



# Performance Tools Hands-On

PAT – Oct/2016



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*



[tools@bsc.es](mailto:tools@bsc.es)

# Extrae features

## Parallel programming models

- MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Java, Python...

## Platforms

- Intel, Cray, BlueGene, MIC, ARM, Android, Fujitsu Sparc...

## Performance Counters

- Using PAPI interface

## Link to source code

- Callstack at MPI routines
- OpenMP outlined routines
- Selected user functions (Dyninst)

## Periodic sampling

## User events (Extrae API)

**No need  
to  
recompile  
/ relink!**

# Extrae overheads

	Average values	Mistral
Event	150 – 200 ns	167 ns
Event + PAPI	750 ns – 1 us	4.7 us
Event + callstack (1 level)	600 ns	626 ns
Event + callstack (6 levels)	1.9 us	2 us

# How does Extrae work?

## « Symbol substitution through LD\_PRELOAD

- Specific libraries for each combination of runtimes
  - MPI
  - OpenMP
  - OpenMP+MPI
  - ...

**Recommended**



## « Dynamic instrumentation

- Based on Dyninst (developed by U.Wisconsin/U.Maryland)
  - Instrumentation in memory
  - Binary rewriting

## « Alternatives

- Static link (i.e., PMPI, Extrae API)

# 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 it!

☞ For further reference check the **Extrae User Guide:**

- Also distributed with Extrae at `$EXTRAE_HOME/share/doc`
- <http://www.bsc.es/computer-sciences/performance-tools/documentation>

# Log in to Mistral

@ your laptop

```
> ssh -Y <USER>@mistral.dkrz.de
```

☞ The following directory in your home folder contains all the examples:

@ mistral.dkrz.de

```
> cp ~k203109/tools-material $HOME  
  
> ls -l $HOME/tools-material  
.. clustering/  
.. extrae/  
.. paraver/  
.. slides/ ←  
.. traces/
```

Here you have a copy of this slides.  
You can copy them to your laptop  
or open remotely with:  
> evince slides/Tools-Hands-On.pdf

# Installing ICON

```
> cd $HOME  
> tar xfzv icon-dev.tgz  
> cd icon-dev  
> module add intel/15.0.6  
> module add mxm/3.3.3002  
> module add fca/2.5.2393  
> module add bullxmpi_mlx/bullxmpi_mlx-1.2.8.3  
> ./configure --with-fortran=intel  
> ./build_command  
> ./make_runscripts atm_amip
```

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

@ mistral.dkrz.de

```
> cd $HOME/icon-dev/run  
> cp exp.atm_amip.run exp.atm_amip.run.extrae  
> vi exp.atm_amip.run.extrae
```

## exp.atm\_amip.run.extrae

```
#SBATCH --account =kg0166  
#SBATCH --job-name=exp.atm_amip.run  
#SBATCH --workdir=/home/dkrz/k203109/icon-dev/run  
#SBATCH --nodes=4  
#SBATCH --threads-per-core=2  
#SBATCH --output=LOG.exp.atm_amip.run.%j.o  
#SBATCH --error=LOG.exp.atm_amip.run.%j.e  
#SBATCH --exclusive  
#SBATCH --time=00:30:00
```

```
...  
end_date="1979-01-01T23:50:00Z"
```

```
...  
${START} ${MODEL}
```

```
...
```

Line 147: Reduce the simulation time

Line 820  
(srun app)



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

## Copy Extrae files to experiment directory

@ mistral.dkrz.de

```
> cp $HOME/tools-material/extrae/* $HOME/icon-dev/run
```

### exp.atm\_amip.run.extrae

```
#SBATCH --account=kg0166
#SBATCH --job-name=exp.atm_amip.run
#SBATCH --workdir=/home/dkrz/k203109/icon-dev/run
#SBATCH --nodes=4
#SBATCH --threads-per-core=2
#SBATCH --output=LOG.exp.atm_amip.run.%j.o
#SBATCH --error=LOG.exp.atm_amip.run.%j.e
#SBATCH --exclusive
#SBATCH --time=00:30:00
...
TRACE=${basedir}/run/trace.sh
EXTRAE_CONFIG=${basedir}/run/extrae.xml
cp ${TRACE} .
cp ${EXTRAE_CONFIG} .
${START} ${TRACE} ${MODEL}
...
```

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

@ mistral.dkrz.de

```
> vi $HOME/icon-dev/run/trace.sh
```

## exp.atm\_amip.run.extrae

```
#SBATCH --account =kg0166
#SBATCH --job-name=exp.atm_amip.run
#SBATCH --workdir=/home/dkrz/k203109/icon-dev/run
#SBATCH --nodes=4
#SBATCH --threads-per-core=2
#SBATCH --output=LOG.exp.atm_amip.run.%j.o
#SBATCH --error=LOG.exp.atm_amip.run.%j.e
#SBATCH --exclusive
#SBATCH --time=00:30:00
...
TRACE=${basedir}/run/trace.sh
EXTRAE_CONFIG=${basedir}/run/extrae.xml
cp ${TRACE} .
cp ${EXTRAE_CONFIG} .
${START} ${TRACE} ${MODEL}
...
```

## trace.sh

```
#!/bin/bash

# Configure Extrae
export EXTRAE_HOME=/sw/.../extrae-3.4.1-bullxmpi-intel14
source ${EXTRAE_HOME}/etc/extrae.sh

export TRACE_NAME=icon.prv
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/libmpitracef.so

# Run the program
$*
```

# Step 1: Which tracing library?

☞ Choose depending on the application type

Library	Serial	MPI	OpenMP	pthread	CUDA
libseqtrace	✓				
libmpitrace[f] <sup>1</sup>		✓			
libomptrace			✓		
libpttrace				✓	
libcudatrace					✓
libompitrace[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 it!

## « Submit your job

@ mistral.dkrz.de

```
> cd $HOME/icon-dev/run  
> sbatch exp.atm_amip.run.extrae
```

## « Easy! 😊

# Step 2: Extrae XML configuration

@ mistral.dkrz.de

```
> vi $HOME/icon-dev/run/extrae.xml
```

```
<mpi enabled="yes">  
  <counters enabled="yes" />  
</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 MPI calls + HW counters

Trace call-stack events @  
MPI calls

# Step 2: Extrae XML configuration (II)

@ mistral.dkrz.de

```
> vi $HOME/icon-dev/run/extrae.xml
```

```
<counters enabled="yes">
  <cpu enabled="yes" starting-set-distribution="cyclic">
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_L1_DCM, PAPI_L3_TCM,
      PAPI_BR_INS, PAPI_BR_MSP
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_SR_INS, PAPI_LD_INS,
      RESOURCES_STALLS:RS, RESOURCE_STALLS:SB
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, RESOURCE_STALLS:ROB, PAPI_L2_DCM
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      ...
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      ...
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>
```

Select which HW  
counters are  
measured

# Step 2: Extrae XML configuration (III)

@ mistral.dkrz.de

```
> vi $HOME/icon-dev/run/extrae.xml
```

```
<buffer enabled="yes">
  <size enabled="yes">5000000</size>
  <circular enabled="no" />
</buffer>

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

<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>
```

**Trace buffer size**

**Enable sampling**

**Merge intermediate files into Paraver trace**

# All done! Check your resulting trace

☞ Once finished (check with “`squeue`”) you will have the trace (3 files):

@ [mistral.dkrz.de](mailto:mistral.dkrz.de)

```
> cd $HOME/icon-dev/experiments/atm_amip
> ls -l
...
icon.pcf
icon.prv
icon.row
```

☞ Compress the trace (takes a little while)

@ [mistral.dkrz.de](mailto:mistral.dkrz.de)

```
> gzip icon.prv
```

☞ Any trouble? Traces already generated here:

@ [mistral.dkrz.de](mailto:mistral.dkrz.de)

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



# Installing Paraver

⌘ Download the Paraver binaries to your laptop

@ your laptop

```
> scp <USER>@mistral.dkrz.de:tools-packages/<VERSION> $HOME
```

Pick your version



Linux 64 bits

```
wxparaver-4.6.2-linux-x86_64.tar.gz
```

Linux 32 bits

```
wxparaver-4.6.2-linux-x86_32.tar.gz
```

Mac

```
wxparaver-4.6.2-mac.zip
```

Windows

```
wxparaver-4.6.2-win.zip
```

# Installing Paraver (II)

## Uncompress the package into your home directory

@ your laptop

```
> tar xvfz wxparaver-4.6.2-linux-x86_64.tar.gz -C $HOME  
> ln -s $HOME/wxparaver-4.6.2-linux-x86_64 $HOME/paraver
```

## Download Paraver tutorials and uncompress into the Paraver directory

@ your laptop

```
> scp <USER>@mistral.dkrz.de:  
    tools-packages/paraver-tutorials-20150526.tar.gz $HOME  
> tar xvfz $HOME/paraver-tutorials-20150526.tar.gz -C $HOME/paraver
```

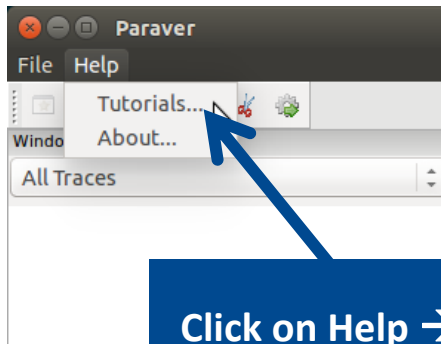
# Check that everything works

## Start Paraver

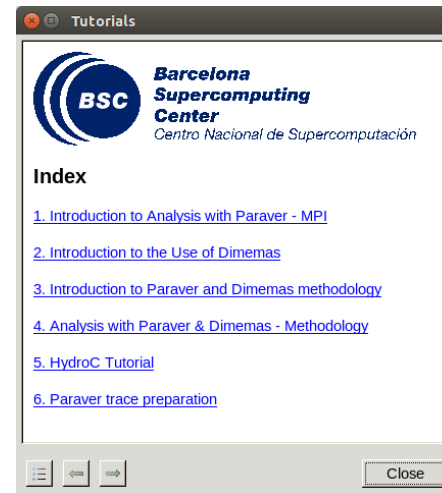
@ your laptop

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

## Check that tutorials are available



Click on Help → Tutorials



## Trouble installing locally? Remote open from Mistral

@ mistral.dkrz.de

```
> ssh -Y <USER>@mistral.dkrz.de  
> cd /sw/rhel6-x64/analysis-tools/wxparaver-4.6.1/bin  
> ./wxparaver
```

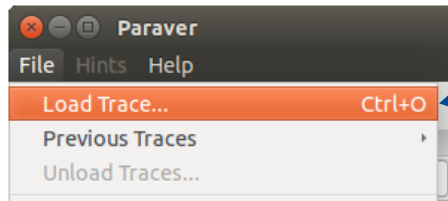
# First steps of analysis

- Copy the trace to your laptop

@ your laptop

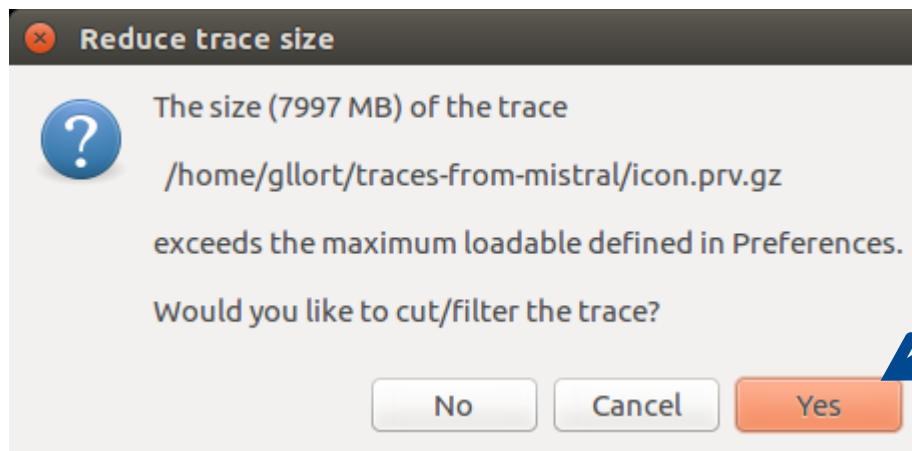
```
> scp <USER>@mistral.dkrz.de:icon-dev/experiments/atm_amip/icon.* $HOME
```

- Load the trace with Paraver



Click on File → Load Trace → Browse to "icon.prv.gz"

- Trace is big: Filter it



Click on "Yes"

# Filter the trace

## What to filter?

- Keep only long computations and flushing events
- Copy this configuration from Mistral

```
> scp <USER>@mistral.dkrz.de:tools-material/paraver/filter.xml $HOME
```

**Traces**

Input:

Output:

Load the processed trace  
 Run application with the processed trace

**Cut/Filter Parameters**

Configuration file:

**Execution chain**

1.- Cutter  
 2.- Filter  
 3.- Software Counters

**Trace Limits**

Cut by time    Begin:   
 Cut by time %    End:

Tasks:

**Trace Options**

Use original time     Remove first state  
 Don't break states     Remove last state

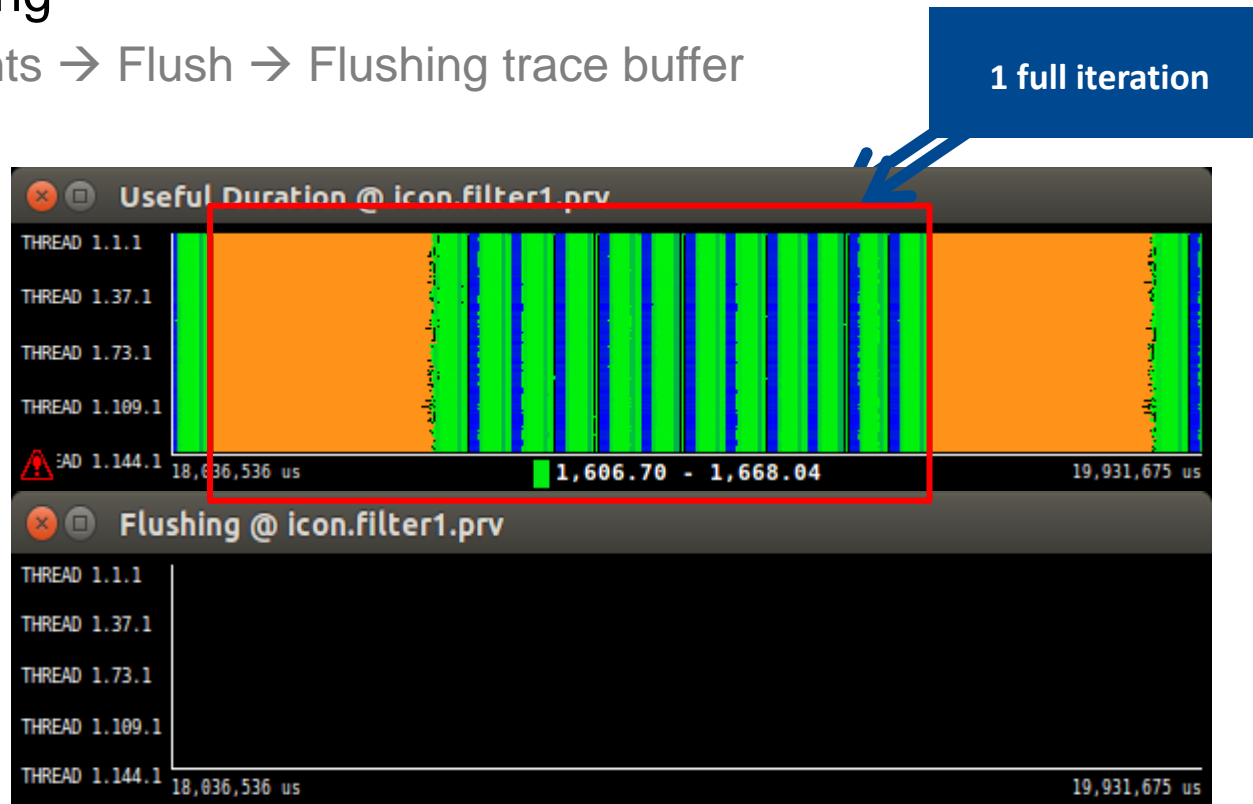
1. Click on "Browse"  
2. Select "filter.xml"

3. Click on "Apply"

# Find a representative region

## ⌘ Load views:

- Useful duration
  - File → Load Configuration → “cfgs/General/views/useful\_duration.cfg”
- Flushing
  - Hints → Flush → Flushing trace buffer



# Cut the trace (I)

Zoom the time interval to cut in the timeline → 1 iteration



# Cut the trace (II)

- Get a subtrace that contains all events only for this iteration
  - Right click → Run → Cutter

**Traces**

Input: icon.filter1.prv

Output: icon.filter1.chop1.prv

Load the processed trace  
 Run application with the processed trace

**Cut/Filter Parameters**

Configuration file:

**Execution chain**

- 1.- Cutter
- 2.- Filter
- 3.- Software Counters

**Cutter** | Filter | Software Counters

**Trace Limits**

Cut by time      Begin: 18128087270

Cut by time %      End: 19459267288

Tasks:

**Trace Options**

Use original time       Remove first state

Don't break states       Remove last state

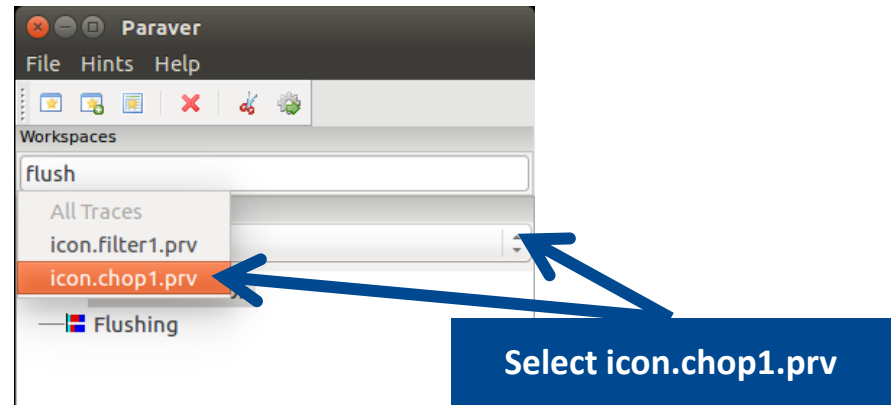
1. Click on "Browse" and select the original (big) trace: icon.prv.gz

2. Click on "Apply"

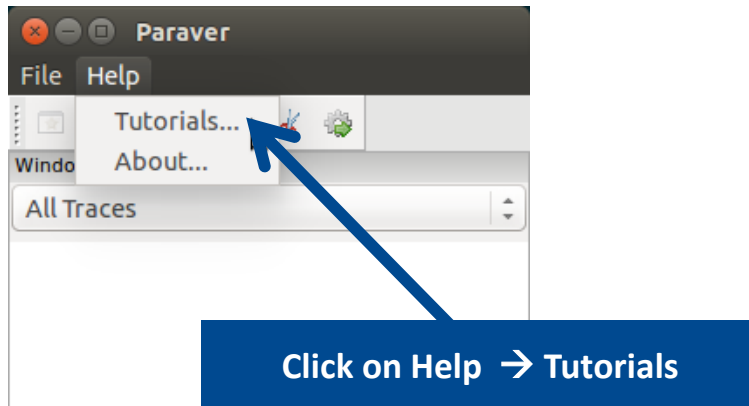


# First steps of analysis

☞ Select the generated cut



☞ Follow Tutorial #3



# Measure the parallel efficiency

## Click on “mpi\_stats.cfg”

- Check the Average for the column labeled “Outside MPI”

**To measure the parallel efficiency** load the configuration file `cfgs/mpi/mpi_stats.cfg`. This configuration pops up a table with %time of 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 recommended to look at the corresponding metric in detail. Control window to identify the phases and iterations of the code.

- **To measure the computation time distribution** load the configuration file `cfgs/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 a call and the entry to the next call. If the histogram does not show vertical lines, it indicates the computation time may be not be distributed. Open the control window to look at the time distribution and correlate both views.
- **To measure the computational load (instructions) distribution** load the configuration file `cfgs/general/2dh_instructions.cfg`. This configuration pops up a histogram of the instructions for the computation regions. The computation regions are delimited by the exit from a call and the entry to the next call. If the histogram does not show vertical lines, it indicates the computational load may be not be distributed. Open the control window to look at the instructions distribution and correlate both views.

THREAD	1.134.1	1.135.1	1.136.1	1.137.1	1.138.1	1.139.1	1.140.1	1.141.1	1.142.1	1.143.1	1.144.1	Total	Average	Maximum	Minimum	StDev	Avg/Max
	79.79%	79.87%	80.95%	80.89%	82.13%	81.31%	80.24%	78.91%	81.43%	81.66%	82.69%	11,550.23%	80.21%	84.43%	77.62%	1.28%	0.95
	7.28%	5.07%	5.11%	5.37%	4.82%	7.57%	5.51%	5.99%	6.46%	4.94%	4.67%	890.42%	6.18%	10.10%	4.18%	1.08%	0.61
	3.29%	3.22%	3.46%	3.51%	3.63%	2.89%	3.26%	3.04%	3.76%	3.20%	3.26%	489.74%	3.40%	4.15%	2.18%	0.39%	0.82
	4.13%	4.81%	4.87%	4.45%	3.95%	2.39%	5.60%	5.23%	2.67%	4.36%	3.89%	634.53%	4.41%	8.75%	1.01%	1.49%	0.50
	5.03%	5.01%	5.20%	5.02%	5.07%	5.40%	4.98%	5.23%	5.27%	5.07%	5.10%	726.98%	5.05%	5.40%	0.46%	0.40%	0.93
	0.08%	1.58%	0.09%	0.43%	0.07%	0.11%	0.08%	1.20%	0.07%	0.41%	0.08%	51.66%	0.36%	2.87%	0.07%	0.49%	0.12
	0.2	0.2	0.1	0.1	0.1	0.1	0.1	0.2	0.1	0.1	0.1	33.1	0.2	0.4	0.0	0.0	0.0



# Cluster-based analysis

- Run the clustering tool on the trace you have generated
  - To avoid copying the cut trace back to Mistral, use a prepared cut

@ mistral.dkrz.de

```
> cd $HOME/tools-material/clustering  
> ./clusterize.sh ../traces/icon.chop1.prv.gz
```

# Cluster-based analysis (II)

## Check the clustering scatter plot

@ mistral.dkrz.de

```
> gnuplot icon.chop1.clustered.IPC.PAPI_TOT_INS.gnuplot
```

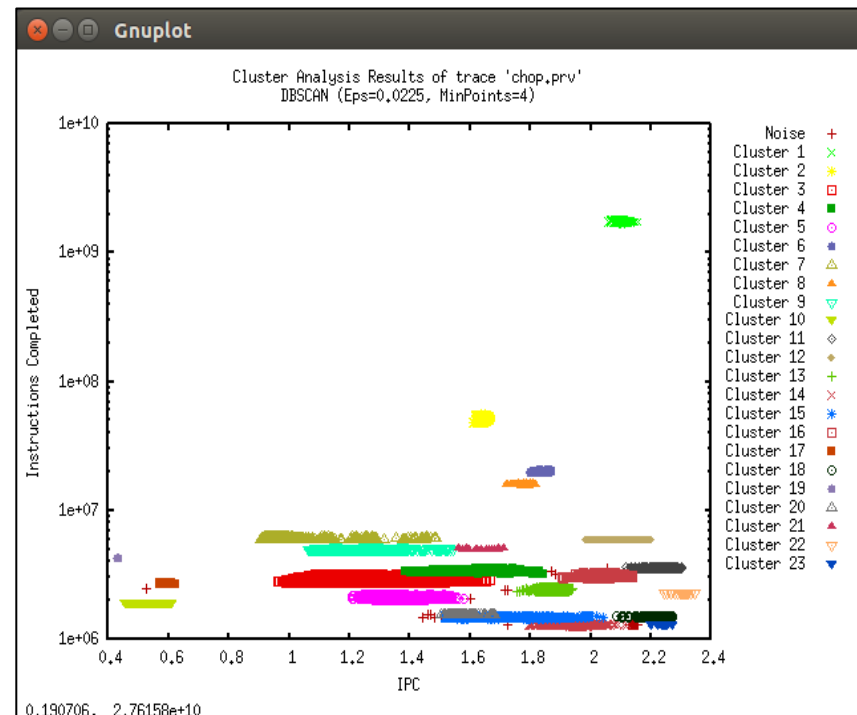
- Press “L” (once) to switch to logarithmic scale

## Identify main computing trends

## Work (Y) vs. Performance (X)

## See the horizontal clusters?

- Large IPC variability
- Indicate potential imbalances



# Cluster-based analysis (III)

## ⌘ Check the clustered trace

- Copy the clustered trace to your laptop

@ your laptop

```
> scp <USER>@mistral.dkrz.de:tools-  
material/clustering/icon.chop1.clustered.* $HOME
```

- Load with Paraver

@ your laptop

```
> $HOME/paraver/bin/wxparaver $HOME/icon.chop1.clustered.prv.gz
```

- Display the distribution of clusters over time

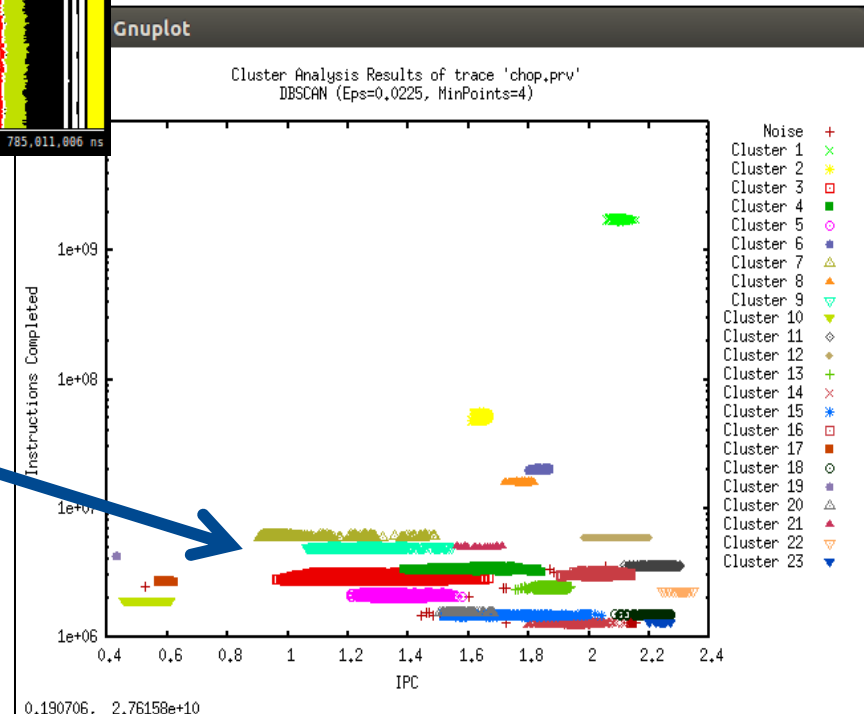
- File → Load configuration
- Select: \$HOME/paraver/cfgs/clustering/clusterID\_window.cfg

# Cluster-based analysis (IV)

Correlate scatter plots & timelines to detect imbalances



Constant work  
+  
Variable performance  
+  
Simultaneously @ different processes  
=  
Imbalances







**Thank you!**



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

 [tools@bsc.es](mailto:tools@bsc.es)