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	Isambard XC50
Event	150 – 200ns	238ns
Event + PAPI	750 – 1000ns	1705ns
Event + callstack (1 level)	1µs	1854ns
Event + callstack (6 levels)	2µs	5700ns

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 Isambard-XCI

```
laptop> ssh isambard-xci
xci100> cp -r /home/ri-emercada/tools-material $HOME
xci100> ls $HOME/tools-material
    apps/
    clustering/
    extrae/
    slides/
    traces/
```

Step 1: Append loader script

```
xci100> vi $HOME/tools-material/extrae/job.pbs
```

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

#PBS -N "lulesh2.0_27p"
#PBS -l select=1
#PBS -l walltime=00:10:00
#PBS -q arm

cd $PBS_O_WORKDIR

export OMP_NUM_THREADS=1

aprun -n 27 ../apps/lulesh2.0 –i 10 –s 65 –p
```

Request resources

Run the program

Step 1: Append loader script

```
xci100> vi $HOME/tools-material/extrae/job.pbs
```

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

#PBS -N "lulesh2.0_27p"
#PBS -l select=1
#PBS -l walltime=00:10:00
#PBS -q arm

cd $PBS_O_WORKDIR

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

aprun -n 27 ./trace.sh ../apps/lulesh2.0 –i 10 –s 65
```

Name of the resulting trace

Activate Extrae during the run

Step 1: Append loader script

```
xci100> vi $HOME/tools-material/extrae/trace.sh
```

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

#PBS -N "lulesh2.0_27p"
#PBS -l select=1
#PBS -l walltime=00:10:00
#PBS -q arm

cd $PBS_O_WORKDIR

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

aprun -n 27 ./trace.sh ./apps/lulesh2.0 -i 10 -s 65
```

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

# Configure Extrae
export EXTRAE_HOME=…/extrae/3.6.1/cray-mpich_7.7.6
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

```
xci100> cd $HOME/tools-material/extrae
xci100> qsub job.pbs
xci100> qstat -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

```
xci100> 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

```
xci100> vi $HOME/tools-material/extrae/extrae.xml
```

```
<counters enabled="yes">
    <cpu enabled="yes" starting-set-distribution="1">
        <set enabled="yes" domain="all" changeat-time="500000us">
            PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_L1_DCM,PAPI_L2_DCM,PAPI_BR_INS,PAPI_SR_INS,PAPI_LD_INS
        </set>
        <set enabled="yes" domain="all" changeat-time="500000us">
            PAPI_TOT_INS,PAPI_TOT_CYC, PAPI_FP_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: Configure what to trace

```
xci100> 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="no"
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)

- Also available in Isambard-XCI
  - `/home/ri-emercada/tools-packages`

Install Paraver tutorials

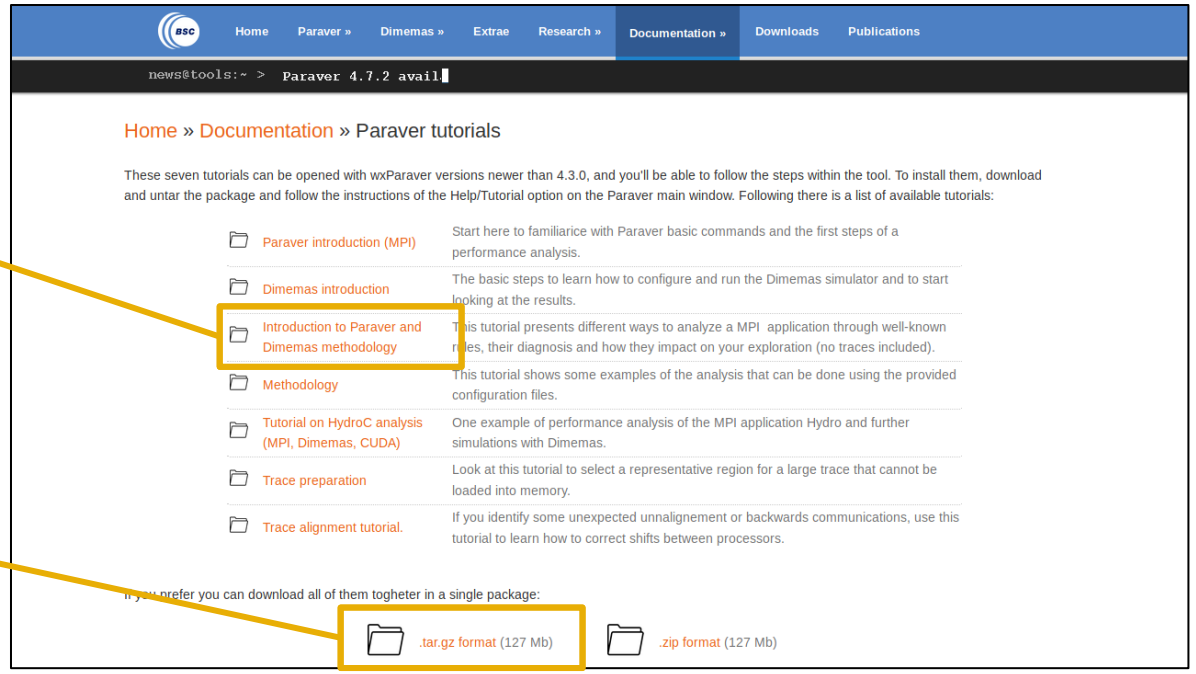
- Download tutorials archive
  - https://tools.bsc.es/paraver-tutorials

Tutorial 3

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

All tutorials

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



- Also available in Isambard-XCI
  - `/home/ri-emercada/tools-packages`

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

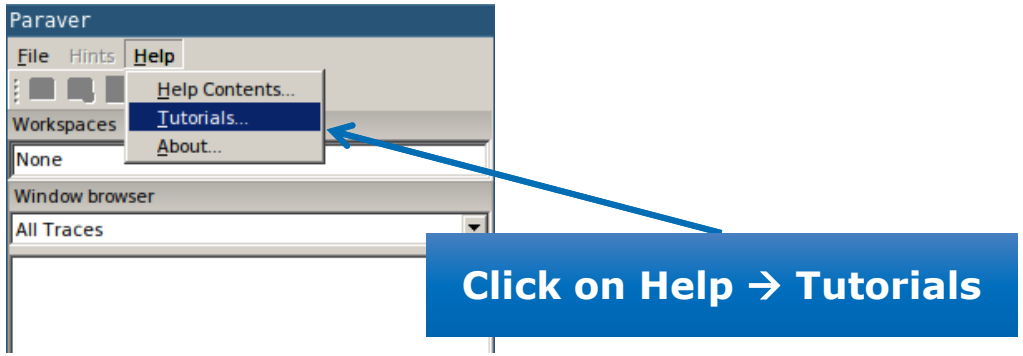
```
laptop> tar xf paraver-tutorials-20150526.tar.bz2
laptop> mv paraver-tutorials-20150526 paraver/tutorials
```

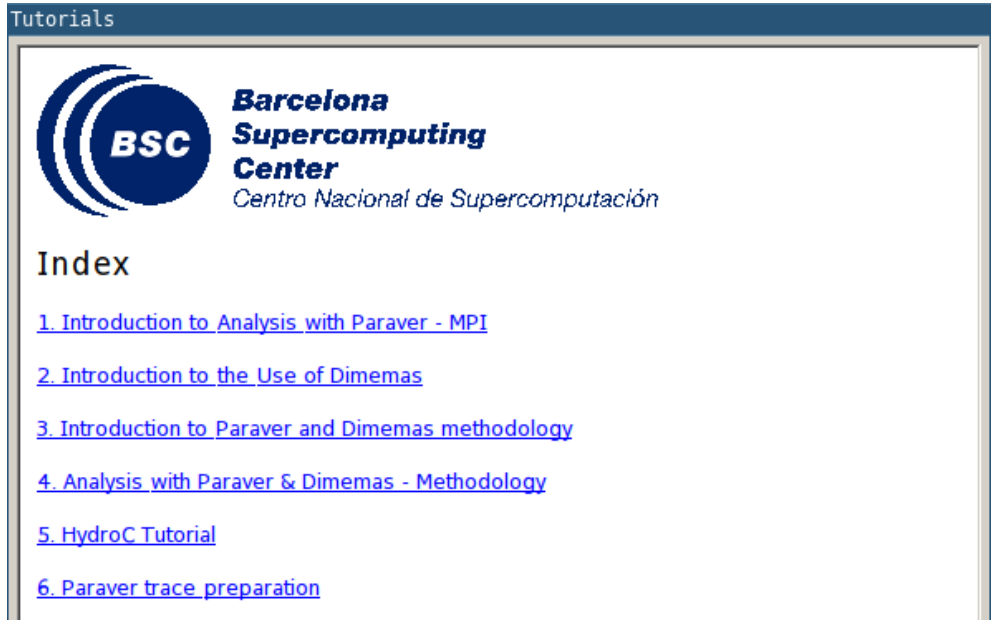
Check that everything works

- Start Paraver

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

- Check the tutorials are available



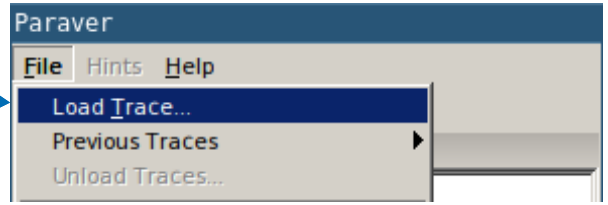


First steps of the analysis

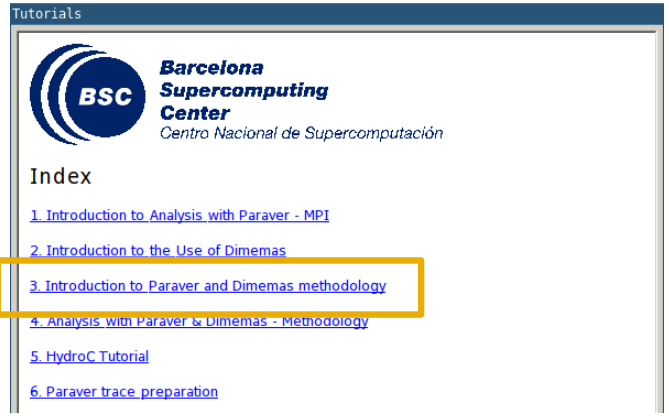
- Copy the trace to your laptop

```
laptop> scp isambard-xci:$HOME/tools-material/extrae/lulesh2.0_27p.{prv,pcf,row} ./
```

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

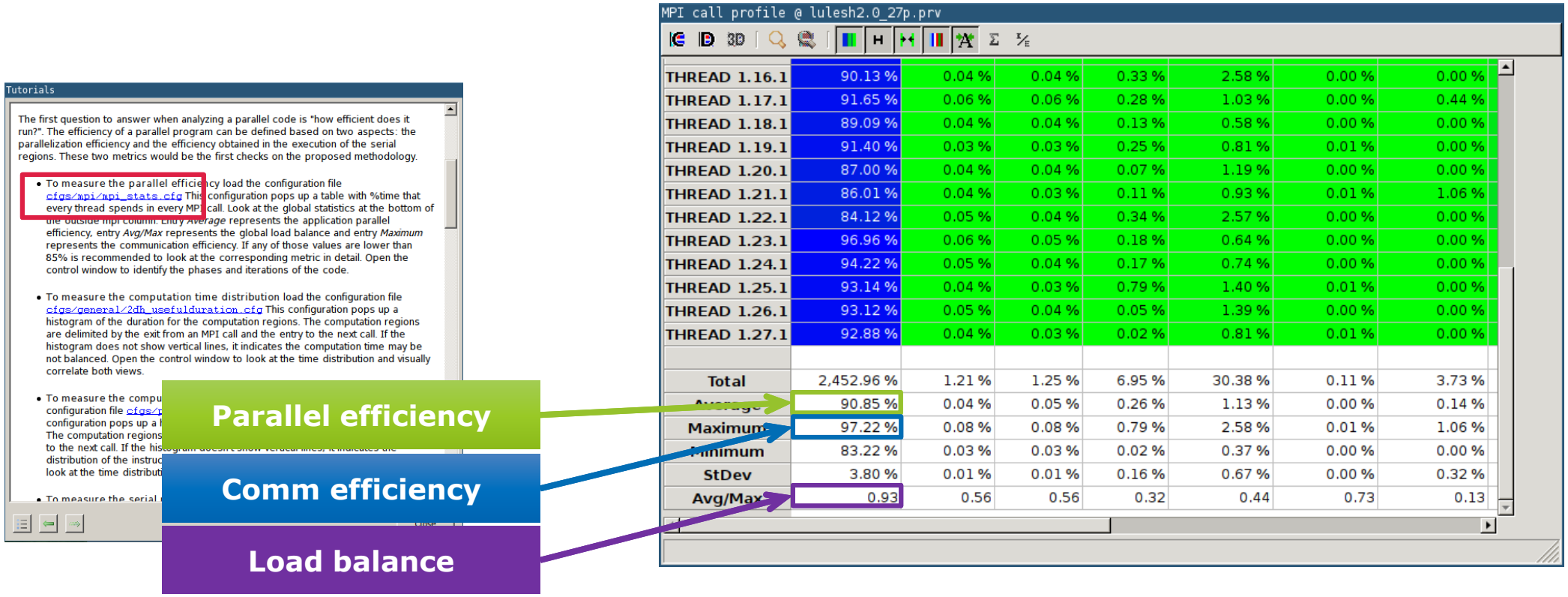


- Follow tutorial #3
  - Help -> Tutorials



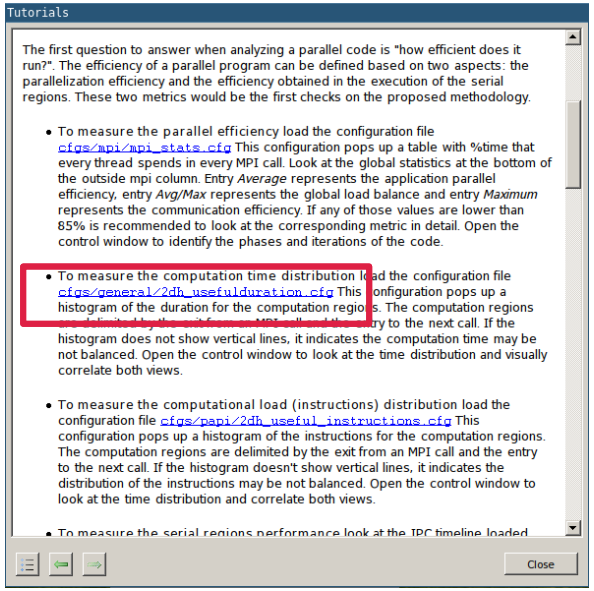
Measure the parallel efficiency

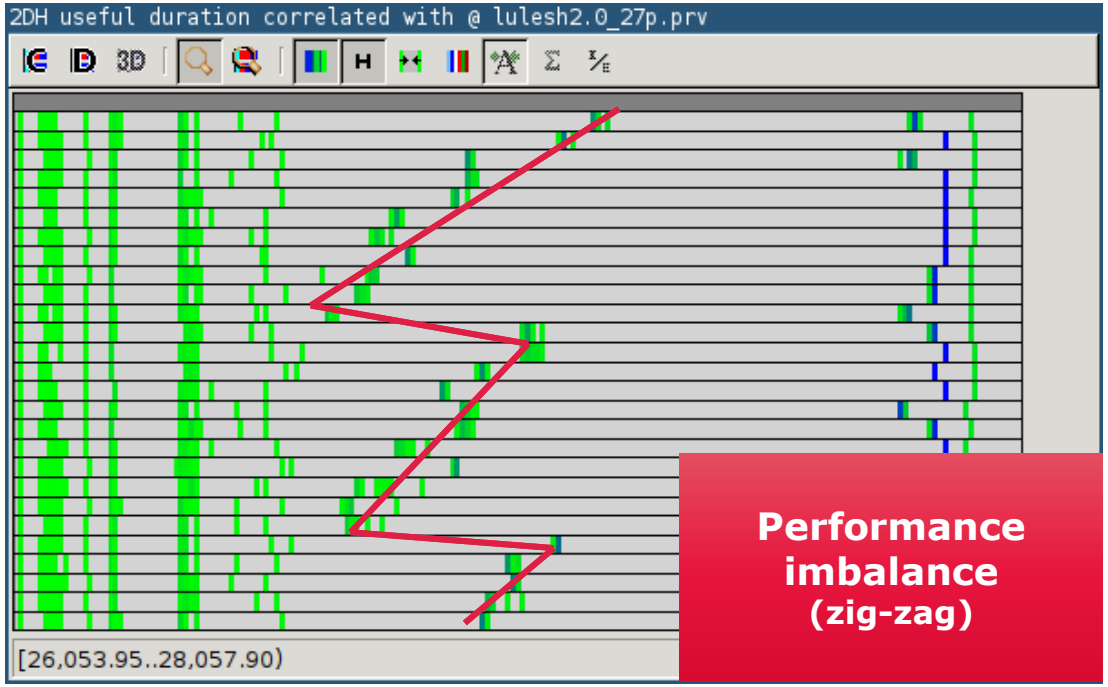
- Click on `mpi_stats.cfg`



Computation time and work distribution

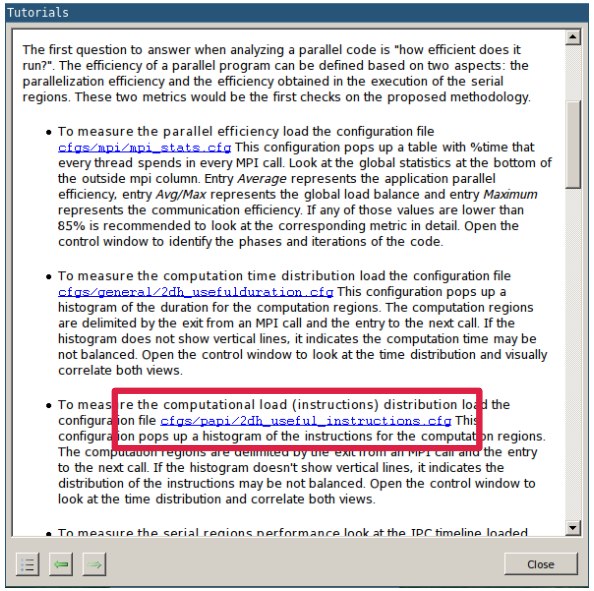
- Click on `2dh_usefulduration.cfg` (2nd link) ➔ Shows **time computing**

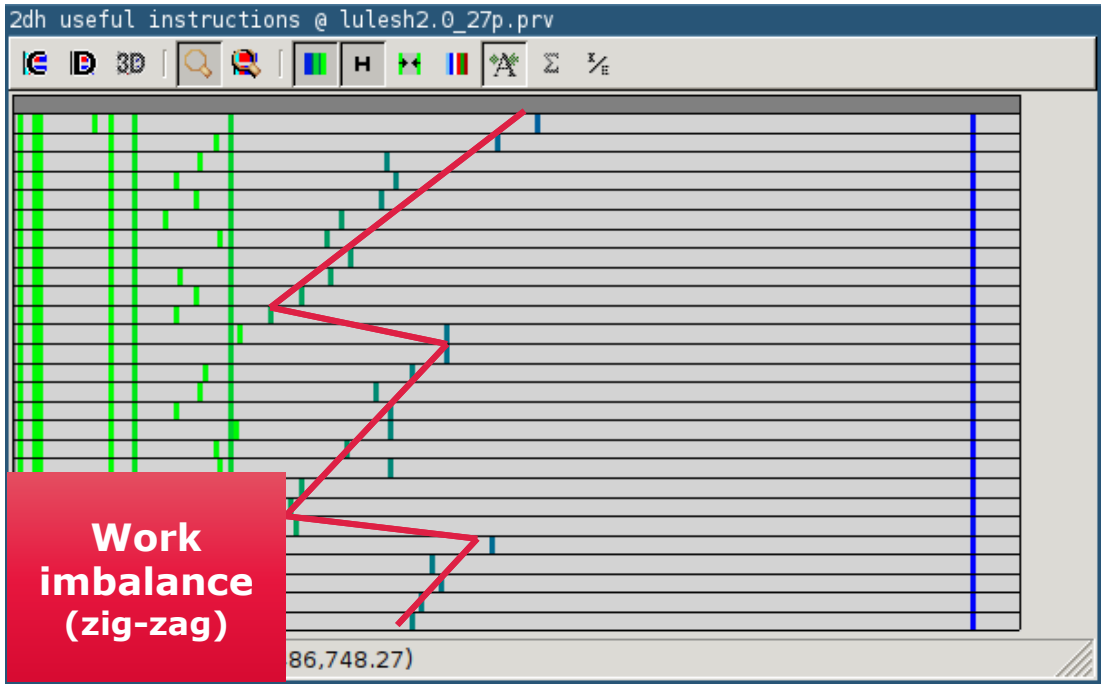




Computation time and work distribution

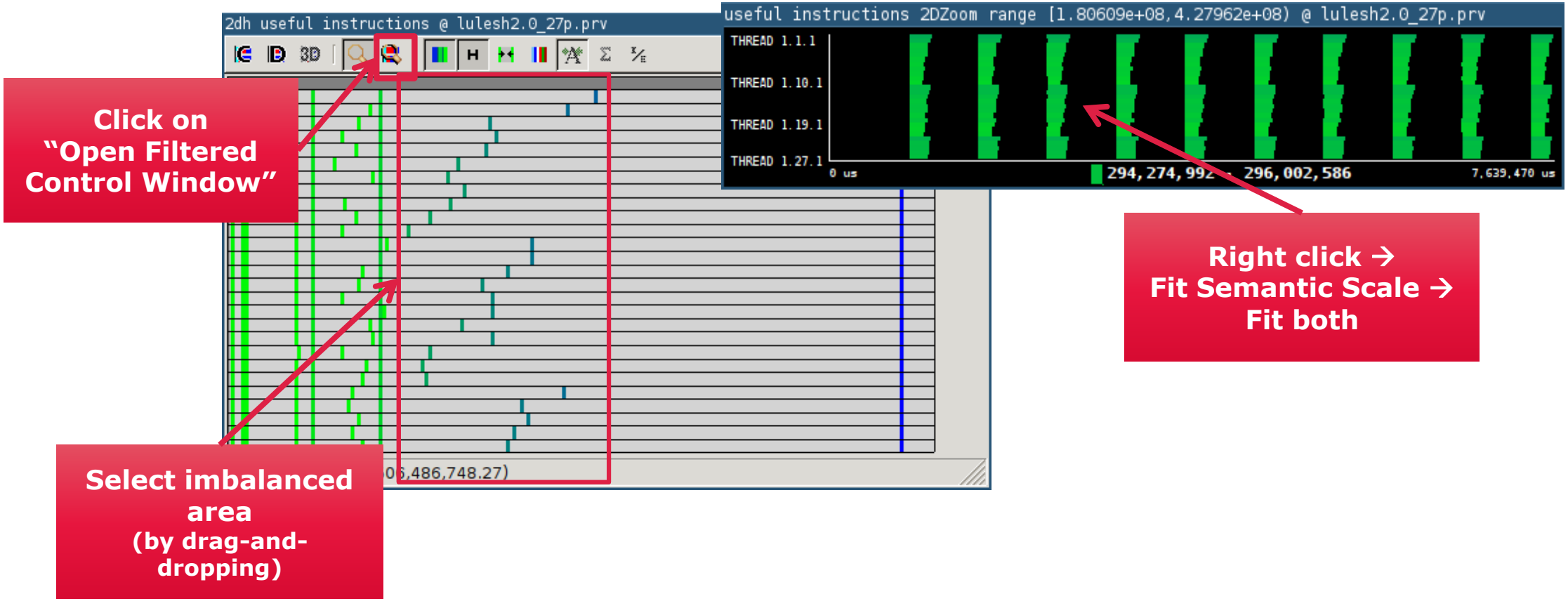
- … and `2dh_useful_instructions.cfg` (3rd link) ➔ Shows **amount of work**





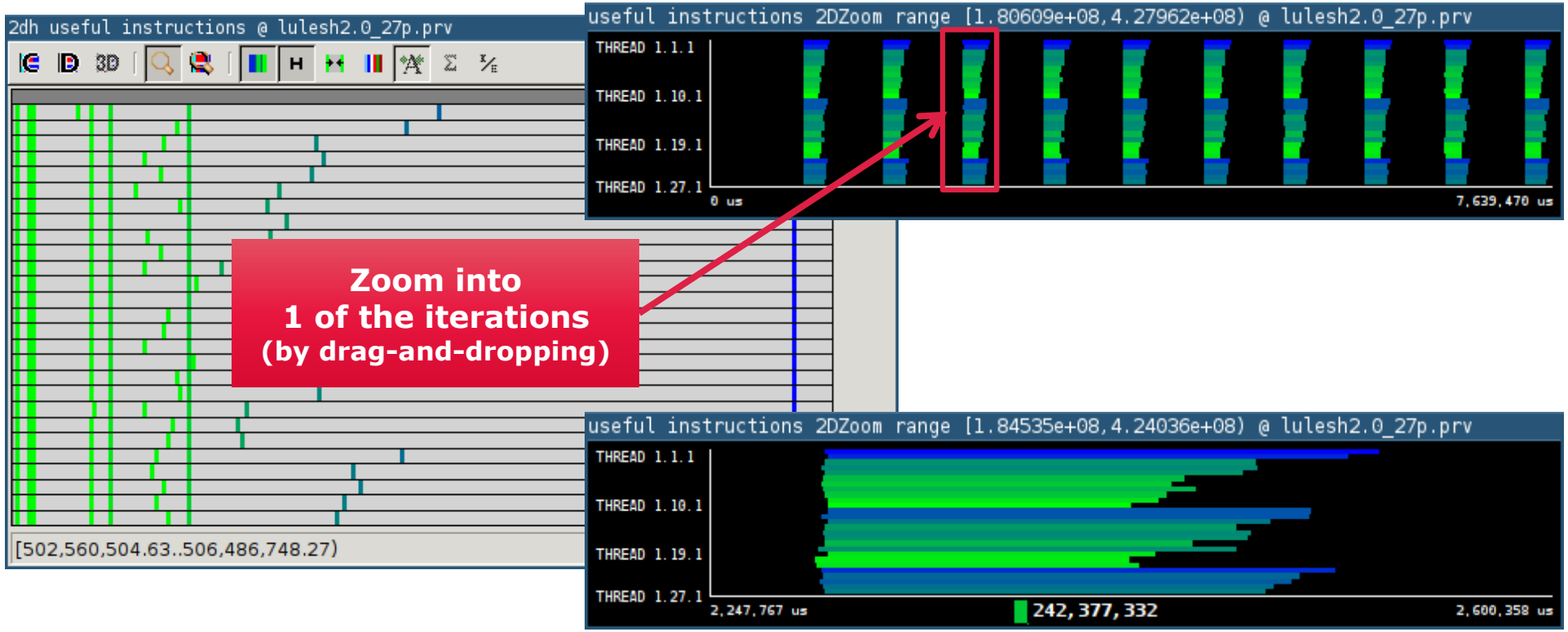
Where does this happen?

- Go from tables to timelines



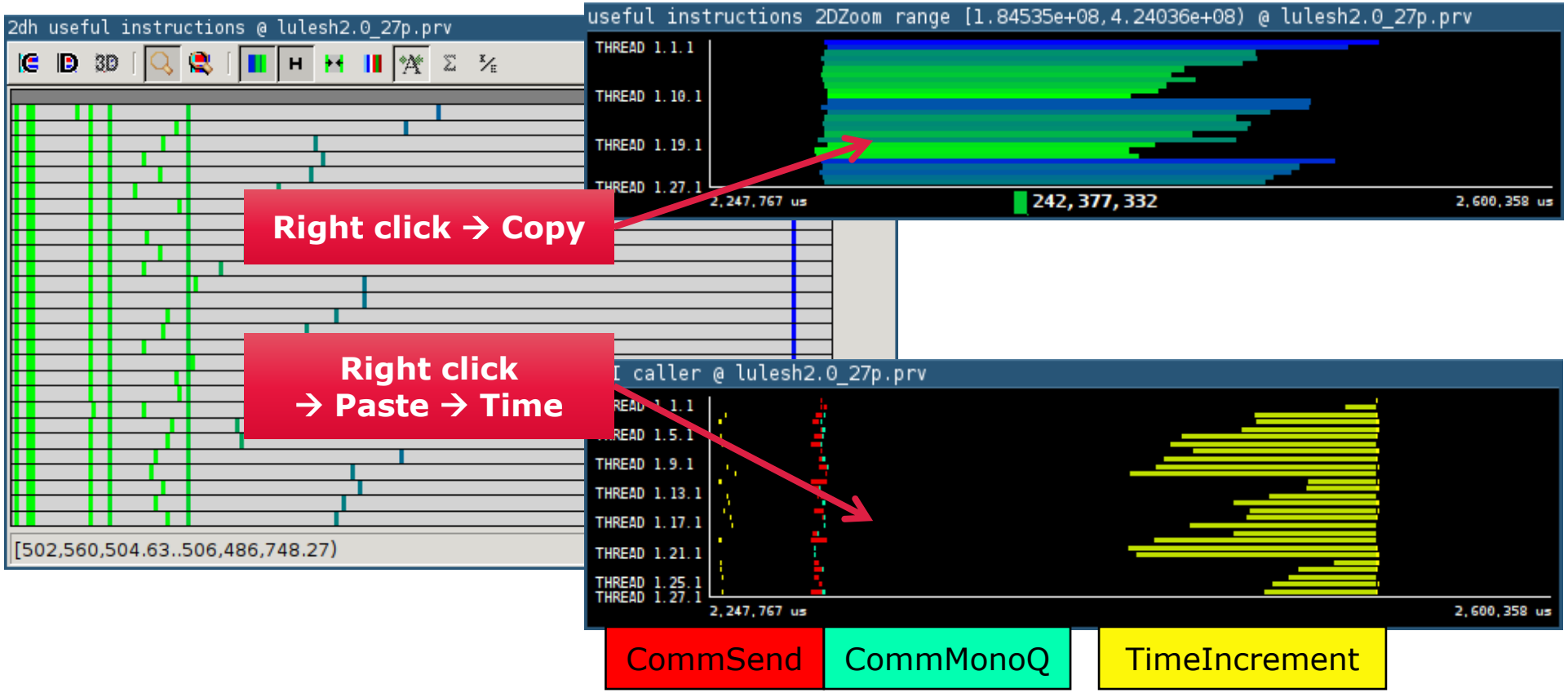
Where does this happen

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

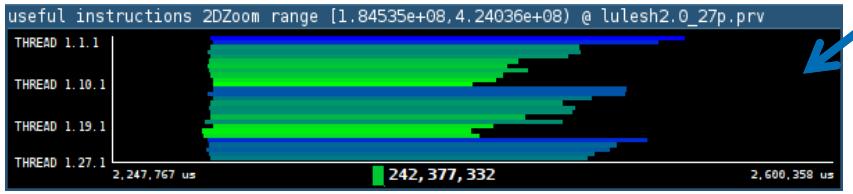


Where does this happen

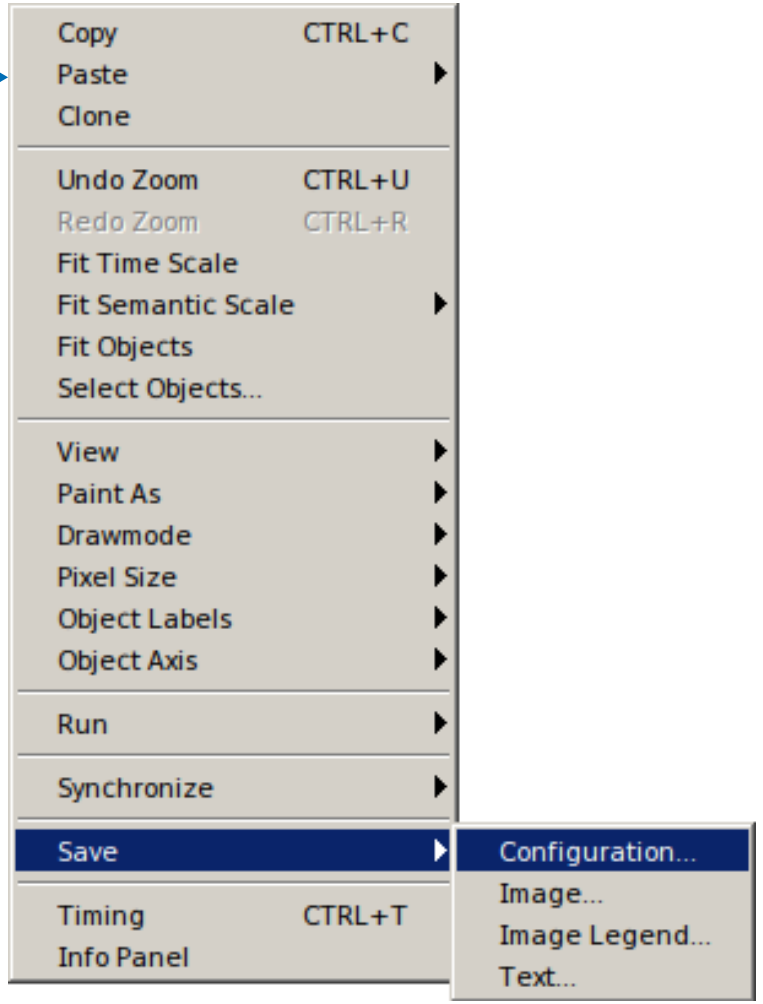
- Hints → Callers → Caller function



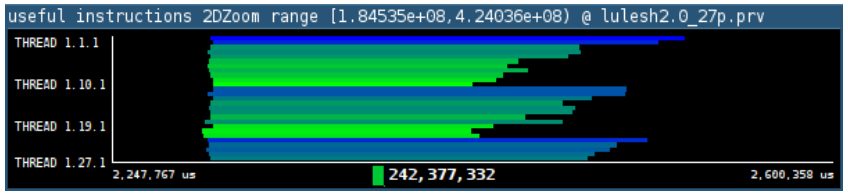
Save CFG's (2 methods)



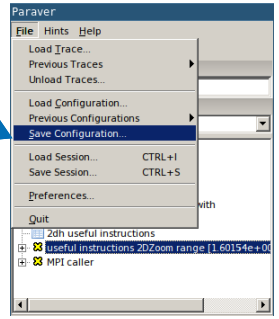
Right click on timeline

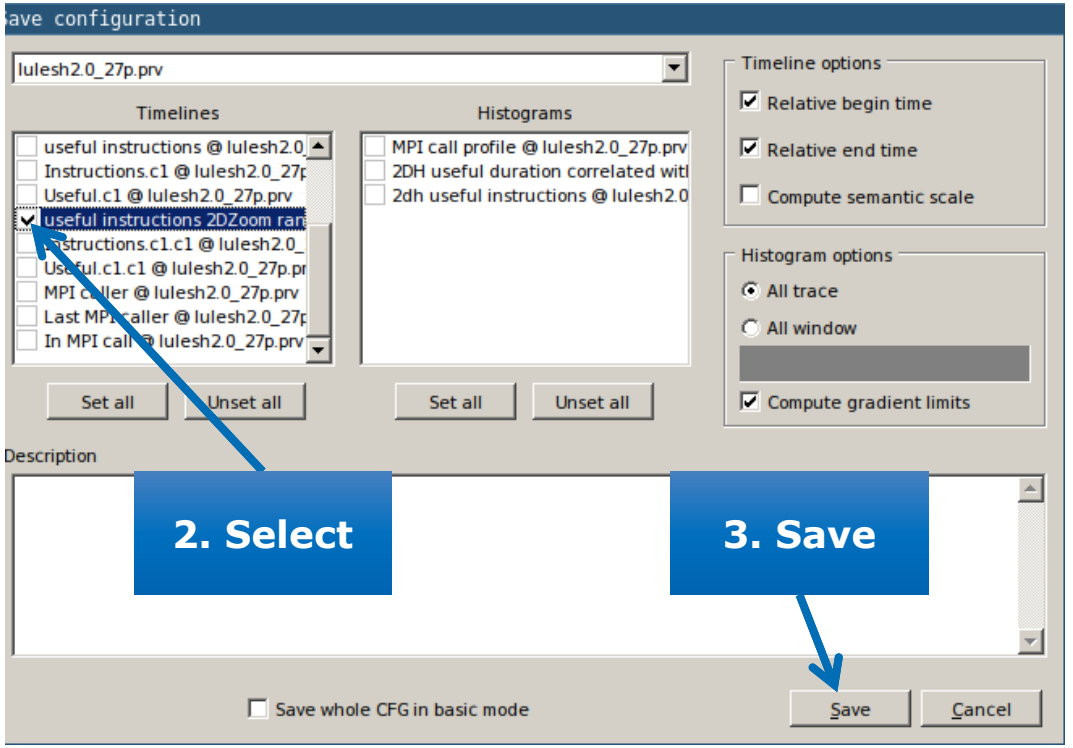


Save CFG's (2 methods)



1. Main Paraver window



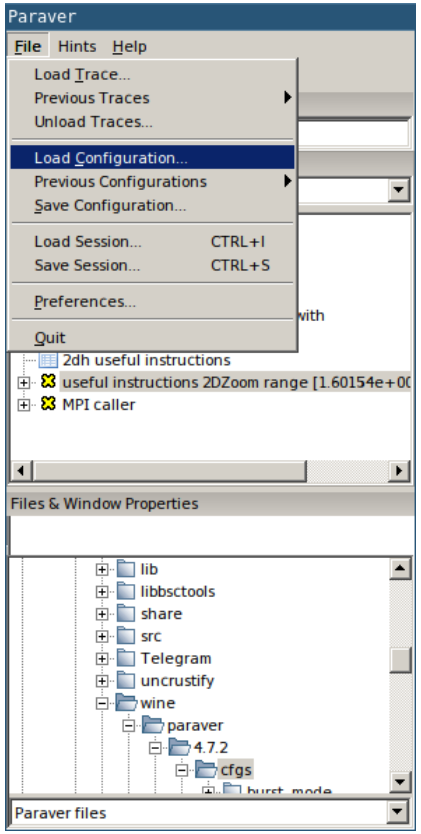


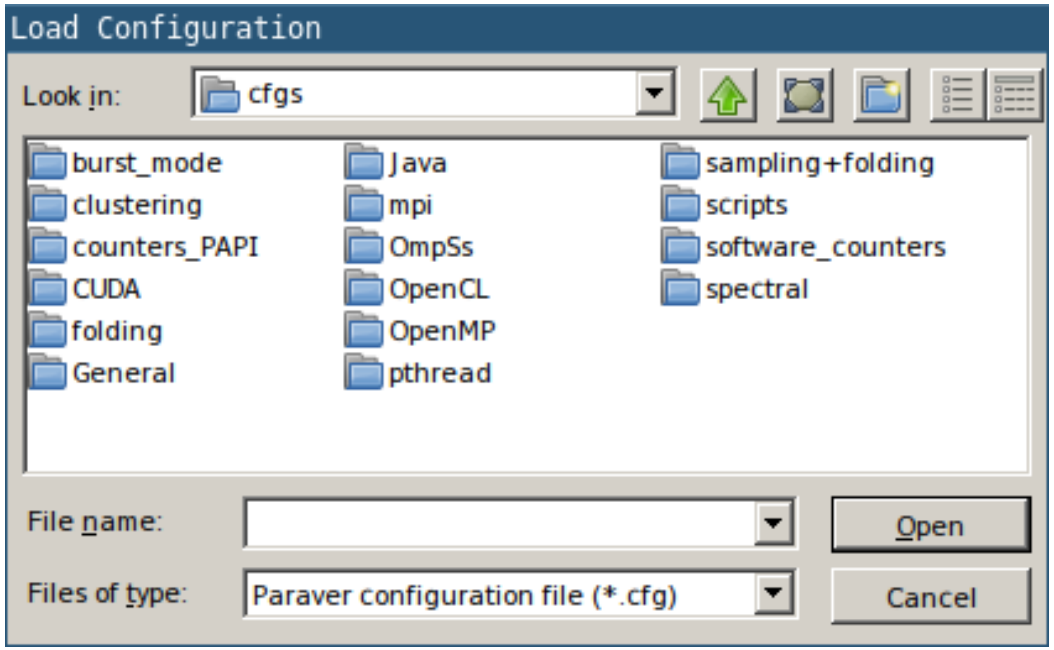
2. Select

3. Save

CFG's distribution

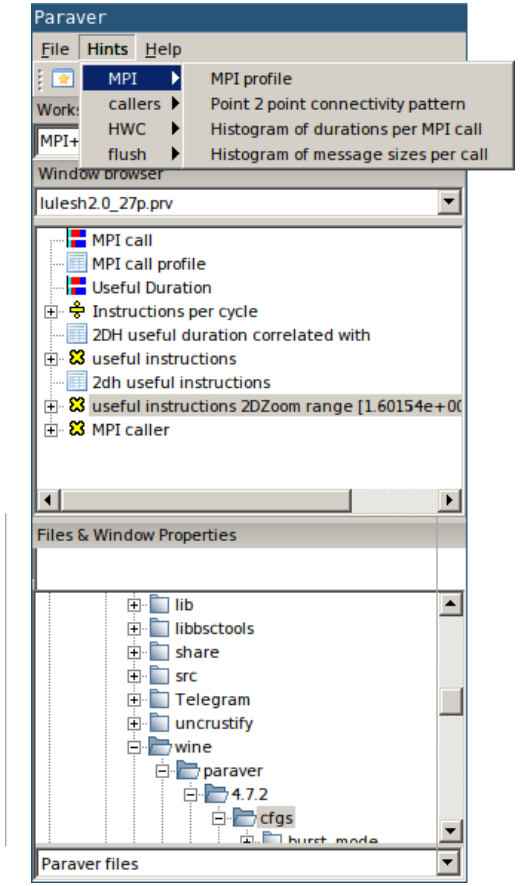
- Paraver comes with many more included CFG's





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 Isambard-XCI
  - `/home/ri-emercada/tools-packages`

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:

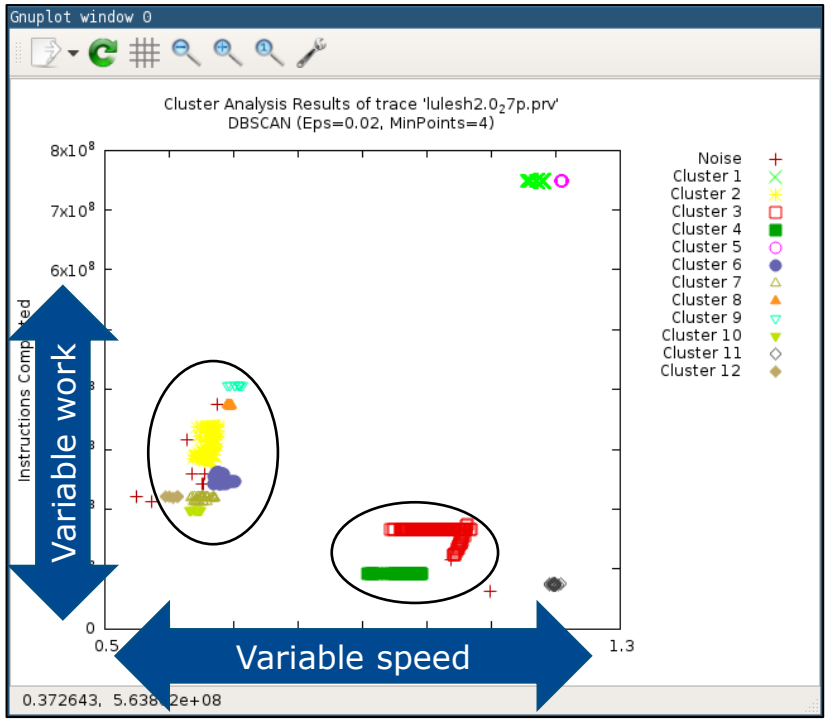
```
xci100> ls $HOME/tools-material/traces/lulesh2.0_27p.prv
```

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**

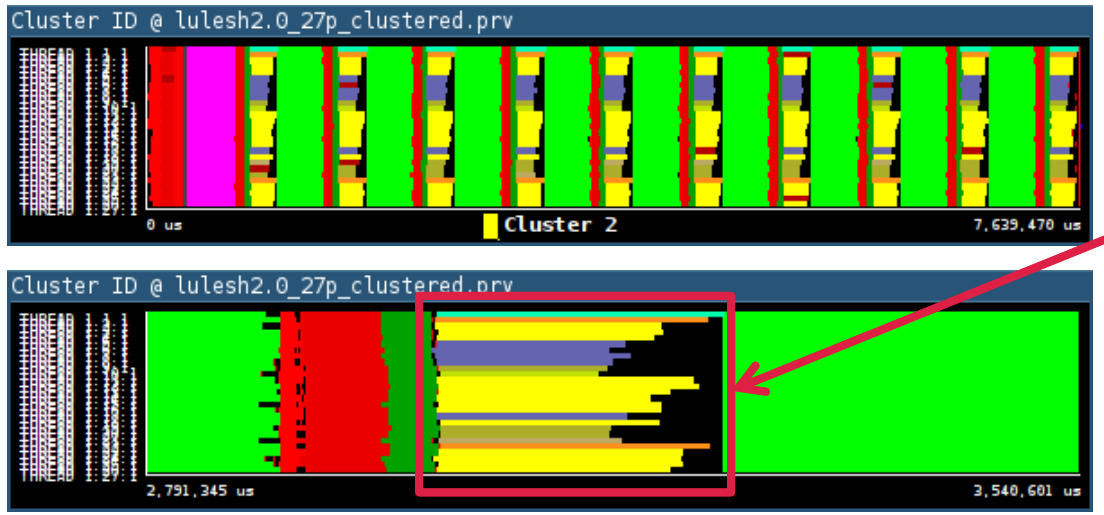


Correlating scatter plot and time distribution

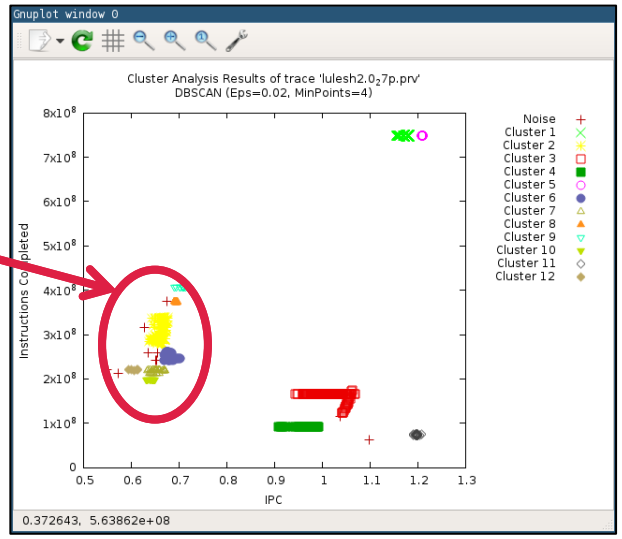
- Open the clustered trace with Paraver and look at it

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