoam+/v1706

(BDW) [user@r000u07l02 ~]$ echo $FOAM_
$FOAM_APP            $FOAM_ETC            $FOAM_LIBBIN         $FOAM_SETTINGS
$FOAM_SRC            $FOAM_USER_LIBBIN    $FOAM_APPBIN         $FOAM_EXT_LIBBIN
$FOAM_MPI            $FOAM_SIGFPE         $FOAM_TUTORIALS      $FOAM_UTILITIES
$FOAM_CINECA_SCRIPT  $FOAM_INST_DIR       $FOAM_RUN            $FOAM_SOLVERS
$FOAM_USER_APPBIN    $FOAM_VERBOSE
```

Access to the system & load the environment

✓ To load the last version of OpenFoam+ in A2-KNL partition

```
[ispisso0@r000u07l02 ~]$ module  purge
[ispisso0@r000u07l02 ~]$ module  load  profile/knl  env-knl  autoload  openfoam+/v1612_knl
(KNL) [ispisso0@r000u07l02 ~]$ module li
Currently Loaded Modulefiles:
 1) profile/base                         5) intelmpi/2017--binary
 2) profile/knl                          6) boost/1.61.0--intelmpi--2017--binary
 3) autoload                             7) fftw/3.3.4--intelmpi--2017--binary
 4) intel/pe-xe-2017—binary              8) openfoam+/v1612_knl

[ispisso0@r000u07l02 ~]$ echo $WM_COMPILER
IccKNL
[ispisso0@r000u07l02 ~]$ module help openfoam+/v1612_knl
-------------------------------------------------------------------
Module Specific Help for /cineca/prod/opt/modulefiles/knl/applications/openfoam+/v1612_knl:
modulefile "openfoam+/v1612_knl"
module name: openfoam+-v1612_knl
creation/update date: 20170328 16:57:13
>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>>
Comment: Compilation Option "IccKNLOpt" == intelmpi/2017--binary + flag for KNL -axMIC-AVX512 -03
forcross compilation on A1+A2 system,
https://wiki.ugov.it/confluence/display/SCAIUS/UG3.1%3A+MARCONI+UserGuide
to allow pre- and post-process operation on the login nodes interactively
```

Set-up of the test-case: 3-D Lid driven cavity flow

- ✓ The test case is a modified version of the standard lid-driven cavity flow
- ✓ https://www.openfoam.com/documentation/tutorial-guide/tutorialse2.php#x6-60002.1
- ✓ 3-D Lid driven cavity flow with 100^3 # of cells
- ✓ Mesh modification and generation

```
[$ module load autoload profile/phys openfoam+/v1706
[$ cd $FOAM_RUN
cp -r $FOAM_TUTORIALS/incompressible/icoFoam/cavity/ .

<Edit the file system/blockMeshDict>

// * * * * * * * * * *  * * * * * * * * * * * * * * //

convertToMeters 1;

vertices
(
(0 0 0)
(1 0 0)
(1 1 0)
(0 1 0)
(0 0 1)
(1 0 1)
(1 1 1)
(0 1 1)
);
blocks
(
hex (0 1 2 3 4 5 6 7) (100 100 100) simpleGrading (1 1 1)
);
```

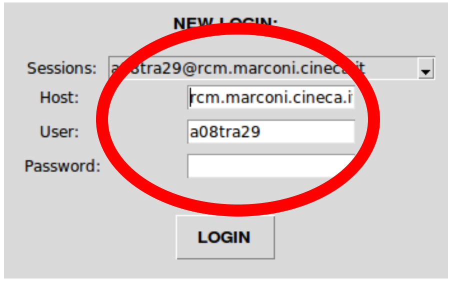
```
<to generate the mesh>
$ blockMesh
…
Writing polyMesh
----------------
Mesh Information
----------------
boundingBox: (0 0 0) (1 1 1)
nPoints: 1030301
nCells: 1000000
nFaces: 3030000
nInternalFaces: 2970000
----------------
Patches
----------------
patch 0 (start: 2970000 size: 10000) name: movingWall
patch 1 (start: 2980000 size: 30000) name: fixedWalls
patch 2 (start: 3010000 size: 20000) name: frontAndBack
End

<Opzional: mesh quality>
checkMesh
```

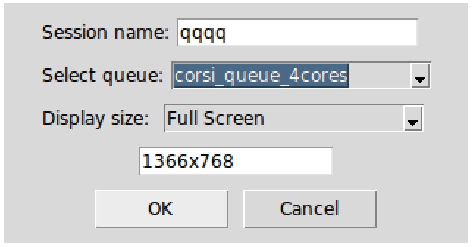
Remote Viz with RCM

- ✓ To visualize the mesh, use the RCM tools provided by CINECA’s staff
- ✓ https://wiki.u-gov.it/confluence/pages/viewpage.action?pageId=68391634
- ✓ open a remote shell on **rcm.marconi.cineca.it**

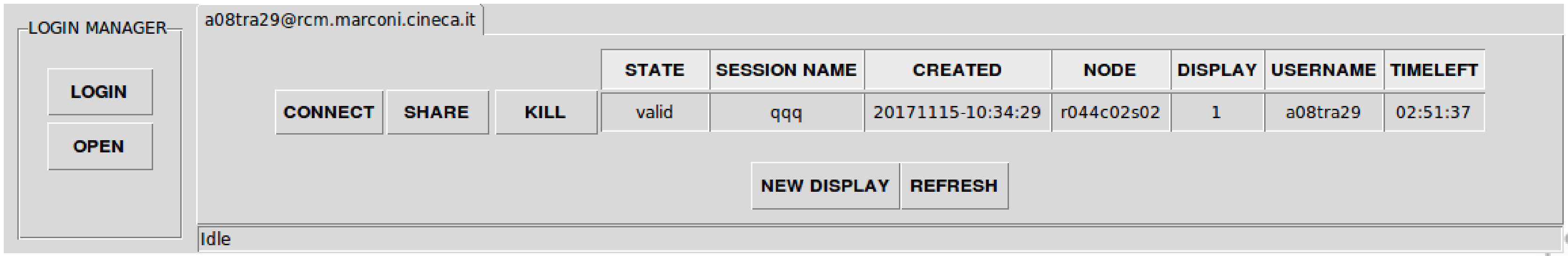
…



Sessions: a08tra29@rcm.marconi.cineca.it
Host: rcm.marconi.cineca.i
User: a08tra29
Password:



Session name: qqqq
Select queue: corsi_queue_4cores
Display size: Full Screen
1366x768

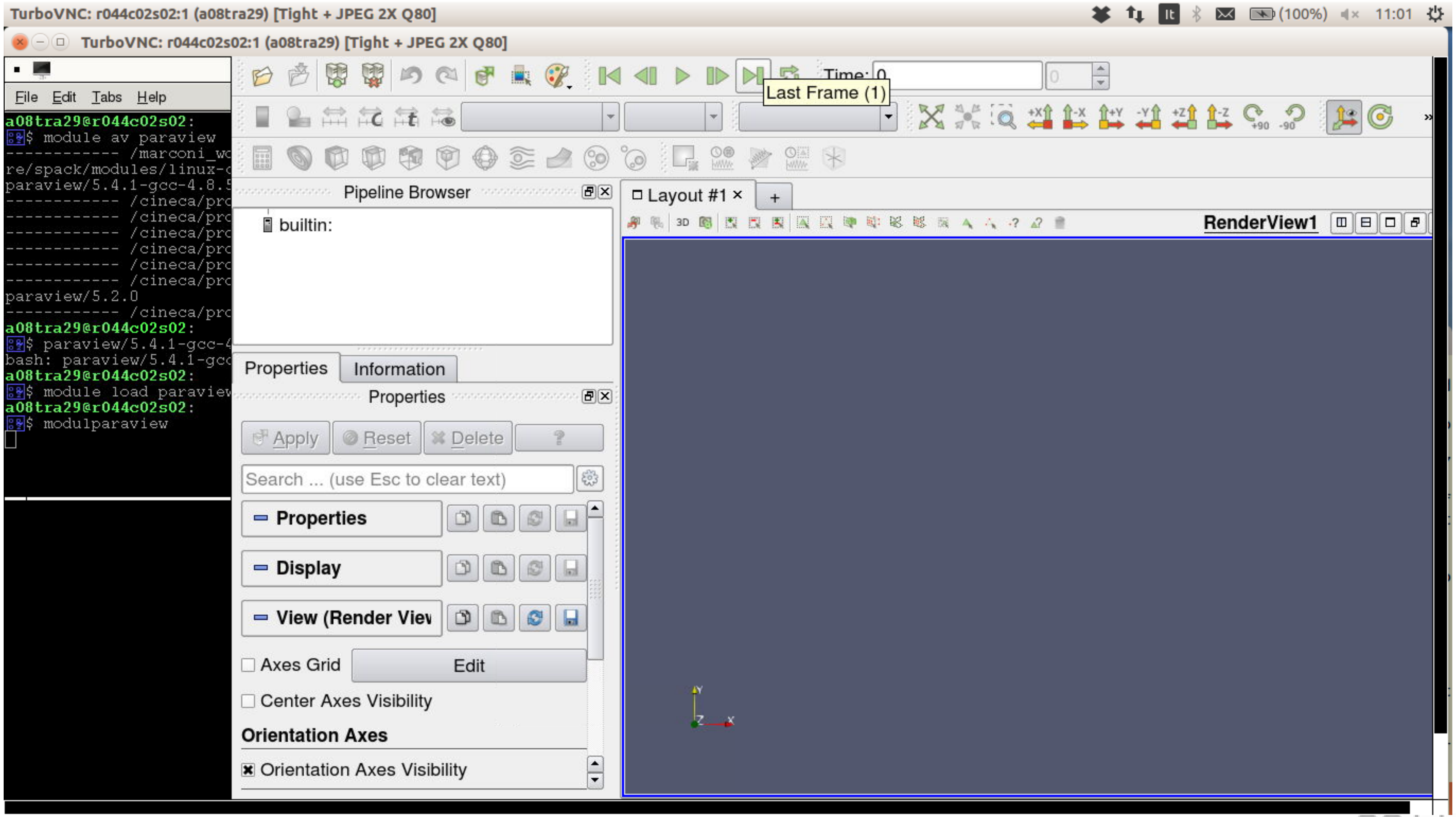


STATE	SESSION NAME	CREATED	NODE	DISPLAY	USERNAME	TIMELEFT
valid	qqq	20171115-10:34:29	r044c02s02	1	a08tra29	02:51:37

Idle

```
~$ module load paraview
~$ paraview
```

Remote Viz with RCM

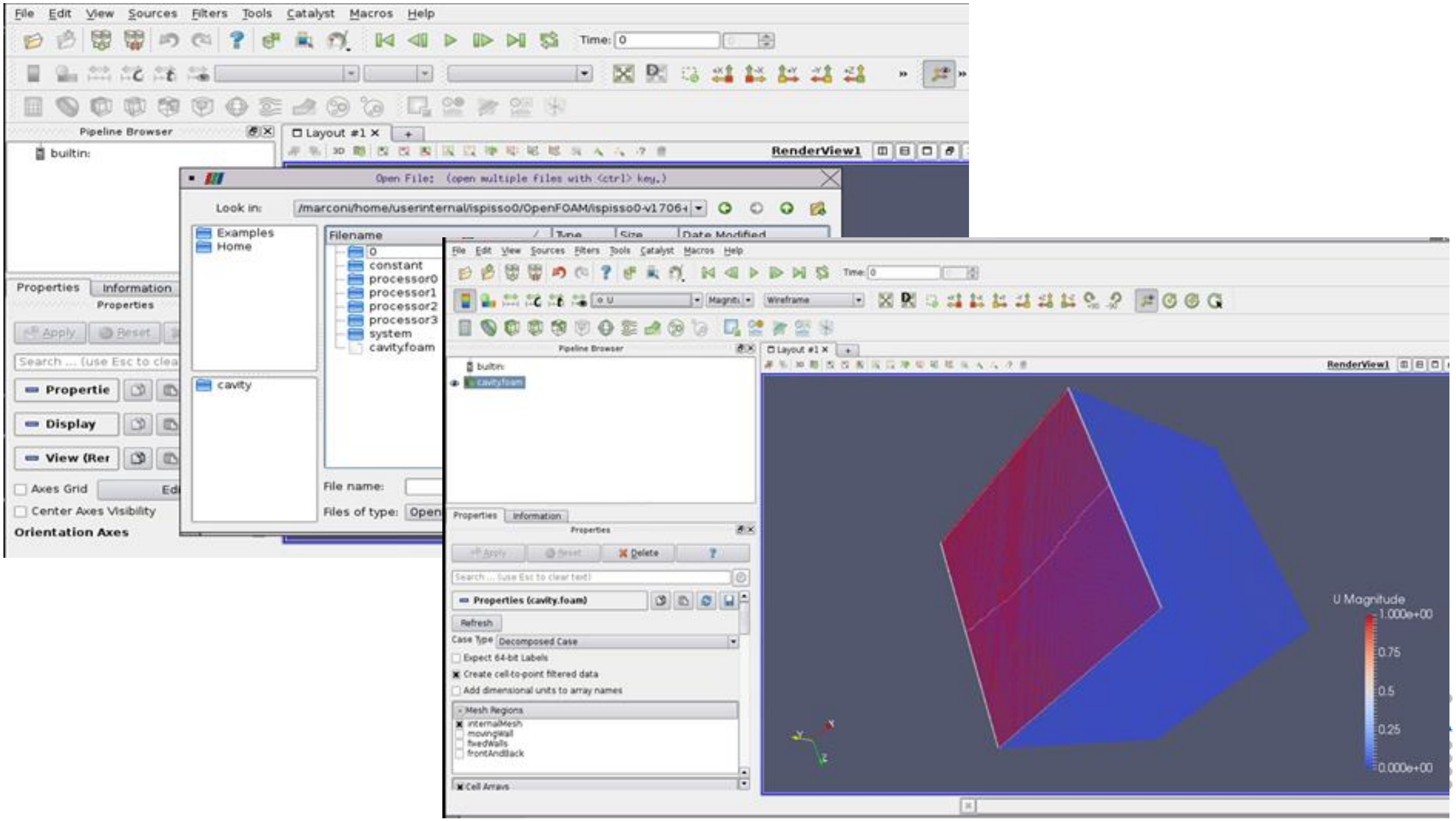


TurboVNC: r044c02s02:1 (a08tra29) [Tight + JPEG 2X Q80]

```
a08tra29@r044c02s02:
~$ module av paraview
------------ /marconi_wo
re/spack/modules/linux-c
paraview/5.4.1-gcc-4.8.5
------------ /cineca/pro
------------ /cineca/pro
------------ /cineca/pro
------------ /cineca/pro
------------ /cineca/pro
------------ /cineca/pro
paraview/5.2.0
------------ /cineca/pro
a08tra29@r044c02s02:
~$ paraview/5.4.1-gcc-4
bash: paraview/5.4.1-gcc
a08tra29@r044c02s02:
~$ module load paraview
a08tra29@r044c02s02:
~$ modulparaview
```

Remote Viz with RCM

- ✓ Create a file .foam in the case dir
- ✓ Open it with foam reader OpenFOAM (*.foam)



Look in: /marconi/home/userinternal/ispisso0/OpenFOAM/ispisso0-v1706

0
constant
processor0
processor1
processor2
processor3
system
cavity.foam

Properties (cavity.foam)

Case Type: Decomposed Case

- [ ] Expect 64-bit Labels
- [x] Create cell-to-point filtered data
- [ ] Add dimensional units to array names

Mesh Regions

- [x] internalMesh
- [ ] movingWall
- [ ] fixedWalls
- [ ] frontAndBack

Run with batch script in serial

- ✓ To run the test-case in serial using the batch script, use the template available in **$FOAM_CINECA_SCRIPT**
- ✓ copy it in your working dir and modify properly
- ✓ set the computational resources with PBS directives

```
$ cp $FOAM_CINECA_SCRIPT/submit.openfoamv1706.pbs  .
$ ls
0  cavity.foam  constant  submit.openfoamv1706.pbs  system
$ cat submit.openfoamv17.06.pbs
#!/bin/bash
#PBS -A <account_number>                          >> insert your account number
#PBS -l walltime=5:00
## on Marconi-A1:
#PBS -l select=2:ncpus=18:mpiprocs=18:mem=118GB
##PBS -l select=1:ncpus=1:mpiprocs=1              >> serial job, node shared
##PBS -l select=1:ncpus=1:mpiprocs=1:mem=118GB    >> serial job, node exclusive
module load profile/phys
module load autoload
module load openfoam+/v1706
..
#mpirun -np $np $solver -parallel > output.$PBS_JOBID >> run in parallel
$solver > output.$PBS_JOBID
```

Run with batch script in serial

- ✓ check **system/controlDict** File
- ✓ 100 iteration = (0.5-0)/0.005
- ✓ write interval 20 iteration
- ✓ submit the job **qsub submit.openfoamv1706.pbs**
- ✓ check log file and time

```
FoamFile
{
    version     2.0;
    format      ascii;
    class       dictionary;
    location    "system";
    object      controlDict;
}
// * * * * * * * * * * * * * *** //
application     icoFoam;
startFrom       startTime;
startTime       0;
stopAt          endTime;
endTime         0.5;
deltaT          0.005;
writeControl    timeStep;
writeInterval   20;
purgeWrite      0;
writeFormat     ascii;
writePrecision  6;
writeCompression off;
timeFormat      general;
timePrecision   6;
runTimeModifiable true;
```

```
drwxr-xr-x 3 ispisso0 interactive  4096 Nov 10 17:29 0.1
drwxr-xr-x 3 ispisso0 interactive  4096 Nov 10 17:33 0.2
drwxr-xr-x 3 ispisso0 interactive  4096 Nov 10 17:36 0.3
-rw-r--r-- 1 ispisso0 interactive   537 Nov 10 17:38 submit.openfoamv1706.pbs
drwxr-xr-x 3 ispisso0 interactive  4096 Nov 10 17:38 0.4
drwxr-xr-x 3 ispisso0 interactive  4096 Nov 10 17:41 0.5
-rw-r--r-- 1 ispisso0 interactive 74143 Nov 10 17:41 output.966786.r000u17l01

[ispisso0@r000u08l03 cavity]$ tailf output.966786.r000u17l01
smoothSolver:  Solving for Ux, Initial residual = 0.00170479, Final residual =
9.79301e-06, No Iterations 5
smoothSolver:  Solving for Uy, Initial residual = 0.00222831, Final residual =
4.687e-06, No Iterations 6
DICPCG:  Solving for p, Initial residual = 0.00162745, Final residual = 8.05425e-
05, No Iterations 75
time step continuity errors : sum local = 9.95287e-09, global = -3.63389e-21,
cumulative = -6.5944e-20
DICPCG:  Solving for p, Initial residual = 0.00123289, Final residual = 9.80305e-
07, No Iterations 222
time step continuity errors : sum local = 1.2066e-10, global = 5.00564e-21,
cumulative = -6.09383e-20
ExecutionTime = 978.83 s  ClockTime = 986 s

End
```

The ExecutionTime is the time spent by the processor and the ClockTime is the wall clock time or "real" time taken from the start of the job to the end

Run with batch script in parallel

✓ First of all, always clean your work directory

```
[ispisso0@r000u08l03 cavity]$ more clean_all.sh
rm -rf 0.*
rm -rf processor*
[ispisso0@r000u08l03 cavity]$ ./clean_all.sh
[ispisso0@r000u08l03 cavity]$
```

Run batch script in parallel: Mesh decomposition

- ✓ To decompose the mesh, use the dictionary **decomposeParDict**.
- ✓ Copy the **decomposeParDict** from the following dir:

```
[ispisso0@r000u07l02 run]$ cp
/cineca/prod/opt/applications/openfoam+/v1706/OpenFOAM-
v1706+/tutorials/multiphase/interFoam/laminar/damBreak/damBreak/system/decomposeP
arDict cavity/system/.

[ispisso0@r000u07l02 run]$ tree -L 2 cavity/
cavity/
├── 0
│   ├── p
│   └── U
├── cavity
│   ├── 0
│   ├── constant
│   └── system
├── constant
│   ├── polyMesh
│   └── transportProperties
└── system
├── controlDict
├── decomposeParDict
├── fvSchemes
└── fvSolution
```

Run batch script in parallel: Mesh decomposition

- ✓ To decompose the mesh, use the dictionary **decomposeParDict**.
- ✓ Modify the dictionary **decomposeParDict**, by selecting the the decomposition method and the optional parameters
- ✓ The user has a choice of four methods of decomposition, specified by keyword
  - **simple,hierarchical,scotch,manual**
- ✓ Type: **decomposePar**

```
[ispisso0@r000u08l03 cavity]$ vim system/decomposeParDict
// * * * * * * * * * * * * ** * * * //
numberOfSubdomains 4;
method          scotch;
simpleCoeffs
{
    n               (2 2 1);
    delta           0.001;
}
hierarchicalCoeffs
{
    n               (1 1 1);
    delta           0.001;
    order           xyz;
}
manualCoeffs
{
    dataFile        "";
}
distributed     no;
```

```
[ispisso0@r000u08l03 cavity]$ tree -L 2
.
├── 0
│   ├── p
│   └── U
├── constant
│   ├── polyMesh
│   └── transportProperties
├── processor0
│   ├── 0
│   └── constant
├── processor1
│   ├── 0
│   └── constant
├── processor2
│   ├── 0
│   └── constant
├── processor3
│   ├── 0
│   └── constant
└── system
    ├── blockMeshDict
    ├── controlDict
    ├── decomposeParDict
    ├── fvSchemes
    └── fvSolution
```

Run batch script in parallel

- ✓ modify your batch script with the adequate computational resources
- ✓ submit the job to the scheduler system
- ✓ check log file and time

```
#!/bin/bash
#PBS -A <account_number> >> insert your account number
#PBS -l walltime=20:00
## on Marconi-A1:
#PBS -l select=2:ncpus=18:mpiprocs=18:mem=118GB
#PBS -l select=1:ncpus=4:mpiprocs=4  >> pure mpi job on 4 cores ncpus=mpiprocs
#PBS -l select=1:ncpus=36:mpiprocs=36:mem=118GB  >> pure mpi job on all available cores in a node exclusive
module load profile/phys
module load autoload
module load openfoam+/v1706
..
mpirun -np $np $solver -parallel > output.$PBS_JOBID >> run in parallel
```

```
[ispisso0@r000u08l03 cavity]$ ls
0                          processor12  processor22  processor32
cavity.foam                processor13  processor23  processor33
System                     processor14  processor24  processor34
constant                   processor15  processor25  processor35
output.966737.r000u17l01   processor16  processor26  processor4
output.966740.r000u17l01   processor17  processor27  processor5
output.966786.r000u17l01   processor18  processor28  processor6
processor0                 processor19  processor29  processor7
processor1                 processor2   processor3   processor8
processor10                processor20  processor30  processor9
processor11                processor21  processor31  submit.openfoamv1706.pbs
[ispisso0@r000u08l03 cavity]$ qsub submit.openfoamv1706.pbs
966854.r000u17l01
[ispisso0@r000u08l03 cavity]$ tailf output.966854.r000u17l01
smoothSolver:  Solving for Uy, Initial residual = 0.00222858, Final residual = 9.42206e-06, No Iterations 6
DICPCG:  Solving for p, Initial residual = 0.00188866, Final residual = 8.96397e-05, No Iterations 91
time step continuity errors : sum local = 1.10891e-08, global = 3.87413e-21, cumulative = -1.13581e-19
DICPCG:  Solving for p, Initial residual = 0.00150688, Final residual = 9.98342e-07, No Iterations 138
time step continuity errors : sum local = 1.38899e-10, global = 1.61527e-21, cumulative = -1.11966e-19
ExecutionTime = 48.9 s  ClockTime = 50 s
End
Finalising parallel run
```

Scaling & Speed-up definition

In the context of HPC, there are two common notions of scalability:

- ✓ The first is ***strong scaling***, which is defined as how the solution time varies with the number of processors for a ***fixed total problem size.***
- ✓ The second is ***weak scaling***, which is defined as how the solution time varies with the number of processors ***for a fixed problem size per processor.***

Scaling & Speed-up definition

Useful definitions:

- Parallel efficiency E_p is parallel speedup S_p divided by the parallelism p , *i.e.*

$$E_p = \frac{S_p}{p}$$

- parallel speedup S_p is the fraction of time-to-reference solution T_r over time-to-parallel-solution T_p for a level p of parallelism:

$$S_p = \frac{T_r}{T_p}$$

- Whether "p" counts number of nodes, processors, cores, processes.

Time-to-reference can refer to single core or single node (or chunks). Sometimes it makes sense to compare to a different baseline than single-processor too, since some problems won't even run on a single unit, regardless of how long you give it.

- Theoretical speed-up equal to parallelism $S_p = p$

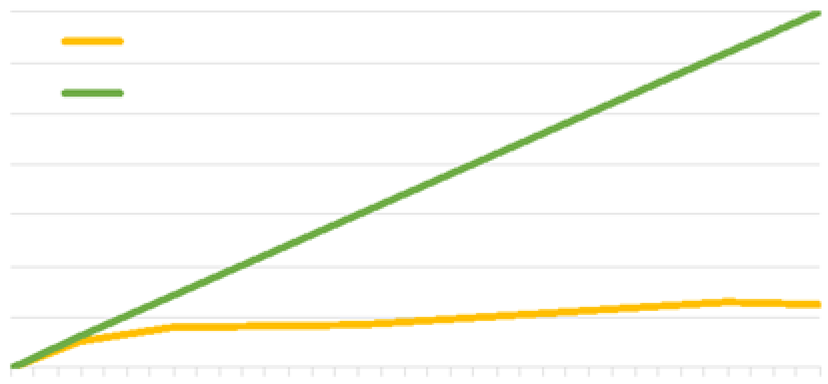
  The idea is that speedup is a comparison of how many times faster a problem can be solved as a function of the number of parallel units, and efficiency is a meter of how big a chunk of that improvement you get per contributing unit.

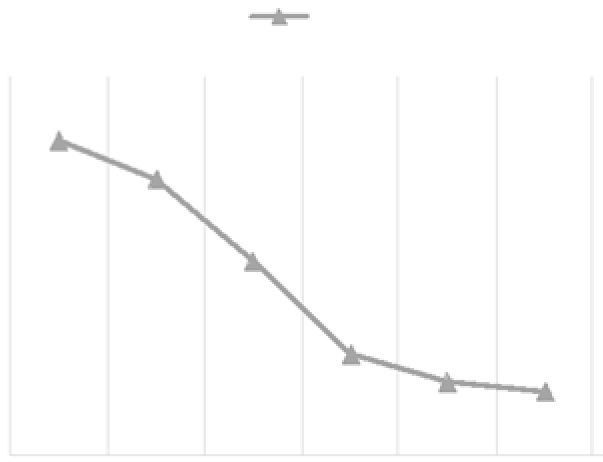
- best-practices: repeat at least twice the measures; use, if possible, full nodes

Strong scaling intra-node

- ✓ Run the 3D lid driven cavity flow in parallel inside a node and measure the speed-up S_p and the parallel efficiency E_p
- ✓ T_r serial time # of cores = 1

Job ID	# of cores	# cell per core	wall clock time (s.)	Speed-up	Efficiency	Theoretical
968599.r000u17l01	1	1.000.000	962	1,00	1	1,0
968610.r000u17l01	4	250.000	274	3,51	0,877737	4,0
968617.r000u17l01	8	125.000	196	4,91	0,61352	8,0
968691.r000u17l01	16	62.500	188	5,12	0,319814	16,0
968696.r000u17l01	32	31.250	130	7,40	0,23125	32,0
968035.r000u17l01	36	27.778	136	7,07	0,196487	36,0





Profiling: Intel MPS

- ✓ Intel MPI Perfomance Snapshothttps://software.intel.com/sites/products/snapshots/mpi-snapshot/documentation/en/help/GUID-5112DC95-80C2-4403-82A3-ED5220936046.html

- ✓ What is that: The first step in analyzing a hybrid MPI/OpenMP* application is getting an overview of the application performance. MPI Performance Snapshot (MPS) that can provide the general performance information about your application. This includes MPI and OpenMP time and load balance information, information about memory and disk usage, most utilized MPI operations, and more.

- ✓ MPI Performance Snapshot is distributed as part of Intel® Trace Analyzer and Collector and is tightly integrated with the Intel® MPI Library. Thus, the analysis is performed by simply adding the -mps option to the launch command.

- ✓ Always check for profiling intrusivity

- ✓ Some other useful profiling tools: gprof, Intel Vtune, scalasca, HPCToolkit, Paraver, etc. etc.

Profiling with Intel MPS

To get profiling stats with Intel MPS, make following changes to the batch submission script:

- ✓ load in your batch the vtune module: **`module load vtune/2017`**
- ✓ source of the file mpsvars.sh: **`source mpsvars.sh -vtune`**
- ✓ add the option mpsrun.sh in the run command line: **`mpirun -np $np mpsrun.sh $solver -parallel > output.$PBS_JOBID`**
- ✓ example of batch script in **`$FOAM_CINECA_SCRIPT/submit.openfoamv1706_prof_mps.pbs`**

```
#!/bin/bash
#PBS -A cin_staff
#PBS -l walltime=00:10:00
#PBS -l select=1:ncpus=4:mpiprocs=4
module load profile/phys
module load autoload
module load openfoam+/v1706
module load vtune/v2017
source mpsvars.sh -vtune
echo "module loaded are the following"
module li
solver=icoFoam
jobid=`echo $PBS_JOBID | sed 's/[.].*//'`
echo "Running on " `hostname`
echo Available resource are `cat $PBS_NODEFILE`
echo "Job started at `date` on nodes: `cat $PBS_NODEFILE` "
cd $PBS_O_WORKDIR
np=`wc -l < $PBS_NODEFILE`
#mpirun -np $np $solver -parallel > output.$PBS_JOBID
mpirun -np $np mpsrun.sh $solver -parallel > output.$PBS_JOBID
```

Profiling with Intel MPS

✓ You need to compile openfoam with the same version of intel used for Intel MPS

```
[ispisso0@r000u07l02 cavity]$ module li
Currently Loaded Modulefiles:
 1) profile/base                         5) intelmpi/2017--binary
 2) autoload                             6) fftw/3.3.4--intelmpi--2017--binary
 3) profile/phys                         7) openfoam+/v1706
 4) intel/pe-xe-2017–binary
qsub submit.openfoamv1706_prof_mps.pbs
```

✓ At the end of the execution, there will be a directory *stat_<date>* with statistics datas gathered from the profiler

```
[ispisso0@r000u07l02 cavity]$ ls stat_20171112/
stat-0.bin  stat-1.bin  stat-2.bin  stat-3.bin  stat-4.bin  stat-5.bin  stat-6.bin  stat-7.bin
```

✓ To generate the report with Intel mps and produce the xhml file, for a single process type:

```
[ispisso0@r000u07l02 cavity]$ mps stat_20171112/ -O report_4cores.htlm
or
[ispisso0@r000u07l02 cavity]$ mps stat_20171112/ -O report_4cores.htlm > a.out
```

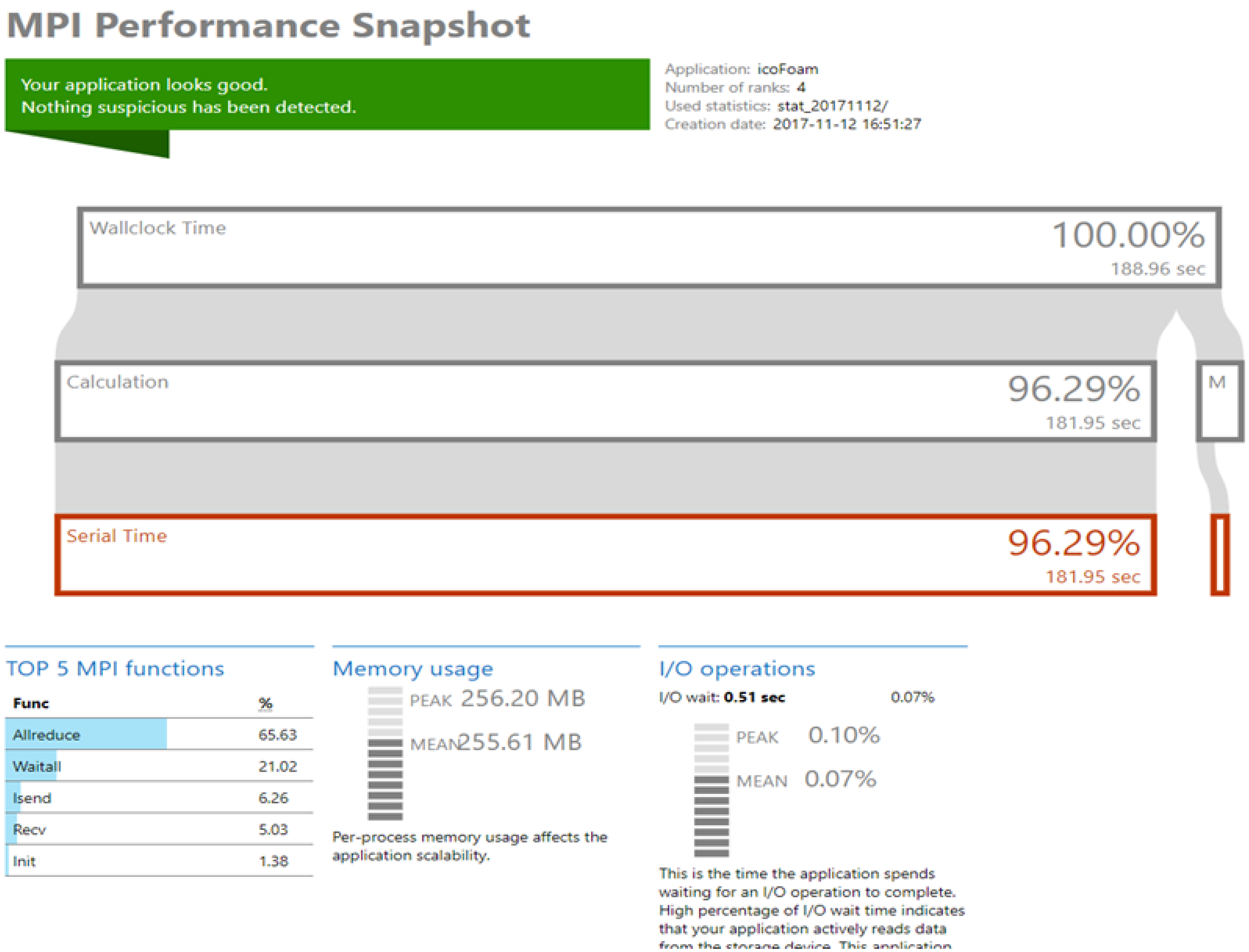
✓ To generate the report with Intel mps and produce the xhml file, for all processes:

```
[ispisso0@r000u07l02 cavity]$ mps -a stat_20171112/ -O report_8.html > 8.out
```

✓ You can read the output file a.out or visualize in a browser the html

CINECA SCAI SuperComputing Applications and Innovation

Profiling with Intel MPS



MPI Performance Snapshot

Your application looks good.
Nothing suspicious has been detected.

Application: icoFoam
Number of ranks: 4
Used statistics: stat_20171112/
Creation date: 2017-11-12 16:51:27

Wallclock Time 100.00% 188.96 sec

Calculation 96.29% 181.95 sec

M

Serial Time 96.29% 181.95 sec

TOP 5 MPI functions

Func	%
Allreduce	65.63
Waitall	21.02
Isend	6.26
Recv	5.03
Init	1.38

Memory usage

PEAK 256.20 MB
MEAN 255.61 MB

Per-process memory usage affects the application scalability.

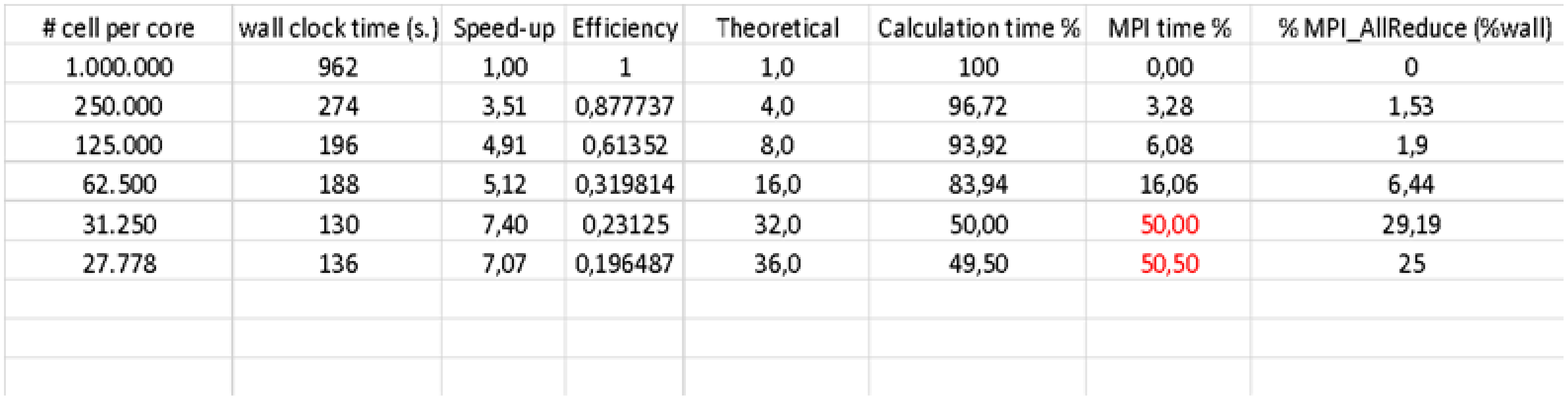
I/O operations

I/O wait: **0.51 sec** 0.07%

PEAK 0.10%
MEAN 0.07%

This is the time the application spends waiting for an I/O operation to complete. High percentage of I/O wait time indicates that your application actively reads data

Strong scaling intra-node with profiling stats



# cell per core	wall clock time (s.)	Speed-up	Efficiency	Theoretical	Calculation time %	MPI time %	% MPI_AllReduce (%wall)
1.000.000	962	1,00	1	1,0	100	0,00	0
250.000	274	3,51	0,877737	4,0	96,72	3,28	1,53
125.000	196	4,91	0,61352	8,0	93,92	6,08	1,9
62.500	188	5,12	0,319814	16,0	83,94	16,06	6,44
31.250	130	7,40	0,23125	32,0	50,00	50,00	29,19
27.778	136	7,07	0,196487	36,0	49,50	50,50	25

Profiling with Intel MPS plus IPM Stats

- ✓ To enable the IPM (Integrated Performance Monitoring) statistics collection, set I_MPI_STATS to *ipm:terse* for a brief summary or to ipm for complete statistics information.
- ✓ https://software.intel.com/en-us/mpi-developer-guide-linux-gathering-statistics
- ✓ export I_MPI_STATS = X
- ✓ X < 6 non intrusive
- ✓ X > 20 complete, (could be very intrusive)
- ✓ For example, to collect statistics in all formats with the maximal level of details, (very verbose)

```
export I_MPI_STATS=ipm
export I_MPI_STATS=all

#!/bin/bash
#PBS -A cin_staff
#PBS -l walltime=00:10:00
#PBS -l select=1:ncpus=4:mpiprocs=4
module load profile/phys
module load autoload
module load openfoam+/v1706
export  I_MPI_STATS=ipm
export I_MPI_STATS_ACCURACY=ALL
module load vtune/v2017
source mpsvars.sh -vtune

[ispisso0@r000u07l02 cavity]$ qsub submit.openfoamv1706_prof_mps_plus_ipm.pbs
968514.r000u17l01
[ispisso0@r000u07l02 cavity]$ ls stat
stat_20171112/ stats.ipm      stats.txt
vim stats.ipm
```

Profiling with Intel MPS plus IPM Stats

```
Intel(R) MPI Library Version 2017 Update 3
Summary MPI Statistics
Stats format: region
Stats scope : full
################################################################################
#
# command : icoFoam (completed)
# host    : r044c02s02/x86_64_Linux         mpi_tasks : 4 on 1 nodes
# start   : 11/12/17/17:02:44               wallclock : 274.533625 sec
# stop    : 11/12/17/17:07:19               %comm     : 3.19
# gbytes  : 0.00000e+00 total               gflop/sec : NA
################################################################################
# region  : *   [ntasks] = 4
#                        [total]       <avg>         min           max
# entries                4             1             1             1
# wallclock              1097.86       274.465       274.424       274.534
# user                   1083.52       270.88        270.359       271.248
# system                 11.4077       2.85193       2.47923       3.42251
# mpi                    35.0049       8.75121       6.56253       11.7063
# %comm                                3.18846       2.39105       4.26551
# gflop/sec              NA            NA            NA            NA
# gbytes                 0             0             0             0
#
#
#                          [time]        [calls]       <%mpi>        <%wall>
# MPI_Allreduce            15.9625       330092        45.60         1.45
# MPI_Waitall              9.66497       124252        27.61         0.88
# MPI_Isend                3.65671       372756        10.45         0.33
# MPI_Recv                 1.75148       3432          5.00          0.16
# MPI_Irecv                1.29933       372756        3.71          0.12
# MPI_Comm_test_inter      0.714333      330112        2.04          0.07
# MPI_Comm_rank            0.698695      330120        2.00          0.06
# MPI_Init                 0.65523       4             1.87          0.06
# MPI_Comm_size            0.556019      330124        1.59          0.05
# MPI_Probe                0.0223291     60            0.06          0.00
# MPI_Send                 0.0133548     3432          0.04          0.00
# MPI_Alltoall             0.00584102    12            0.02          0.00
# MPI_Get_count            0.00296474    3492          0.01          0.00
# MPI_Comm_create          0.000835419   4             0.00          0.00
# MPI_Comm_free            0.000126123   4             0.00          0.00
# MPI_Finalize             5.19753e-05   4             0.00          0.00
# MPI_Comm_group           2.59876e-05   20            0.00          0.00
# MPI_Buffer_attach        1.5974e-05    4             0.00          0.00
# MPI_Buffer_detach        1.14441e-05   4             0.00          0.00
…
# MPI_TOTAL                35.0049       2.2007e+06    100.00        3.19
################################################################################
```

Further tutorial

- ✓ Weak scalability tests 3d Lid-driven cavity 100^3 cells
- ✓ Inter-node strong strong scalability tests 200^3 (8 M) o 300^3 (27 M) of cells
- ✓ Compare with KNL
- ✓ Compare with different openfoam version
- ✓ Automatize the tests with python script or dakota

Some issues about scaling

- ✓ Using up-to-date CPU in serial performance could be unfair (low frequency)
- ✓ Always look for the best configuration (task/node)
- ✓ Rule of thumb:
  - always use more the 10K/20/50 K elements per task
  - Sometimes use #task less then #core
- ✓ Weak scaling (Scale-up) is “problematic” for implicit solvers
- ✓ Play with convergence
- ✓ Play with decomposition

Some issues about scaling

Twelve Ways to fool the masses when giving performance results on parallel computing (http://crd-legacy.lbl.gov/~dhbailey/dhbpapers/twelve-ways.pdf)

what you have NOT to do

1. Quote only 32-bit performance results, not 64-bit results
2. Present performance figures for an inner kernel, and then represent these figures as the performance of the entire application.
3. Quietly employ assembly code and other low-level language constructs.
4. Scale up the problem size with the number of processors, but omit any mention of this fact.
5. Quote performance results projected to a full system.
6. Compare your results against scalar, unoptimized code on Crays
7. When direct run time comparisons are required, compare with an old code on an obsolete system.
8. If MFLOPS rates must be quoted, base the operation count on the parallel implementation, not on the best sequential implementation
9. Quote performance in terms of processor utilization, parallel speedups or MFLOPS per dollar.
10. Mutilate the algorithm used in the parallel implementation to match the architecture.
11. Measure parallel run times on a dedicated system, but measure conventional run times in a busy environment.
12. If all else fails, show pretty pictures and animated videos, and don't talk about performance