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)	1µs	2µs
Event + callstack (6 levels)	2µs	5µs

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)

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

module load gcc/8.1.0 openmpi/2.0.4.1.1_gnu

export OMP_NUM_THREADS=1

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

Request resources

Select MPI version

Run the program

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

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

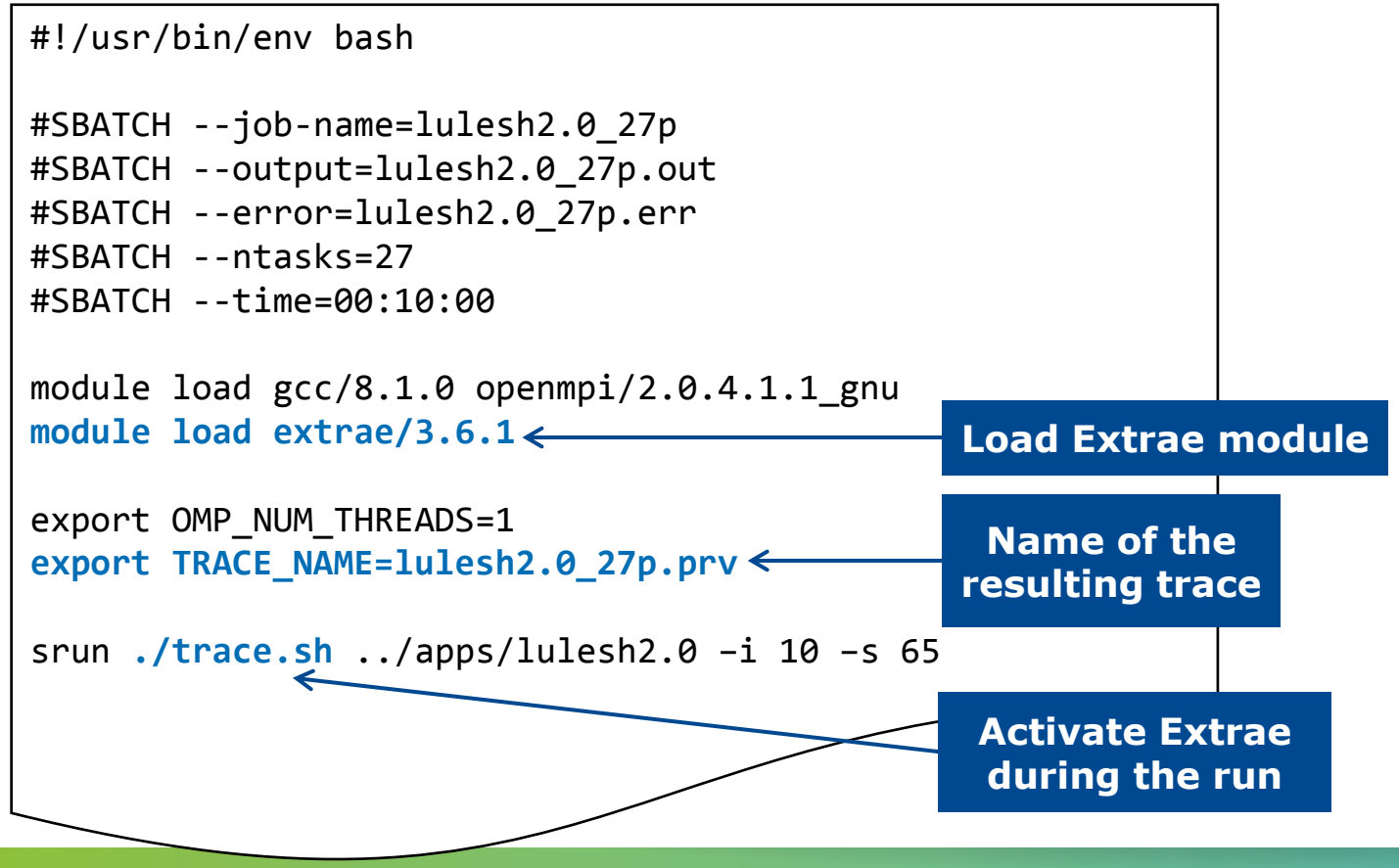
```
#!/usr/bin/env bash

#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
```



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/libmpitracef.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]		✓			
libomptrace			✓		
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>

<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)

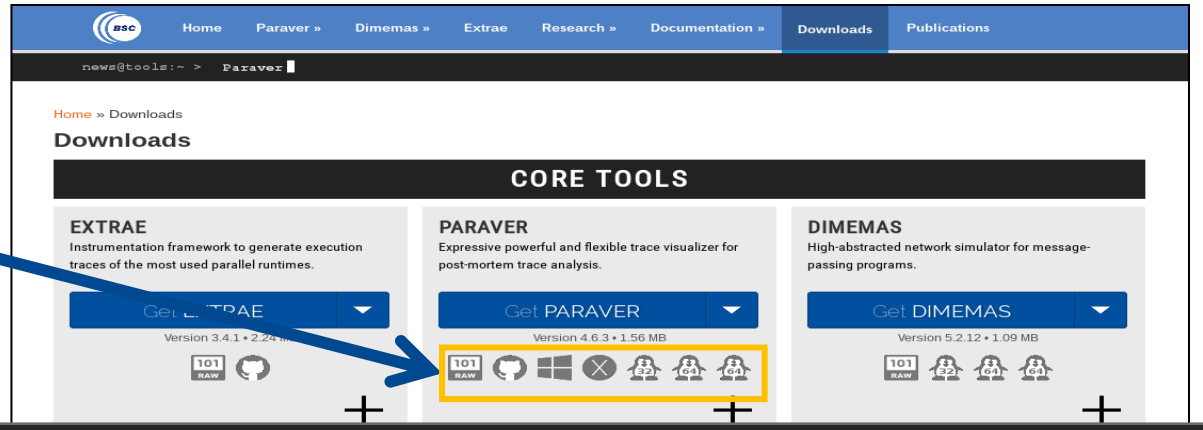
Enable sampling (Want more details?)

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

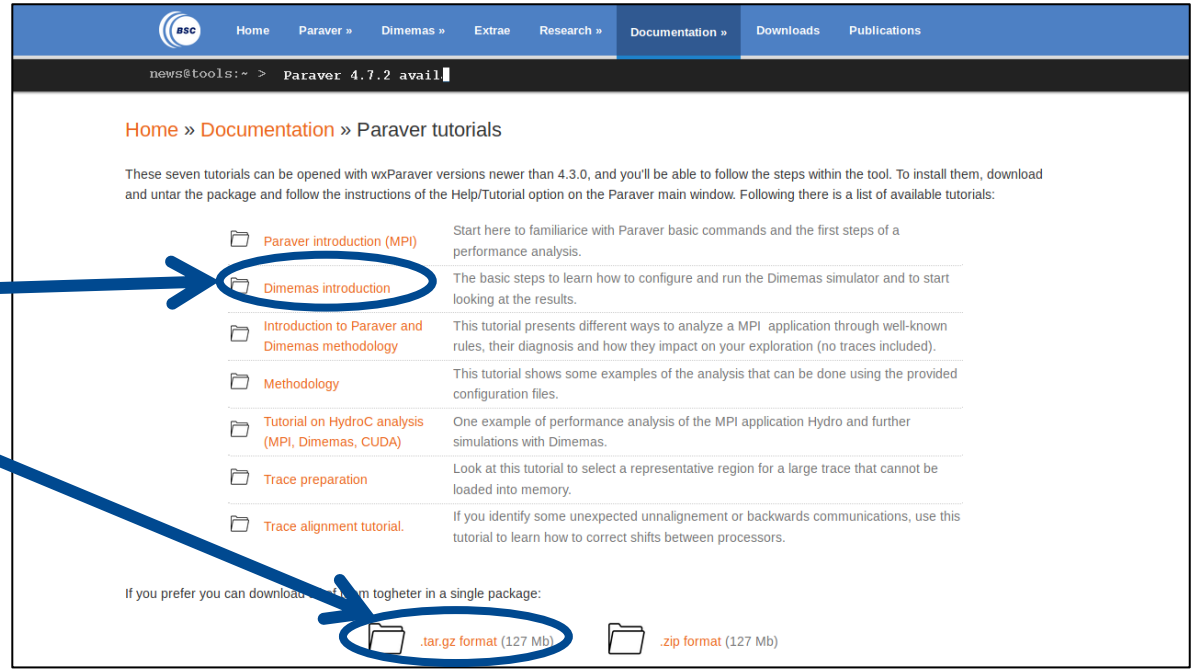
wxparaver-4.7.2-Linux_i686.tar.gz (32-bits)
wxparaver-4.7.2-Linux_x86_64.tar.gz (64-bits)

```
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

```
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

- Remotely available in Romeo

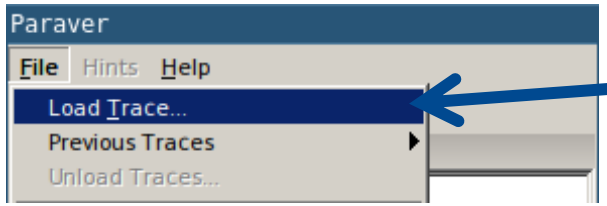
```
laptop> ssh -Y <USER>@romeologin1.univ-reims.fr
romeo> /home/laumercadal/wxparaver/latest/bin/wxparaver
```

First steps of analysis

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

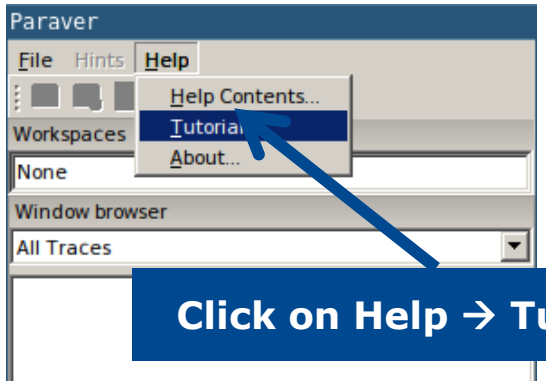
```
laptop> scp <USER>@romeologin1.univ-reims.fr:$HOME/tools-material/extrae/lulesh2.0* ./
```

- Load the trace

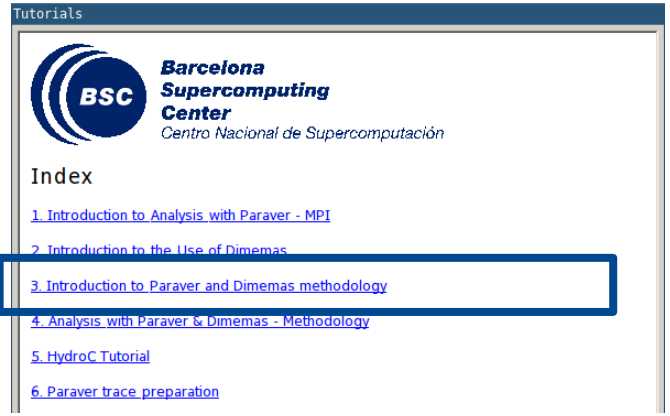


Click on File → Load Trace → Browse to the *.prv file

- Follow Tutorial #3

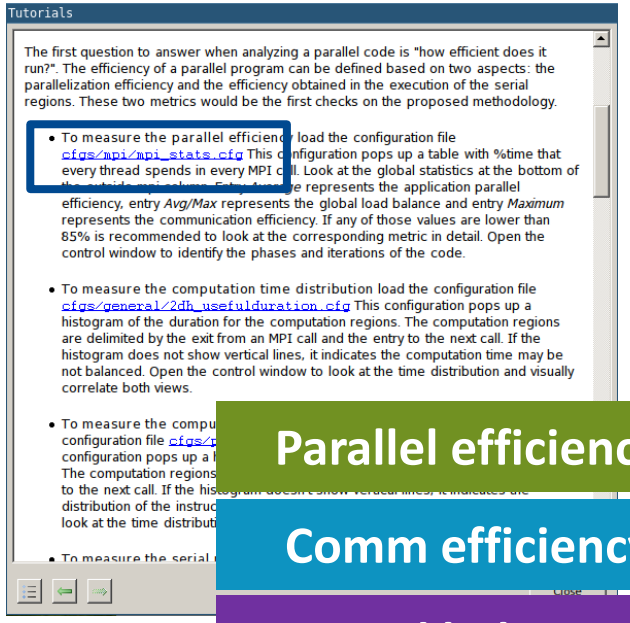


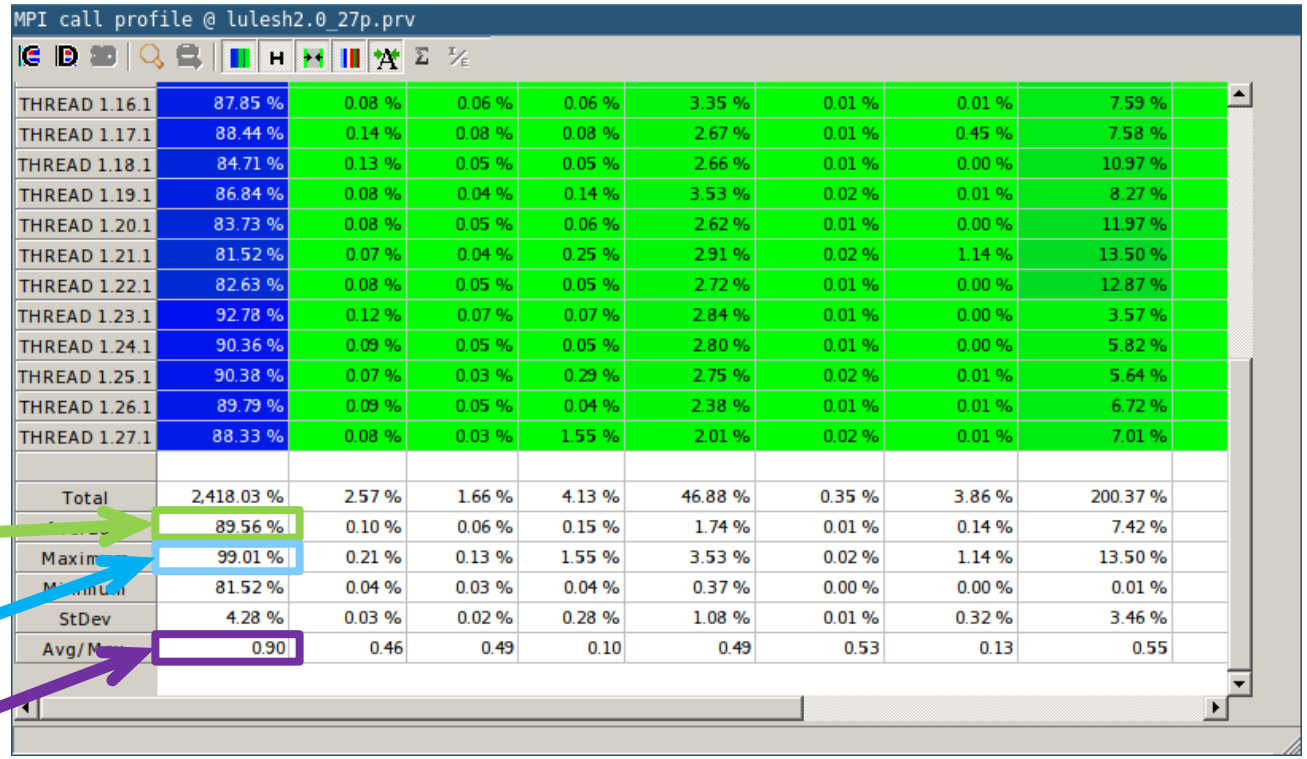
Click on Help → Tutorials



Measure the parallel efficiency

- Click on “mpi_stats.cfg”





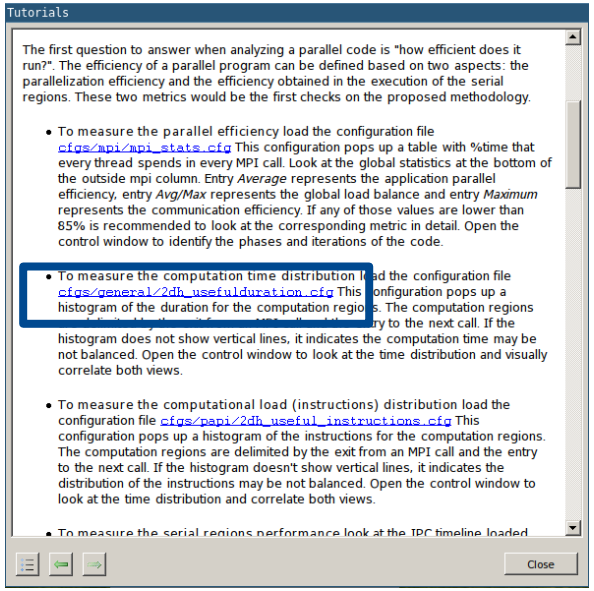
Parallel efficiency

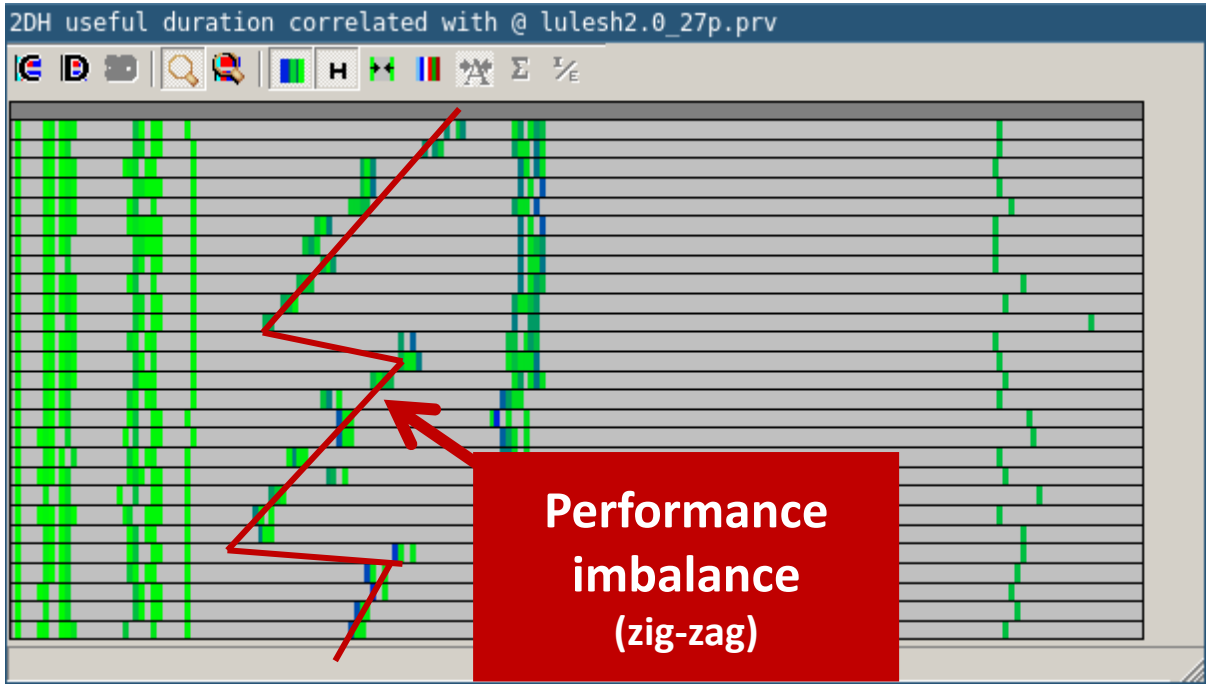
Comm efficiency

Load balance

Computation time and work distribution

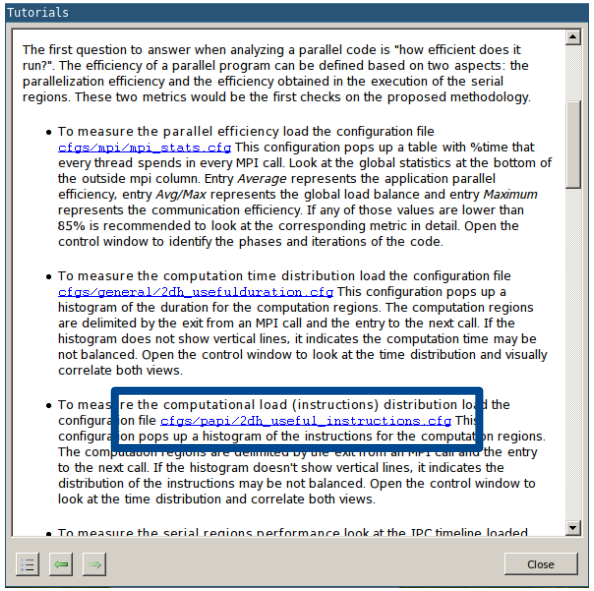
- Click on “2dh_usefulduration.cfg” (2nd link) ➔ Shows **time computing**

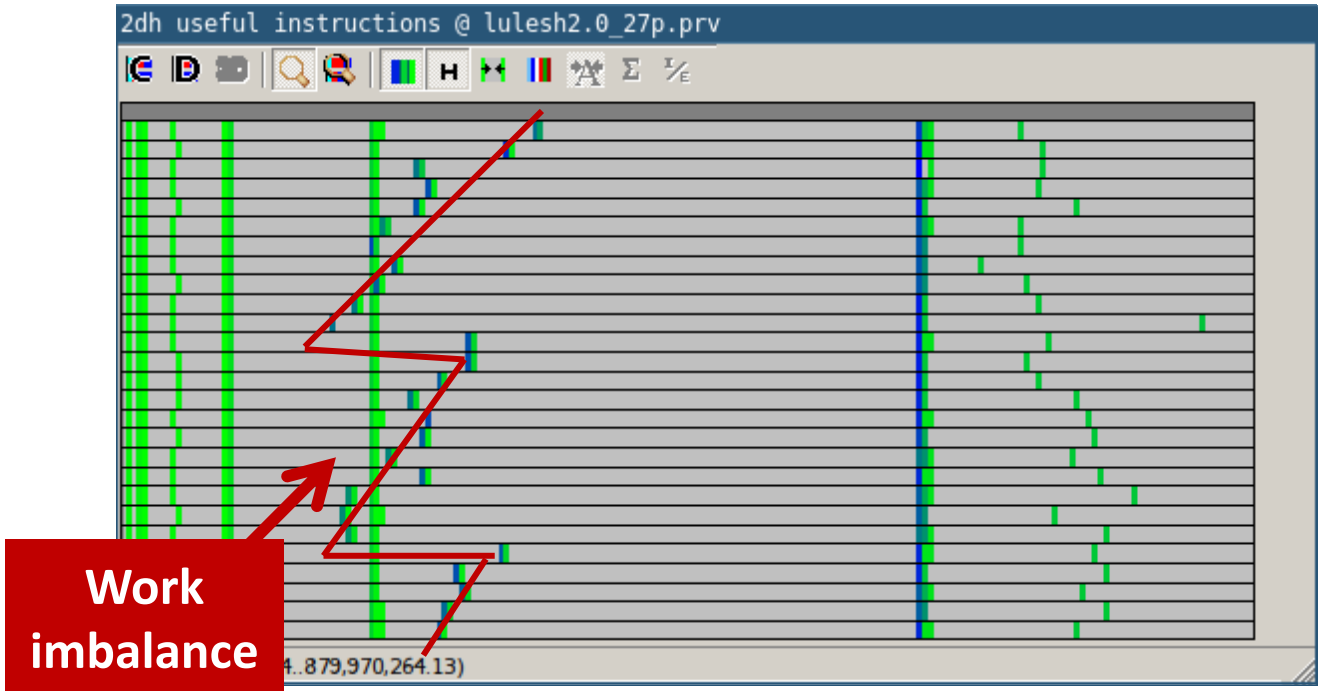




Computation time and work distribution

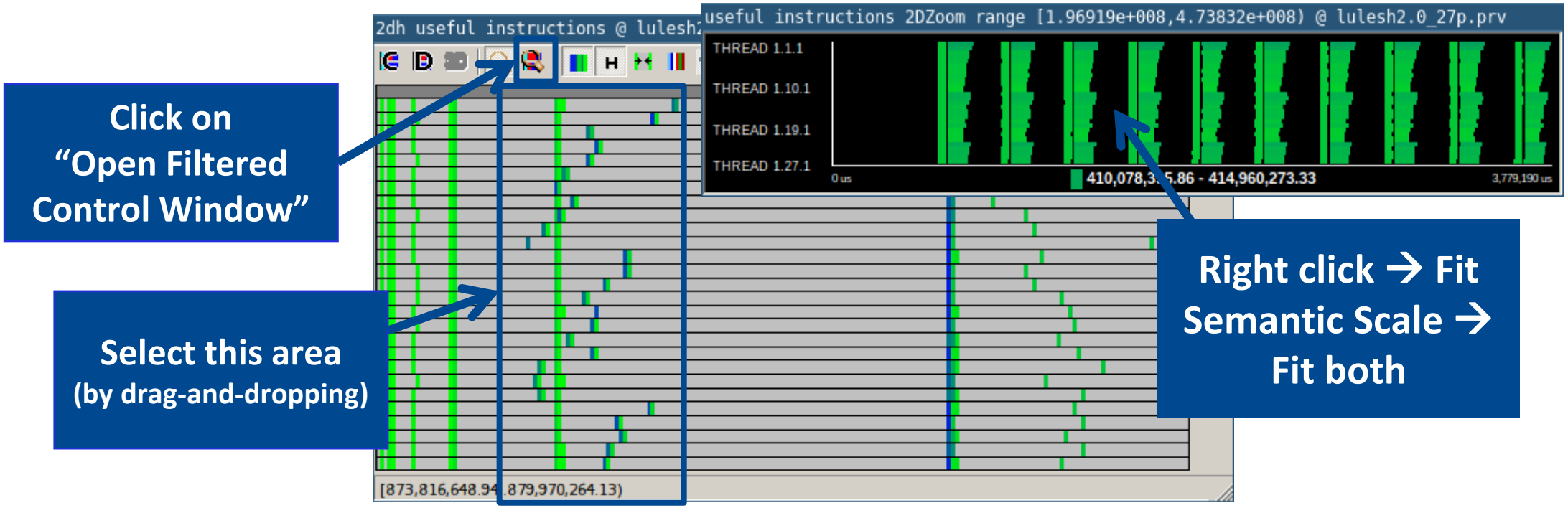
- … and "2dh_useful_instructions.cfg" (3rd link) ➔ Shows **amount of work**





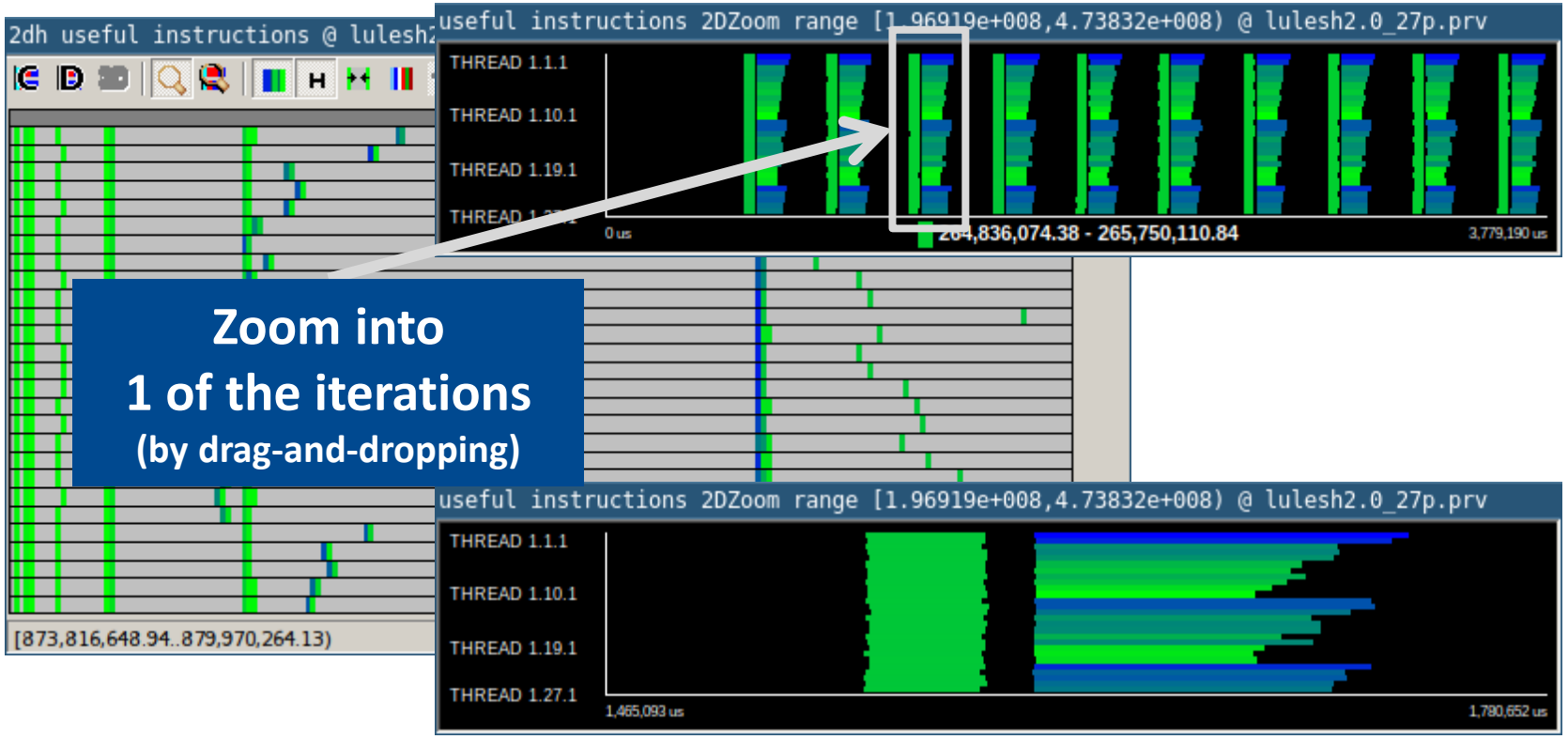
Where does this happen?

- Go from the table to the timeline



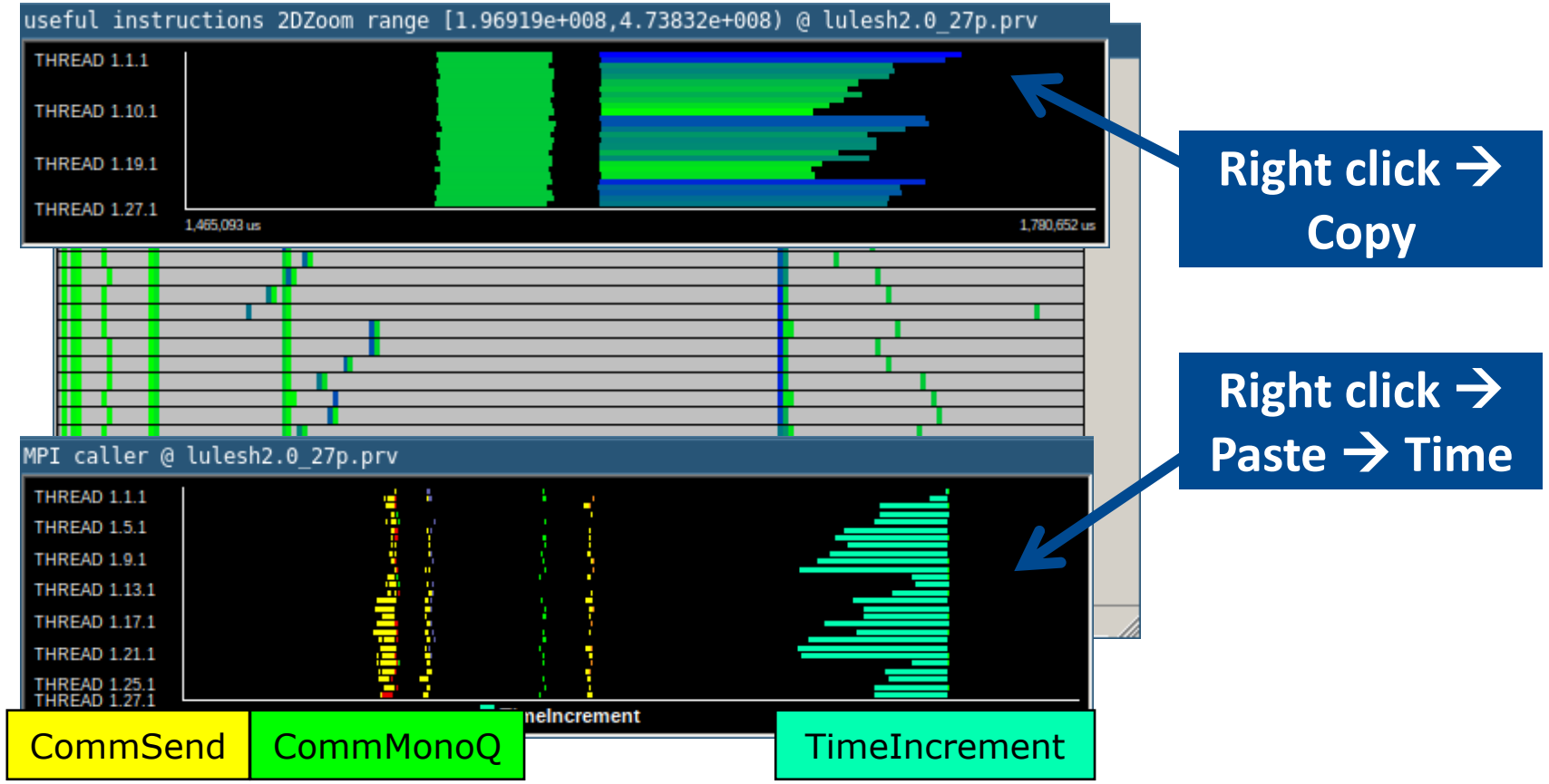
Where does this happen?

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

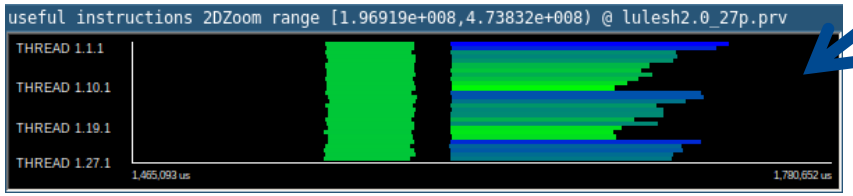


Where does this happen?

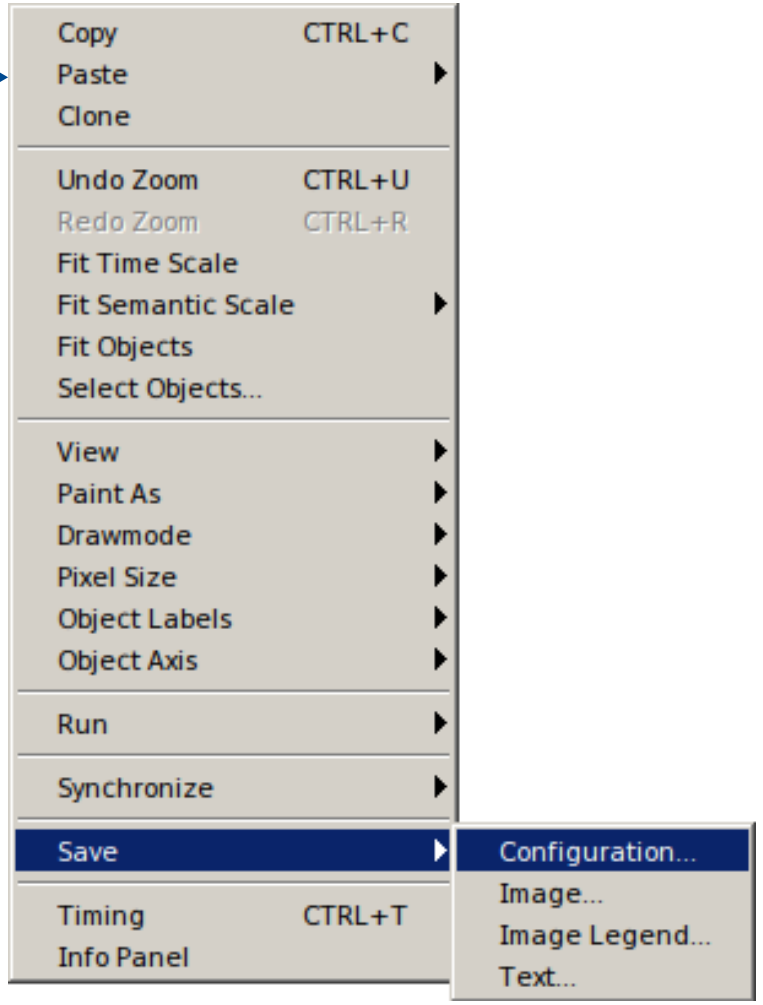
- Hints → Callers → Caller function



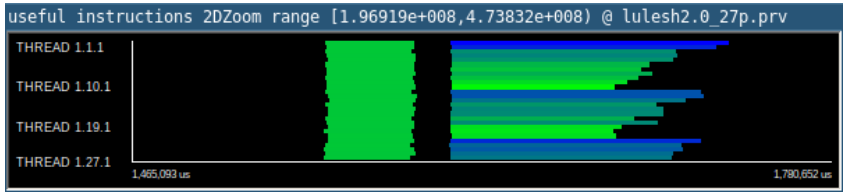
Save CFG's (2 methods)



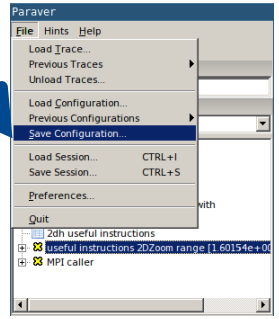
Right click on timeline

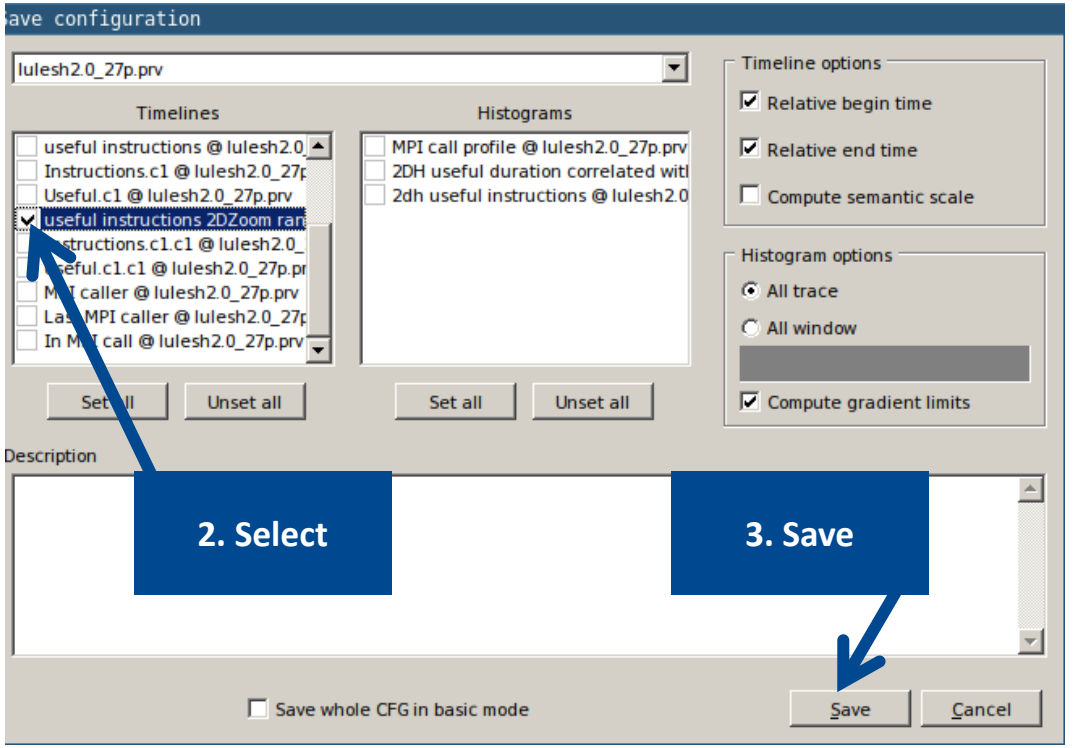


Save CFG's (2 methods)



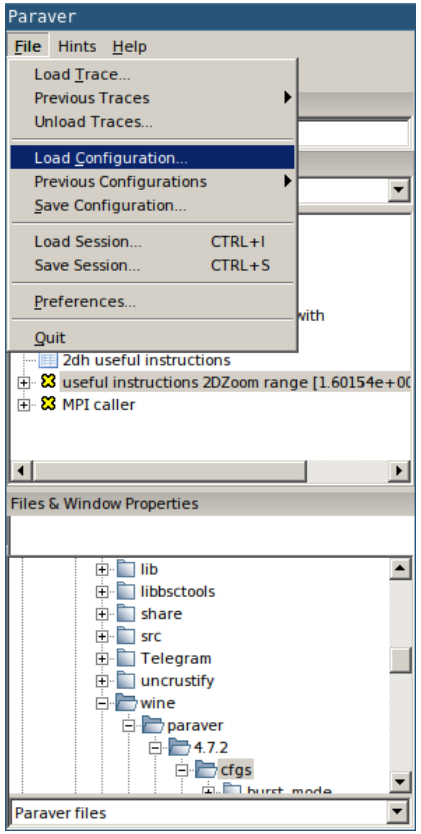
1. Main Paraver window

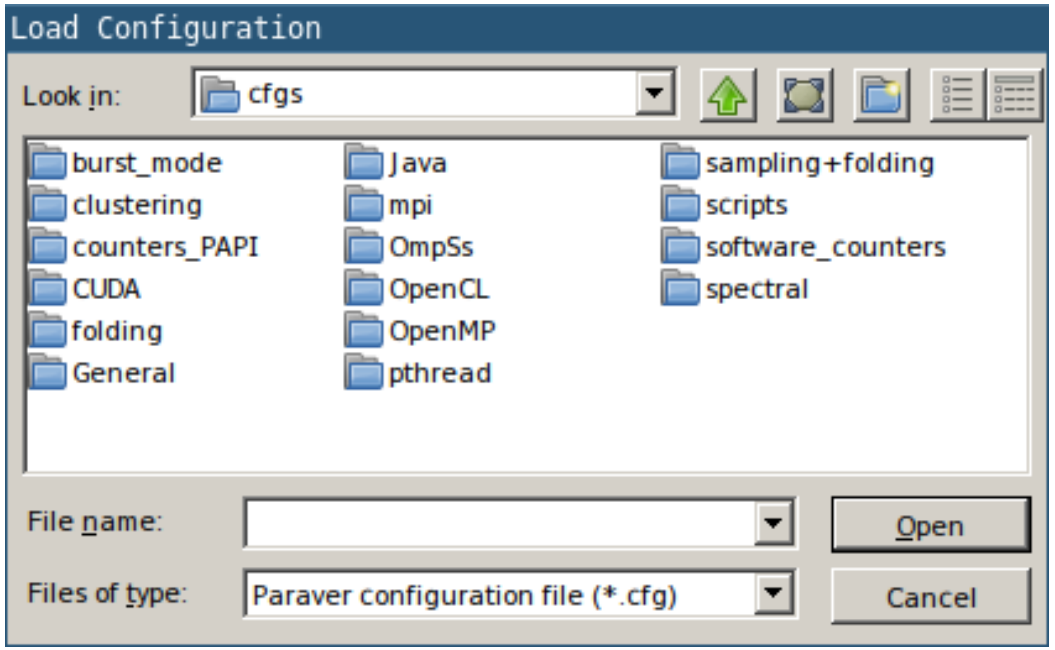




CFG's distribution

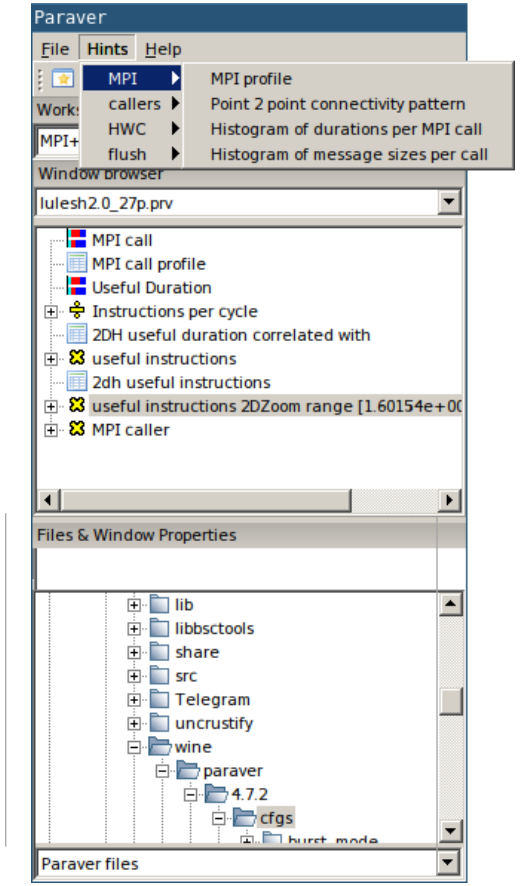
- Paraver comes with many more included CFG's





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

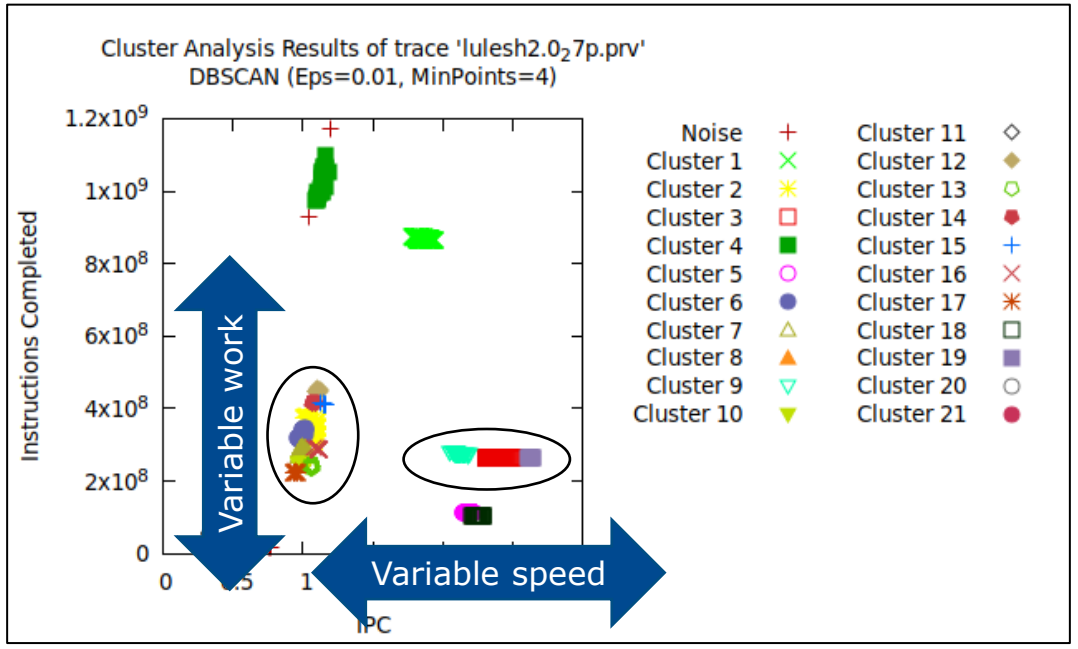
```
laptop> scp <USER>@romeologin1.univ-reims.fr:$HOME/tools-material/clustering/* ./
```

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**

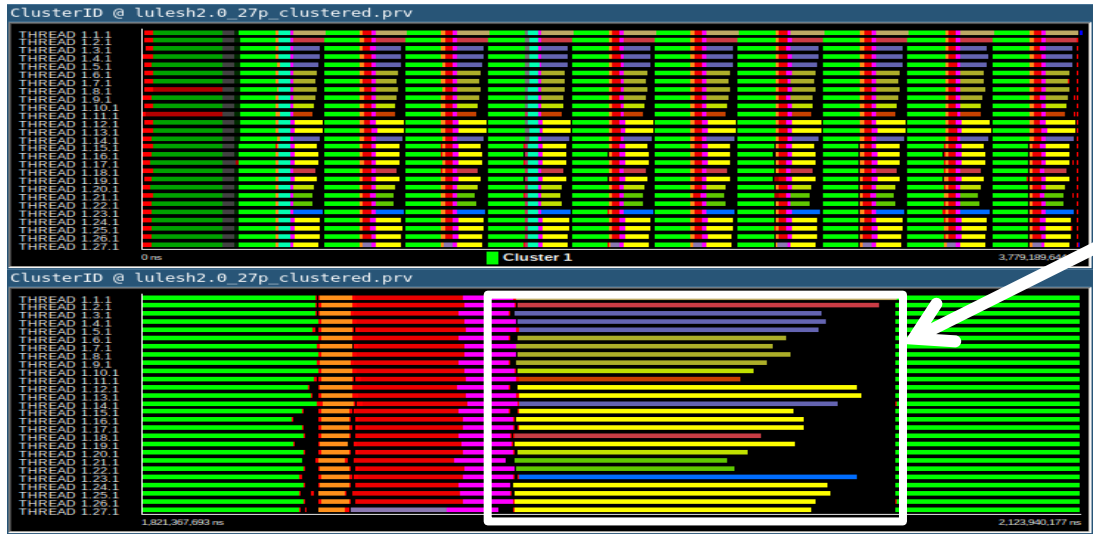


Correlating scatter plot and time distribution

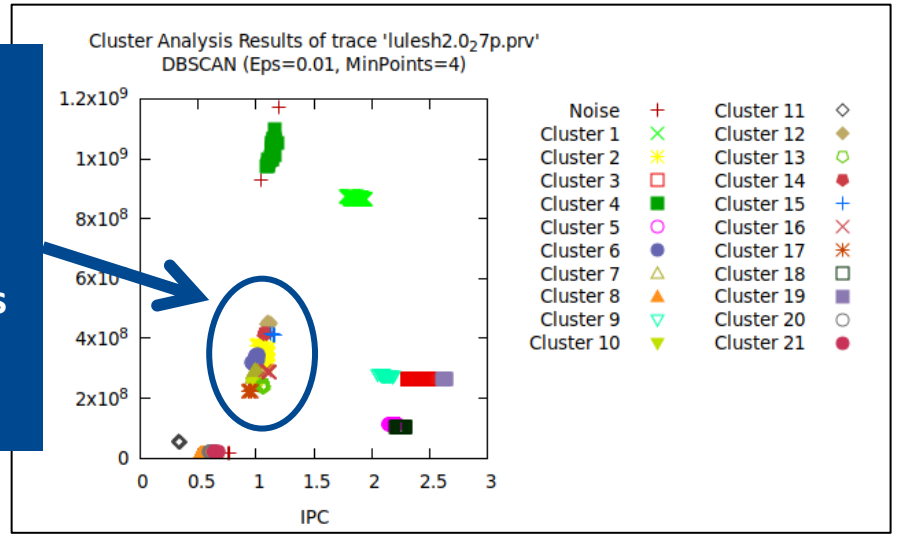
- Open the clustered trace with Paraver and look at it

```
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