

## BSC Tools Hands-On

---

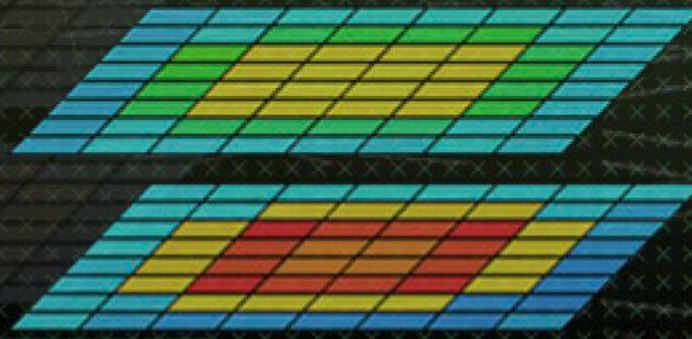
---

Judit Giménez, Lau Mercadal ([lau.mercadal@bsc.es](mailto:lau.mercadal@bsc.es))

Barcelona Supercomputing Center

---

---



# Extrae

## Extrae features

---

- Parallel programming models
  - MPI, OpenMP<sup>(\*)</sup>, pthreads, OmpSs, CUDA, CUPTI, OpenCL, Java, Python...
- Platforms: Intel, Cray, BlueGene, Fujitsu Sparc, MIC, ARM, Android...
- Performance Counters
  - Using PAPI and PMAPI interfaces
- Link to source code
  - Callstack at MPI routines
  - OpenMP outlined routines and their containers
  - Selected user functions
- And more: Sampling, IO, memory allocation...
- User events (Extrae API)



**No need to  
recompile / relink!**

## Extrae overheads

---

|                              | Average values | CoolMUC-3 |
|------------------------------|----------------|-----------|
| Event                        | 150-200 ns     | 130 ns    |
| Event + PAPI                 | 750 ns – 1 us  | 850 ns    |
| Event + callstack (1 level)  | 600 ns         | 1.3 us    |
| Event + callstack (6 levels) | 1.9 us         | 3.1 us    |

# How does Extrae work?

---

- Symbol substitution through LD\_PRELOAD
  - Specific libraries for each combination of runtimes
    - MPI
    - OpenMP
    - OpenMP+MPI
    - OmpSs
    - ...
- Dynamic instrumentation
  - Based on DynInst (developed by U.Wisconsin/U.Maryland)
    - Instrumentation in memory
    - Binary rewriting
- Static link (i.e., PMPI, Extrae API)



**Recommended**

# Using Extrae in 3 steps


---

1. Adapt the job submission script
  2. [Optional] Tune the Extrae XML configuration file
    - Examples distributed with Extrae at \$EXTRAE\_HOME/share/example
  3. Run with instrumentation
- For further reference check the **Extrae User Guide:**
    - [https://tools.bsc.es/tools\\_manuals](https://tools.bsc.es/tools_manuals)

## Log in and copy the examples to your work directory

@ your laptop

```
> ssh -Y <USER>@lxlogin8.lrz.de  
  
> cp -r /home/hpc/a2c06/lu23vet/tools-material ./  
  
> ls -l $HOME/tools-material  
...apps  
...extrae  
...slides  
...traces
```



Here's a copy  
of these slides

## Step 1: Adapt the job script to load Extrae with LD\_PRELOAD

```
> vi $HOME/tools-material/extrae/job.slurm
```

```
#!/bin/sh

#SBATCH -J lulesh2
#SBATCH -o lulesh2_%j.out
#SBATCH -e lulesh2_%j.err
#SBATCH --ntasks=64
#SBATCH --ntasks-per-core=1
#SBATCH --time=00:15:00
#SBATCH --cluster=mpp3
#SBATCH --reservation=TuningWorkshop
#SBATCH --exclusive

export OMP_NUM_THREADS=1

# run the script
mpiexec ../apps/lulesh2.0 -i 10 -s 65 -p
```



## Step 1: Adapt the job script to load Extrae with LD\_PRELOAD

```
> vi $HOME/tools-material/extrae/job.slurm
```

```
#!/bin/sh

#SBATCH -J lulesh2
#SBATCH -o lulesh2_%j.out
#SBATCH -e lulesh2_%j.err
#SBATCH --ntasks=64
#SBATCH --ntasks-per-core=1
#SBATCH --time=00:15:00
#SBATCH --cluster=mpp3
#SBATCH --reservation=TuningWorkshop
#SBATCH --exclusive

export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh2_64p.prv

# run the script
mpixexec ./trace.sh ../apps/lulesh2.0
```

## Step 1: Adapt the job script to load Extrae with LD\_PRELOAD

```
> vi $HOME/tools-material/extrae/trace.sh
```

```
#!/bin/sh

#SBATCH -J lulesh2
#SBATCH -o lulesh2_%j.out
#SBATCH -e lulesh2_%j.err
#SBATCH --ntasks=64
#SBATCH --ntasks-per-core=1
#SBATCH --time=00:15:00
#SBATCH --cluster=mpp3
#SBATCH --reservation=TuningWorkshop
#SBATCH --exclusive

export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh2_64p_piv

# run the script
mpixec ./trace.sh ../apps/lulesh2.0
```

```
#!/bin/bash

# Configure Extrae
export EXTRAE_HOME=/home/hpc/a2c06/lu23vet/ \
                    bsctools/extrae/3.5.2-impi_2017
source ${EXTRAE_HOME}/etc/extrae.sh
export EXTRAE_CONFIG_FILE=./extrae.xml

# Load the tracing library (choose C/fortran)
export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitrace.so
#export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitracer.so

# Run the program
$*
```

Pick a tracing library

## Step 1: LD\_PRELOAD library selection

| Library                          | Serial | MPI | OpenMP | OmpSs | pthread | CUDA |
|----------------------------------|--------|-----|--------|-------|---------|------|
| libseqtrace                      | ✓      |     |        |       |         |      |
| libmpitrace[f] <sup>1</sup>      |        | ✓   |        |       |         |      |
| libomptrace                      |        |     | ✓      |       |         |      |
| libnanotracer                    |        |     |        | ✓     |         |      |
| libpttrace                       |        |     |        |       | ✓       |      |
| libcudatracer                    |        |     |        |       |         | ✓    |
| libompitrace[f] <sup>1</sup>     |        | ✓   | ✓      |       |         |      |
| libnanosmpitrace[f] <sup>1</sup> |        | ✓   |        | ✓     |         |      |
| libptmpitrace[f] <sup>1</sup>    |        | ✓   |        |       | ✓       |      |
| libcudampitrace[f] <sup>1</sup>  |        | ✓   |        |       |         | ✓    |

<sup>1</sup> include suffix "f" in Fortran codes

## Step 3: Run with instrumentation

---

@ CoolMUC-3

```
> cd $HOME/tools-material/extrae  
> sbatch job.slurm
```

- Check the state with:

@ CoolMUC-3

```
> squeue -u ${USER}
```

## Step 2: Extrae XML configuration: extrae.xml

```
<mpi enabled="yes">
```

```
  <counters enabled="yes" />
```

Trace MPI calls + HW counters

```
</mpi>
```

```
<openmp enabled="yes">
```

```
  <locks enabled="no" />
```

```
  <counters enabled="yes" />
```

```
</openmp>
```

```
<pthread enabled="no">
```

```
  <locks enabled="no" />
```

```
  <counters enabled="yes" />
```

```
</pthread>
```

```
<callers enabled="yes">
```

```
  <mpi enabled="yes">1-3</mpi>
```

```
  <sampling enabled="no">1-5</sampling>
```

```
</callers>
```

Trace call-stack events @ MPI calls

## Step 2: Extrae XML configuration: extrae.xml (II)

```
<counters enabled="yes">
  <cpu enabled="yes" starting-set-distribution="cyclic">
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM
    </set>
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_BR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_BR_MSP
    </set>
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_SR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_LD_INS
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>
```

**Define which HW counters  
are measured**

## Step 2: Extrae XML configuration: extrae.xml (III)

```
<buffer enabled="yes">
```

```
  <size enabled="yes">5000000</size>
```

```
  <circular enabled="no" />
```

```
</buffer>
```

Trace buffer size

```
<sampling enabled="no" type="default" period="50m" variability="10m" />
```

Enable sampling

```
<merge enabled="yes">
```

```
  synchronization="default"
```

```
  tree-fan-out="16"
```

```
  max-memory="512"
```

```
  joint-states="yes"
```

```
  keep-mpits="yes"
```

```
  sort-addresses="yes"
```

```
  overwrite="yes">
```

```
  $TRACE_NAME$
```

```
</merge>
```

Merge intermediate files into Paraver trace

## Check the resulting trace

---

- After the execution you will get the trace (3 files):

@ CoolMUC-3

```
> ls -l $HOME/tools-material/extrae
...
lulesh2_64p.pcf
lulesh2_64p.prv
lulesh2_64p.row
```

- Copy these files to your computer

@ your computer

```
> scp <USER>@lxlogin8.lrz.de:$HOME/tools-material/extrae/ \  
*.{pcf,prv,row} $HOME
```



# Paraver

# Installing Paraver

- Download the Paraver binaries from our website

The screenshot shows the BSC website's Downloads page. The navigation bar includes Home, Paraver, Dimemas, Extrae, Research, Documentation, Downloads, and Publications. A notification at the top reads "news@tools:~ > Paraver 4.7.2 available!". The main content is divided into two sections: CORE TOOLS and PERFORMANCE ANALYTICS. Under CORE TOOLS, there are three tool cards: EXTRAE (Version 3.5.2 • 1.21 MB), PARAVAR (Version 4.7.2 • 1.57 MB), and DIMEMAS (Version 5.3.4 • 0.9 MB). Each card has a "Get" button and icons for supported platforms (101 RAW, Linux, Windows, macOS, and various architectures). Under PERFORMANCE ANALYTICS, there are three tool cards: CLUSTERING (Version 2.6.6 • 1.97 MB), TRACKING (Version 2.6.7 • 1.87 MB), and FOLDING (Version 1.3.1 • 12.67 MB). Each card also has a "Get" button and platform icons.

# Installing Paraver

- Or copy them from CoolMUC-3:

@ your computer

```
> scp <USER>@lxlogin8.lrz.de:/home/hpc/a2c06/lu23vet/  
tools-packages/<VERSION> $HOME
```

Pick your version

Linux 64 bits

`wxparaver-4.7.2-Linux_x86_64.tar.bz2`

Linux 32 bits

`wxparaver-4.7.2-Linux_i686.tar.bz2`

Mac

`wxparaver-4.7.2-mac.zip`

Windows

`wxparaver-4.7.2-win.zip`

# Installing Paraver

---

- Uncompress the package into your home directory:

@ your computer

```
> tar xf wxparaver-4.7.2-Linux_x86_64.tar.bz2  
> ln -s $HOME/wxparaver-4.7.2-Linux_x86_64 $HOME/paraver
```

- Download Paraver tutorials and uncompress into the Paraver directory

- <https://tools.bsc.es/sites/default/files/documentation/paraver-tutorials-20150526.tar.gz>

@ your computer

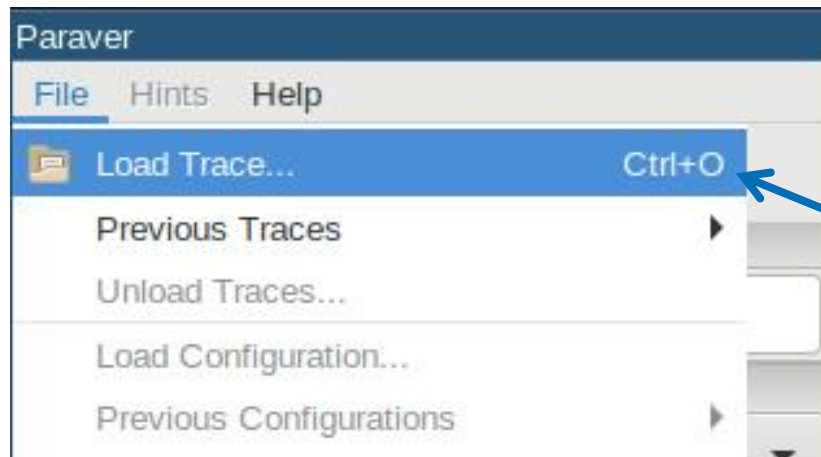
```
> tar xf $HOME/paraver-tutorials-20150526.tar.gz  
> mv paraver-tutorials-20150526 $HOME/paraver/tutorials
```

## Check that everything works

- Start Paraver

```
> $HOME/paraver/bin/wxparaver
```

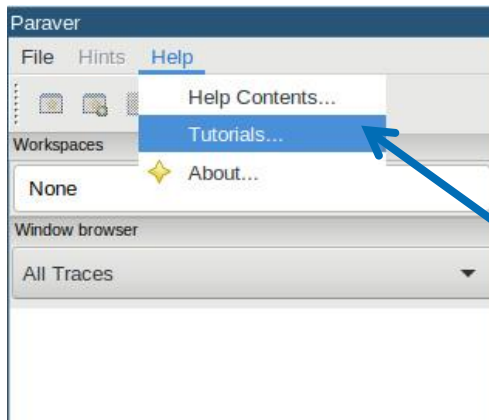
- Load the trace



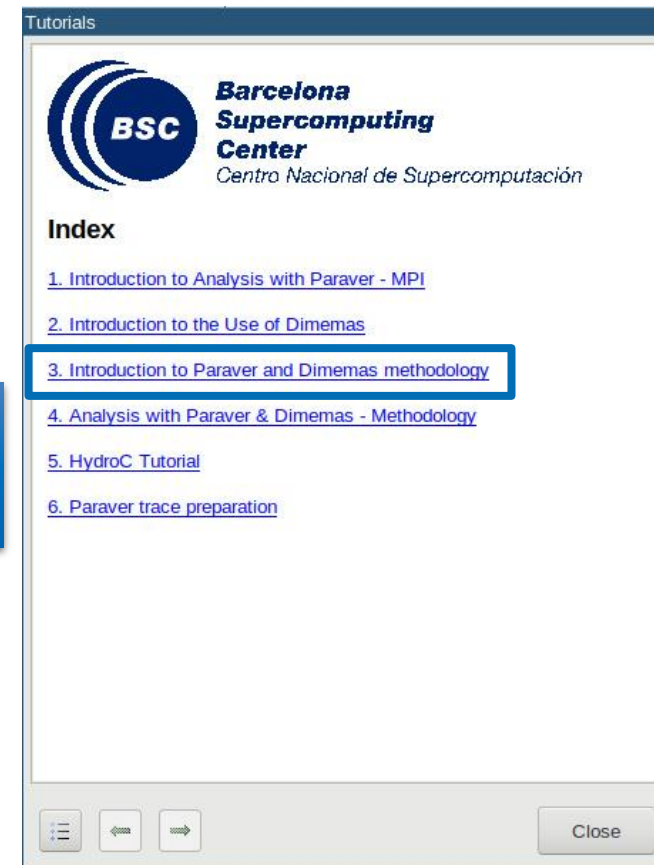
**Click on File → Load Trace  
Browse to "lulesh2\_64p.prv"**

# Check that everything works

- Check that tutorials are available & follow #3



Click on Help →  
Tutorials



# Measure the parallel efficiency

- Click on the “mpi\_stats.cfg”
  - Check the Average for the column labeled “Outside MPI”

Tutorials

To measure the parallel efficiency load the configuration file [cfigs/mpi/mpi\\_stats.cfg](#). This configuration pops up a table with %time that every thread spends in every MPI call. Look at the global statistics at the bottom of the outside mpi column. Entry *Average* represents the application parallel efficiency, entry *Avg/Max* represents the global load balance and entry *Maximum* represents the communication efficiency. If any of those values are lower than 85% is recommended to look at the corresponding metric in detail. Open the control window to identify the phases and iterations of the code.

- To measure the computation time distribution load the configuration file [cfigs/general/2dh\\_usefulduration.cfg](#). This configuration pops up a histogram of the duration for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram does not show vertical lines, it indicates the computation time may be not balanced. Open the control window to look at the time distribution and visually correlate both views.
- To measure the computational load (instructions) distribution load the configuration file [cfigs/papi/2dh\\_useful\\_instructions.cfg](#). This configuration pops up a histogram of the instructions for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram doesn't show vertical lines, it indicates the distribution of the instructions may be not balanced. Open the control window to look at the time distribution and visually correlate both views.
- To measure the serial regions performance look at the IPC timeline loaded with [cfigs/general/2dh\\_useful\\_instructions.cfg](#). This configuration would depend on the machine used to run the application. If the value is lower than 1 identify poor performance. Open the control window to modify the computation time modifying the Statistic of the useful duration histogram to use correlate with metric and verify that the selected Metric is Instructions per cycle. Now the cell color corresponds to the value. The color between duration (position) and IPC (Instructions per cycle) region of the histogram would allow you to identify the regions with different IPC. Change the Metric to Instructions per cycle.

Close

MPI call profile @ lulesh2\_64p.prv

|               |            |        |        |         |         |        |         |          |
|---------------|------------|--------|--------|---------|---------|--------|---------|----------|
| THREAD 1.54.1 | 86.07 %    | 0.12 % | 0.05 % | 0.31 %  | 0.49 %  | 0.02 % | 1.02 %  | 11.55 %  |
| THREAD 1.55.1 | 84.55 %    | 0.11 % | 0.05 % | 0.08 %  | 0.97 %  | 0.23 % | 1.13 %  | 12.50 %  |
| THREAD 1.56.1 | 93.17 %    | 0.08 % | 0.03 % | 0.08 %  | 1.13 %  | 0.24 % | 0.26 %  | 4.66 %   |
| THREAD 1.57.1 | 93.35 %    | 0.08 % | 0.03 % | 0.29 %  | 0.85 %  | 0.02 % | 0.27 %  | 4.74 %   |
| THREAD 1.58.1 | 92.91 %    | 0.12 % | 0.05 % | 0.30 %  | 0.81 %  | 0.02 % | 0.31 %  | 5.12 %   |
| THREAD 1.59.1 | 90.69 %    | 0.11 % | 0.05 % | 0.07 %  | 0.73 %  | 0.23 % | 0.55 %  | 7.22 %   |
| THREAD 1.60.1 | 91.91 %    | 0.09 % | 0.03 % | 0.06 %  | 0.64 %  | 0.24 % | 0.43 %  | 6.23 %   |
| THREAD 1.61.1 | 87.71 %    | 0.06 % | 0.02 % | 0.29 %  | 1.04 %  | 0.02 % | 0.82 %  | 9.68 %   |
| THREAD 1.62.1 | 86.91 %    | 0.09 % | 0.03 % | 0.26 %  | 0.63 %  | 0.02 % | 0.94 %  | 10.77 %  |
| THREAD 1.63.1 | 88.03 %    | 0.09 % | 0.03 % | 0.06 %  | 0.71 %  | 0.24 % | 0.81 %  | 9.67 %   |
| THREAD 1.64.1 | 85.00 %    | 0.07 % | 0.02 % | 0.05 %  | 1.33 %  | 0.24 % | 1.06 %  | 11.86 %  |
| Total         | 5,768.95 % | 6.32 % | 3.19 % | 11.73 % | 44.08 % | 9.62 % | 46.40 % | 493.47 % |
| Average       | 90.14 %    | 0.10 % | 0.05 % | 0.18 %  | 0.69 %  | 0.15 % | 0.73 %  | 7.71 %   |
| Maximum       | 98.11 %    | 0.17 % | 0.09 % | 0.50 %  | 1.75 %  | 0.24 % | 1.43 %  | 13.18 %  |
| Minimum       | 84.13 %    | 0.04 % | 0.02 % | 0.05 %  | 0.14 %  | 0.00 % | 0.00 %  | 0.03 %   |
| StDev         | 3.53 %     | 0.03 % | 0.02 % | 0.11 %  | 0.31 %  | 0.09 % | 0.37 %  | 3.19 %   |
| Avg/Max       | 0.92       | 0.59   | 0.55   | 0.36    | 0.39    | 0.62   | 0.51    | 0.59     |

MPI\_Irecv

**Parallel efficiency** → Average

**Comm efficiency** → Maximum

**Load balance** → Avg/Max

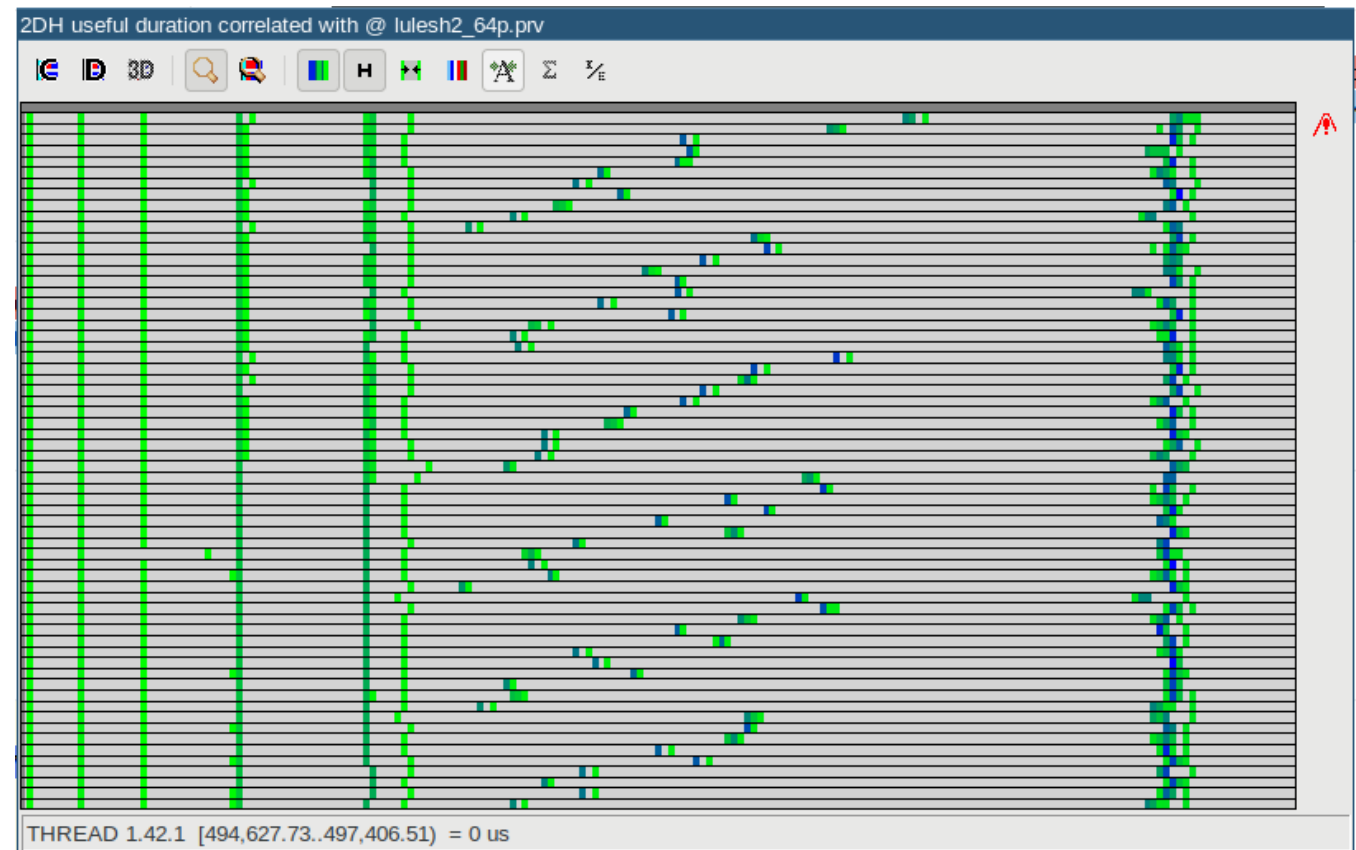
# Measure the computation time distribution

- Click on the “2dh\_usefulduration.cfg”

Tutorials

- To **measure the parallel efficiency** load the configuration file [cfs/mpi/mpi\\_stats.cfg](#) This configuration pops up a table with %time that every thread spends in every MPI call. Look at the global statistics at the bottom of the outside mpi column. Entry *Average* represents the application parallel efficiency, entry *Avg/Max* represents the global load balance and entry *Maximum* represents the communication efficiency. If any of those values are lower than 85% is recommended to look at the corresponding metric in detail. Open the control window to identify the phases and iterations of the code.
- To **measure the computation time distribution** load the configuration file [cfs/general/2dh\\_usefulduration.cfg](#) This configuration pops up a histogram of the duration for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram does not show vertical lines, it indicates the computation time may be not balanced. Open the control window to look at the time distribution and visually correlate both views.
- To **measure the computational load (instructions) distribution** load the configuration file [cfs/papi/2dh\\_useful\\_instructions.cfg](#) This configuration pops up a histogram of the instructions for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram doesn't show vertical lines, it indicates the distribution of the instructions may be not balanced. Open the control window to look at the time distribution and correlate both views.
- To **measure the serial regions performance** look at the IPC timeline loaded with [cfs/general/2dh\\_usefulduration.cfg](#). What it's a reasonable IPC would depend on the machine used to run the application, but typically values lower than 1 identify poor performance sections. You can correlate the IPC with the computation time modifying the Statistic of the useful duration histogram to use correlate with metric and verify that the selected Metric is Instructions per cycle. Now the cell color corresponds to the IPC showing the correlation between duration (position) and IPC (color). Zooming into an unbalanced region of the histogram would allow you to verify if the unbalance is related to a different IPC. Change the Metric to Instructions to correlate the duration with

Close





# Measure the computation time distribution

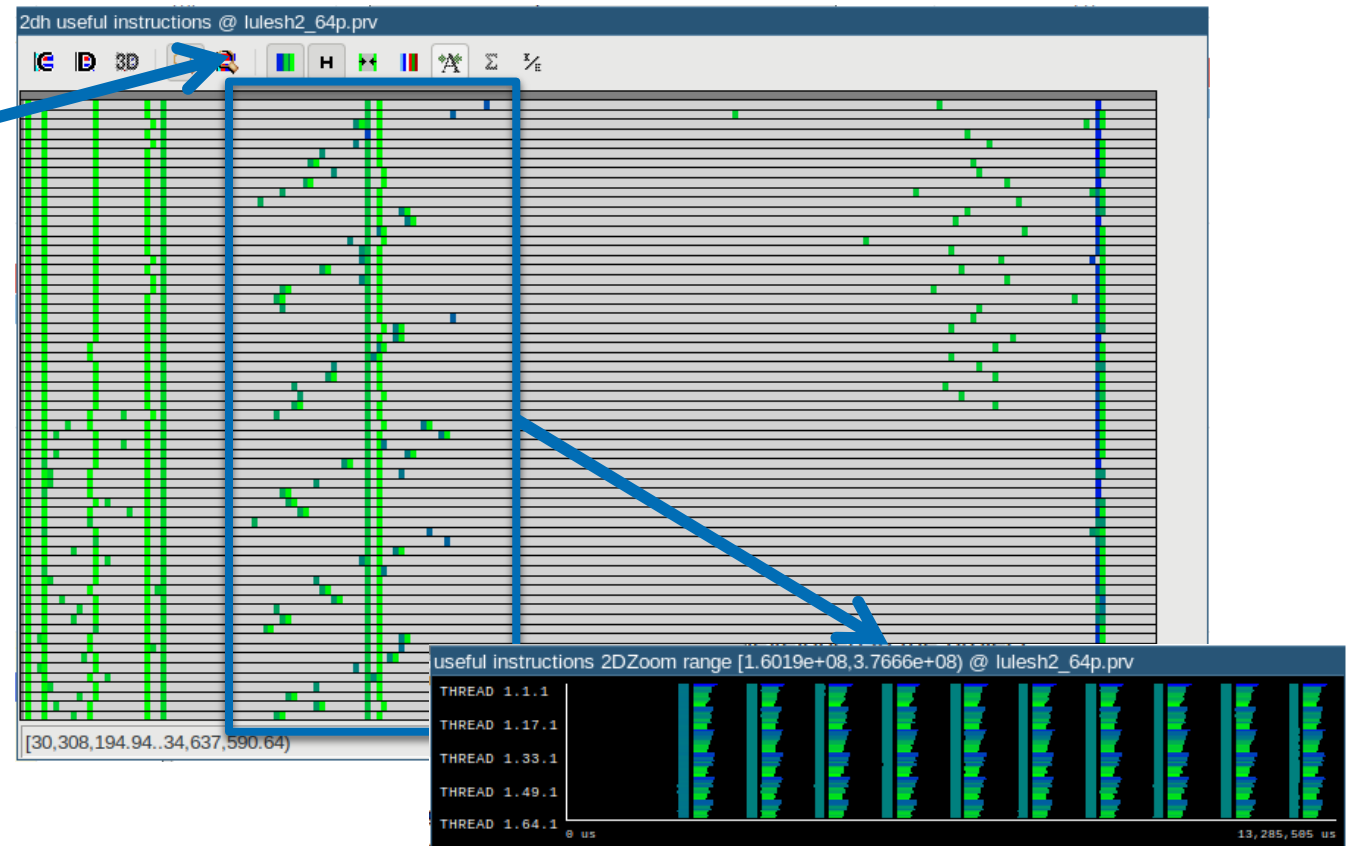
- Click on the “2dh\_useful\_instructions.cfg”

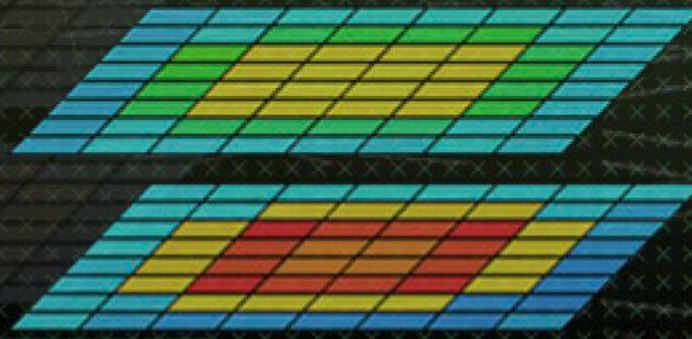
Tutorials

- To measure the parallel efficiency, load the configuration file `cfgs/mpi/mpi_stats.cfg`. This configuration file shows the time each thread spends in every MPI call. The time spent in the outside mpi column. Efficiency is calculated as  $\text{Efficiency} = \frac{\text{Avg/Max}}{\text{Maximum}}$ . An efficiency lower than 85% is recommended. Open the control window to look at the time distribution and visually correlate both views.
- To measure the computation time distribution load the configuration file `cfgs/general/2dh_usefulduration.cfg`. This configuration file pops up a histogram of the duration for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram does not show vertical lines, it indicates the computation time may be not balanced. Open the control window to look at the time distribution and visually correlate both views.
- To measure the computational load (instructions) distribution load the configuration file `cfgs/papi/2dh_useful_instructions.cfg`. This configuration file pops up a histogram of the instructions for the computation regions. The computation regions are delimited by the exit from an MPI call and the entry to the next call. If the histogram does not show vertical lines, it indicates the distribution of the instructions may be not balanced. Open the control window to look at the time distribution and correlate both views.
- To measure the serial regions performance look at the IPC timeline loaded with `cfgs/general/2dh_usefulduration.cfg`. What is a reasonable IPC would depend on the machine used to run the application, but typically values lower than 1 identify poor performance sections. You can correlate the IPC with the computation time modifying the Statistic of the useful duration histogram to use correlate with metric and verify that the selected Metric is Instructions per cycle. Now the cell color corresponds to the IPC showing the correlation between duration (position) and IPC (color). Zooming into an unbalanced region of the histogram would allow you to verify if the unbalance is related to a different IPC. Change the Metric to Instructions to correlate the duration with

Click on “Open Filtered Control Window” and select this area

Close





# Cluster analysis

# Cluster-based analysis

---

- Run clustering

@ CoolMUC-3

```
> cd $HOME/tools-material/clustering  
> ./clusterize.sh ../extrae/lulesh2_64p.prv
```

- If you didn't get your own trace, find one at:

@ CoolMUC-3

```
> cd $HOME/tools-material/traces
```

# Looking at the clusters

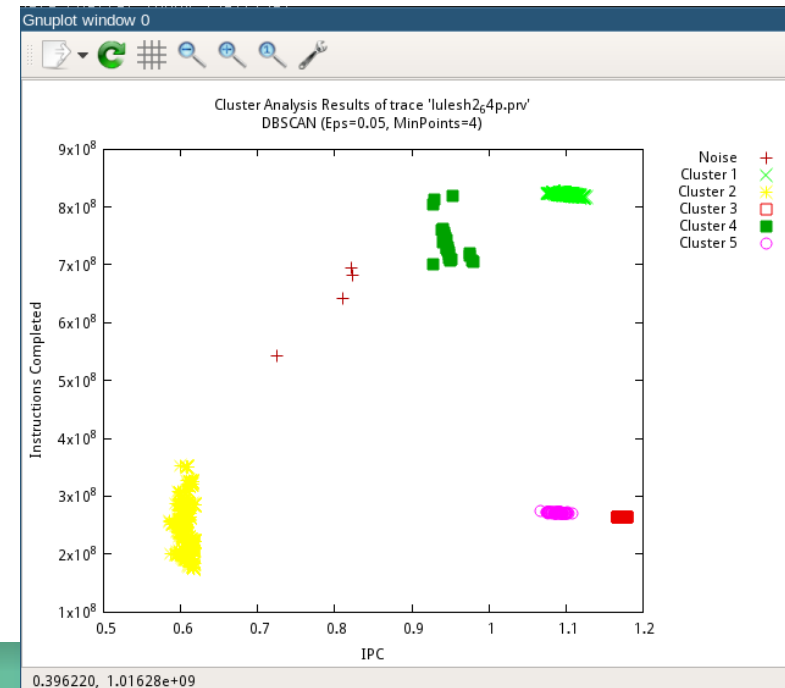
- Check the clustering scatter plot

@ CoolMUC-3

```
> module load gnuplot
```

```
> gnuplot lulesh2_64.clustered.IPC.PAPI_TOT_INS.gnuplot
```

- Identify main computing trends
  - Work (Y-axis), Performance (X-axis)
- See the elongated clusters?
  - Large IPC variability
  - Wide range of instructions
  - Indicate potential imbalances

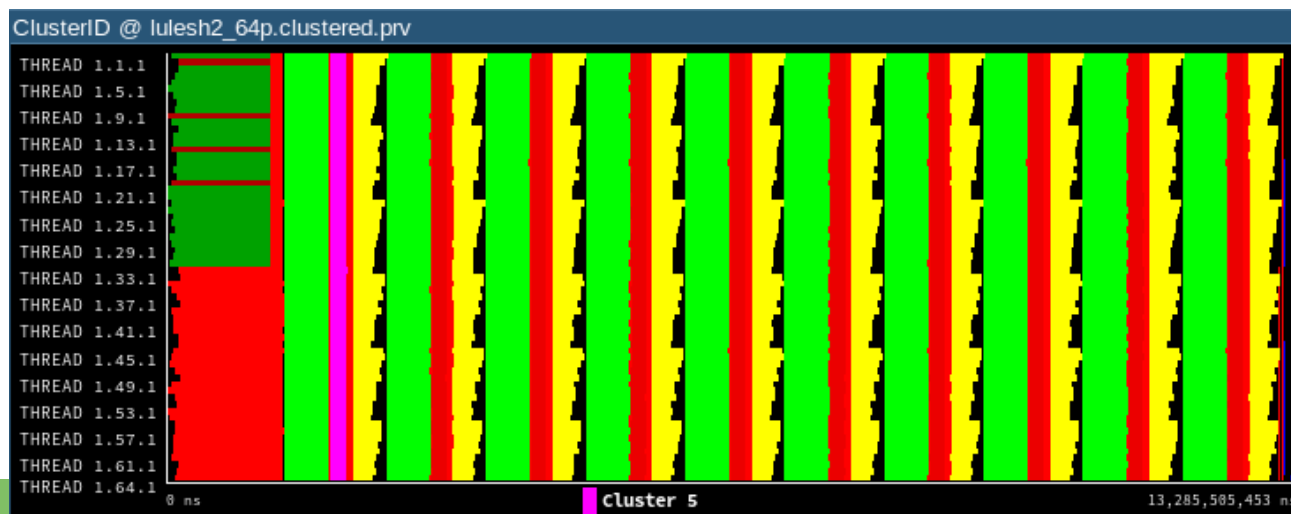


## Looking at the clustered trace

- Copy and load the clustered trace with Paraver

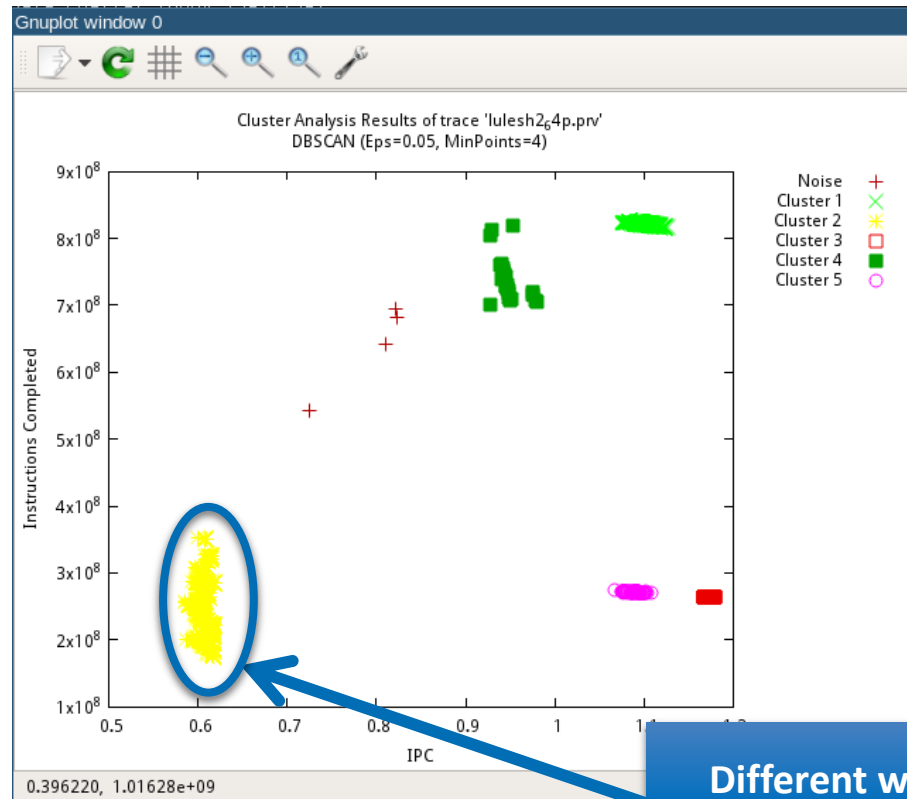
```
> scp <USER>@lxlogin8.lrz.de:$HOME/tools-material/clustering/ \  
*.{pcf,prv,row} $HOME
```

- File → Load Trace → Browse to **\$HOME/tools-material/clustering/lulesh2\_64.clustered.prv**
- Display the distribution of clusters over time
  - File → Load configuration → Browse to **\$PARAVER\_HOME/cfgs/clustering/clusterID\_window.cfg**

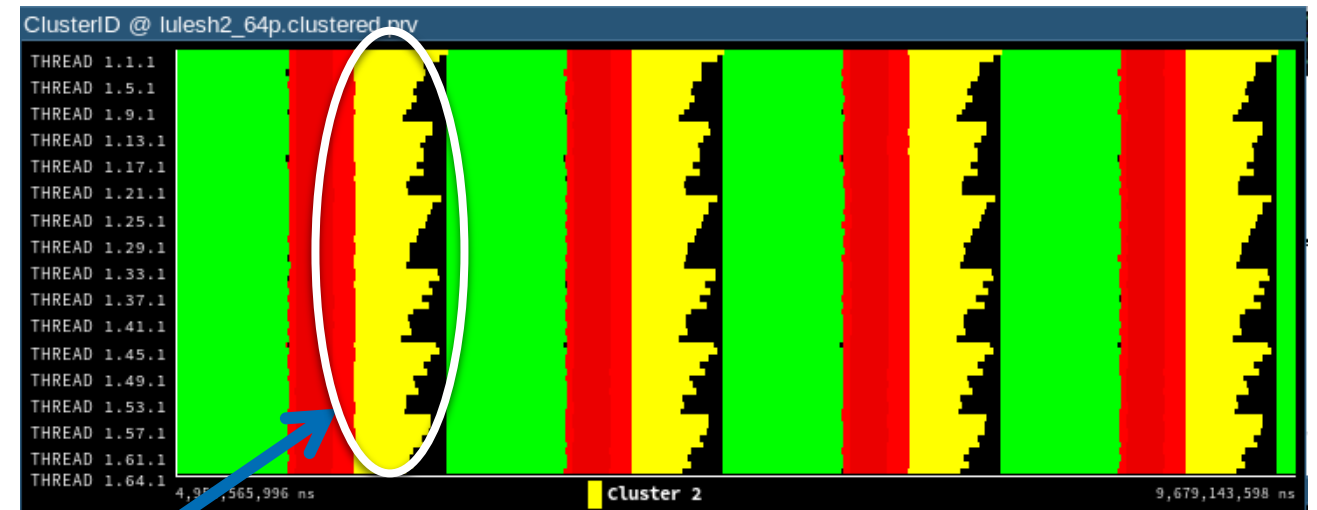


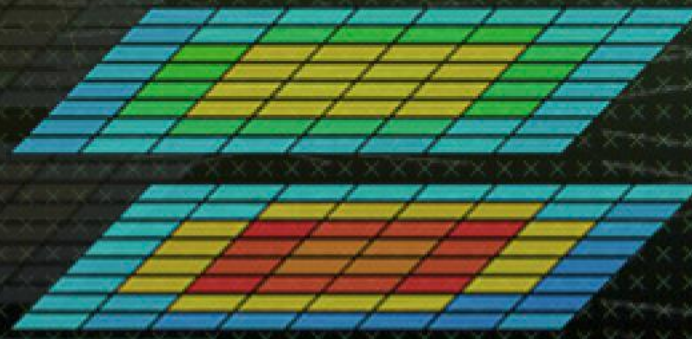
# Looking at the clustered trace

- Correlate scatter-plots and timelines to detect imbalances



Different work,  
same speed





# Dimemas

## Simulating with Dimemas

---

- Simulate an ideal machine

@ CoolMUC-3

```
> cd $HOME/tools-material/dimemas  
> ./dimemas.sh ../extrae/lulesh2_64p ideal.cfg
```

- Copy the simulated trace to your computer

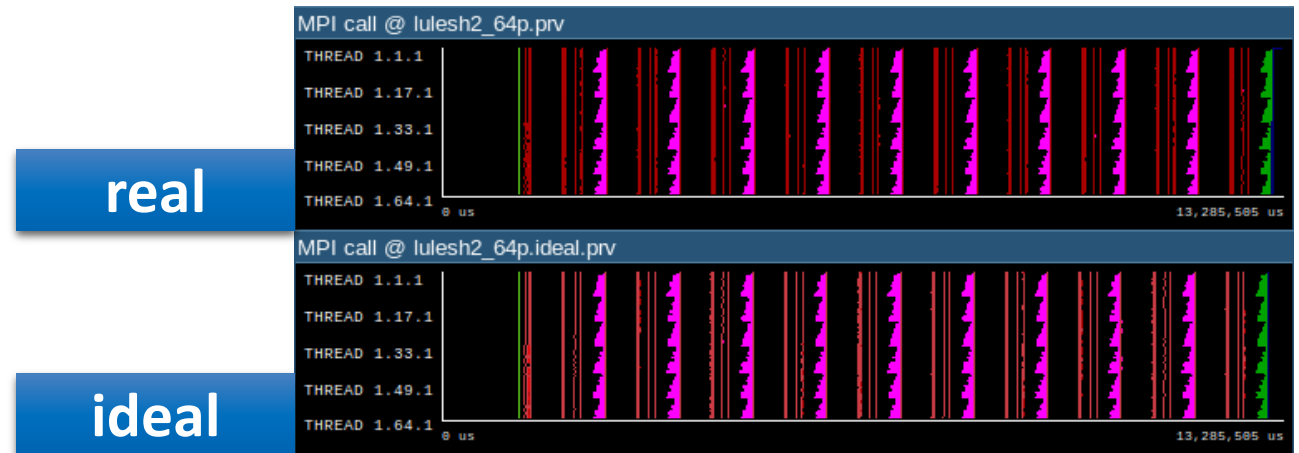
@ your computer

```
➤ scp <USER>@lxlogin8.lrz.de:$HOME/tools-material/dimemas/ \  
  *.{pcf,prv,row} $HOME
```



# Simulating with Dimemas

- Improves?
  - Compare the original and simulated trace with Paraver
    - Hints -> MPI -> MPI profile
      - Open control window
    - Copy and paste time from real to ideal



- No improvement: Not limited by network → Check serializations

## BSC Tools Hands-On

---

---

Judit Giménez, Lau Mercadal ([lau.mercadal@bsc.es](mailto:lau.mercadal@bsc.es))

Barcelona Supercomputing Center

---

---