



**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación



# Extrae & Paraver Hands-On

Judit Giménez, Germán Llort, Lau Mercadal,  
Sandra Méndez

✉ [tools@bsc.es](mailto:tools@bsc.es)

19/05/2022

POP Performance Analysis

# Extræ features

- Platforms
  - Intel, Cray, BlueGene, MIC, ARM, Android, Fujitsu Sparc ...
- Parallel programming models
  - 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 (Dyninst)
- Periodic sampling
- User events (Extræ API)

No need  
to  
recompile  
or relink!

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



# Using Extrae in 3 steps

1. **Adapt** your job submission scripts
  2. **Configure** what to trace
    - XML configuration file
    - Example configurations at `$EXTRAE_HOME/share/example`
  3. **Run it!**
- For further reference check the **Extrae User Guide:**
    - <https://tools.bsc.es/doc/html/extrae>
    - Also distributed with Extrae at `$EXTRAE_HOME/share/doc`

# Log in to Jusuf

```
laptop$ ssh -Y <USER>@jusuf.fz-juelich.de
```

- The following directory contains all the examples:

```
jusuf$ cp -r /p/project/training2214/tools-material  
$HOME
```

```
jusuf$ ls -l $HOME/tools-material
```

```
bin/
```

```
clustering/
```

```
extrae/
```

```
slides/ ←
```

```
traces/
```

Here you have a copy of this slides.  
Copy them to your laptop  
or open remotely with:

```
> evince slides/Extrae-Paraver-Hands-  
On.pdf
```

# Step 1: Adapt the job script to load Extrae

```
jusuf$ vi $HOME/tools-material/extrae/job_27p.sh
```

```
#!/bin/bash -x
#SBATCH --account=training2214
#SBATCH --nodes=1
#SBATCH --ntasks=27
#SBATCH --cpus-per-task=1
#SBATCH --output=lulesh2.0_27p.%j
#SBATCH --error=lulesh2.0_27p.%j
#SBATCH --time=00:10:00
#SBATCH --partition=batch

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

srun --cpu-bind=cores ../bin/lulesh2.0 -i 10 -s 65 -p
```

Request resources

Run the program

# Step 1: Adapt the job script to load Extrae

```
jusuf$ vi $HOME/tools-material/extrae/job_27p.sh
```

```
#!/bin/bash -x
#SBATCH --account=training2214
#SBATCH --nodes=1
#SBATCH --ntasks=27
#SBATCH --cpus-per-task=1
#SBATCH --output=lulesh2.0_27p.%j
#SBATCH --error=lulesh2.0_27p.%j
#SBATCH --time=00:10:00
#SBATCH --partition=batch

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

srun --cpu-bind=cores ./trace.sh ../bin/lulesh2.0_i
...
```

Run with Extrae

# Step 1: Adapt the job script to load Extrae

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

```
#!/bin/bash -x
#SBATCH --account=training2214
#SBATCH --nodes=1
#SBATCH --ntasks=27
#SBATCH --cpus-per-task=1
#SBATCH --output=lulesh2.0_27p.%j
#SBATCH --error=lulesh2.0_27p.%j
#SBATCH --time=00:10:00
#SBATCH --partition=batch

export OMP_NUM_THREADS=${SLURM_CPUS_PER_TASK}

srun --cpu-bind=cores ./trace.sh ../bin/lulesh2.
...
```

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

module load GCC ParaStationMPI Extrae

export
EXTRAE_HOME=/p/software/jusuf/stages/2022/software/Extrae/3.8.3-gpsmpi-2021b

export TRACE_NAME=lulesh2.0_27p.prv

# 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/

# Run the program
$*
```

Load Extrae

What to trace?

Type of application



# Step 1: Which tracing library?

- Choose depending on the application type

Library	Serial	MPI	OpenMP	pthread	CUDA
libseqtrace	X				
libmpitrace[f] <sup>1</sup>		X			
libompitrace			X		
libpttrace				X	
libcudatrace					X
libompitrace[f] <sup>1</sup>		X	X		
libptmpitrace[f] <sup>1</sup>		X		X	
libcudampitrace[f] <sup>1</sup>		X			X

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

# Step 2: Extrae XML configuration

```
jusuf$ vim $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?)

## Step 2: Extrae XML configuration (II)

```
jusuf$ vi $HOME/tools-material/extrae/extrae.xml
```

```
<counters enabled="yes">  
  <cpu enabled="yes" starting-set-distribution="1">  
    <set enabled="yes" domain="all" changeat-time="0">  
      PAPI_TOT_INS,PAPI_TOT_CYC  
    </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)

```
jusuf$ vi $HOME/tools-material/extrae/extrae.xml
```

```
<buffer enabled="yes">  
  <size enabled="yes">500000</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**

# Step 3: Run it!

- Submit your job

```
jusuf$ cd $HOME/tools-material/extrae
```

```
jusuf$ sbatch job_27p.sh
```

- Easy!

# All done! Check your resulting trace

- Once finished (check with “queue”) you will have the trace (3 files):

```
jusuf$ ls -l $HOME/tools-material/extrae  
lulesh2.0_i_27p.pcf  
lulesh2.0_i_27p.prv  
lulesh2.0_i_27p.row
```

- Any trouble? Traces already generated here:

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

- Now let's look into it !

# Install Paraver

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

Pick your version



wxparaver-4.10.0-win.zip



wxparaver-4.10.0-mac.zip



wxparaver-4.10.0-Linux\_i686.tar.gz (32-bits)

wxparaver-4.10.0-Linux\_x86\_64.tar.gz (64-bits)

The screenshot shows a grid of tool options for Paraver. Each option is presented in a light gray box with a blue header button labeled 'Get [Tool Name]'. Below the header, the version and size are listed. Icons for supported operating systems (101 RAW, i686, x86\_64) are shown. A plus sign is located at the bottom right of each box. The tools shown are:

- CLUSTERING**: Version 2.6.6 • 2 MB. Supports 101 RAW, i686, and x86\_64.
- TRACKING**: Version 2.6.5 • 1.9 MB. Supports 101 RAW, i686, and x86\_64.
- FOLDING**: Version 1.0.2 • 11.06 MB. Supports 101 RAW, i686, and x86\_64.
- SPECTRAL**: Version 3.4.0 • 0.31 MB. Supports 101 RAW, i686, and x86\_64. Description: Signal processing techniques to select representative regions from Paraver traces.
- BASIC ANALYSIS**: Version 0.2 • 10.89 MB. Supports 101 RAW. Description: Framework for automatic extraction of fundamental factors for Paraver traces.

# Install Paraver (II)

```
laptop$ tar xf wxparaver-4.10.0-linux-  
x86_64.tar.gz
```

```
laptop$ mv wxparaver-4.10.0-linux-x86_64 paraver
```

- Start Paraver

```
laptop$ paraver/bin/wxparaver
```

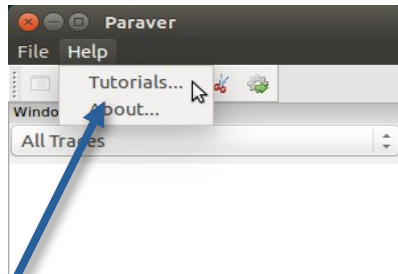
- Trouble installing locally? Remote open from Jusuf

```
jusuf$ wxparaver
```



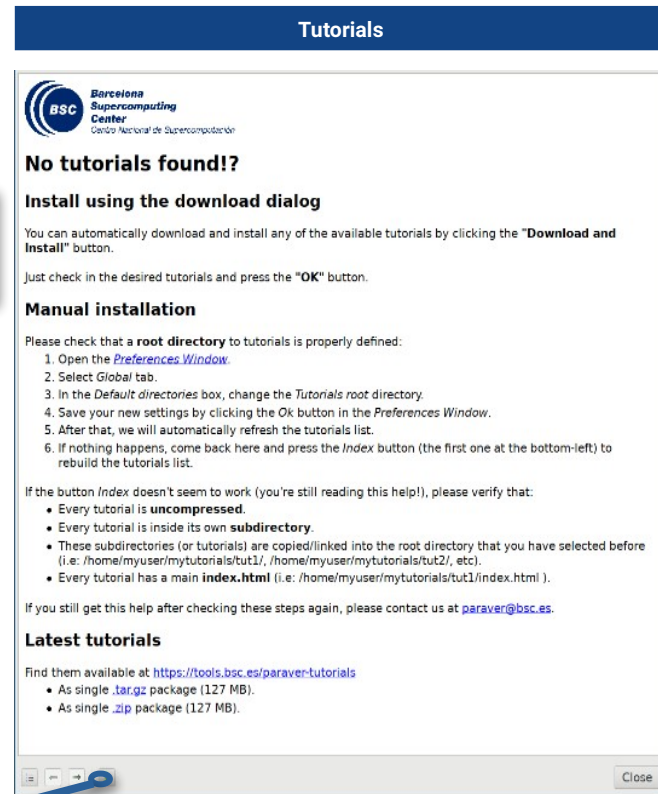
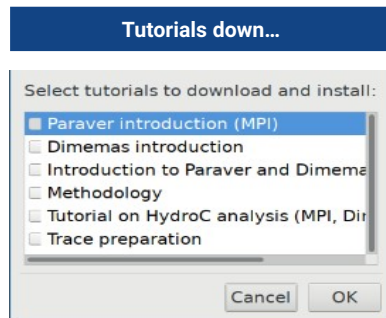
# Install Paraver tutorials (I)

- Download tutorials:



Click on Help  
Tutorials

Tutorials window  
will pop-up










Follow these tutorials by clicking on the hyperlinks and reading the explanations. When you click on a link, multiple views will open.

# Install Paraver tutorials – alternative methods(II)



- Download tutorials archive
  - <https://tools.bsc.es/paraver-tutorials>

Home » Documentation » Paraver tutorials

These seven tutorials can be opened with wxParaver versions newer than 4.3.0, and you'll be able to follow the steps within the tool. To install them, download and untar the package and follow the instructions of the Help/Tutorial option on the Paraver main window. Following there is a list of available tutorials:

 <a href="#">Paraver introduction (MPI)</a>	Start here to familiarize with Paraver basic commands and the first steps of a performance analysis.
 <a href="#">Dimemas introduction</a>	The basic steps to learn how to configure and run the Dimemas simulator and to start looking at the results.
 <a href="#">Introduction to Paraver and Dimemas methodology</a>	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).
 <a href="#">Methodology</a>	This tutorial shows some examples of the analysis that can be done using the provided configuration files.
 <a href="#">Tutorial on HydroC analysis (MPI, Dimemas, CUDA)</a>	One example of performance analysis of the MPI application Hydro and further simulations with Dimemas.
 <a href="#">Trace preparation</a>	Look at this tutorial to select a representative region for a large trace that cannot be loaded into memory.
 <a href="#">Trace alignment tutorial.</a>	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 all of them together in a single package:

 [.tar.gz format \(127 Mb\)](#)  [.zip format \(127 Mb\)](#)

All tutorials

paraver-tutorials-20150526.tar.gz

# Install Paraver tutorials – alternative methods(III)

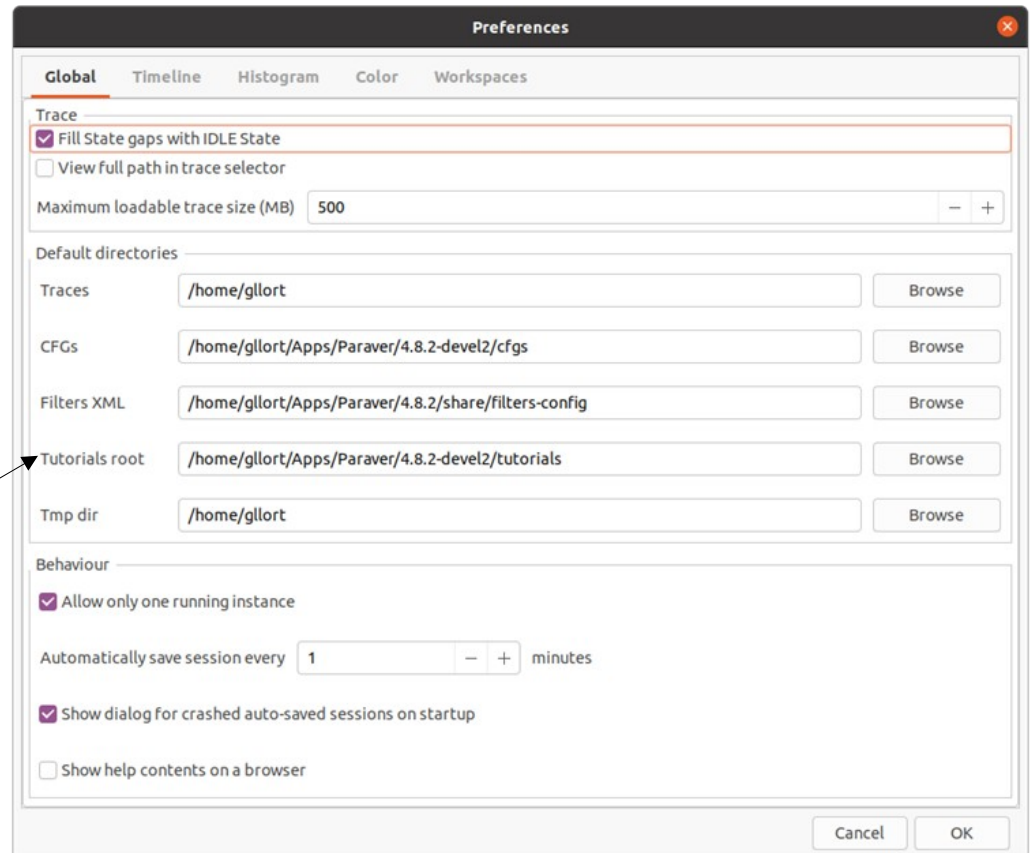
- Start Paraver:
  - Linux: Run the command:

```
laptop$ paraver/bin/wxparaver
```

- Windows: Double-click on paraver/wxparaver.exe
- MAC: Double click on paraver/wxparaver.app

- Open File → Preferences  
Setup the “Tutorials root” pointing to your folder “tutorials”

Click Browse and select your folder “tutorials”



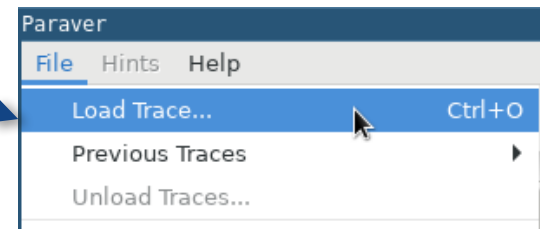
# First steps of analysis

- Copy the trace to your laptop

```
laptop$ scp <USER>@jusuf.fz-juelich.de: \
tools-material/extrae/lulesh2.0_i_27p.* $HOME
```

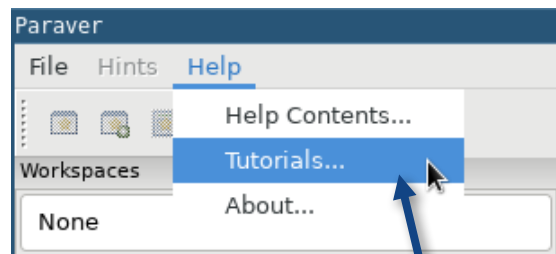
- Load the trace with Paraver

Click on File → Load Trace  
→ Browse to  
“lulesh2.0\_i\_27p.prv”

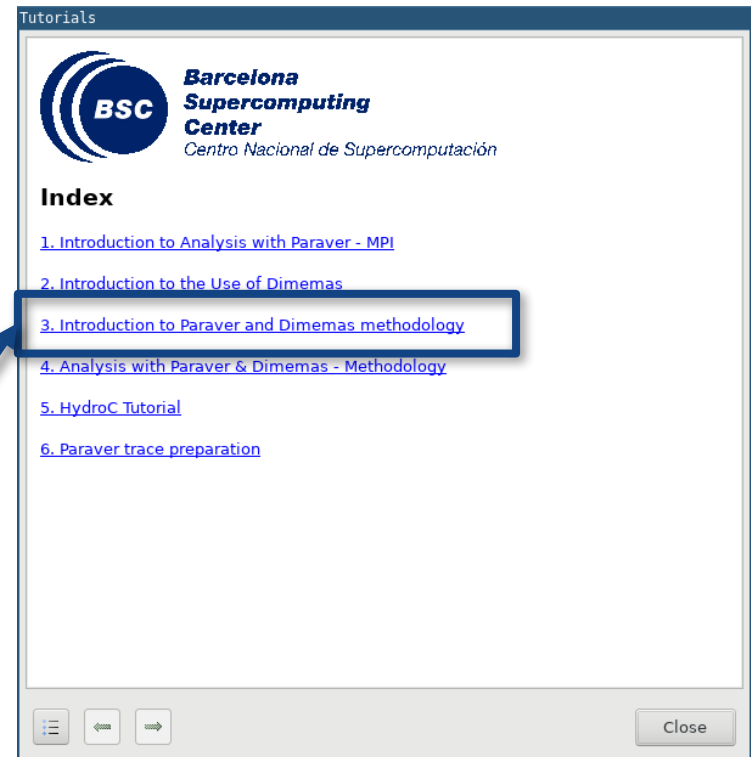


# First steps of analysis

- Follow Tutorial #3
  - Introduction to Paraver and Dimemas methodology



Click on Help → Tutorials



# Measure the parallel efficiency

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

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: parallelization efficiency and the efficiency obtained in the execution of the code in the computation regions. These two metrics would be the first checks on the proposed metrics.

- To measure the parallel efficiency load the configuration file `cfgs/mpi/mpi_stats.cfg`. This configuration pops up a table with %time spent in every thread in every MPI call. Look at the global statistics at the bottom of the outside mpi column. Entry *Average* represents the average parallel efficiency, entry *Avg/Max* represents the global load balance, entry *Maximum* represents the communication efficiency. If any of the values are lower than 85% is recommended to look at the corresponding metric in detail. Open the control window to identify the phases and 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 computational load (instructions) distribution load the configuration file `cfgs/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 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.

THREAD	Parallel Efficiency	Comm Efficiency	Load Balance	...	...	...	...	...	...
1.18.1	94.41 %	0.06 %	0.05 %	0.09 %	0.30 %	0.01 %	0.00 %	4	
1.19.1	94.77 %	0.05 %	0.04 %	0.05 %	0.46 %	0.02 %	0.00 %	4	
1.20.1	87.72 %	0.06 %	0.05 %	0.17 %	3.97 %	0.01 %	0.00 %	7	
1.21.1	86.55 %	0.05 %	0.03 %	2.50 %	1.83 %	0.02 %	0.49 %	8	
1.22.1	87.11 %	0.08 %	0.05 %	0.22 %	4.09 %	0.01 %	0.00 %	7	
1.23.1	91.30 %	0.09 %	0.06 %	0.59 %	3.63 %	0.01 %	0.00 %	3	
1.24.1	90.49 %	0.07 %	0.05 %	0.66 %	3.42 %	0.01 %	0.00 %	4	
1.25.1	97.48 %	0.05 %	0.03 %	0.24 %	0.29 %	0.03 %	0.00 %	1	
1.26.1	96.21 %	0.08 %	0.04 %	0.06 %	0.35 %	0.01 %	0.00 %	2	
1.27.1	95.49 %	0.05 %	0.04 %	0.04 %	0.58 %	0.01 %	0.00 %	3	
<b>Total</b>	2,483.17 %	1.79 %	1.50 %	12.83 %	55.38 %	0.33 %	1.03 %	128	
<b>Average</b>	91.97 %	0.07 %	0.06 %	0.48 %	2.05 %	0.01 %	0.04 %	4	
<b>Maximum</b>	98.92 %	0.13 %	0.11 %	3.42 %	4.13 %	0.03 %	0.49 %	8	
<b>Minimum</b>	86.55 %	0.03 %	0.03 %	0.04 %	0.22 %	0.00 %	0.00 %	0	
<b>StDev</b>	3.54 %	0.02 %	0.02 %	0.74 %	1.61 %	0.01 %	0.10 %	1	
<b>Avg/Max</b>	0.93	0.52	0.50	0.14	0.50	0.43	0.08		

Parallel efficiency

Comm efficiency

Load balance

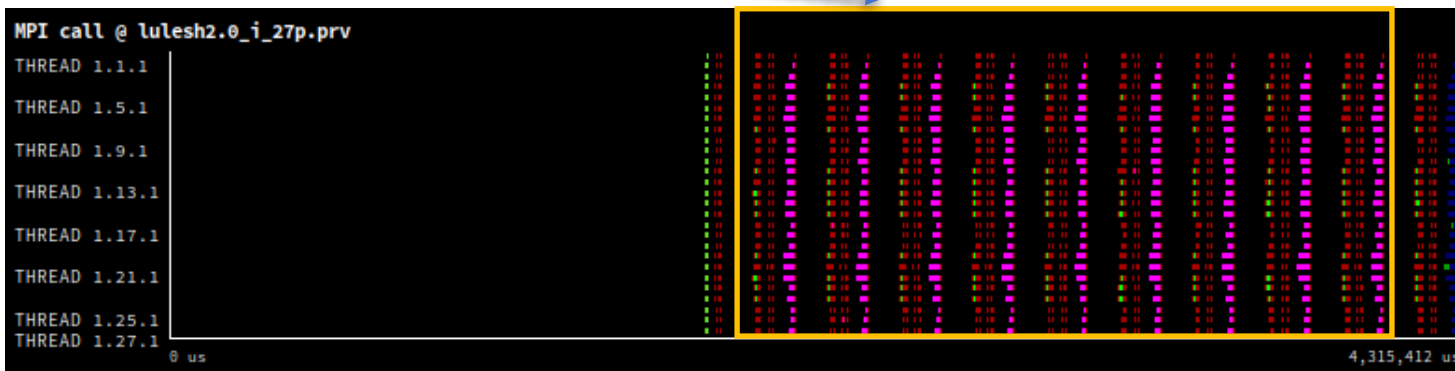
# Focus on the iterative part

Click on "Open Control Window"

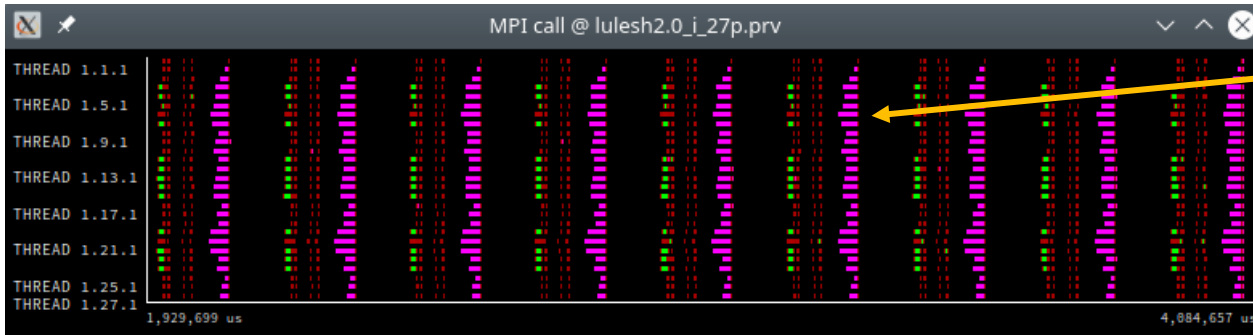
Zoom iterative part

MPI call profile @ lulesh2.0\_i\_27p.prv

THREAD 1.18.1	94.41 %	0.06 %	0.05 %	0.09 %	0.30 %	0.01 %	0.00 %	4
THREAD 1.19.1	94.77 %	0.05 %	0.04 %	0.05 %	0.46 %	0.02 %	0.00 %	4
THREAD 1.20.1	87.72 %	0.06 %	0.05 %	0.17 %	3.97 %	0.01 %	0.00 %	7
THREAD 1.21.1	86.55 %	0.05 %	0.03 %	2.50 %	1.83 %	0.02 %	0.49 %	8
THREAD 1.22.1	87.11 %	0.08 %	0.05 %	0.22 %	4.09 %	0.01 %	0.00 %	7
THREAD 1.23.1	91.30 %	0.09 %	0.06 %	0.59 %	3.63 %	0.01 %	0.00 %	3
THREAD 1.24.1	90.49 %	0.07 %	0.05 %	0.66 %	3.42 %	0.01 %	0.00 %	4
THREAD 1.25.1	97.48 %	0.05 %	0.03 %	0.24 %	0.29 %	0.03 %	0.00 %	1
THREAD 1.26.1	96.21 %	0.08 %	0.04 %	0.06 %	0.35 %	0.01 %	0.00 %	2
THREAD 1.27.1	95.49 %	0.05 %	0.04 %	0.04 %	0.58 %	0.01 %	0.00 %	3
<b>Total</b>	2,483.17 %	1.79 %	1.50 %	12.83 %	55.38 %	0.33 %	1.03 %	128
<b>Average</b>	91.97 %	0.07 %	0.06 %	0.48 %	2.05 %	0.01 %	0.04 %	4
<b>Maximum</b>	98.92 %	0.13 %	0.11 %	3.42 %	4.13 %	0.03 %	0.49 %	8
<b>Minimum</b>	86.55 %	0.03 %	0.03 %	0.04 %	0.22 %	0.00 %	0.00 %	0
<b>StDev</b>	3.54 %	0.02 %	0.02 %	0.74 %	1.61 %	0.01 %	0.10 %	1
<b>Avg/Max</b>	0.93	0.52	0.50	0.14	0.50	0.43	0.08	



# Recalculate efficiency of iterative region



Right click -> Copy

Right click -> Paste -> Time

Thread	Parallel efficiency	Comm efficiency	Load balance	Other 1	Other 2	Other 3	Other 4
THREAD 1.20.1	78.31 %	0.10 %	0.08 %	0.26 %	6.91 %	14.27 %	0.08 %
THREAD 1.21.1	76.25 %	0.08 %	0.05 %	4.53 %	2.94 %	16.04 %	0.10 %
THREAD 1.22.1	77.21 %	0.12 %	0.08 %	0.39 %	7.09 %	15.00 %	0.10 %
THREAD 1.23.1	84.78 %	0.15 %	0.10 %	1.08 %	6.19 %	7.62 %	0.07 %
THREAD 1.24.1	83.36 %	0.11 %	0.08 %	1.14 %	5.87 %	9.36 %	0.08 %
THREAD 1.25.1	95.64 %	0.08 %	0.05 %	0.41 %	0.49 %	3.26 %	0.08 %
THREAD 1.26.1	93.31 %	0.12 %	0.07 %	0.11 %	0.59 %	5.72 %	0.07 %
THREAD 1.27.1	92.10 %	0.08 %	0.06 %	0.07 %	0.91 %	6.71 %	0.07 %
Total	2,318.40 %	2.68 %	2.48 %	22.79 %	94.91 %	256.50 %	2.25 %
Average	85.87 %	0.10 %	0.09 %	0.84 %	3.52 %	9.50 %	0.08 %
Maximum	98.23 %	0.19 %	0.19 %	6.12 %	7.16 %	16.04 %	0.11 %
Minimum	76.25 %	0.05 %	0.05 %	0.07 %	0.38 %	0.02 %	0.06 %
StDev	6.25 %	0.03 %	0.03 %	1.33 %	2.80 %	3.83 %	0.01 %
Avg/Max	0.87	0.52	0.49	0.14	0.49	0.59	0.74

Parallel efficiency

Comm efficiency

Load balance



# Computation time distribution

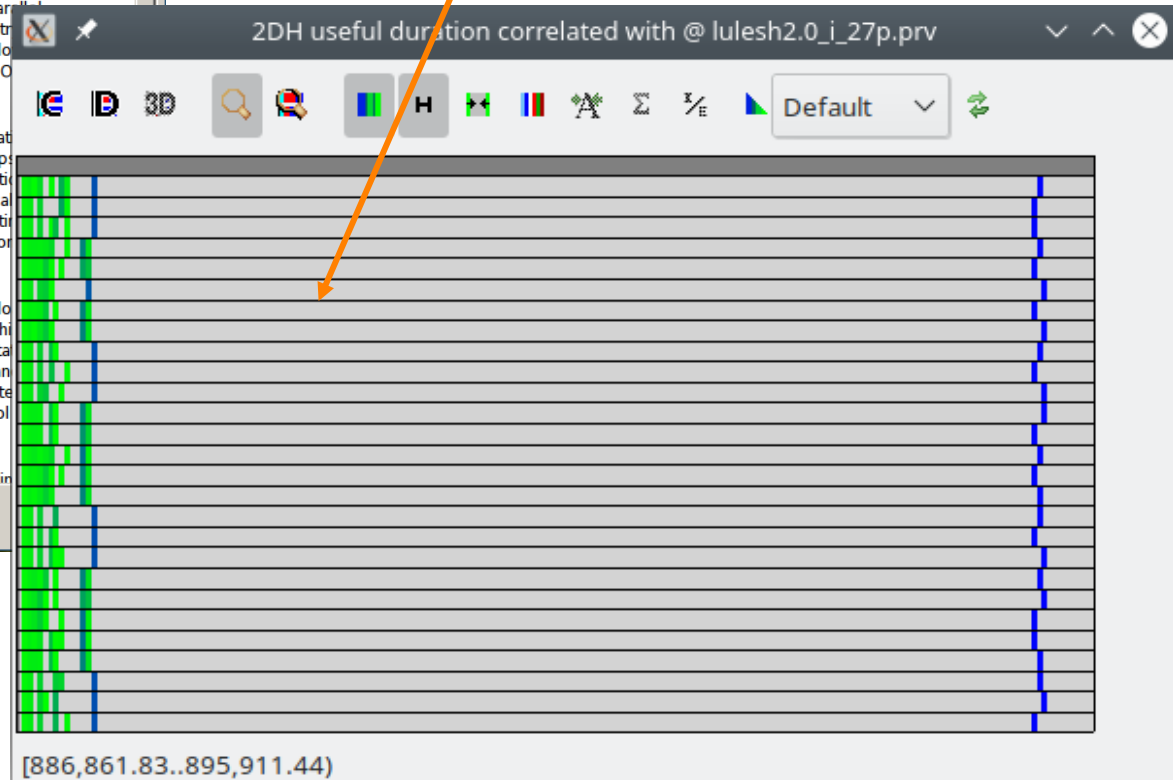
- Click on “2dh\_usefulduration.cfg” (2nd link) ☒ Shows **time computing**

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 [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 parallelization efficiency, entry *Avg/Max* represents the global load balance and entry *Efficiency* 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 is not balanced. 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 [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 correlate both views.
- To measure the serial regions performance look at the IPC timeline

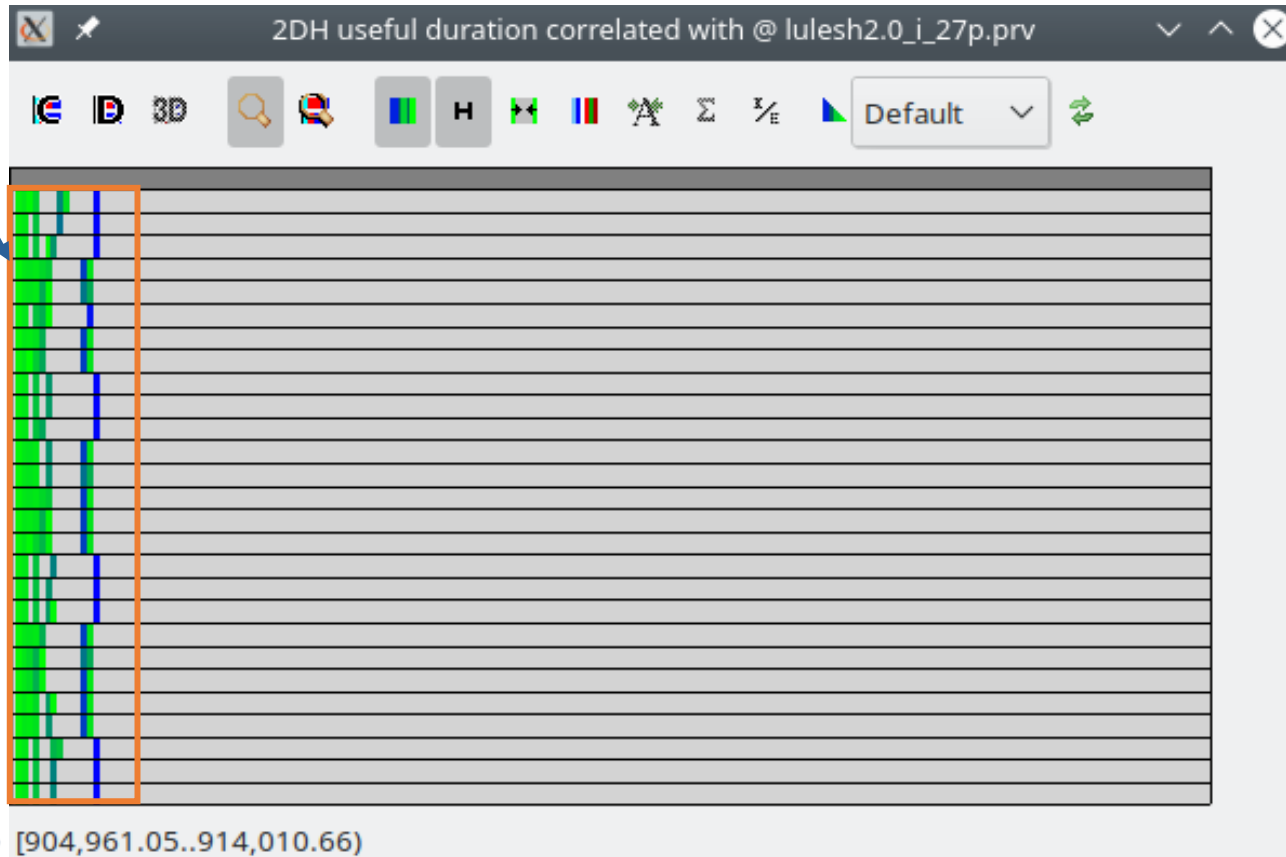
Right click -> Paste -> Time  
(to focus on the iterative part)



# Computation time distribution

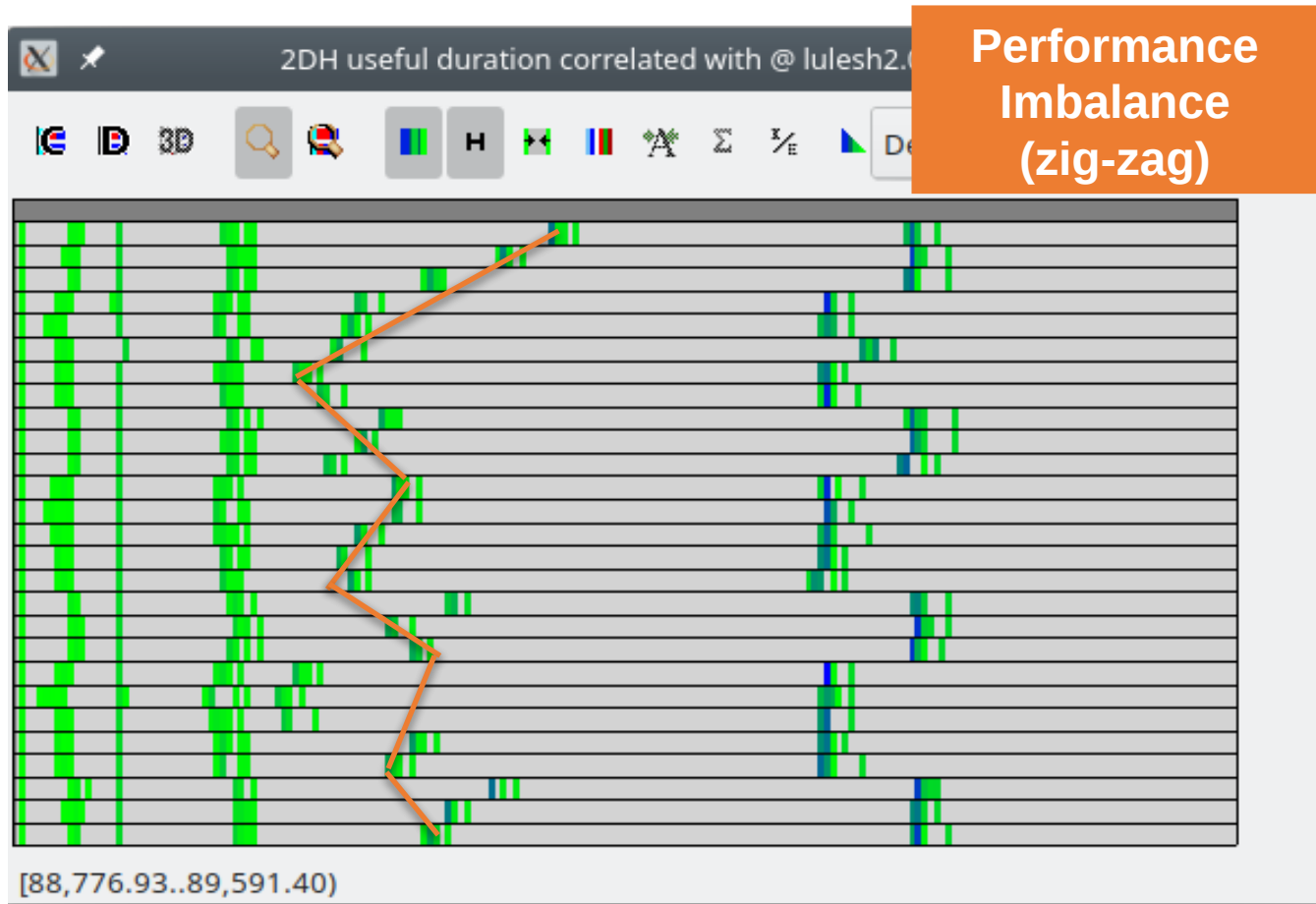
- Click on “2dh\_usefulduration.cfg” (2nd link) ☒ Shows **time computing**

Zoom in



# Computation time distribution

- Click on “2dh\_usefulduration.cfg” (2nd link) ☒ Shows **time computing**



# Computation load distribution

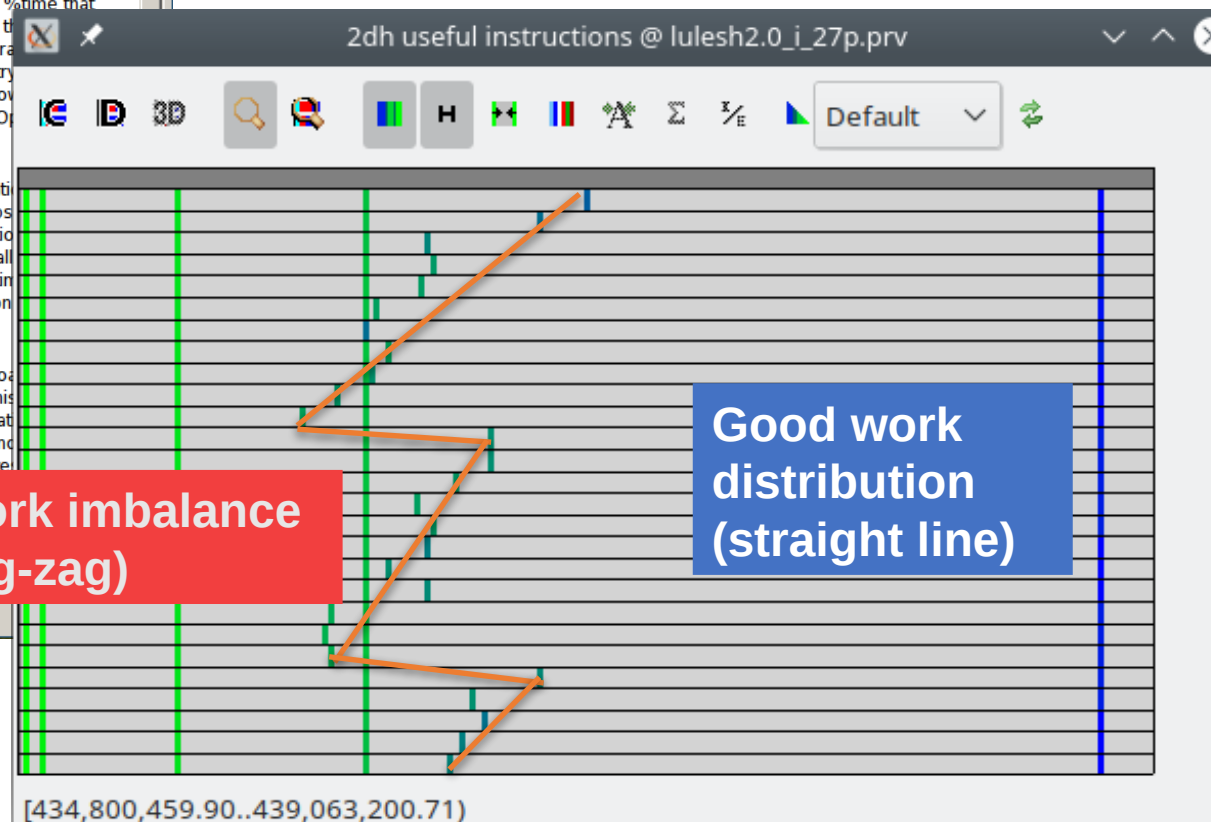
- Click on “2dh\_useful\_instructions.cfg” (3rd link) ☒ Shows amount of work

Focus only on the iteration part  
(copy time and zoom done)

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 [cifs/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 outside mpi column. Entry *Average* represents the application parallel efficiency, entry *Avg/Max* represents the global load balance and entry *Efficiency* represents the communication efficiency. If any of those values are low 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 [cifs/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 is not balanced. 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 [cifs/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



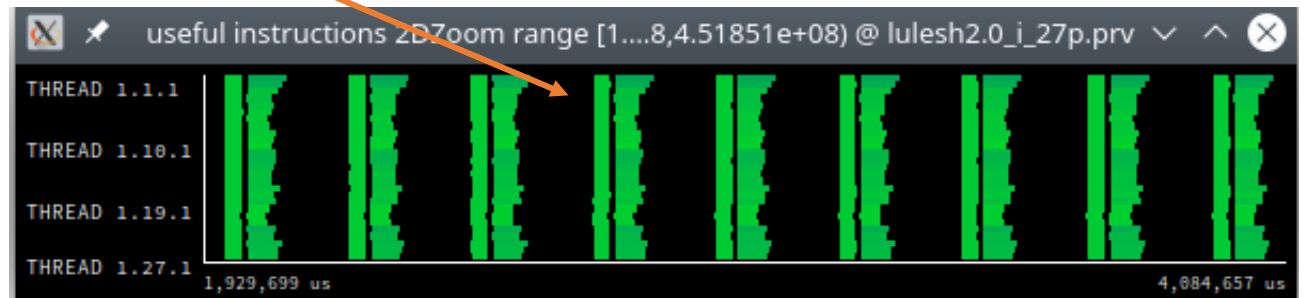
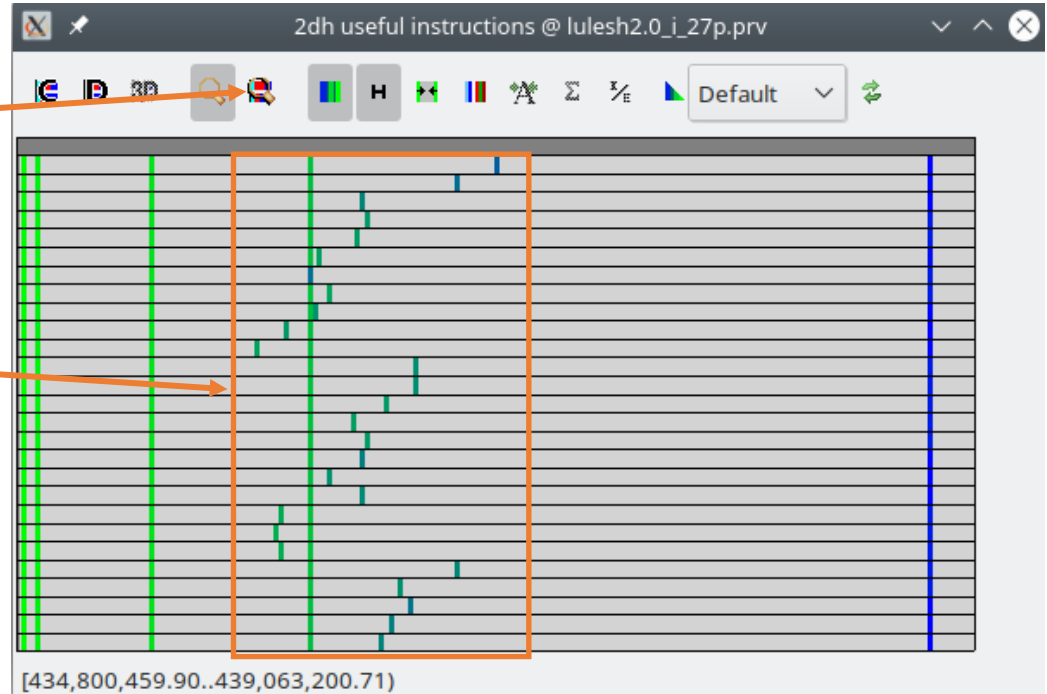
# Where does this happen?

- Go from the table to the timeline

1. Click on  
“Open Filtered Control  
Window”

2. Select this area  
(by drag-and-dropping)

Right click  
→ Fit Semantic Scale  
→ Fit both



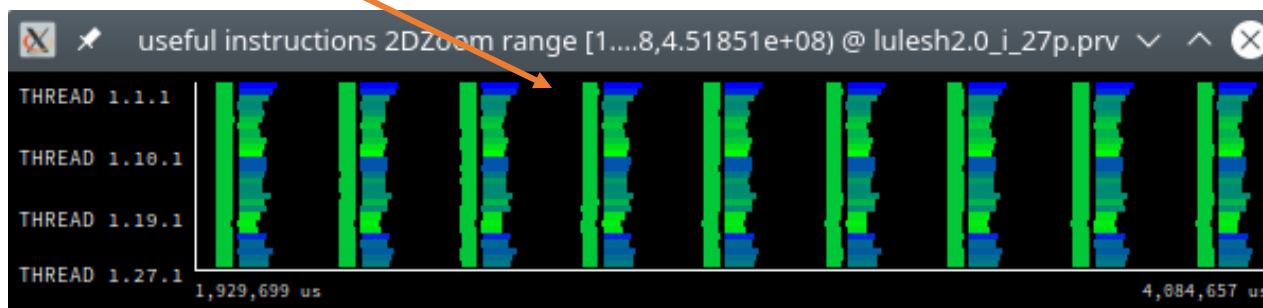
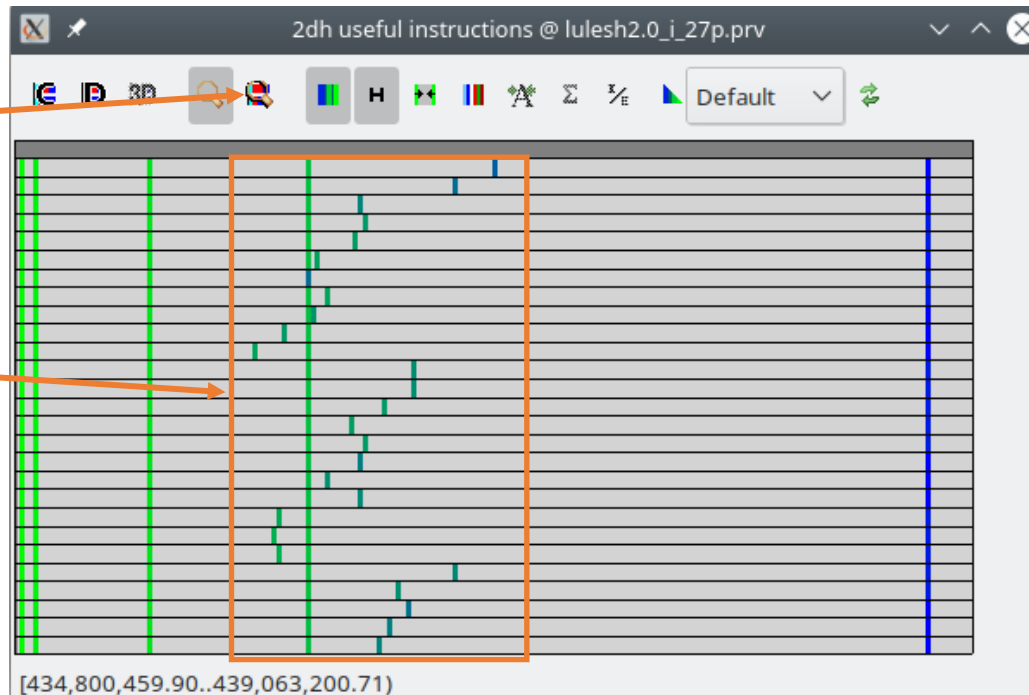
# Where does this happen?

- Go from the table to the timeline

1. Click on  
“Open Filtered Control  
Window”

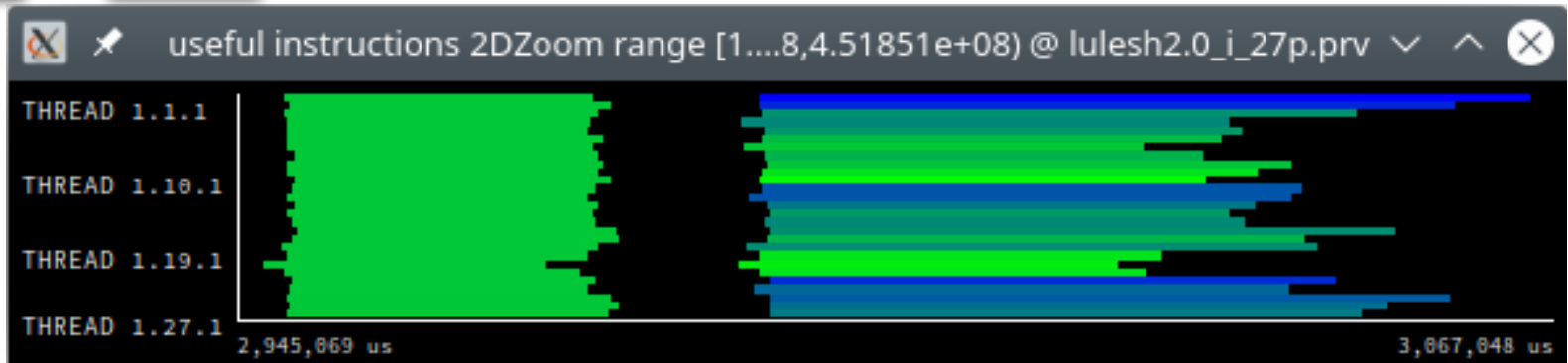
2. Select this area  
(by drag-and-dropping)

Right click  
→ Fit Semantic Scale  
→ Fit both

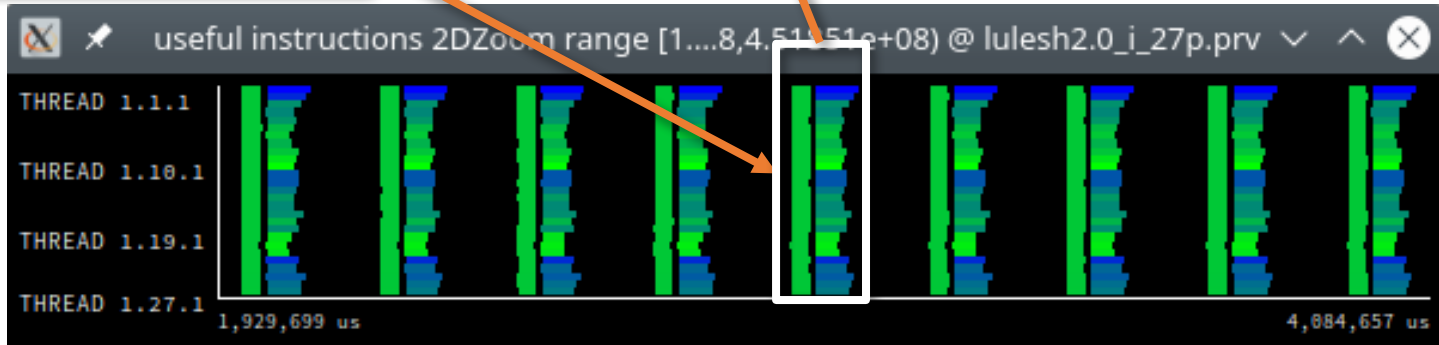


# Where does this happen?

- **Slow** & **Fast** at the same time? ☒ Imbalance

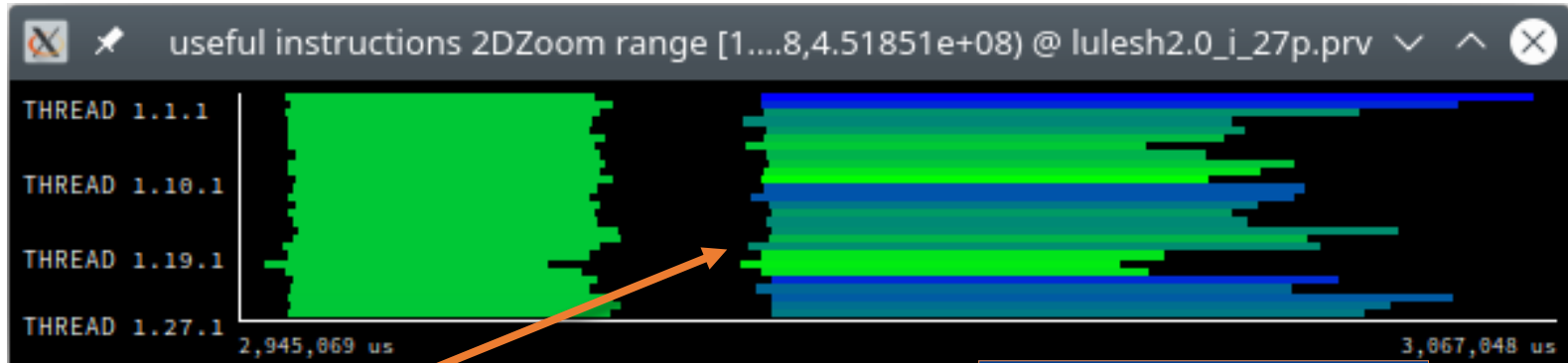


Zoom into  
1 of the iterations  
(by drag-and-dropping)



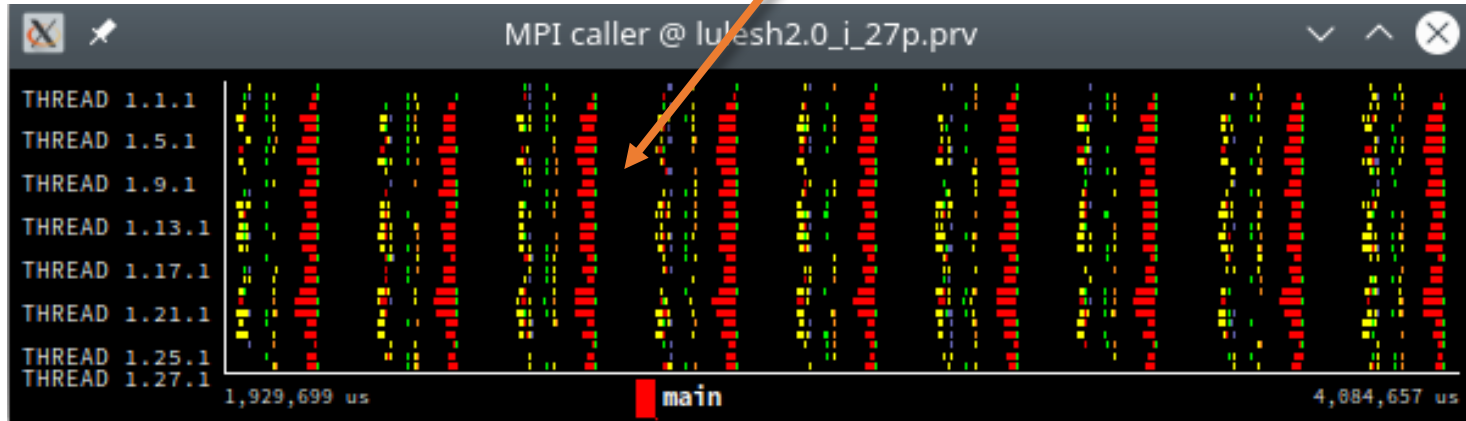
# Where does this happen?

- Hints → Call stack references → Caller function



Right click → Copy

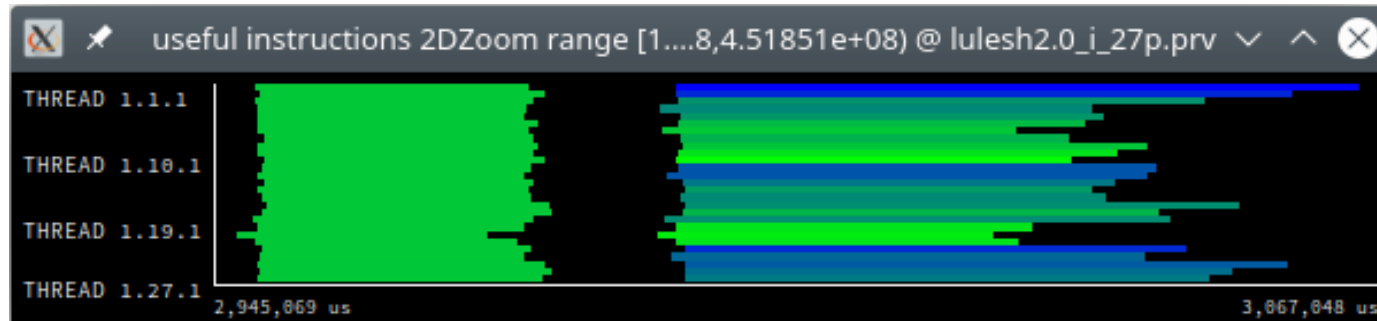
Right click → Paste → Time



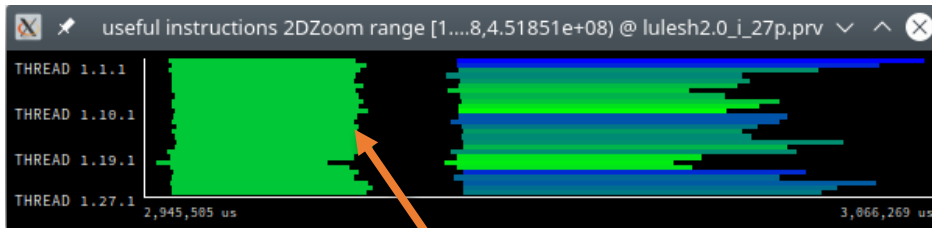


# Where does this happen?

- Hints → Call stack references → Caller function



# Save CFG's (2 methods)



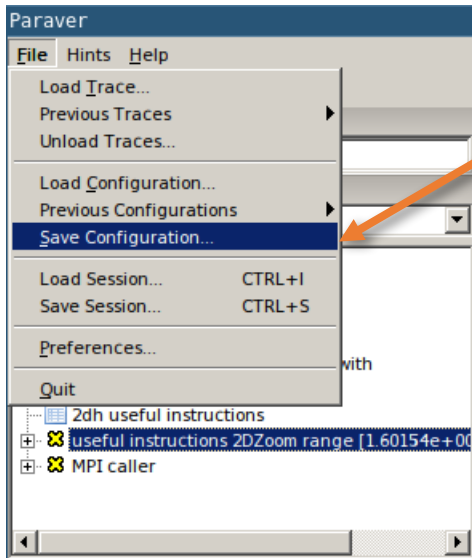
Right click on  
timeline

A screenshot of a context menu for a timeline visualization. The menu is open, showing various options. The "Save" option is highlighted in blue, and its sub-menu is also open, showing "Configuration...", "Image...", "Image Legend...", and "Text...".

Copy	CTRL+C
Paste	
Clone	
Undo Zoom	CTRL+U
Redo Zoom	CTRL+R
Fit Time Scale	
Fit Semantic Scale	
Fit Objects	
Select Objects...	
View	
Paint As	
Drawmode	
Pixel Size	
Object Labels	
Object Axis	
Run	
Synchronize	
<b>Save</b>	
Timing	CTRL+T
Info Panel	

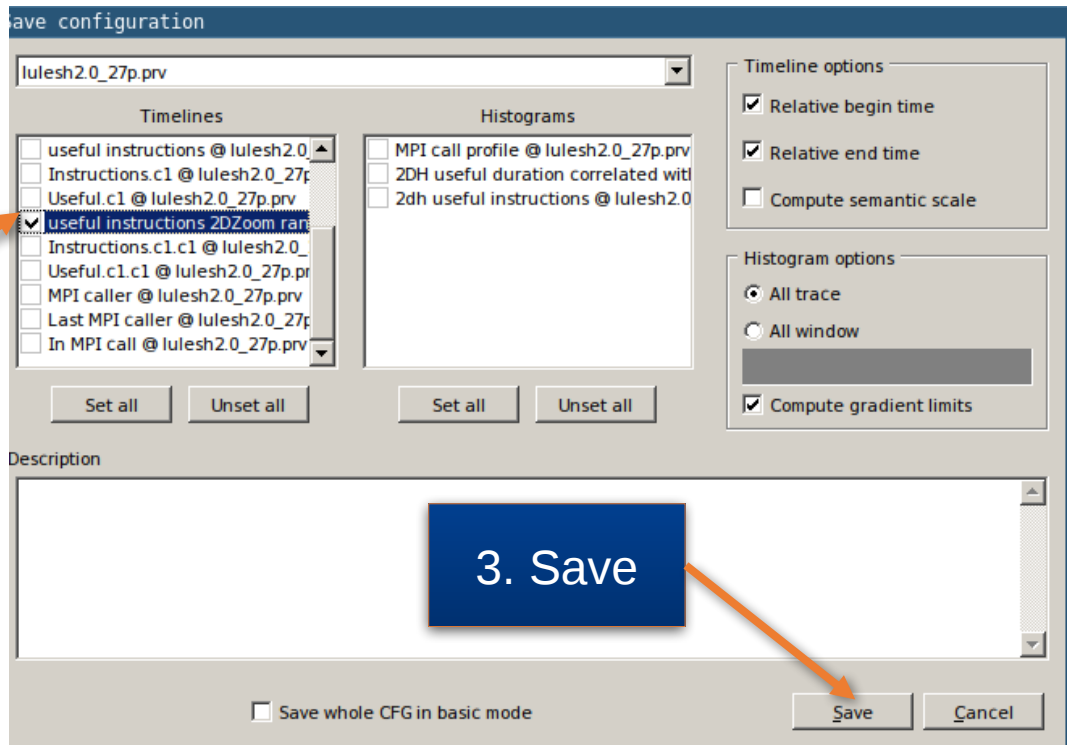
Configuration...
Image...
Image Legend...
Text...

# Save CFG's (2 methods)



1. Main Paraver window

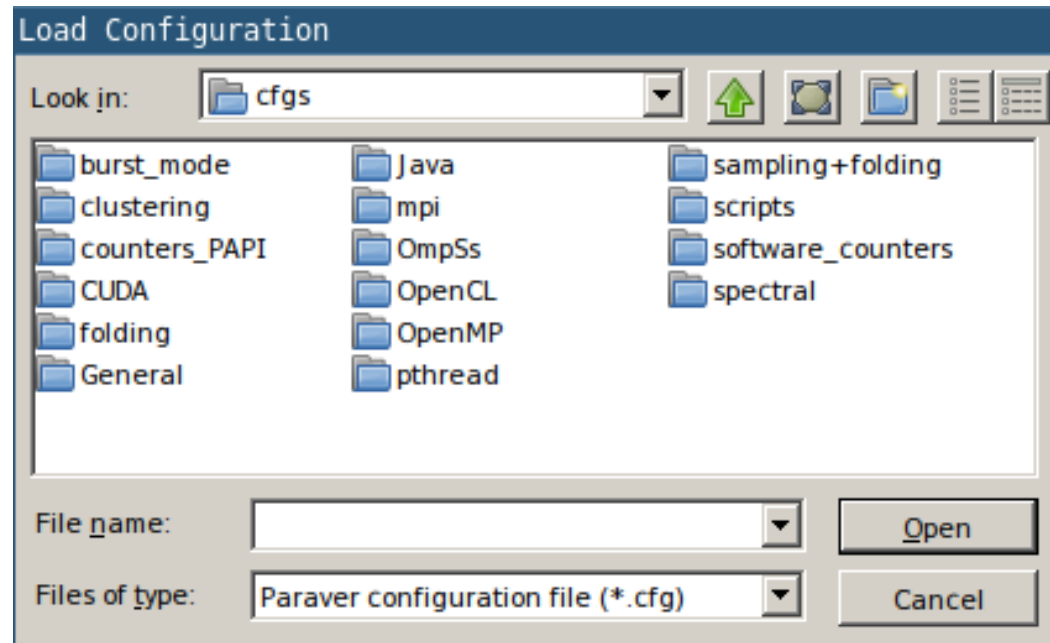
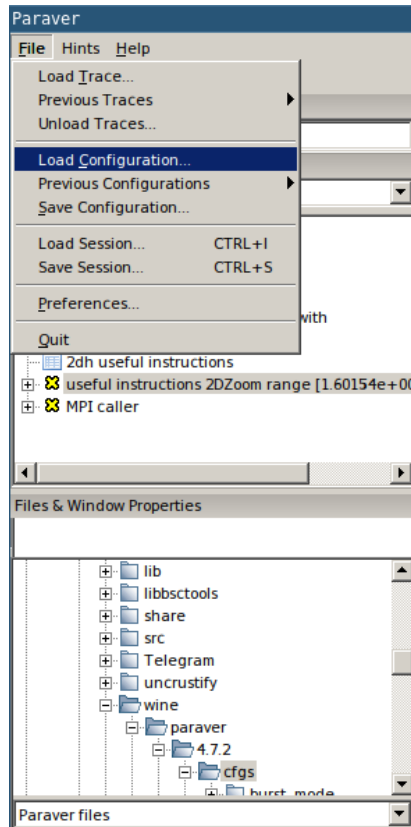
2. Select



3. Save

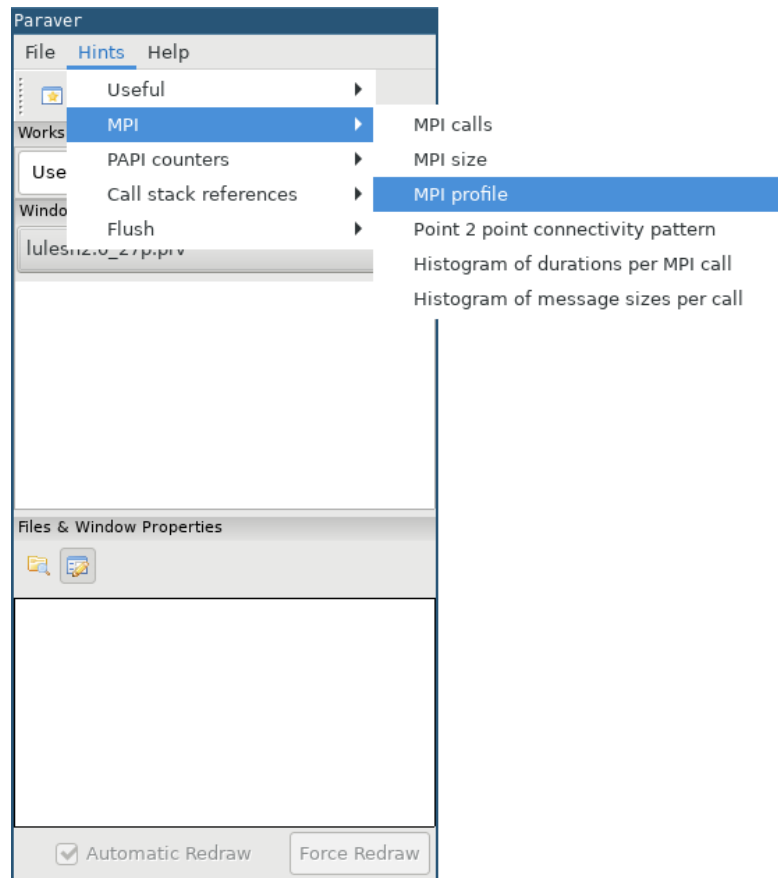
# CFG's distribution

- Paraver comes with many more included CFG's



# Hints: a good place to start!

- Paraver suggests CFG's based on the contents of the trace





**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación



# Takeaway: Analyse efficiencies and unbalances

Judit Giménez, Germán Llort, Lau Mercadal,  
Sandra Méndez

✉ [tools@bsc.es](mailto:tools@bsc.es)

19/04/2021

POP Performance Analysis



**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación



# Clustering Hands-On

Judit Giménez, Germán Llort, Lau Mercadal,  
Sandra Méndez

✉ [tools@bsc.es](mailto:tools@bsc.es)

19/04/2021

POP Performance Analysis

# Cluster-based analysis (I)

- Run the clustering tool on the trace you generated

```
jusuf$ source /p/project/training2214/setup.sh
jusuf$ cd $HOME/tools-material/clustering
jusuf$ BurstClustering \
-d cluster.xml \
-i ../extrae/lulesh2.0_i_27p.prv \
-o lulesh2.0_i_27p-clustered.prv
```

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

```
jusuf$ ls
$HOME/tools-material/traces/lulesh2.0_i_27p.prv
```

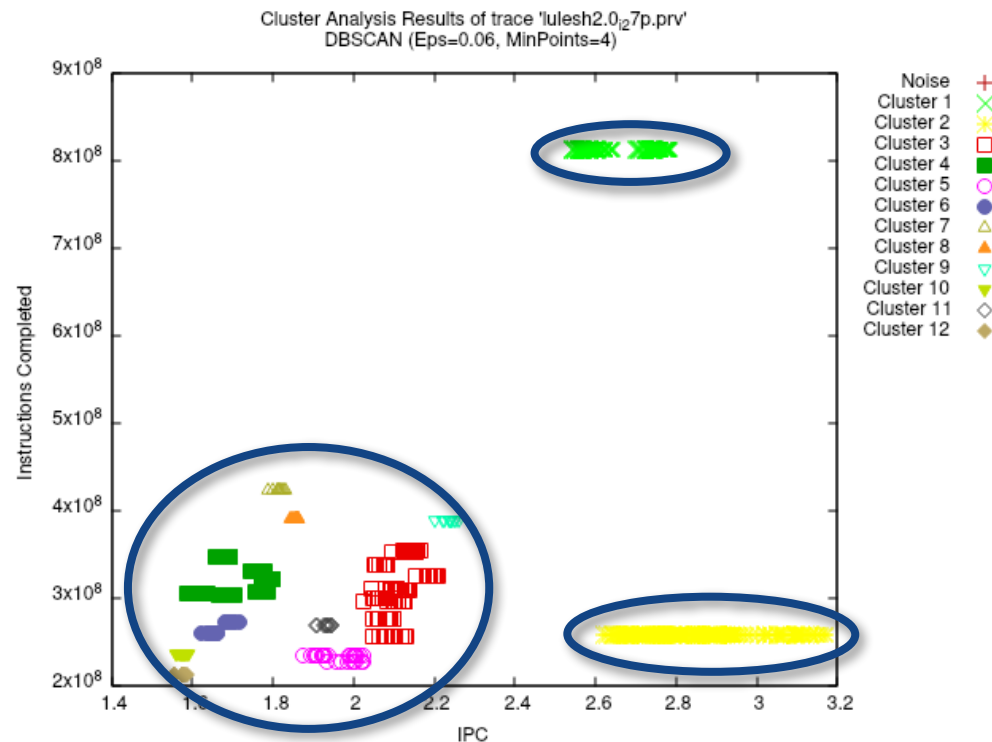


# Cluster-based analysis (II)

- Check the clustering scatter plot

```
jusuf$ gnuplot \  
lulesh2.0_i_27p-clustered.IPC.PAPI_TOT_INS.gnuplot
```

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



# Cluster-based analysis (III)

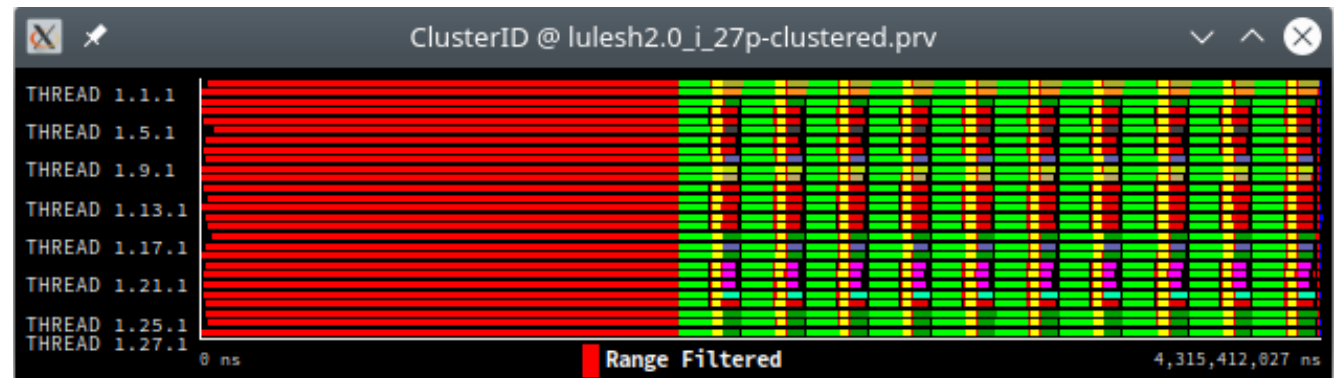
- Check the clustered trace
  - Copy the trace to your laptop

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

- Load with Paraver

```
laptop$ paraver/bin/wxparaver \
$HOME/lulesh2.0_i_27p_clustered.prv
```

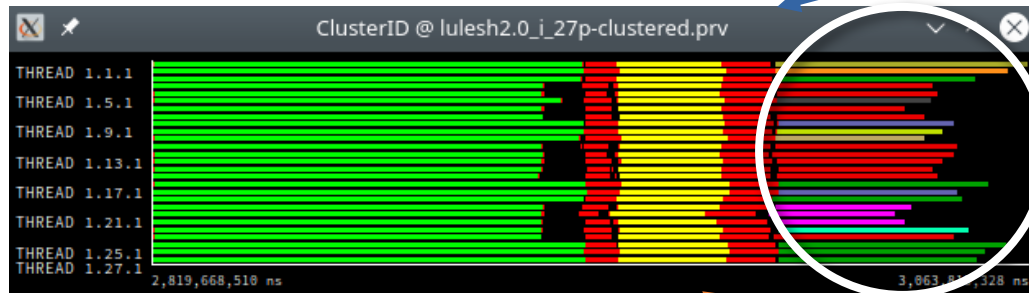
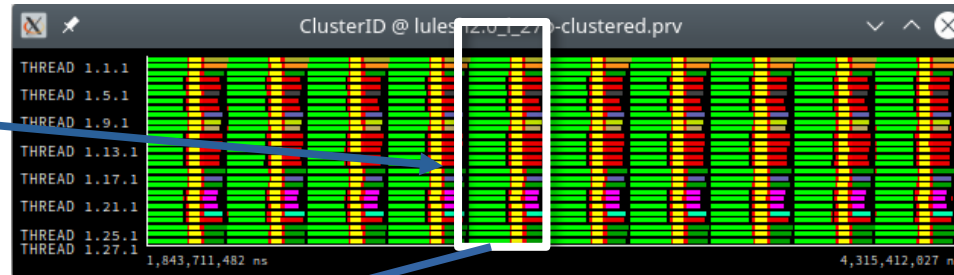
- Display the distribution of clusters over time
  - File `paraver/cfgs/clustering/clusterID_window.cfg`



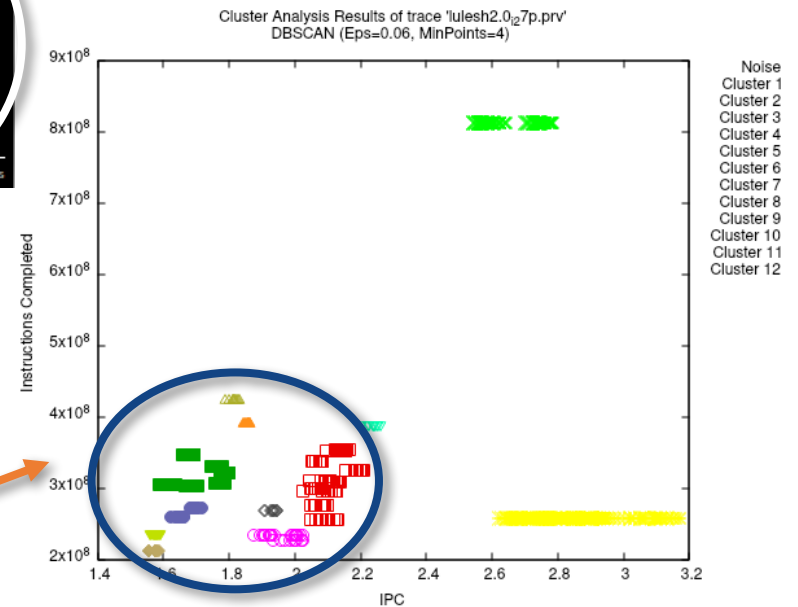
# Cluster-based analysis (III)

- Correlate scatter plots & timelines to detect imbalances

Zoom into  
1 of the iterations  
(by drag-and-  
dropping)



Variable work  
and/or  
Variable speed  
+  
Simultaneously @ different processes  
=  
Imbalances





**Barcelona  
Supercomputing  
Center**  
Centro Nacional de Supercomputación



# End Clustering Hands-On

Judit Giménez, Germán Llort, Lau Mercadal,  
Sandra Méndez

✉ [tools@bsc.es](mailto:tools@bsc.es)

19/04/2021

POP Performance Analysis