

BSC Tools Hands-On

Lau Mercadal, Judit Giménez
(tools@bsc.es)
Barcelona Supercomputing Center

Getting a trace with Extrae

Extrae features

- Platforms
 - Intel, Cray, BlueGene, Intel MIC, ARM, Android, Fujitsu Sparc...
- Parallel programming model
 - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Java, Python...
- Performance Counters
 - Using PAPI interface
- Link to source code
 - Callstack at MPI routines
 - OpenMP outlined routines
 - Selected user functions
- Periodic samples
- User events (Extrae API)

**No need to
recompile
or relink!**

Extrae overheads

	Average values	Romeo
Event	150 – 200ns	133ns
Event + PAPI	750 – 1000ns	1000ns
Event + callstack (1 level)	1 μ s	2 μ s
Event + callstack (6 levels)	2 μ s	5 μ s

How does Extrae work?

- Symbol substitution through LD_PRELOAD
 - Specific libraries for each combination of runtimes
 - MPI
 - OpenMP
 - OpenMP+MPI
 - ...
- 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 your job submission script

2. Configure what to trace

- XML configuration file
- Example configurations at \$EXTRAE_HOME/share/example


3. Run it!

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

- <https://tools.bsc.es/sites/default/files/documentation/html/extrae/index.html>
- Also distributed with Extrae at \$EXTRAE_HOME/share/doc

Login to Romeo and copy the examples

```
laptop> ssh -Y <USER>@romeologin1.univ-reims.fr  
  
romeo> cp -r /home/laumercadal/tools-material $HOME  
  
romeo> ls $HOME/tools-material  
... apps/  
... clustering/  
... extrae/  
... slides/ ←  
... traces/
```



Here you have
a copy of this slides

Step 1: Adapt the job script to load Extrae with LD_PRELOAD

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

```
#!/usr/bin/env bash
```

```
#SBATCH --job-name=lulesh2.0_27p  
#SBATCH --output=lulesh2.0_27p.out  
#SBATCH --error=lulesh2.0_27p.err  
#SBATCH --ntasks=27  
#SBATCH --time=00:10:00
```

**Request
resources**

```
module load gcc/8.1.0 openmpi/2.0.4.1.1_gnu
```

**Select MPI
version**

```
export OMP_NUM_THREADS=1
```

```
srunk ../apps/lulesh2.0 -i 10 -s 65 -p
```

**Run the
program**

Step 1: Adapt the job script to load Extrae with LD_PRELOAD

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

```
#!/usr/bin/env bash

#SBATCH --job-name=lulesh2.0_27p
#SBATCH --output=lulesh2.0_27p.out
#SBATCH --error=lulesh2.0_27p.err
#SBATCH --ntasks=27
#SBATCH --time=00:10:00

module load gcc/8.1.0 openmpi/2.0.4.1.1_gnu
module load extrae/3.6.1

export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh2.0_27p.prv

srun ./trace.sh ../apps/lulesh2.0 -i 10 -s 65
```

Load Extrae module

Name of the
resulting trace

Activate Extrae
during the run

Step 1: Adapt the job script to load Extrae with LD_PRELOAD

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

```
#!/usr/bin/env bash

#SBATCH --job-name=lulesh2.0_27p
#SBATCH --output=lulesh2.0_27p.out
#SBATCH --error=lulesh2.0_27p.err
#SBATCH --ntasks=27
#SBATCH --time=00:10:00

module load gcc/8.1.0 openmpi/2.0.4.1.1_gnu
module load extrae/3.6.1

export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh2.0_27p.prv

srun ./trace.sh ./apps/lulesh2.0 -i 10 -s 65
```

```
#!/usr/bin/env bash

# Configure Extrae
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/libmpitracecf.so

# Run the program
$*
```

Select
"what to trace"

Select your
type of application

Step 1: Which tracing library?

- Choose depending on the application type

Library	Serial	MPI	OpenMP	pthread	CUDA
libseqtrace	✓				
libmpitrace[f] ¹		✓			
libompitrace			✓		
libpttrace				✓	
libcudatrace					✓
libompitrace[f] ¹		✓	✓		
libptmpitrace[f] ¹		✓		✓	
libcudampitrace[f] ¹		✓			✓

¹ include suffix "f" in Fortran codes

Step 3: Run it!

- Submit your job

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

- Once finished the trace will be in the same folder: `lulesh2.0_27p.{pcf,prv,row}`
 - Check the status of your job with:

```
romeo> squeue -u $USER
```

- Any issue?
 - Already generated at `$HOME/tools-material/traces`

Step 2: Extrae XML configuration

```
romeo> vi $HOME/tools-material/extrae/extrae.xml
```

```
<mpi enabled="yes">  
  <counters enabled="yes" />  
</mpi>  
  
<openmp enabled="no">  
  <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 the MPI calls
(What's the program doing?)**

**Trace the call-stack
(Where in my code?)**

Compile with debug!(-g)

Step 2: Extrae XML configuration (II)

```
romeo> vi $HOME/tools-material/extrae/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
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_BR_MSP, RESOURCE_STALLS, PAPI_SR_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_L2_DCM, PAPI_LD_INS
    </set>
  </cpu>
  <network enabled="no" />
  <resource-usage enabled="no" />
  <memory-usage enabled="no" />
</counters>
```

**Select which HW counters
are measured**

(How's the machine doing?)

Step 2: Extrae XML configuration (III)

```
romeo> vi $HOME/tools-material/extrae/extrae.xml
```

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

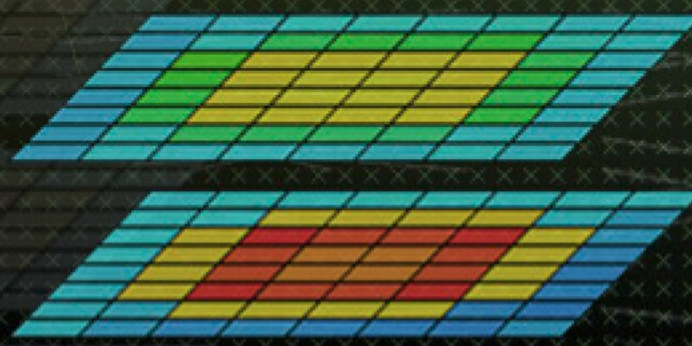
**Trace buffer size
(Flush/memory trade-off)**

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

**Enable sampling
(Want more details?)**

```
<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>
```

**Automatic
post-processing
to generate the
Paraver trace**



Installing Paraver & First analysis steps





Install Paraver in your laptop

- Download from <http://tools.bsc.es/downloads>

Pick your version

- Also available @Romeo
 - /home/laumercadal/tools-packages



-  wxparaver-4.7.2-win.zip
-  wxparaver-4.7.2-mac.zip
-  wxparaver-4.7.2-Linux_i686.tar.gz (32-bits)
-  wxparaver-4.7.2-Linux_x86_64.tar.gz (64-bits)

```
laptop> scp <USER>@romeologin1.univ-remis-fr: \
/home/laumercadal/tools-packages/<PACKAGE> $HOME
```

Install Paraver (II)

- Download tutorials:
 - Documentation -> Tutorial guidelines

Download links

- Also available @Romeo
 - /home/laumercadal/tools-packages

The screenshot shows the Paraver documentation page. The navigation bar includes Home, Paraver, Dimemas, Extrae, Research, Documentation, Downloads, and Publications. The main content area is titled 'Paraver tutorials' and lists seven tutorials. Two items are circled in blue: 'Dimemas introduction' and the download options at the bottom. A blue box labeled 'Download links' has arrows pointing to these circled items. A yellow box contains the text 'Also available @Romeo' and the path '/home/laumercadal/tools-packages'.

Tutorial Name	Description
Paraver introduction (MPI)	Start here to familiarize with Paraver basic commands and the first steps of a performance analysis.
Dimemas introduction	The basic steps to learn how to configure and run the Dimemas simulator and to start looking at the results.
Introduction to Paraver and Dimemas methodology	This tutorial presents different ways to analyze a MPI application through well-known rules, their diagnosis and how they impact on your exploration (no traces included).
Methodology	This tutorial shows some examples of the analysis that can be done using the provided configuration files.
Tutorial on HydroC analysis (MPI, Dimemas, CUDA)	One example of performance analysis of the MPI application Hydro and further simulations with Dimemas.
Trace preparation	Look at this tutorial to select a representative region for a large trace that cannot be loaded into memory.
Trace alignment tutorial	If you identify some unexpected unalignment or backwards communications, use this tutorial to learn how to correct shifts between processors.

If you prefer you can download them together in a single package:

- .tar.gz format (127 Mb)
- .zip format (127 Mb)

```
laptop> scp <USER>@romeologin1.univ-reims.fr:  
/home/laumercadal/tools-packages/paraver-tutorials-20150526.tar.gz $HOME
```

Uncompress, rename & move

▪ Paraver

```
laptop> tar xf wxparaver-4.7.2-linux-x86_64.tar.gz  
laptop> mv wxparaver-4.7.2-linux-x86_64 paraver
```

▪ Tutorials

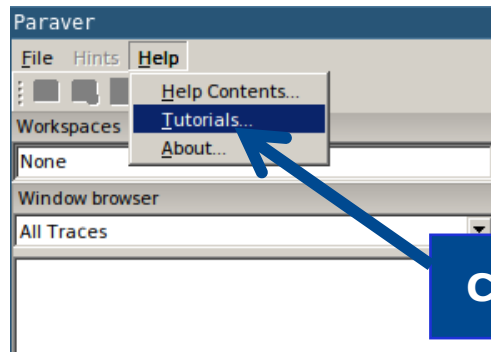
```
laptop> tar xf paraver-tutorials-20150526.tar.gz  
laptop> mv paraver-tutorials-20150526 paraver/tutorials
```

Check that everything works

- Start Paraver

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

- Check that tutorials are available



Click on Help → Tutorials

- Remotely available in Romeo

```
laptop> ssh -Y <USER>@romeologin1.univ-reims.fr  
romeo> /home/laumercadal/wxparaver/latest/bin/wxparaver
```

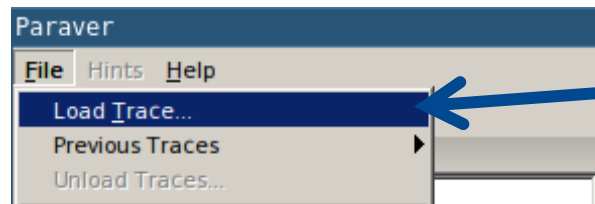


First steps of analysis

- Copy the trace to your laptop (All 3 files: *.prv, *.pcf, *.row)

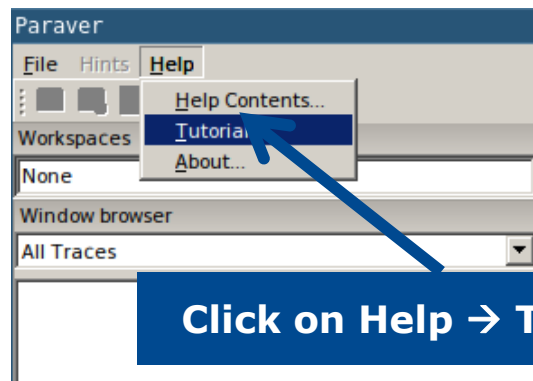
```
laptop> scp <USER>@romeologin1.univ-reims.fr:$HOME/tools-material/extrae/lulesh2.0* ./
```

- Load the trace

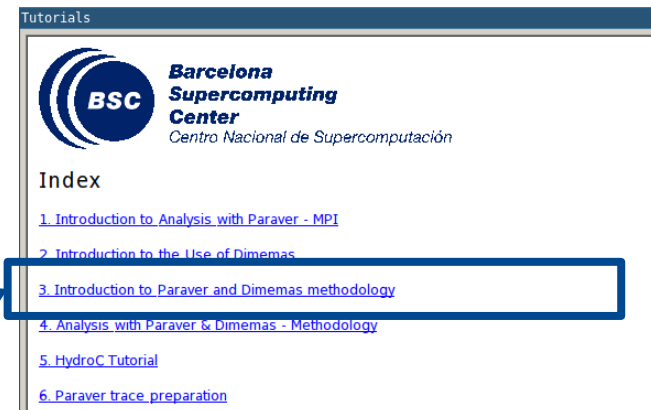


Click on File → Load Trace
→ Browse to the *.prv file

- Follow Tutorial #3



Click on Help → Tutorials



Measure the parallel efficiency

- Click on "mpi_stats.cfg"

Tutorials

The first question to answer when analyzing a parallel code is "how efficient does it run?". The efficiency of a parallel program can be defined based on two aspects: the parallelization efficiency and the efficiency obtained in the execution of the serial regions. These two metrics would be the first checks on the proposed methodology.

- To measure the parallel efficiency, load the configuration file `cfgs/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 table. `Avg/Max` 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 `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 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 serial...

Parallel efficiency

Comm efficiency

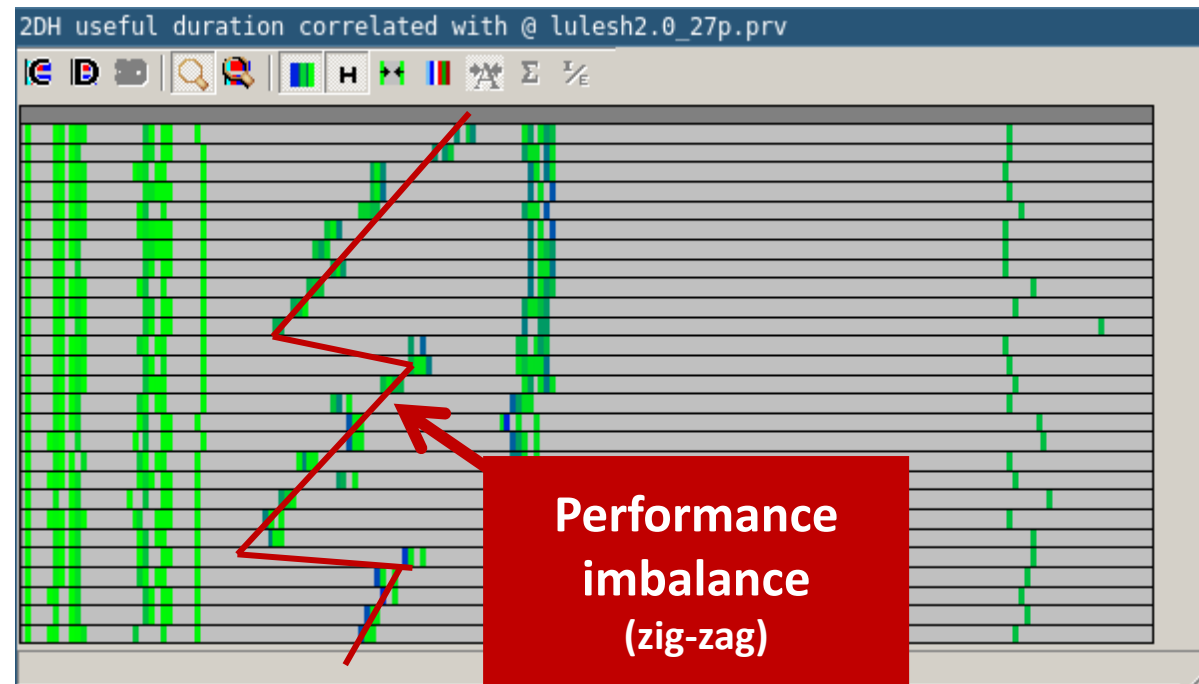
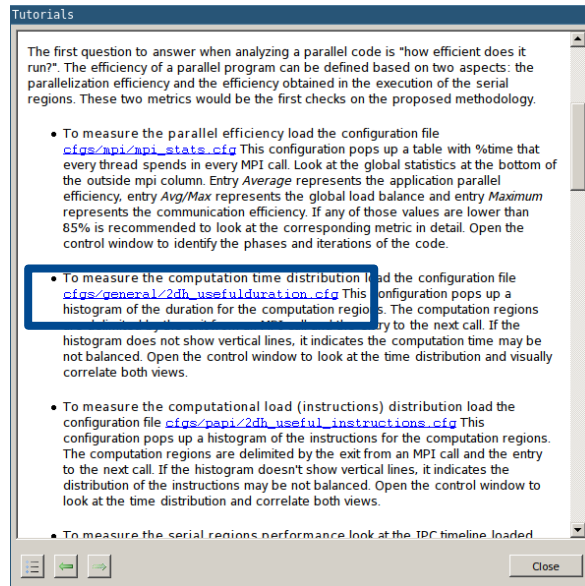
Load balance

MPI call profile @ lulesh2.0_27p.prv

THREAD 1.16.1	87.85 %	0.08 %	0.06 %	0.06 %	3.35 %	0.01 %	0.01 %	7.59 %
THREAD 1.17.1	88.44 %	0.14 %	0.08 %	0.08 %	2.67 %	0.01 %	0.45 %	7.58 %
THREAD 1.18.1	84.71 %	0.13 %	0.05 %	0.05 %	2.66 %	0.01 %	0.00 %	10.97 %
THREAD 1.19.1	86.84 %	0.08 %	0.04 %	0.14 %	3.53 %	0.02 %	0.01 %	8.27 %
THREAD 1.20.1	83.73 %	0.08 %	0.05 %	0.06 %	2.62 %	0.01 %	0.00 %	11.97 %
THREAD 1.21.1	81.52 %	0.07 %	0.04 %	0.25 %	2.91 %	0.02 %	1.14 %	13.50 %
THREAD 1.22.1	82.63 %	0.08 %	0.05 %	0.05 %	2.72 %	0.01 %	0.00 %	12.87 %
THREAD 1.23.1	92.78 %	0.12 %	0.07 %	0.07 %	2.84 %	0.01 %	0.00 %	3.57 %
THREAD 1.24.1	90.36 %	0.09 %	0.05 %	0.05 %	2.80 %	0.01 %	0.00 %	5.82 %
THREAD 1.25.1	90.38 %	0.07 %	0.03 %	0.29 %	2.75 %	0.02 %	0.01 %	5.64 %
THREAD 1.26.1	89.79 %	0.09 %	0.05 %	0.04 %	2.38 %	0.01 %	0.01 %	6.72 %
THREAD 1.27.1	88.33 %	0.08 %	0.03 %	1.55 %	2.01 %	0.02 %	0.01 %	7.01 %
Total	2,418.03 %	2.57 %	1.66 %	4.13 %	46.88 %	0.35 %	3.86 %	200.37 %
Parallel efficiency	89.56 %	0.10 %	0.06 %	0.15 %	1.74 %	0.01 %	0.14 %	7.42 %
Maximum	99.01 %	0.21 %	0.13 %	1.55 %	3.53 %	0.02 %	1.14 %	13.50 %
Minimum	81.52 %	0.04 %	0.03 %	0.04 %	0.37 %	0.00 %	0.00 %	0.01 %
StDev	4.28 %	0.03 %	0.02 %	0.28 %	1.08 %	0.01 %	0.32 %	3.46 %
Avg/Max	0.90	0.46	0.49	0.10	0.49	0.53	0.13	0.55

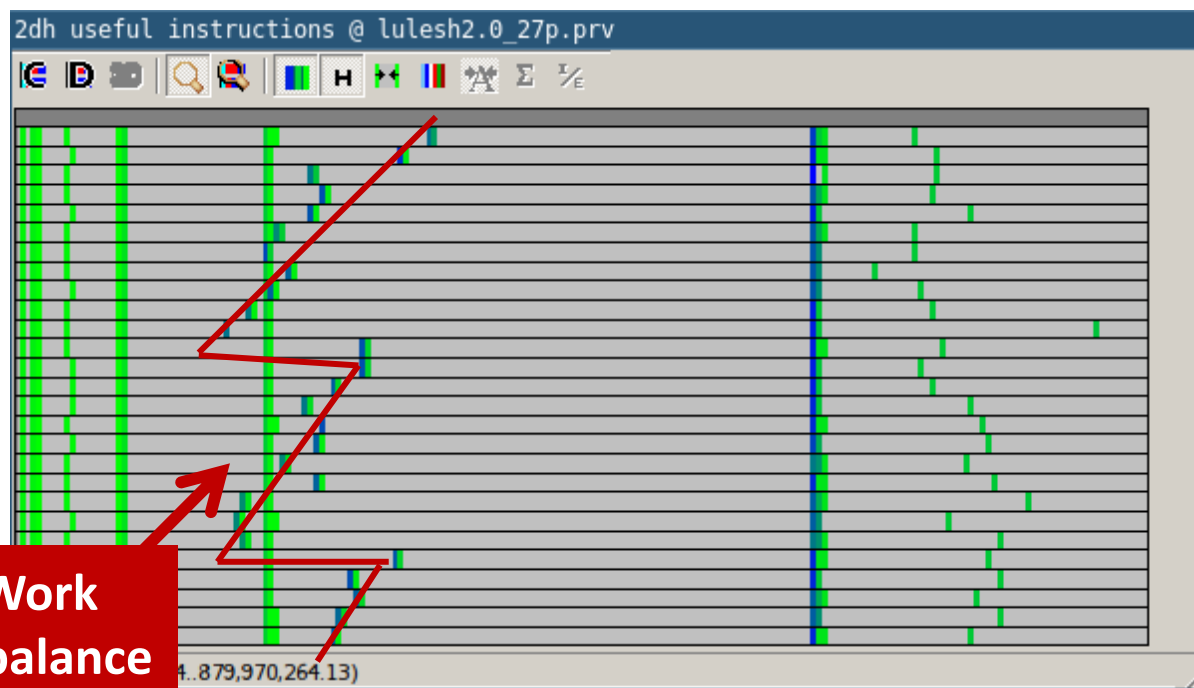
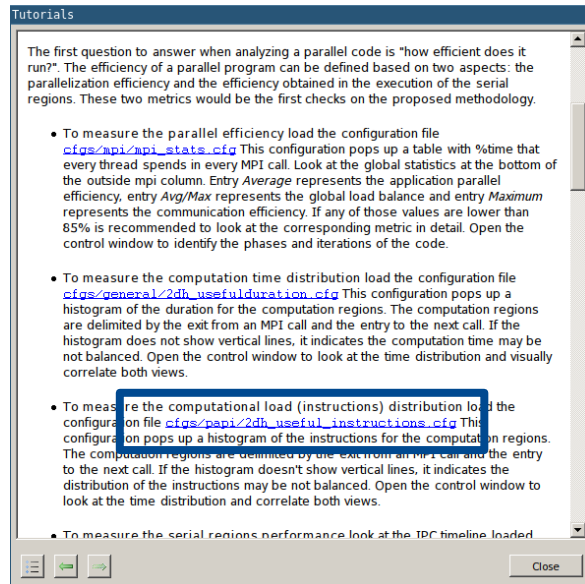
Computation time and work distribution

- Click on “2dh_usefulduration.cfg” (2nd link) → Shows **time computing**



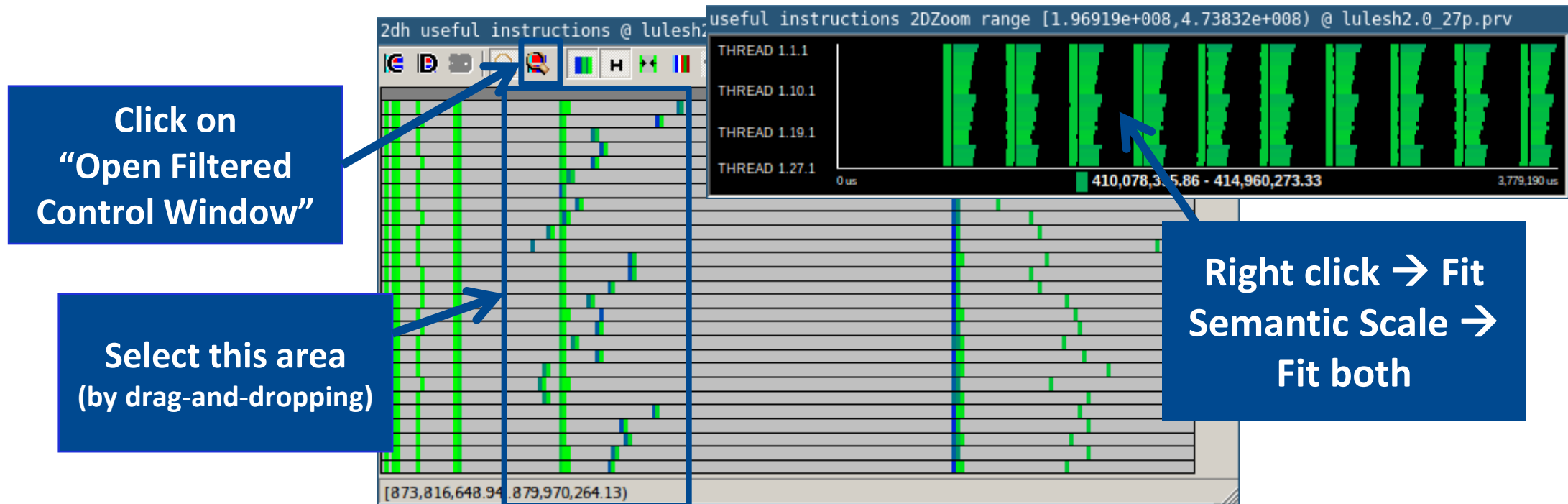
Computation time and work distribution

- ... and “2dh_useful_instructions.cfg” (3rd link) → Shows **amount of work**



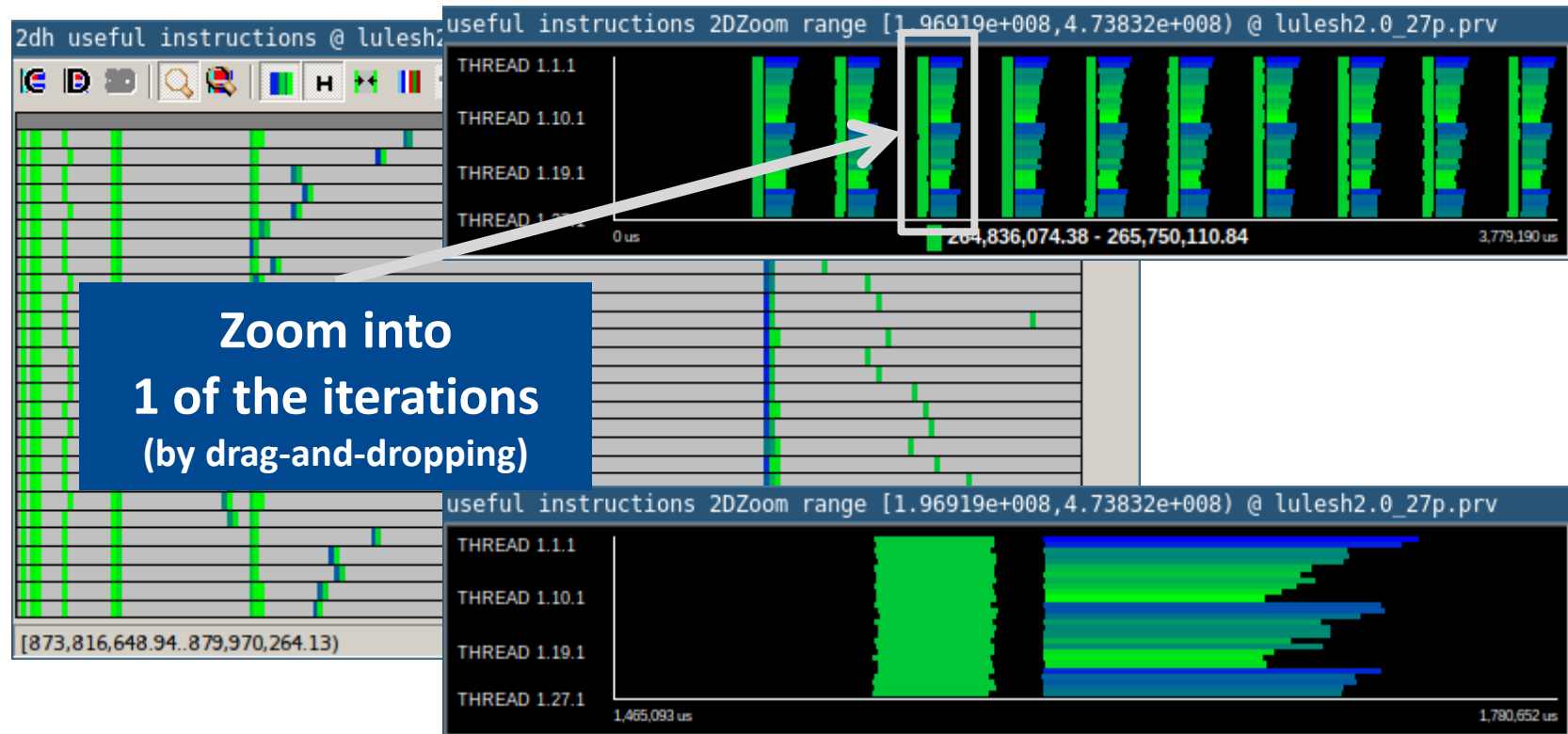
Where does this happen?

- Go from the table to the timeline



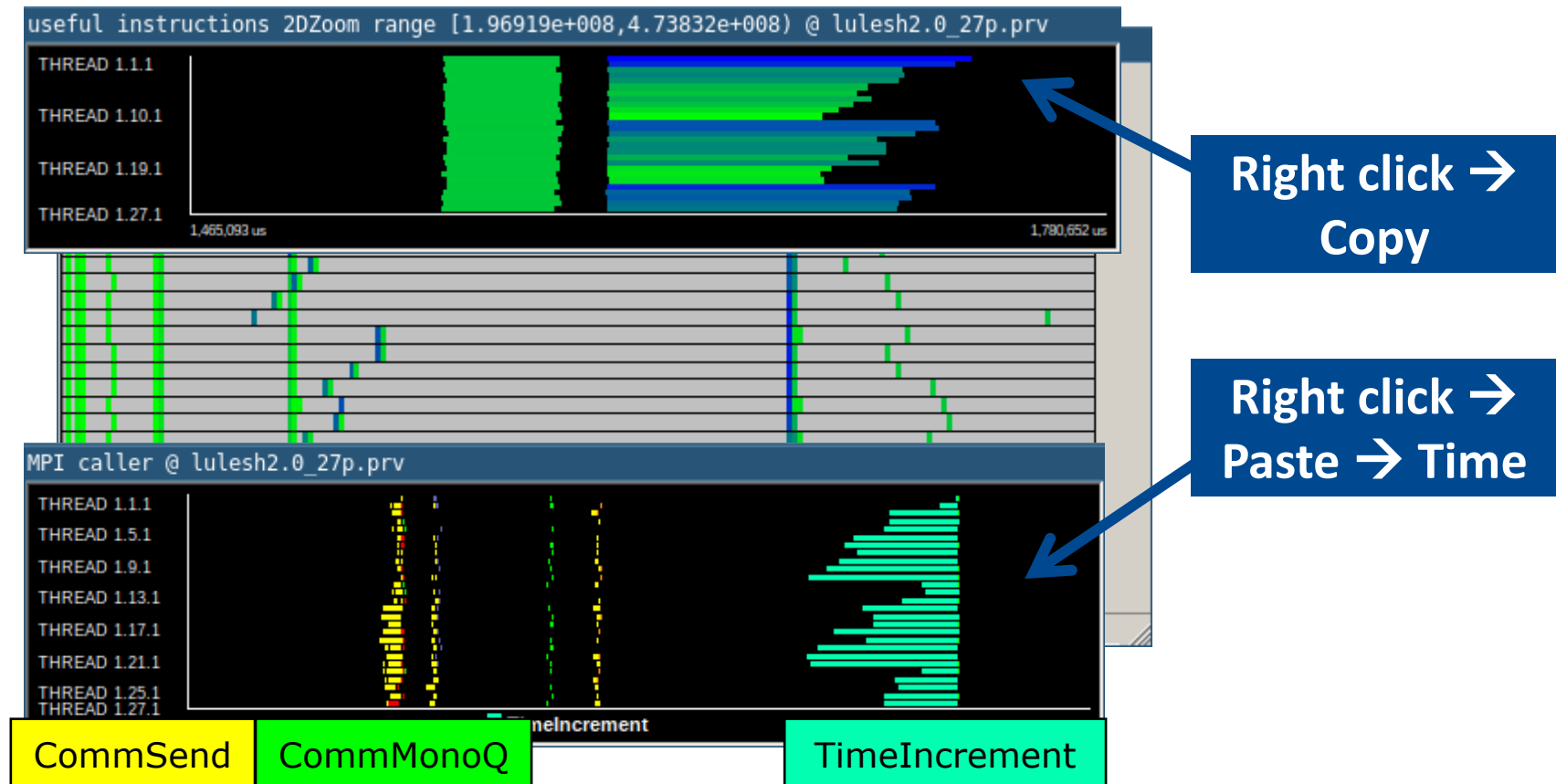
Where does this happen?

- **Slow** & **Fast** at the same time → Imbalance

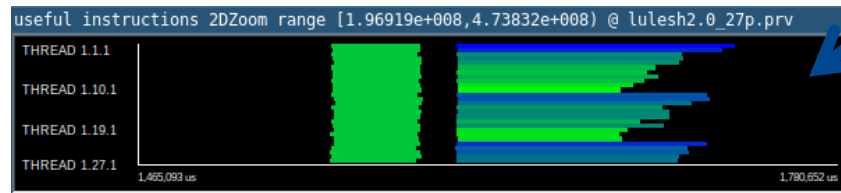


Where does this happen?

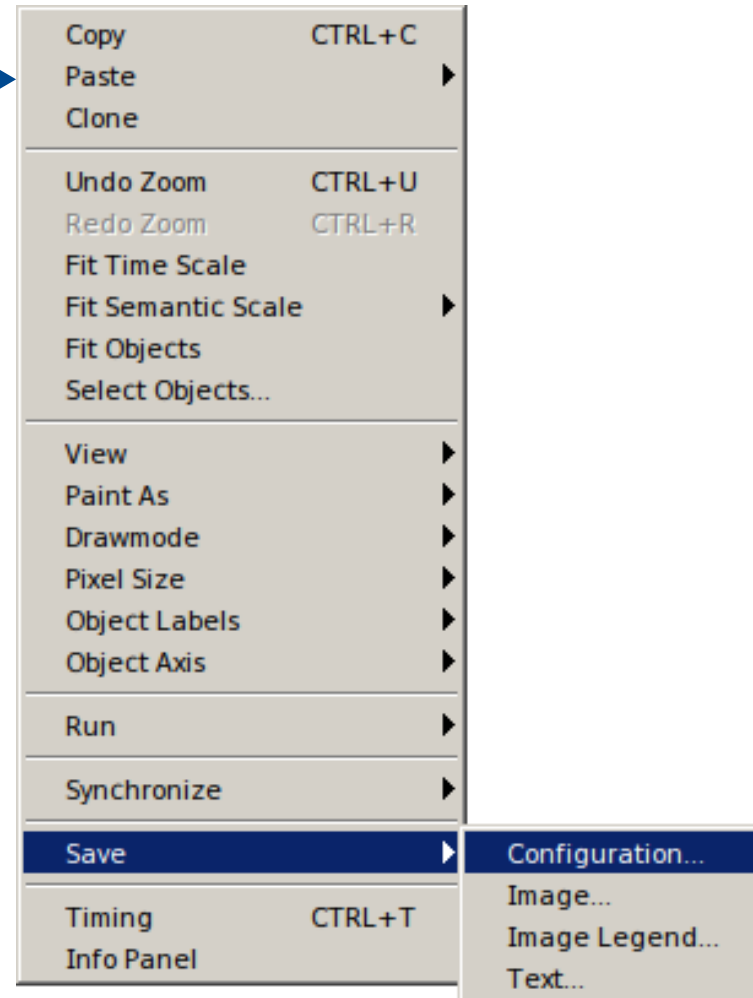
- Hints → Callers → Caller function



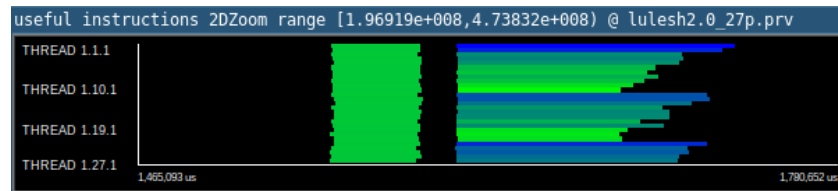
Save CFG's (2 methods)



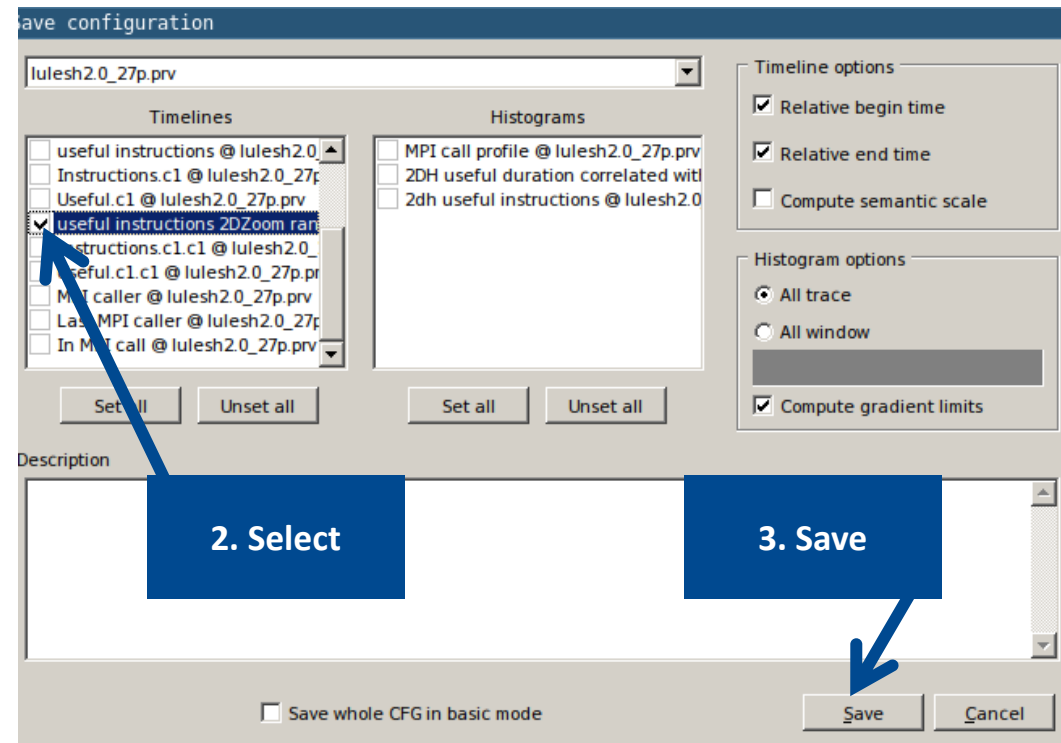
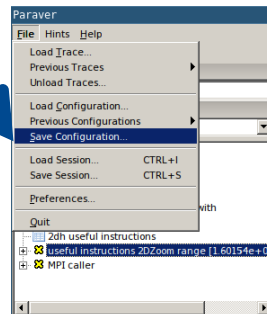
Right click on
timeline



Save CFG's (2 methods)

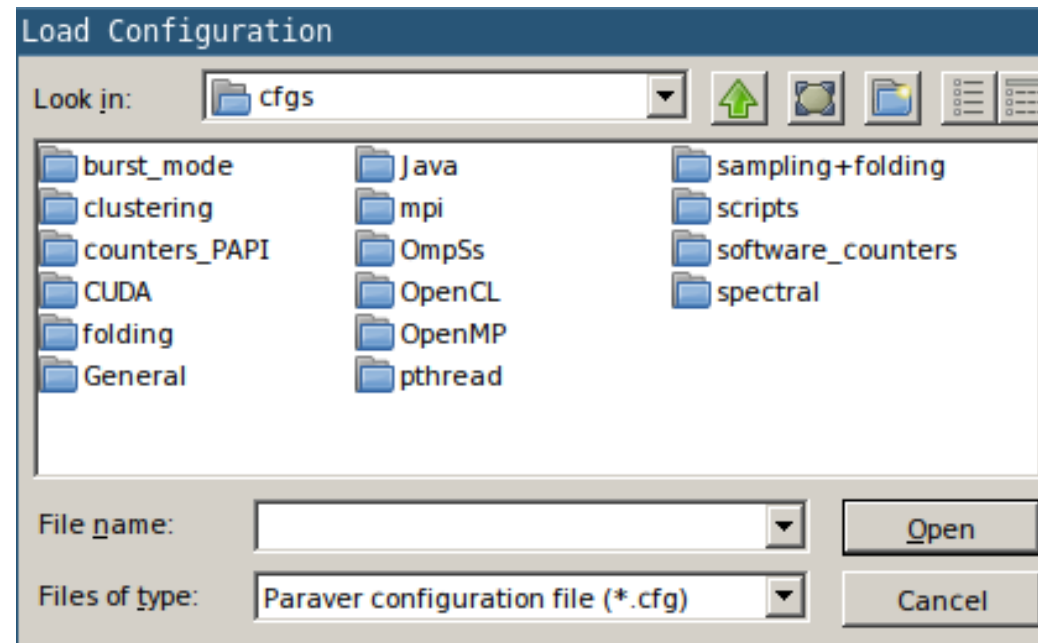
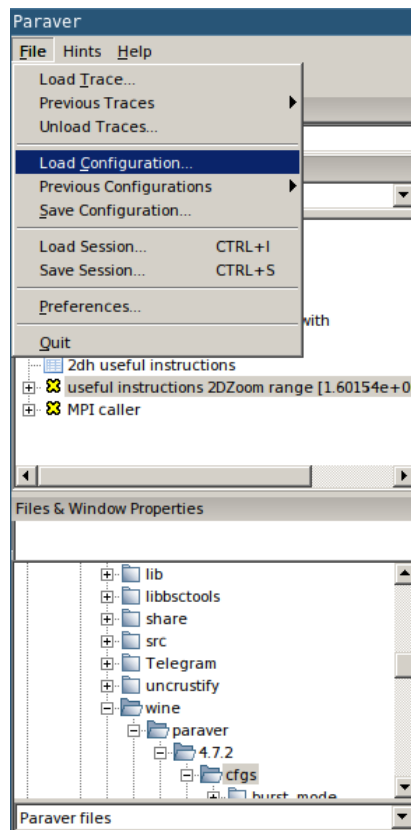


1. Main Paraver window



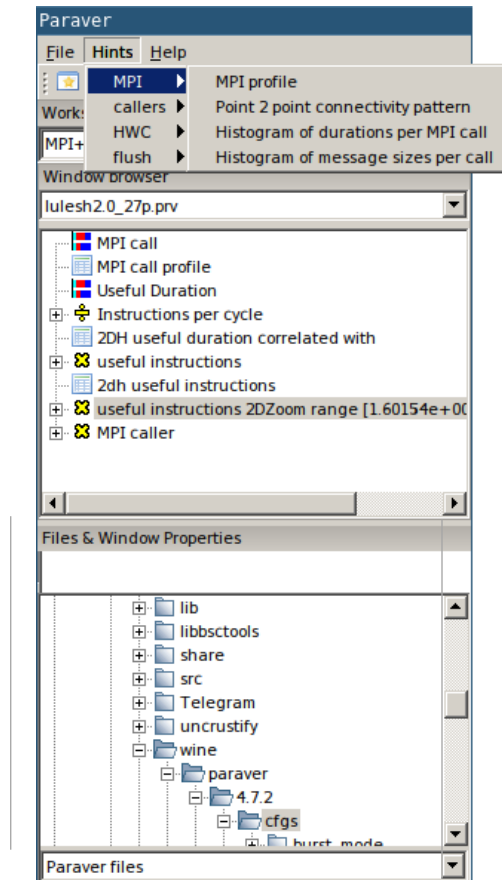
CFG's distribution

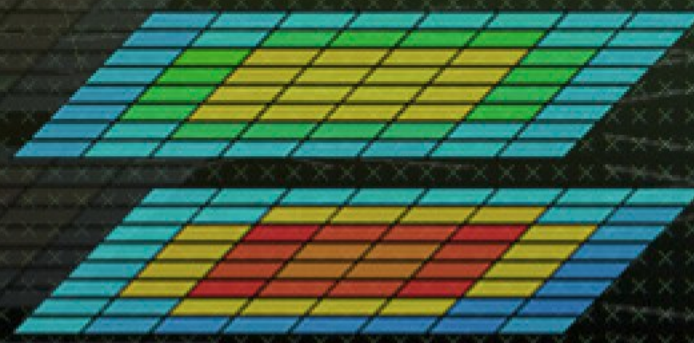
- Paraver comes with many more included CFG's



Hints: a good place to start!

- Paraver suggests CFG's based on the information present in the trace





Cluster-based analysis

Use clustering analysis

- Run clustering

```
romeo> module load clustering/2.6.8
romeo> cd $HOME/tools-material/clustering
romeo> BurstClustering -d cluster.xml -i ../extrae/lulesh2.0_27p.prv \
    -o lulesh2.0_27p_clustered.prv
```

- If you didn't get your own trace, use a prepared one from:

```
romeo> ls $HOME/tools-material/traces/lulesh2.0_27p.prv
```

- Copy the results to your computer

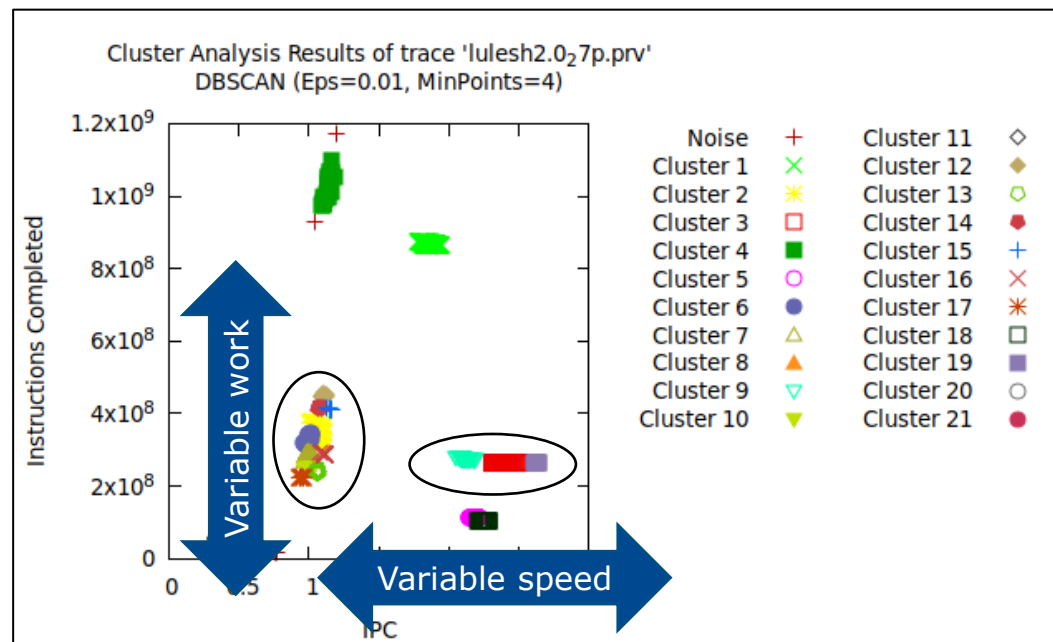
```
laptop> scp <USER>@romeologin1.univ-reims.fr:$HOME/tools-material/clustering/* ./
```

Cluster-based analysis

- Check the resulting scatter plot

```
laptop> gnuplot lulesh2.0_27p_clustered.IPC.PAPI_TOT_INS.gnuplot
```

- Identify main computing trends
- Work (Y) vs. Speed (X)
- Look at the clusters shape
 - Variability in both axes indicate **potential imbalances**

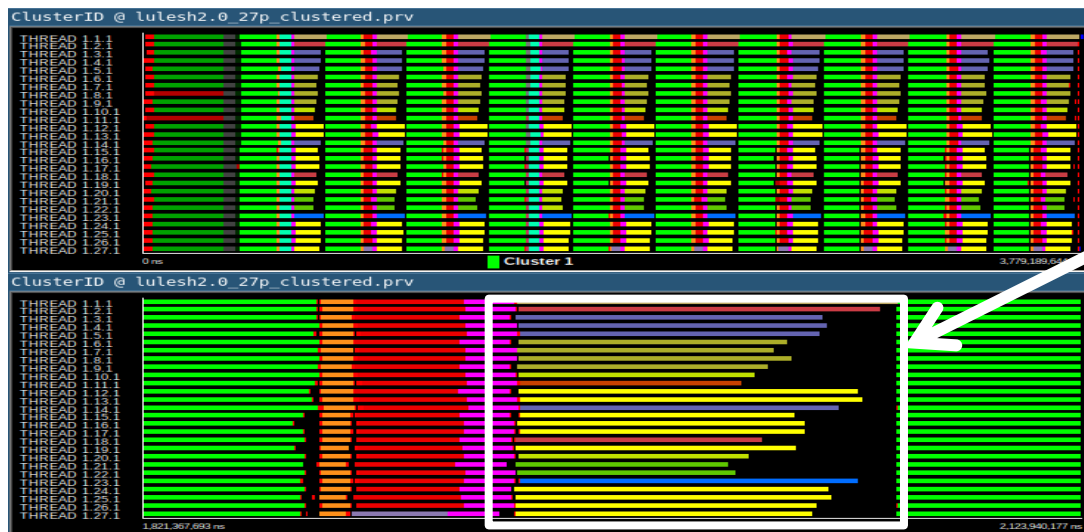


Correlating scatter plot and time distribution

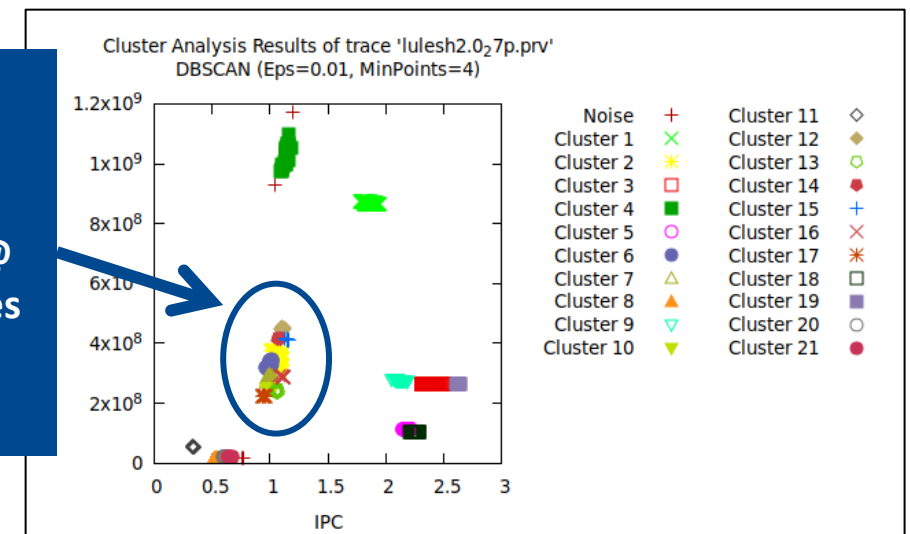
- Open the clustered trace with Paraver and look at it

```
laptop> $HOME/paraver/bin/wxparaver <path-to>/lulesh_27p_clustered.prv
```

- Display the distribution of clusters over time
 - File → Load configuration → \$HOME/paraver/cfgs/clustering/clusterID_window.cfg



Variable work /
speed
+
Simultaneously @
different processes
=
Imbalances



BSC Tools Hands-On

Lau Mercadal, Judit Giménez
(tools@bsc.es)
Barcelona Supercomputing Center
