BSC Tools
Hands-On

Lau Mercadal, Germán Llort
(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	Jureca Cluster
Event	150 – 200ns	123ns
Event + PAPI	750 – 1000ns	920ns
Event + callstack (1 level)	1µs	1.439µs
Event + callstack (6 levels)	2µs	3.447µs

How does Extrae work?

- Symbol substitution through LD_PRELOAD
  - Specific libraries for each combination of runtimes
    - MPI, OpenMP, MPI+OpenMP, …

Recommended

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

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

Using Extrae in 3 steps

1. **Adapt** your job submission script
   - Append Extrae loader script

2. **Configure** what to trace
   - Modify `extrae.xml`

3. **Run** it!

- For further reference check the Extrae User Guide
  - https://tools.bsc.es -> Documentation -> Tools manuals
  - Also distributed with Extrae in `$EXTRAE_HOME/share/doc`

Using Extrae in JURECA

```
laptop> ssh <USER>@jureca.fz-juelich.de
jrlXX> cp -r tools-material $HOME
jrlXX> ls $HOME/tools-material
     apps/
     clustering/
     extrae/
     slides/
     traces/
```

Step 1: Append loader script

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

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

#SBATCH --ntasks=27
#SBATCH --nodes=2
#SBATCH --output=lulesh2.0_27p.out
#SBATCH --error=lulesh2.0_27p.err
#SBATCH --time=00:15:00
#SBATCH --partition=batch
#SBATCH --account=jzam11

module load Intel IntelMPI

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

Request resources

Run the program

Step 1: Append loader script

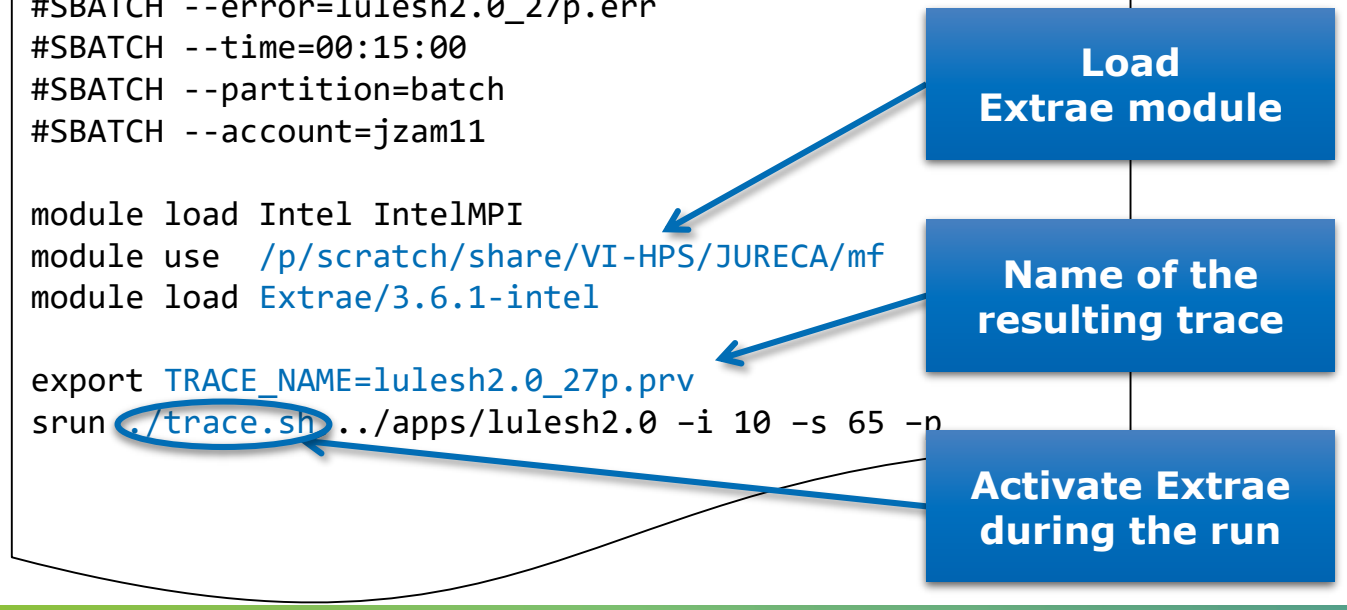
```
jrlXX> vi $HOME/tools-material/extrae/job.slurm
```

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

#SBATCH --ntasks=27
#SBATCH --nodes=2
#SBATCH --output=lulesh2.0_27p.out
#SBATCH --error=lulesh2.0_27p.err
#SBATCH --time=00:15:00
#SBATCH --partition=batch
#SBATCH --account=jzam11

module load Intel IntelMPI
module use  /p/scratch/share/VI-HPS/JURECA/mf
module load Extrae/3.6.1-intel

export TRACE_NAME=lulesh2.0_27p.prv
srun ./trace.sh ../apps/lulesh2.0 -i 10 -s 65 -p
```



Step 1: Append loader script

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

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

#SBATCH --ntasks=27
#SBATCH --nodes=2
#SBATCH --output=lulesh2.0_27p.out
#SBATCH --error=lulesh2.0_27p.err
#SBATCH --time=00:15:00
#SBATCH --partition=batch
#SBATCH --account=jzam11

module load Intel IntelMPI
module use  /p/scratch/share/VI-HPS/JURECA/mf
module load Extrae/3.6.1-intel

export TRACE_NAME=lulesh2.0_27p.prv
srun ./trace.sh ../apps/lulesh2.0 -i 10 -s 65 -p
```

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

# Configure Extrae
export EXTRAE_CONFIG_FILE=./extrae.xml


# Load the tracing library
export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitrace.so

# Run the program
$*
```

What to trace

Type of application

Step 1: Append loader script

- Choose depending on application type

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

[1] Append "f" suffix for Fortran codes

Step 3: Run it!

- Submit your job & check status

```
jrlXX> cd $HOME/tools-material/extrae
jrlXX> sbatch job.slurm
jrlXX> squeue -u $USER
```

- Once finished the trace will be in the same folder
  - `lulesh2.0_27p.{pcf,prv,row}`

- Any issue?
  - Traces already generated in `$HOME/tools-material/traces`

Step 2: Configure what to trace

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

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

<openmp enabled="no">
    <locks enabled="no" />
    <counters enabled="yes" />
</openmp>

<callers enabled="yes">
    <mpi enabled="yes">1-3</mpi>
    <sampling enabled="no">1-5</sampling>
</callers>
```

Trace MPI calls
(What is the program doing?)

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

Compile with debug (-g)

Step 2: Configure what to trace

```
jrlXX> 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,PAPI_L1_DCM,PAPI_L2_DCM
        </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: Configure what to trace

```
jrlXX> vi $HOME/tools-material/extrae/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 (Flush / memory trade-off)

Enable periodic sampling (More details)

Automatic post-processing to generate the Paraver trace

Installing Paraver
&
First analysis steps

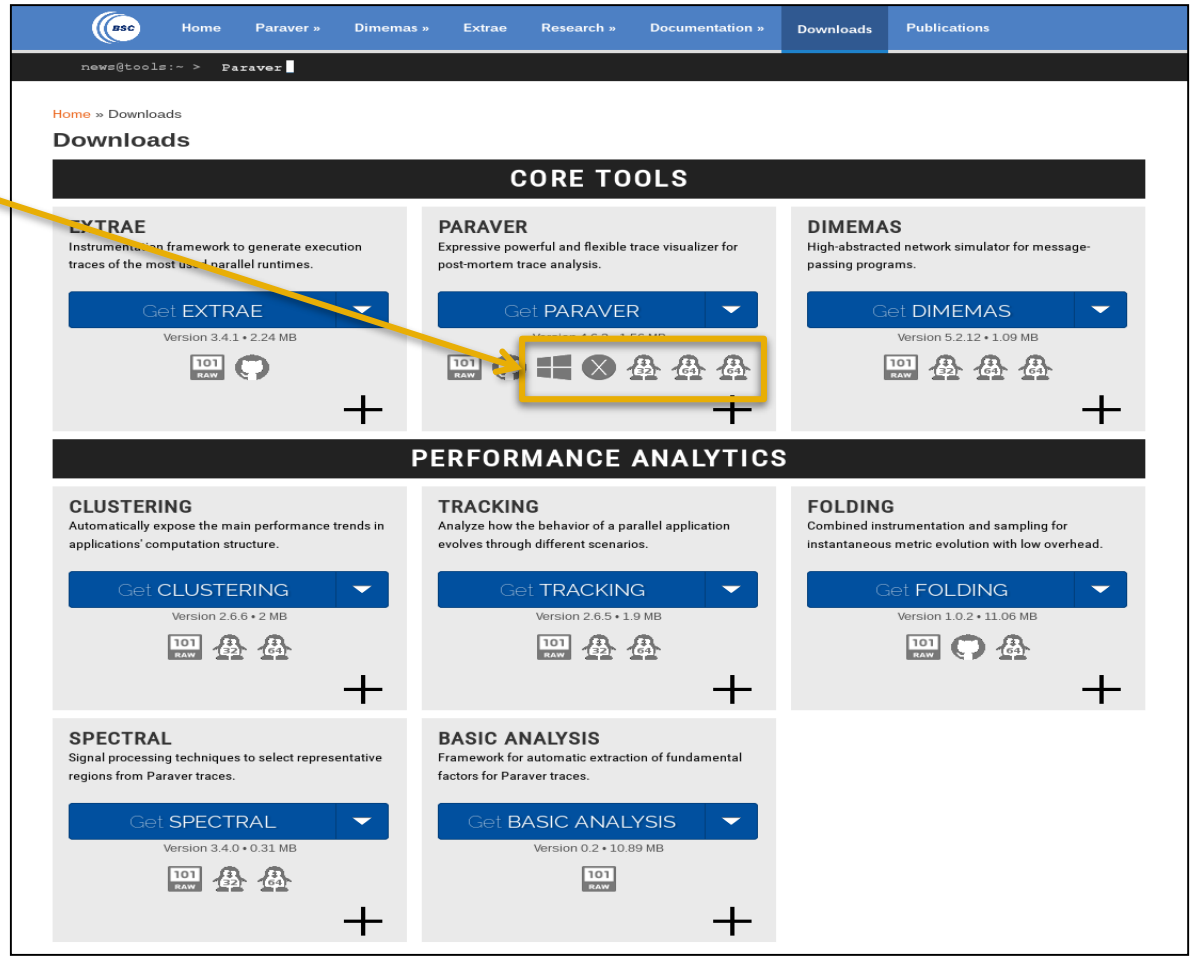
Install Paraver in your laptop

- Download a binary for your OS
  - https://tools.bsc.es/downloads

wxparaver-4.8.1-win.zip

wxparaver-4.8.1-mac.zip

wxparaver-4.8.1-Linux_i686.tar.gz (32bit)
wxparaver-4.8.1-Linux_x86_64.tar.gz (64bit)



Install Paraver tutorials

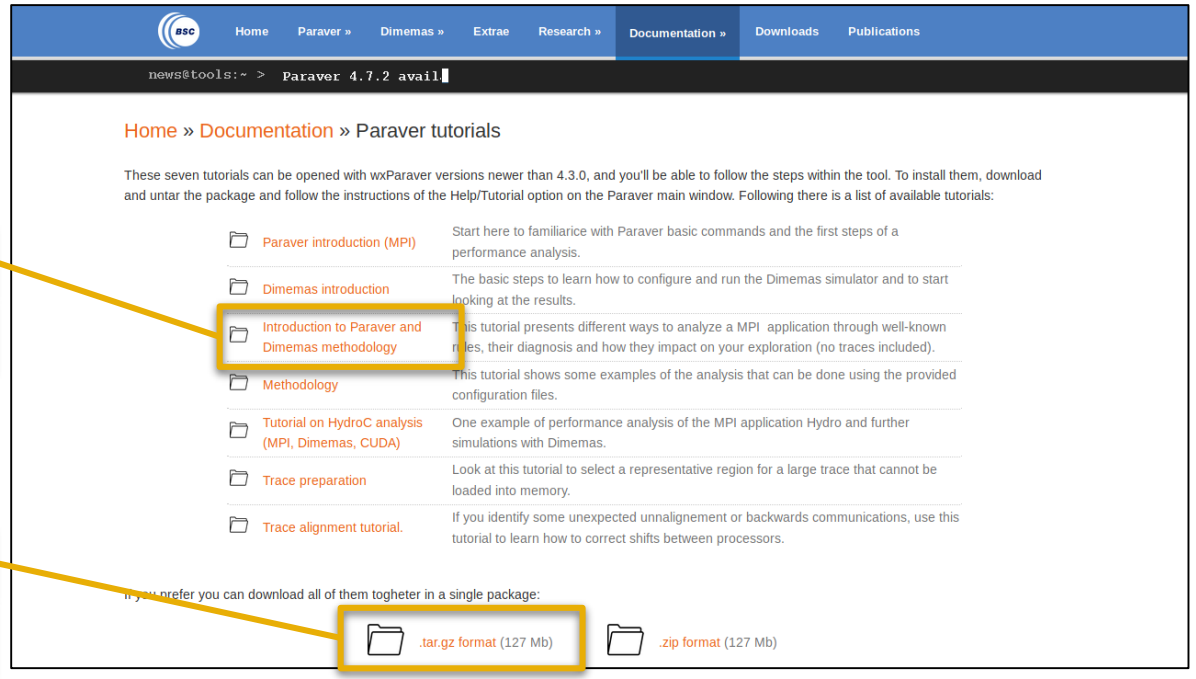
- Download tutorials archive
  - https://tools.bsc.es/paraver-tutorials

Tutorial 3

```
3.introduction_to_paraver_and_dimemas_methodology.tar.gz
```

All tutorials

```
paraver-tutorials-20150526.tar.gz
```



Uncompress, rename & move

- Paraver

```
laptop> tar xf wxparaver-4.8.1-Linux_x86_64.tar.bz2
laptop> mv wxparaver-4.8.1-Linux_x86_64 paraver
```

- Tutorials

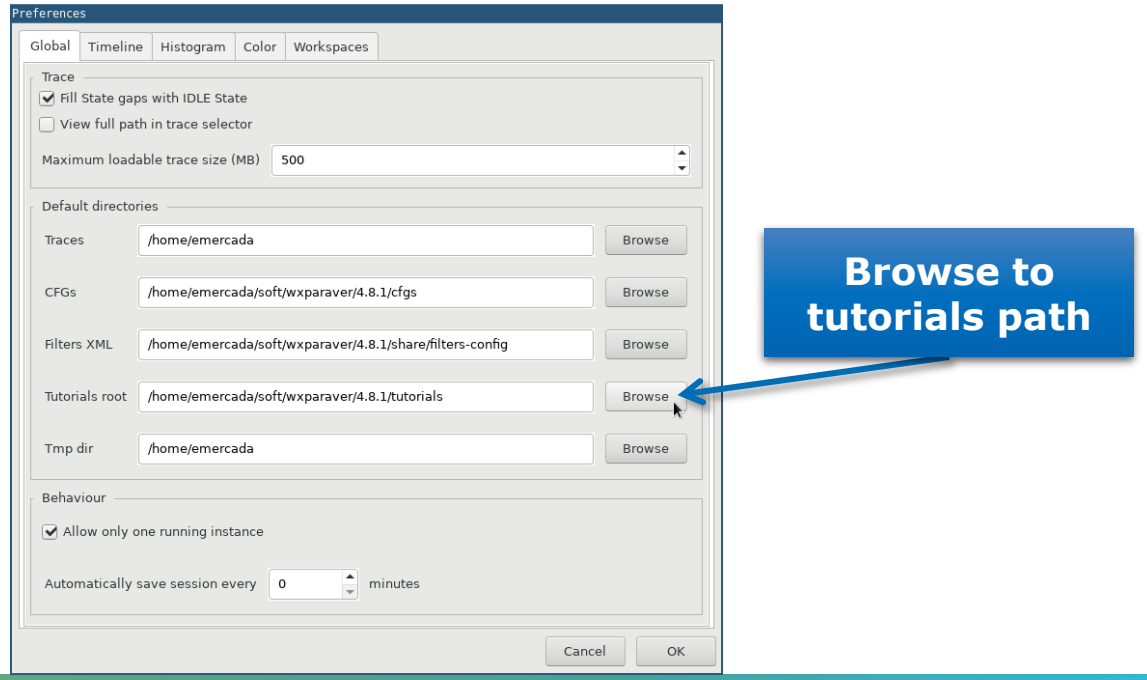
```
laptop> tar xf paraver-tutorials-20150526.tar.bz2
laptop> mv paraver-tutorials-20150526 paraver/tutorials
```

Tell Paraver where to find the tutorials

- Start Paraver

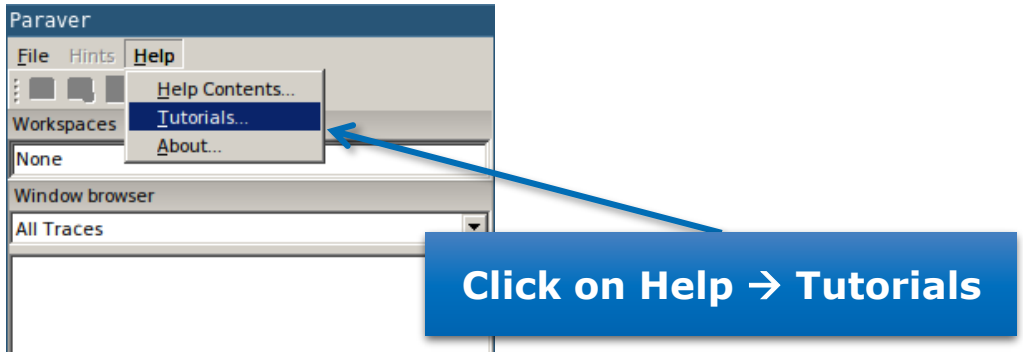
```
laptop> $HOME/paraver/bin/wxparaver &
```

- Check the tutorials are available



Check that everything works

- Check the tutorials are available



- Also available in JURECA

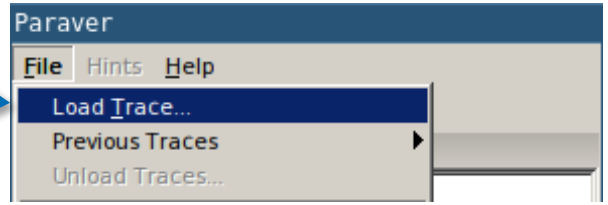
```
jrlXX> module load Paraver/4.8.1
```

First steps of the analysis

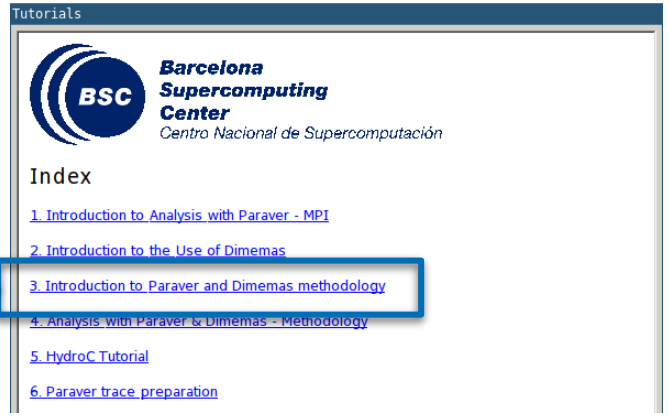
- Copy the trace to your laptop

```
laptop> scp <USER>@jureca:$HOME/tools-material/extrae/lulesh2.0_27p.{prv,pcf,row} ./
```

- Load the trace
  - File -> Load trace -> Browse to `.prv` file

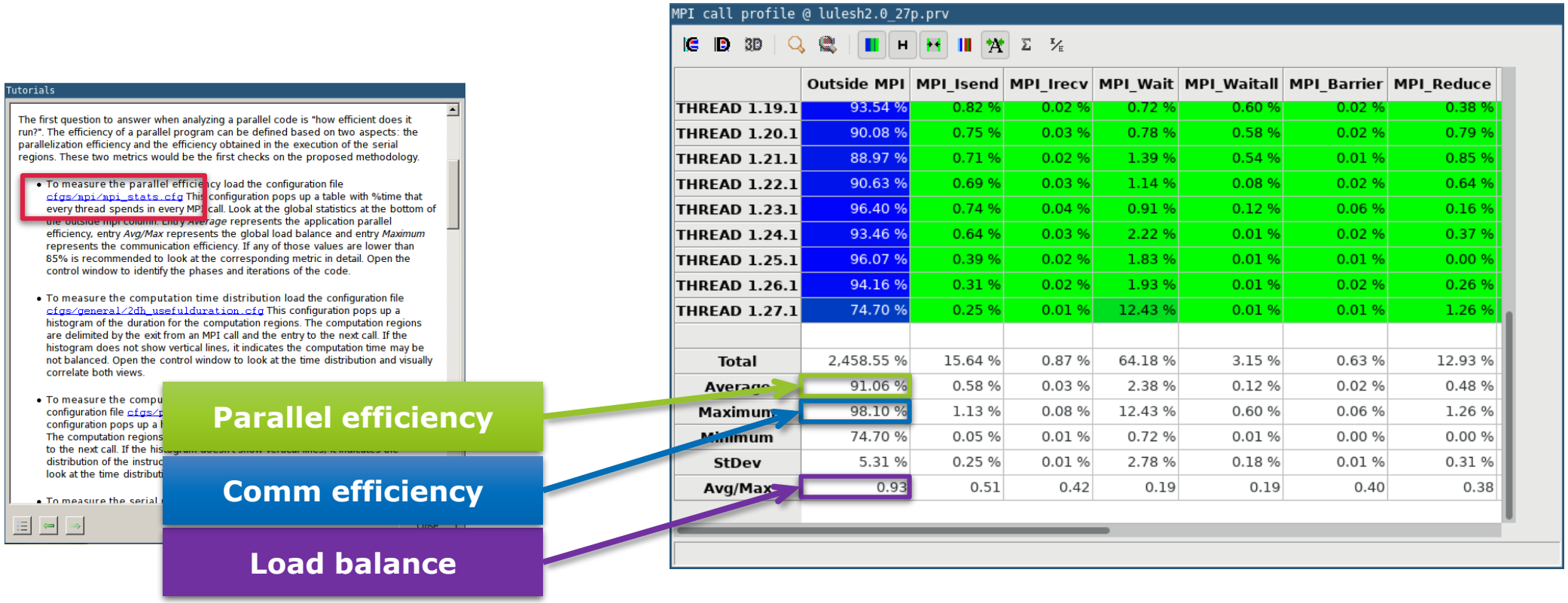


- Follow tutorial #3
  - Help -> Tutorials



Measure the parallel efficiency

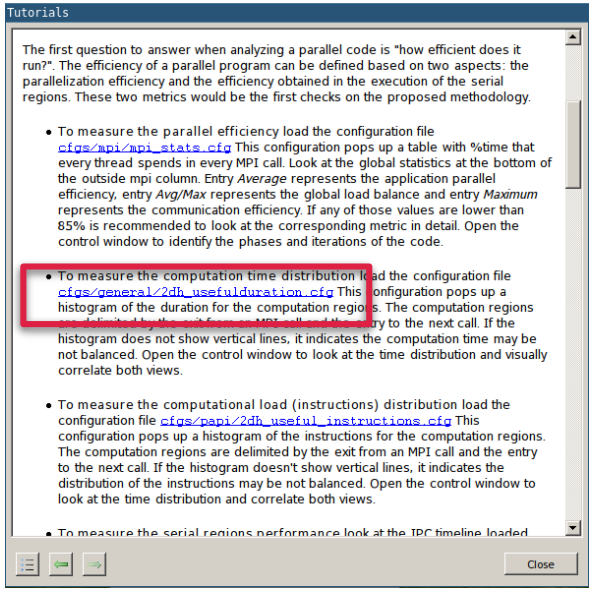
- Click on `mpi_stats.cfg`

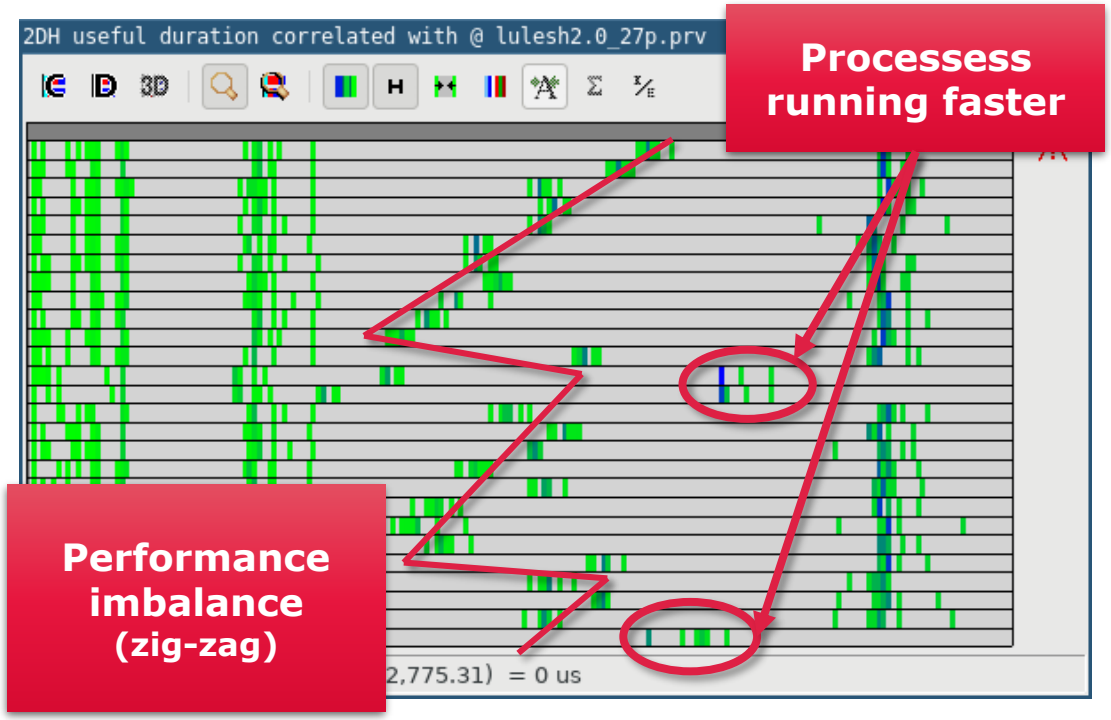


	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Barrier	MPI_Reduce
THREAD 1.19.1	93.54 %	0.82 %	0.02 %	0.72 %	0.60 %	0.02 %	0.38 %
THREAD 1.20.1	90.08 %	0.75 %	0.03 %	0.78 %	0.58 %	0.02 %	0.79 %
THREAD 1.21.1	88.97 %	0.71 %	0.02 %	1.39 %	0.54 %	0.01 %	0.85 %
THREAD 1.22.1	90.63 %	0.69 %	0.03 %	1.14 %	0.08 %	0.02 %	0.64 %
THREAD 1.23.1	96.40 %	0.74 %	0.04 %	0.91 %	0.12 %	0.06 %	0.16 %
THREAD 1.24.1	93.46 %	0.64 %	0.03 %	2.22 %	0.01 %	0.02 %	0.37 %
THREAD 1.25.1	96.07 %	0.39 %	0.02 %	1.83 %	0.01 %	0.01 %	0.00 %
THREAD 1.26.1	94.16 %	0.31 %	0.02 %	1.93 %	0.01 %	0.02 %	0.26 %
THREAD 1.27.1	74.70 %	0.25 %	0.01 %	12.43 %	0.01 %	0.01 %	1.26 %
Total	2,458.55 %	15.64 %	0.87 %	64.18 %	3.15 %	0.63 %	12.93 %
Average	91.06 %	0.58 %	0.03 %	2.38 %	0.12 %	0.02 %	0.48 %
Maximum	98.10 %	1.13 %	0.08 %	12.43 %	0.60 %	0.06 %	1.26 %
Minimum	74.70 %	0.05 %	0.01 %	0.72 %	0.01 %	0.00 %	0.00 %
StDev	5.31 %	0.25 %	0.01 %	2.78 %	0.18 %	0.01 %	0.31 %
Avg/Max	0.93	0.51	0.42	0.19	0.19	0.40	0.38

Computation time and work distribution

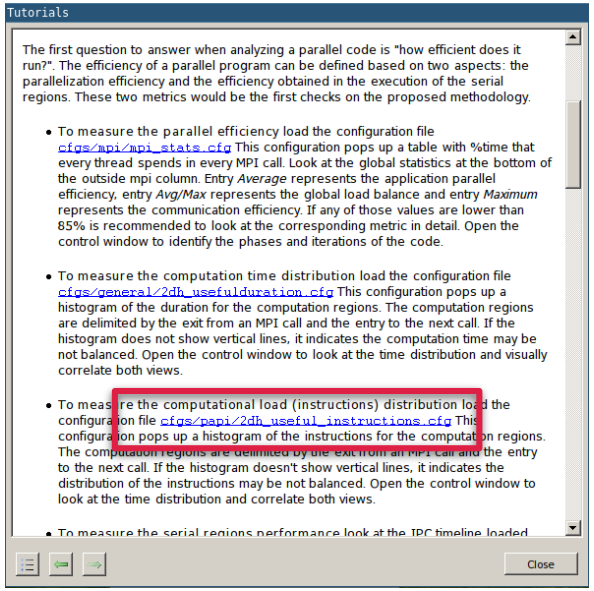
- Click on `2dh_usefulduration.cfg` (2nd link) ➔ Shows **time computing**

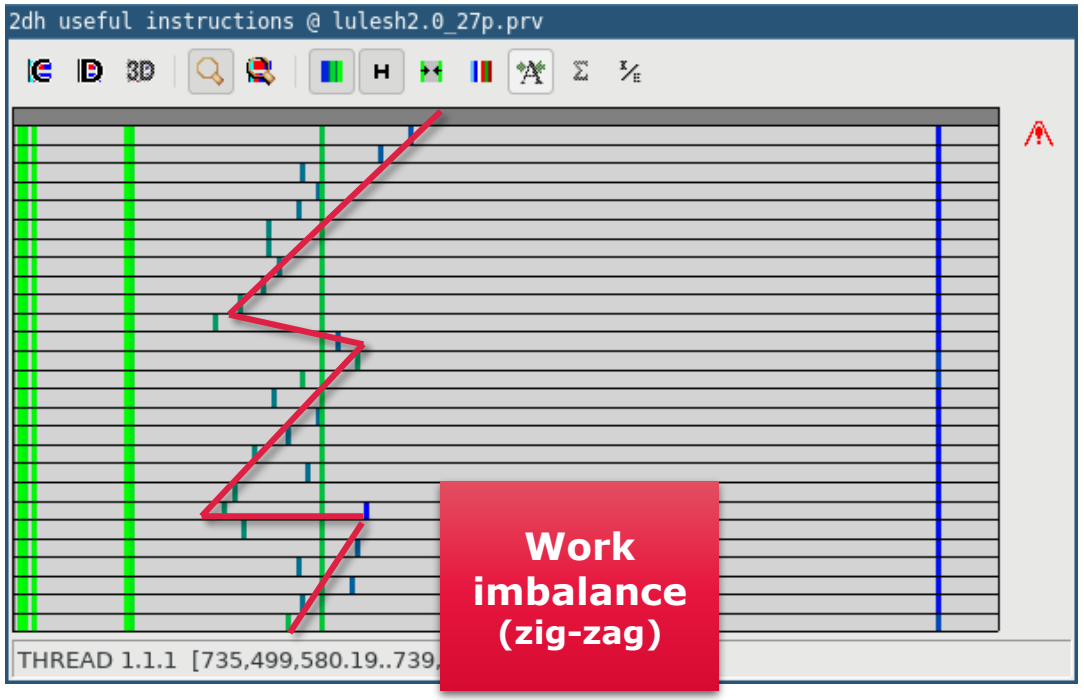




Computation time and work distribution

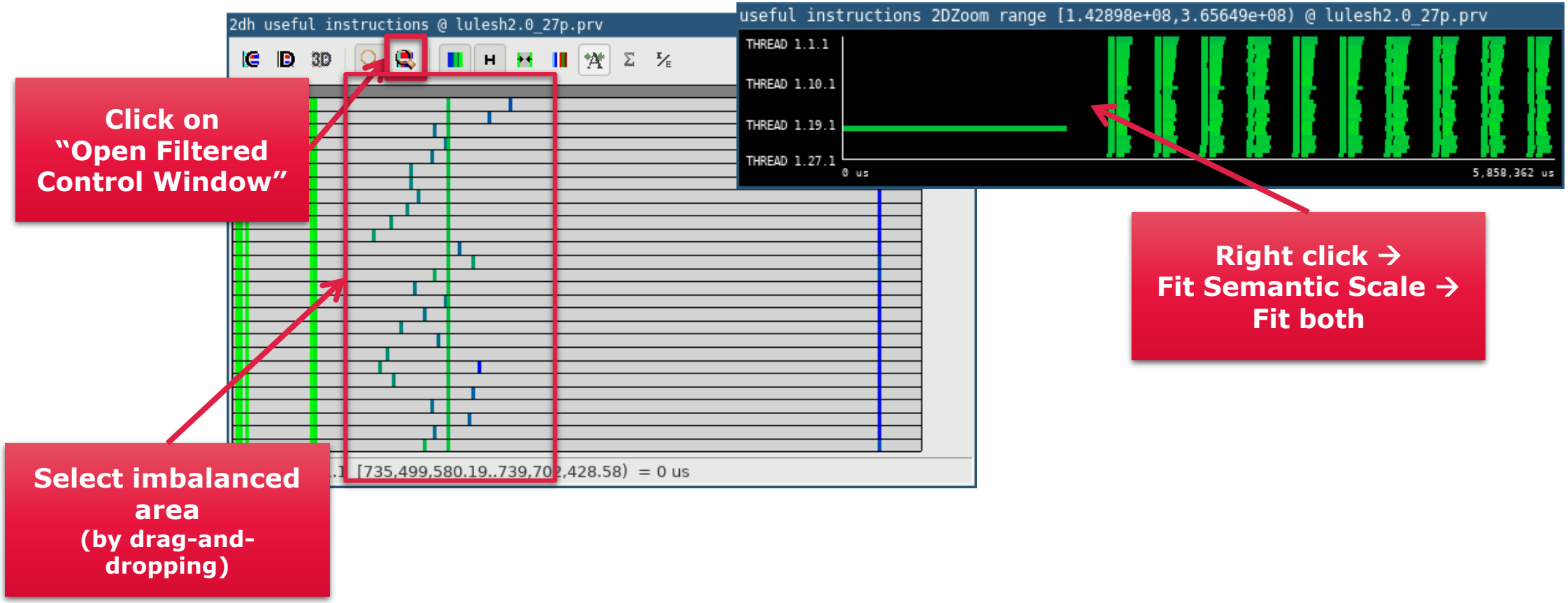
- … and `2dh_useful_instructions.cfg` (3rd link) ➔ Shows **amount of work**





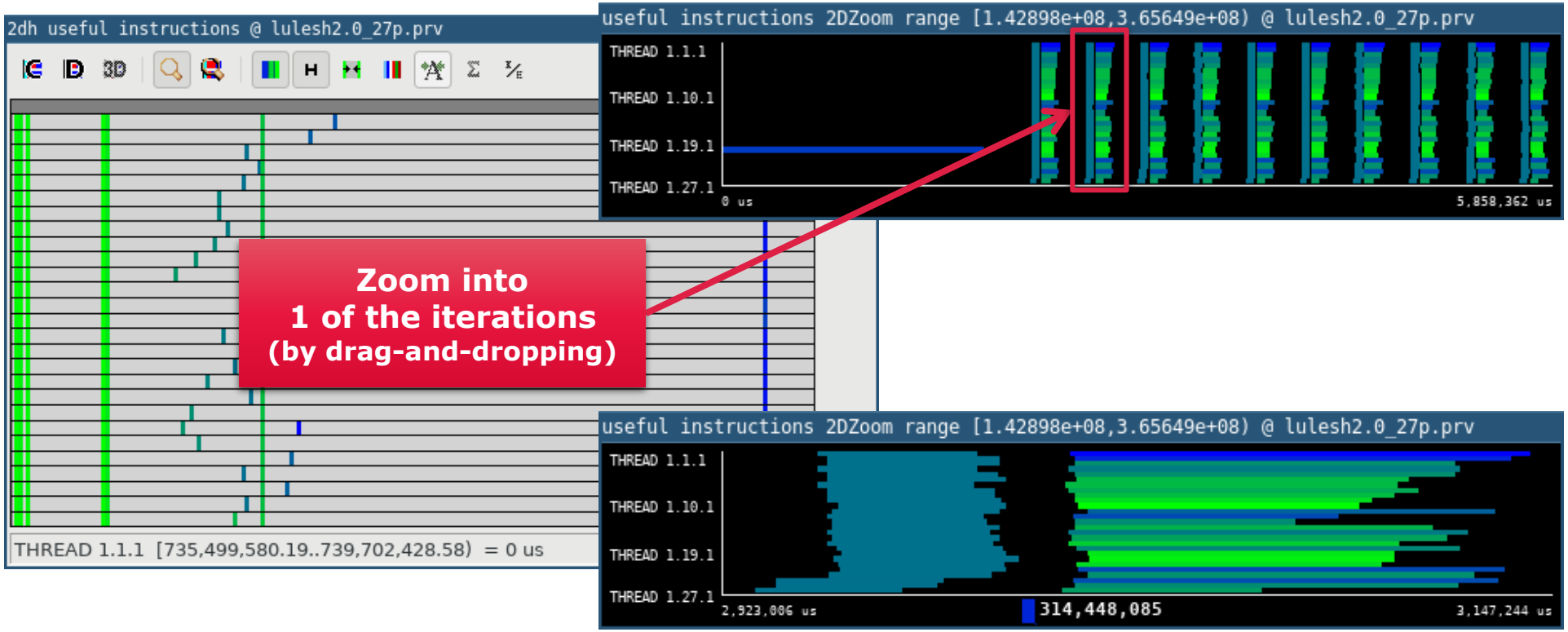
Where does this happen?

- Go from tables to timelines



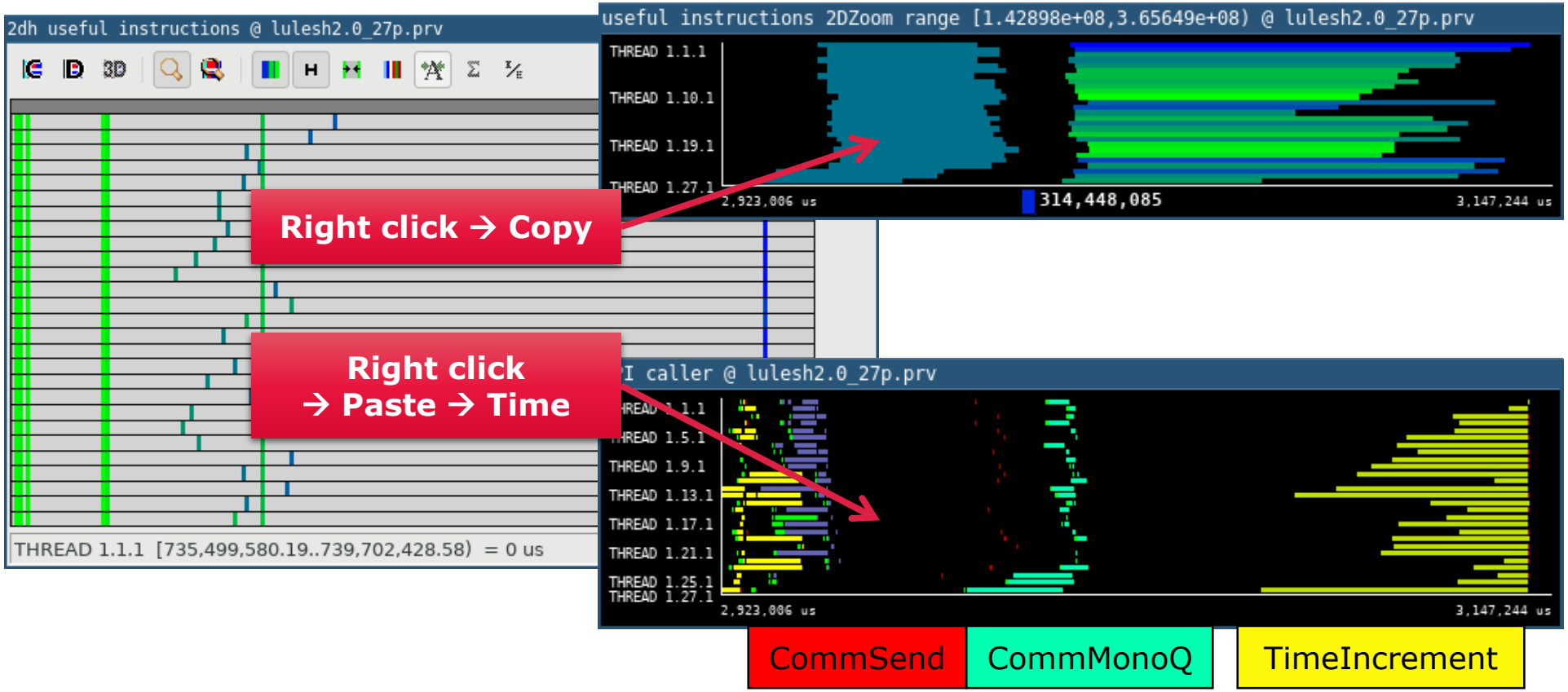
Where does this happen

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

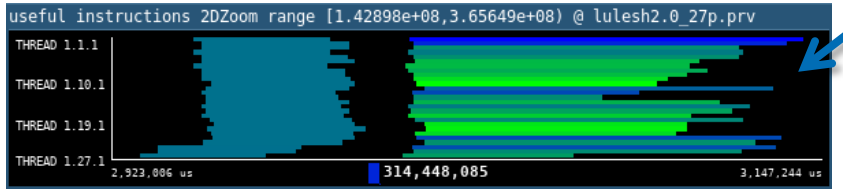


Where does this happen

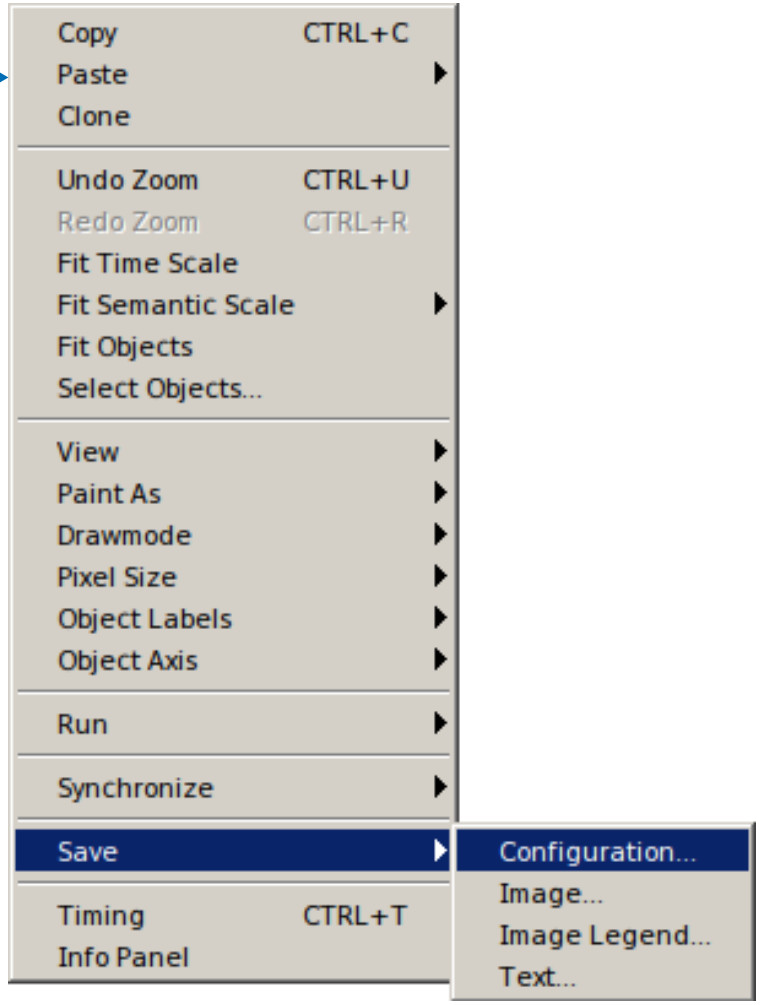
- Hints → Callers → Caller function



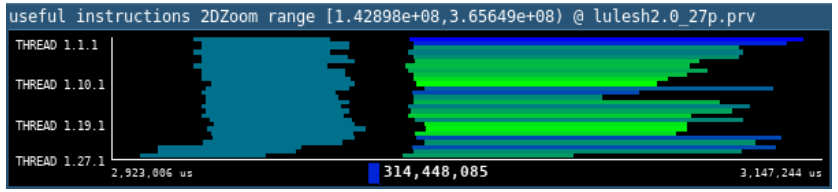
Save CFG's (2 methods)



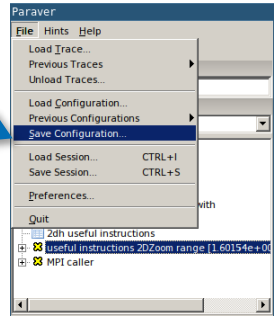
Right click on timeline

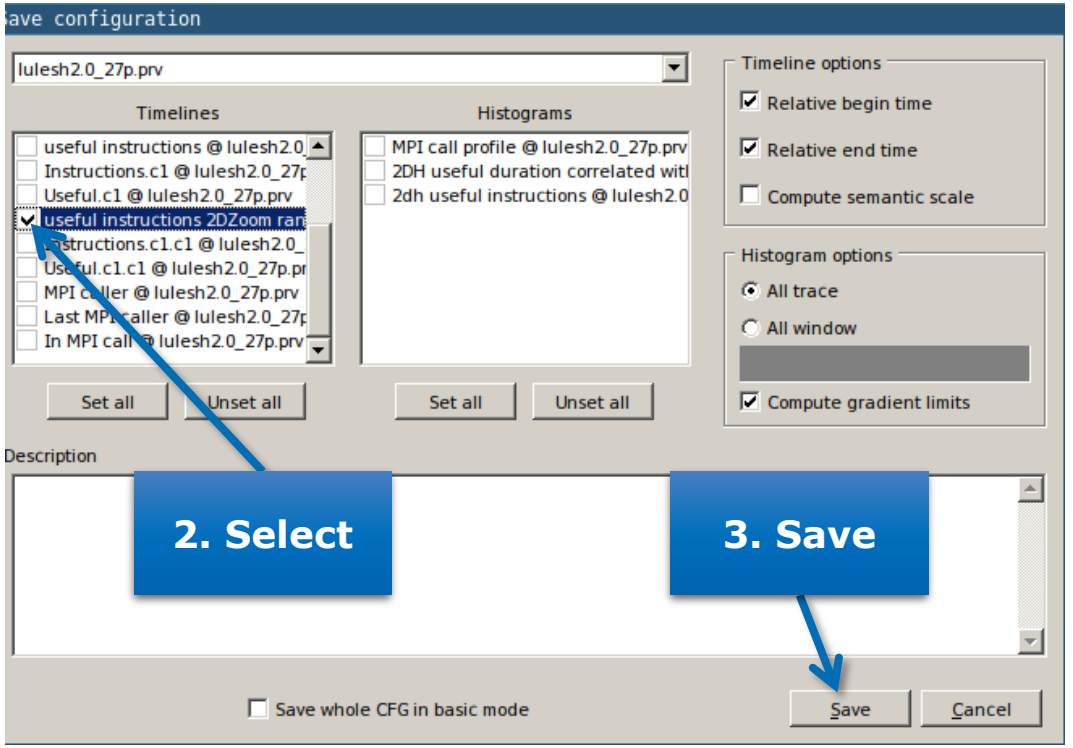


Save CFG's (2 methods)



1. Main Paraver window



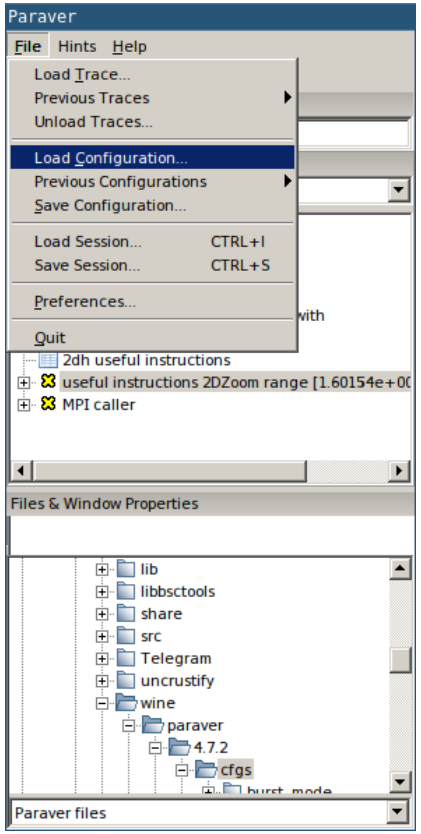


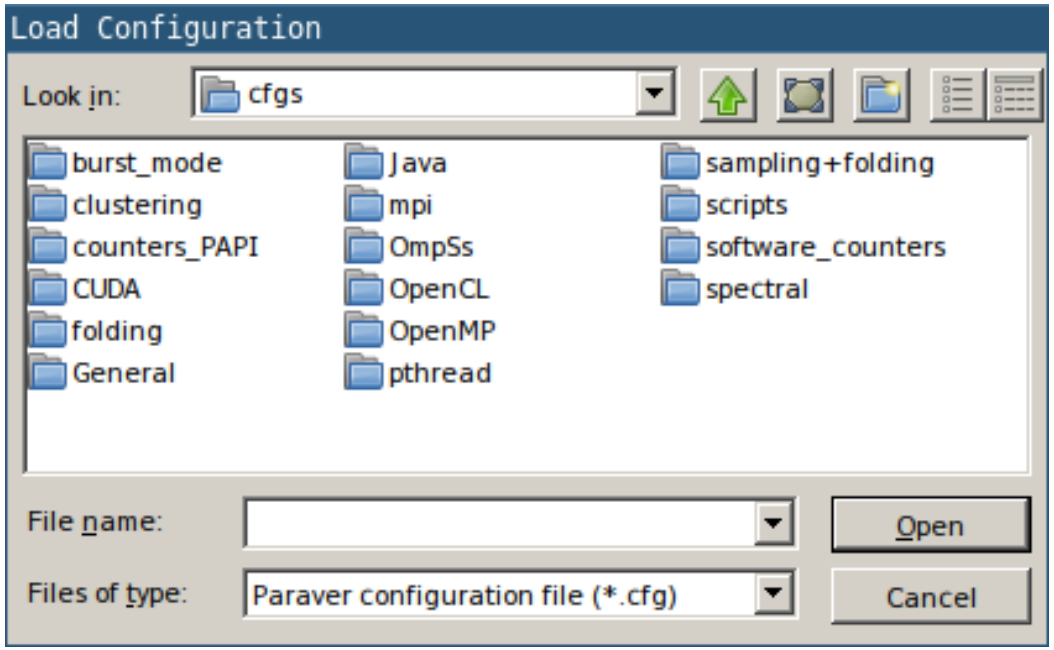
2. Select

3. Save

CFG's distribution

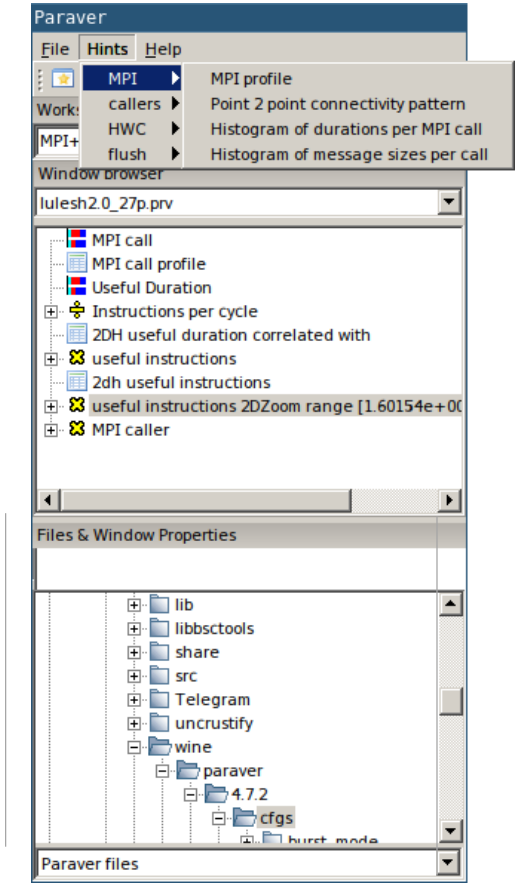
- Paraver comes with many more included CFG's





Hints: a good place to start!

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



Cluster-based analysis

Install Clustering in your laptop

- Download a binary for your OS
  - https://tools.bsc.es/downloads

```
laptop> tar xf clusteringsuite-2.6.8-Linux_x86_64.tar.bz2
laptop> mv clusteringsuite-2.6.8-Linux_x86_64 clustering
```

- Also available in JURECA

```
jrlXX> module load ClusteringSuite/2.6.8
```

Use clustering analysis

- Run clustering

```
laptop> cd $HOME/tools-material/clustering
laptop> $HOME/clustering/bin/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:

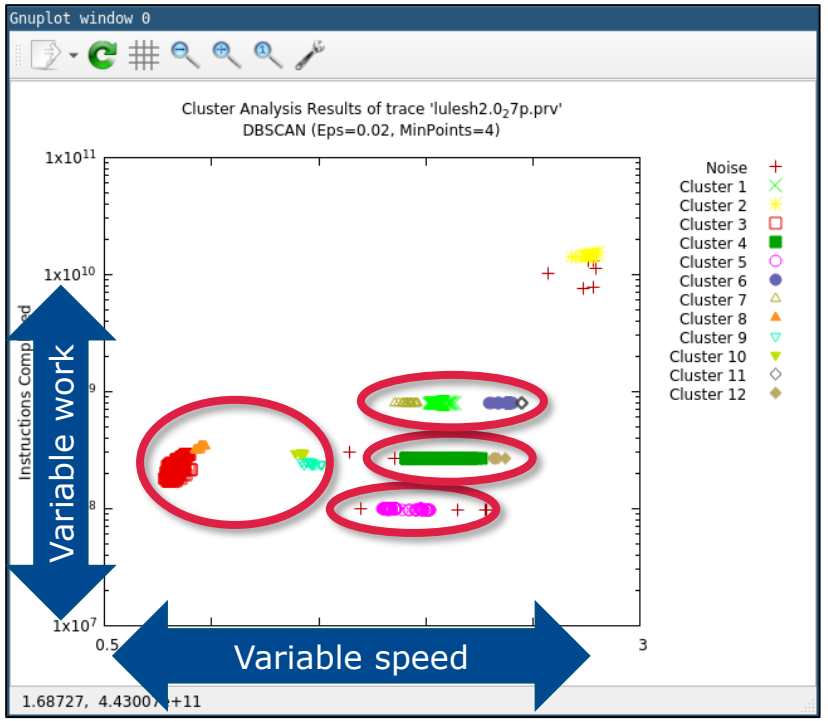
```
jrlXX> ls $HOME/tools-material/traces/lulesh2.0_27p.prv
```

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**

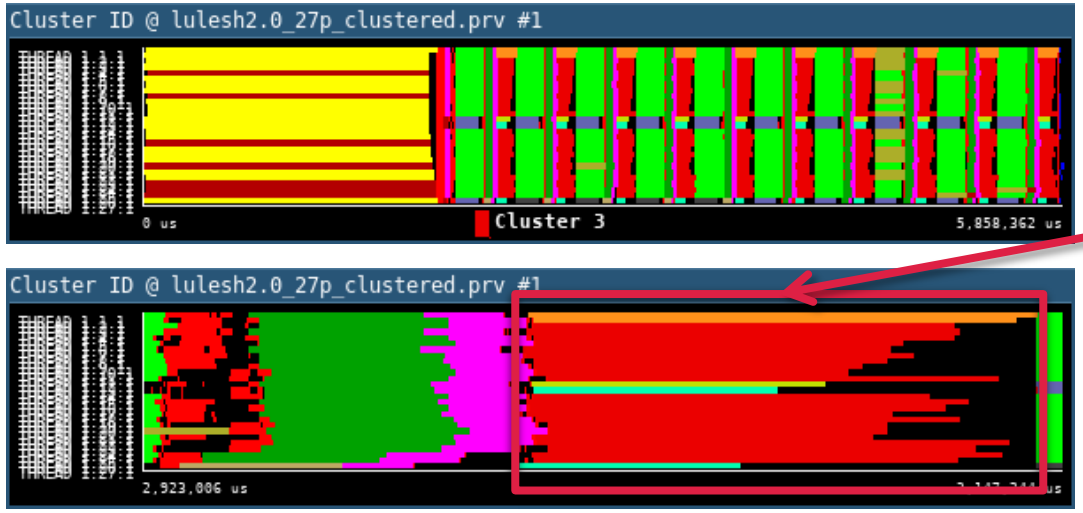


Correlating scatter plot and time distribution

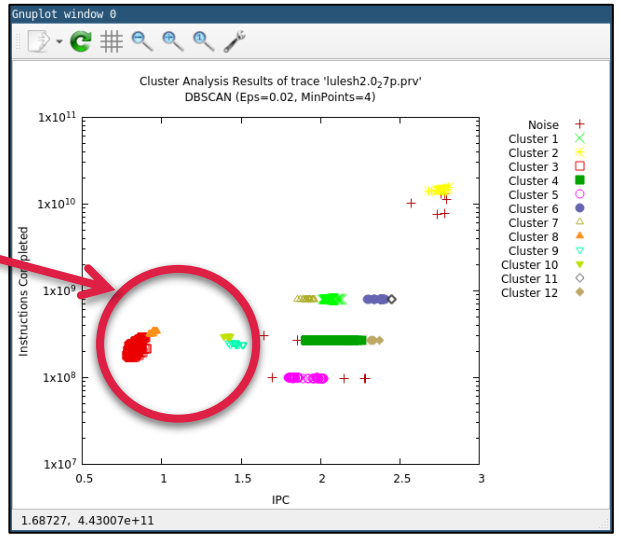
- Open the clustered trace with Paraver and look at it

```
laptop> $HOME/paraver/bin/wxparaver <path-to>/lulesh2.0_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, Germán Llort
(tools@bsc.es)
Barcelona Supercomputing Center