BSC Tools
Hands-On

Lau Mercadal Melià
(tools@bsc.es)
Barcelona Supercomputing Center

Getting a trace with Extrae

Extrae features

- Platforms
  - Intel, Cray, BlueGene, Intel MIC, ARM, Android, Fujitsu Sparc, IBM POWER…
- 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

	Avg. values	MN4	CTE-POWER
Event	150 – 200ns	171ns	173ns
Event + PAPI	750 – 1000ns	755ns	1.4µs
Event + callstack (1 level)	1µs	1µs	5.3µs
Event + callstack (6 levels)	2µs	2.5µs	12µs

How does Extrae work?

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

Recommended

- Dynamic instrumentation
  - Based on DynInst (developed by U.Wisconsin/U.Maryland)
    - Instrumentation in memory
    - Binary rewriting
- Function instrumentation through "-finstrument-functions" compiler option
- 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 in $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 in $EXTRAE_HOME/share/doc

Login to MareNostrum4

```
laptop> ssh -Y <USER>@mn1.bsc.es

login1> ls $HOME/tools-material
   ... apps/
   ... clustering/
   ... extrae_cte-power/
   ... extrae_mn4/
   ... slides/
   ... src/
   ... traces/
```

Here you have a copy of the slides

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

```
login1> vi $HOME/tools-material/extrae_mn4/job.sh
```

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

#SBATCH --job-name=lulesh_mn4_27p
#SBATCH --output=lulesh_mn4_27p.out
#SBATCH --error=lulesh_mn4_27p.err
#SBATCH --ntasks=27
#SBATCH --cpus-per-task=1
#SBATCH --time=00:05:00
#SBATCH --reservation=PATCVIHPS

export OMP_NUM_THREADS=1

srun ../apps/lulesh_mn4 -i 10 -s 65 -p
```

Request resources

Run the program

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

```
login1> vi $HOME/tools-material/extrae_mn4/job.sh
```

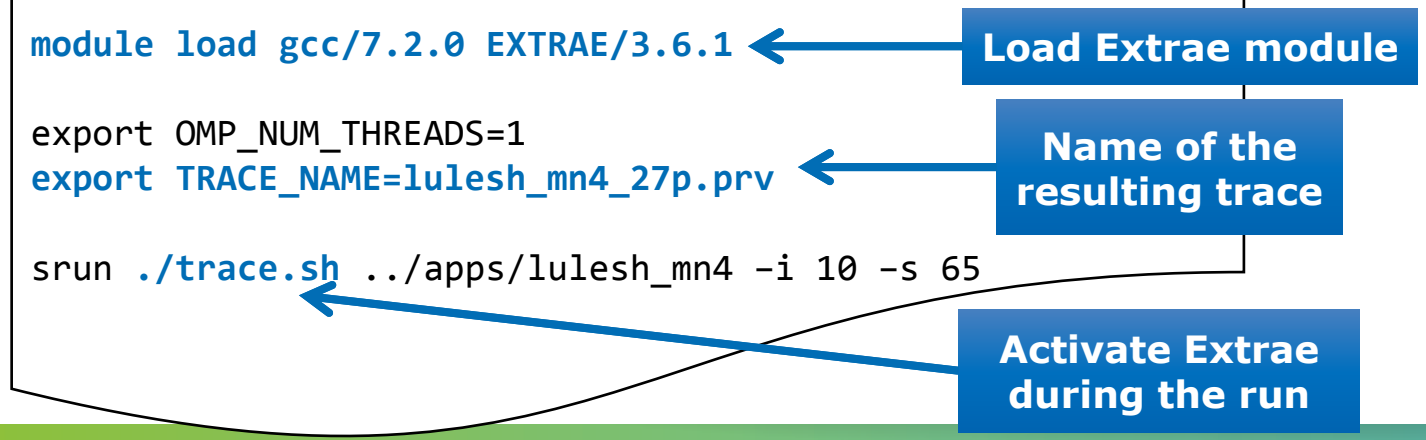
```
#!/usr/bin/env bash

#SBATCH --job-name=lulesh_mn4_27p
#SBATCH --output=lulesh_mn4_27p.out
#SBATCH --error=lulesh_mn4_27p.err
#SBATCH --ntasks=27
#SBATCH --cpus-per-task=1
#SBATCH --time=00:05:00
#SBATCH --reservation=PATCVIHPS

module load gcc/7.2.0 EXTRAE/3.6.1

export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh_mn4_27p.prv

srun ./trace.sh ../apps/lulesh_mn4 -i 10 -s 65
```



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

```
login1> vi $HOME/tools-material/extrae_mn4/trace.sh
```

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

#SBATCH --job-name=lulesh_mn4_27p
#SBATCH --output=lulesh_mn4_27p.out
#SBATCH --error=lulesh_mn4_27p.err
#SBATCH --ntasks=27
#SBATCH --cpus-per-task=1
#SBATCH --time=00:05:00
#SBATCH --reservation=PATCVIHPS

module load gcc/7.2.0 EXTRAE/3.6.1

export OMP_NUM_THREADS=1
export TRACE_NAME=lulesh_mn4_27p.prv

srun ./trace.sh ../apps/lulesh_mn4 –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

```
login1> cd $HOME/tools-material/extrae_mn4

login1> sbatch job.sh
```

- Once finished the trace will be in the same folder: `lulesh_mn4_27p.{pcf,prv,row}`
  - Check the status of your job with:

```
login1> squeue
```

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

Step 2: Extrae XML configuration

```
login1> vi $HOME/tools-material/extrae_mn4/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)

```
login1> vi $HOME/tools-material/extrae_mn4/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_L3_TCM,PAPI_BR_INS,PAPI_BR_MSP,RESOURCE_STALLS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_SR_INS,PAPI_LD_INS
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS,PAPI_TOT_CYC,PAPI_VEC_SP
    </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)

```
login1> vi $HOME/tools-material/extrae_mn4/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)

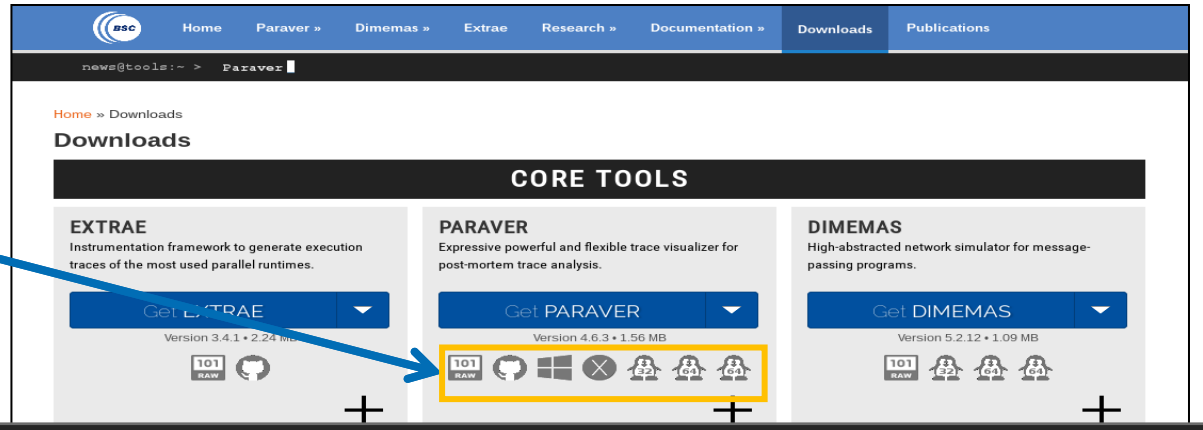
Enable sampling (Want more details?)

Automatic post-processing to generate the Paraver trace

Installing Paraver
&
First analysis steps

Install Paraver in your laptop

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



Pick your version

- Also available in MN under
  - $HOME/tools-packages

wxparaver-4.8.1-win.zip

wxparaver-4.8.1-mac.zip

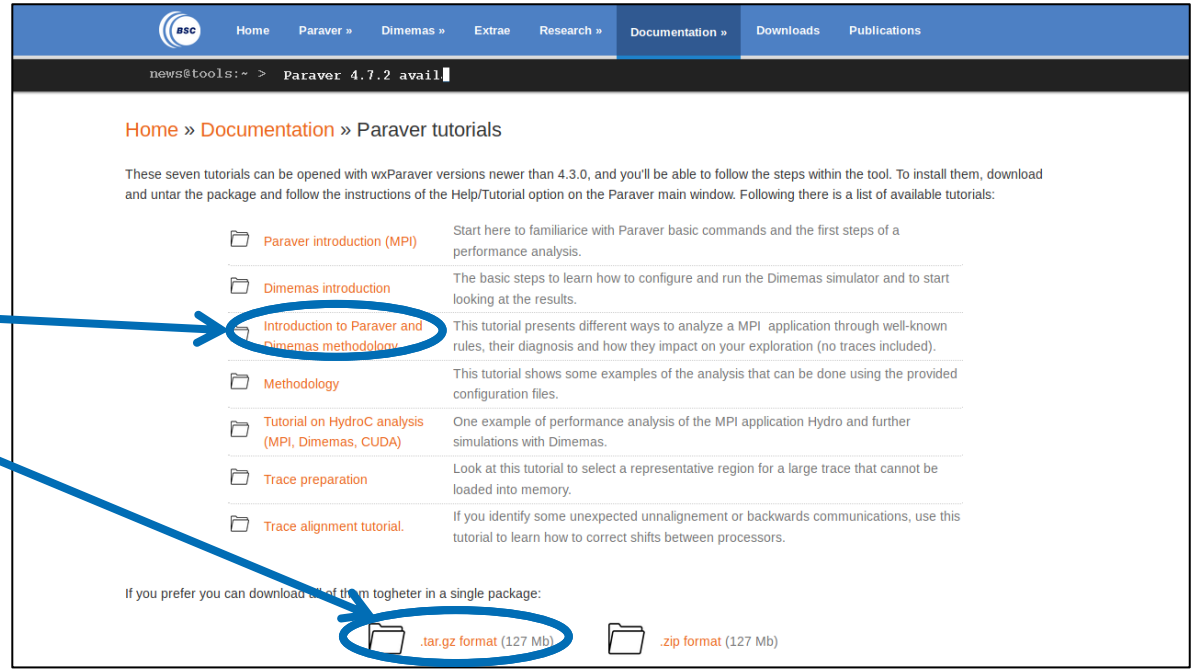
wxparaver-4.8.1-Linux_i686.tar.gz (32-bits)
wxparaver-4.8.1-Linux_x86_64.tar.gz (64-bits)

```
laptop> scp mn1:$HOME/tools-packages/<version> $HOME
```

Install Paraver (II)

- Download tutorials:
  - Documentation -> Tutorial guidelines

Download links



- Also available in your $HOME
  - $HOME/tools-packages

```
laptop> scp mn1:$HOME/tools-packages/paraver-tutorials-20150526.tar.gz $HOME
```

Uncompress, rename & move

- Paraver

```
laptop> tar xf wxparaver-4.8.1-linux-x86_64.tar.gz
laptop> mv wxparaver-4.8.1-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 MareNostrum

```
laptop> ssh -Y <USER>@mn1.bsc.es
login1> module load paraver
login1> wxparaver &
```

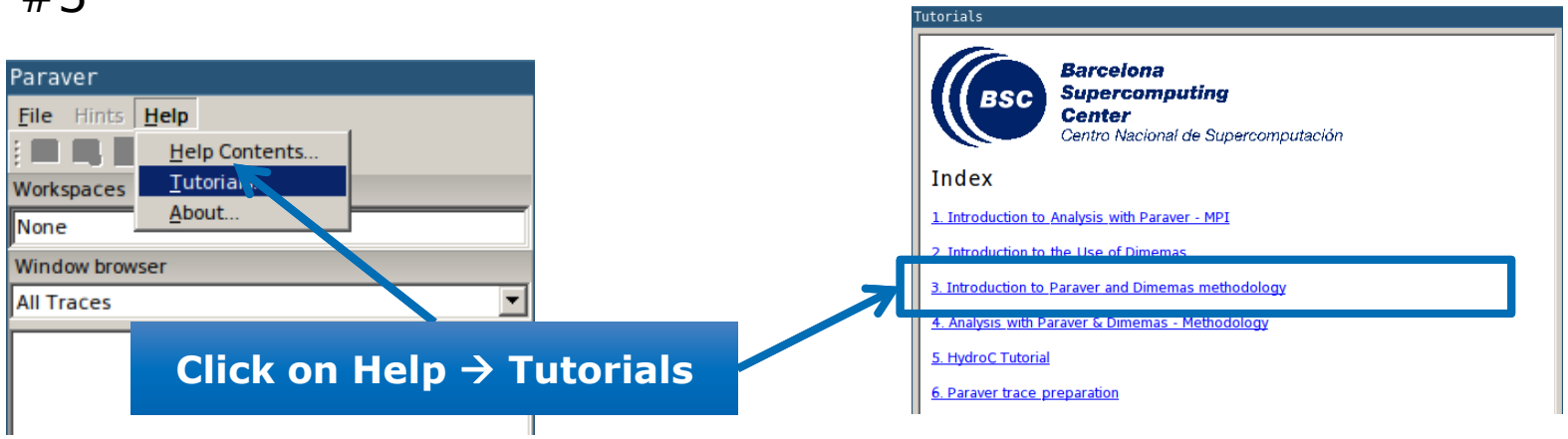
First steps of analysis

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

```
laptop> scp <USER>@mn1.bsc.es:$HOME/tools-material/extrae_mn4/lulesh* ./
```

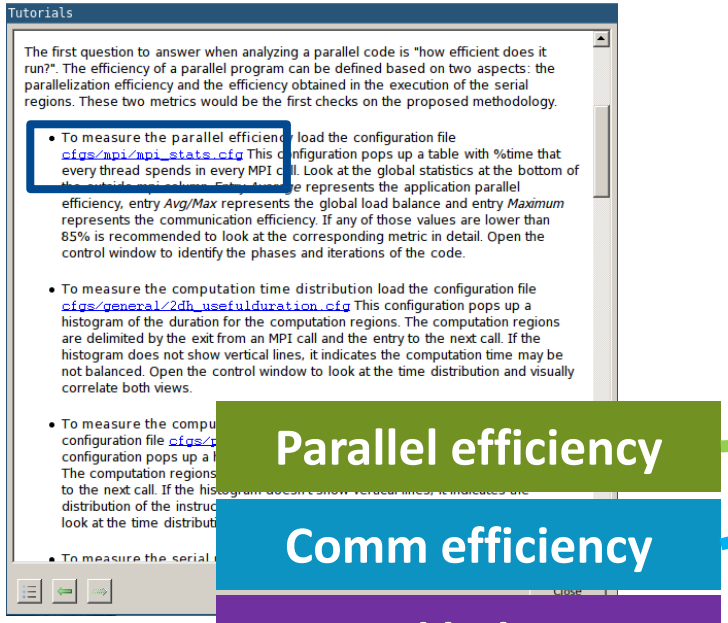
- Load the trace

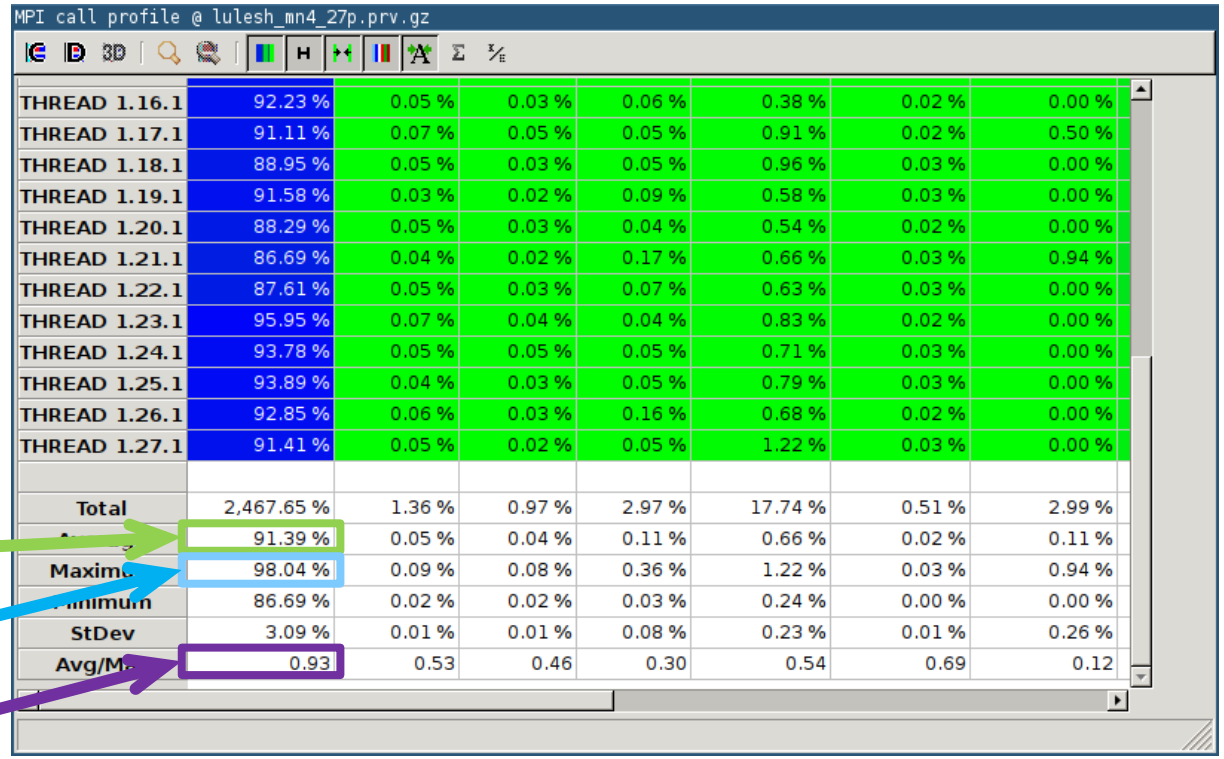
- Follow Tutorial #3



Measure the parallel efficiency

- Click on “mpi_stats.cfg”





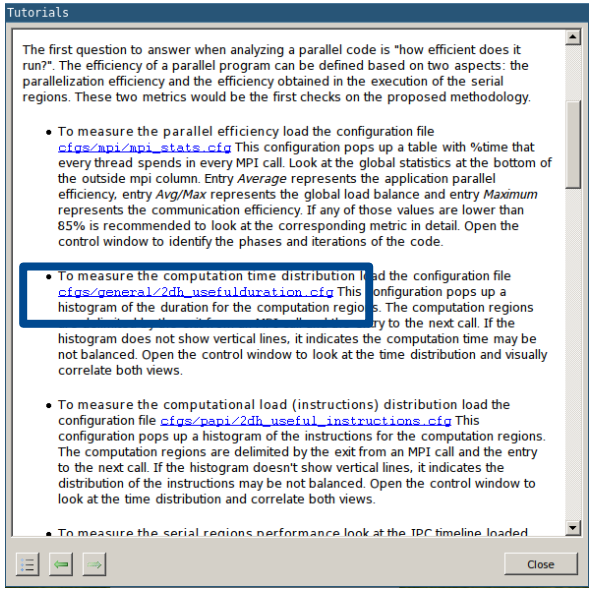
Parallel efficiency

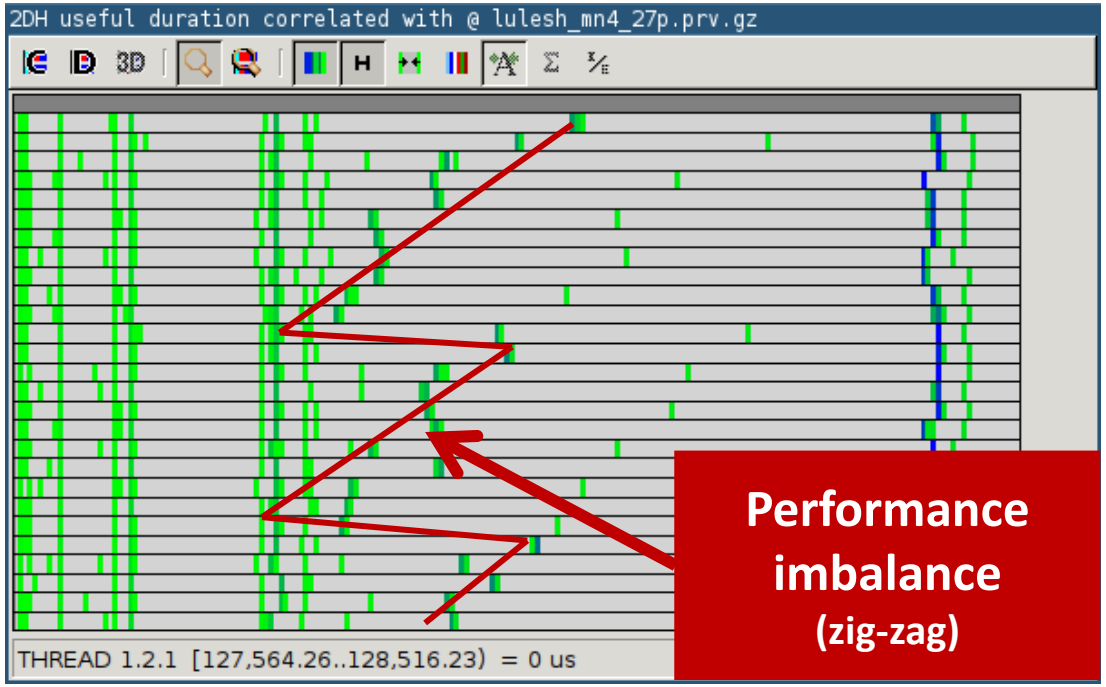
Comm efficiency

Load balance

Computation time and work distribution

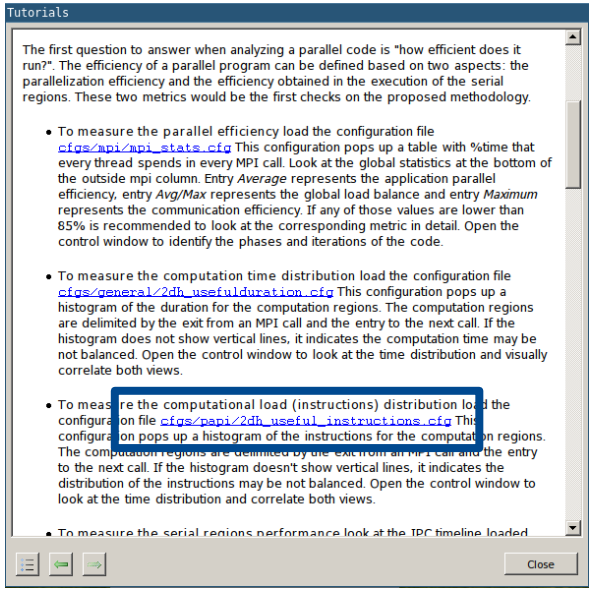
- Click on “2dh_usefulduration.cfg” (2nd link) ➔ Shows **time computing**

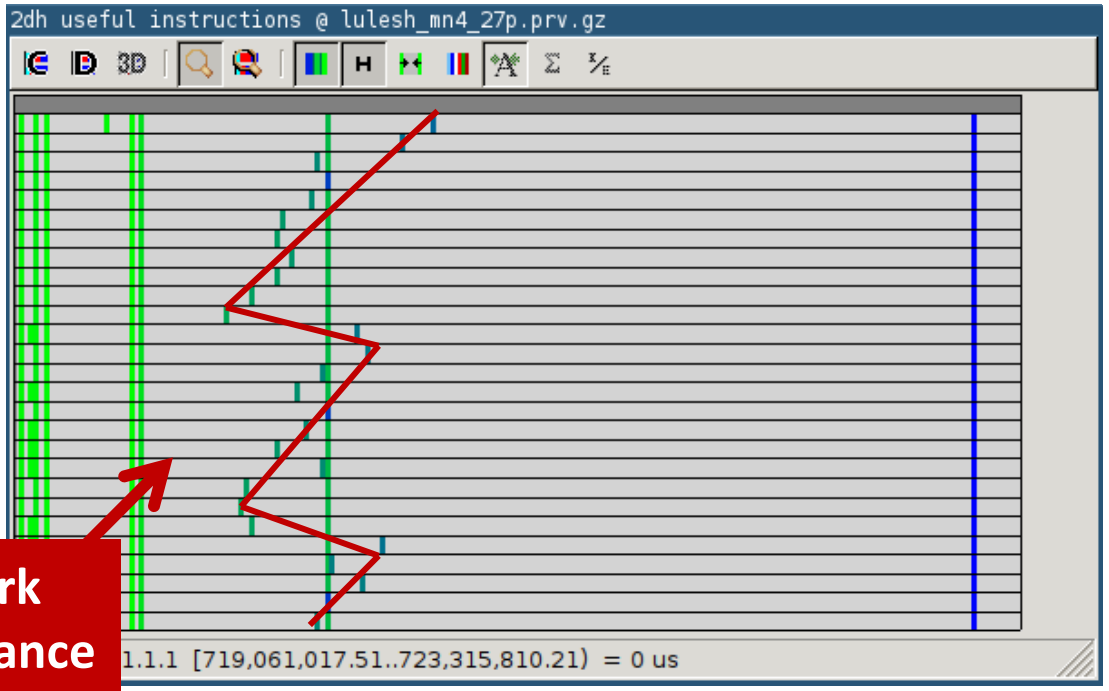




Computation time and work distribution

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

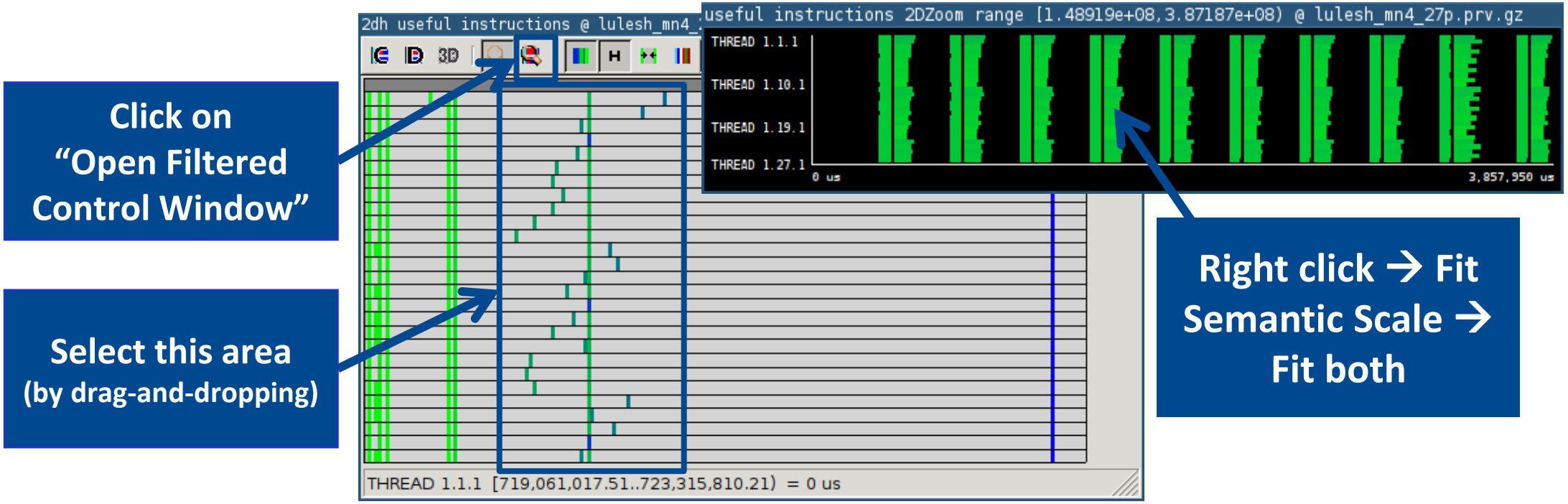




Work imbalance (zig-zag)

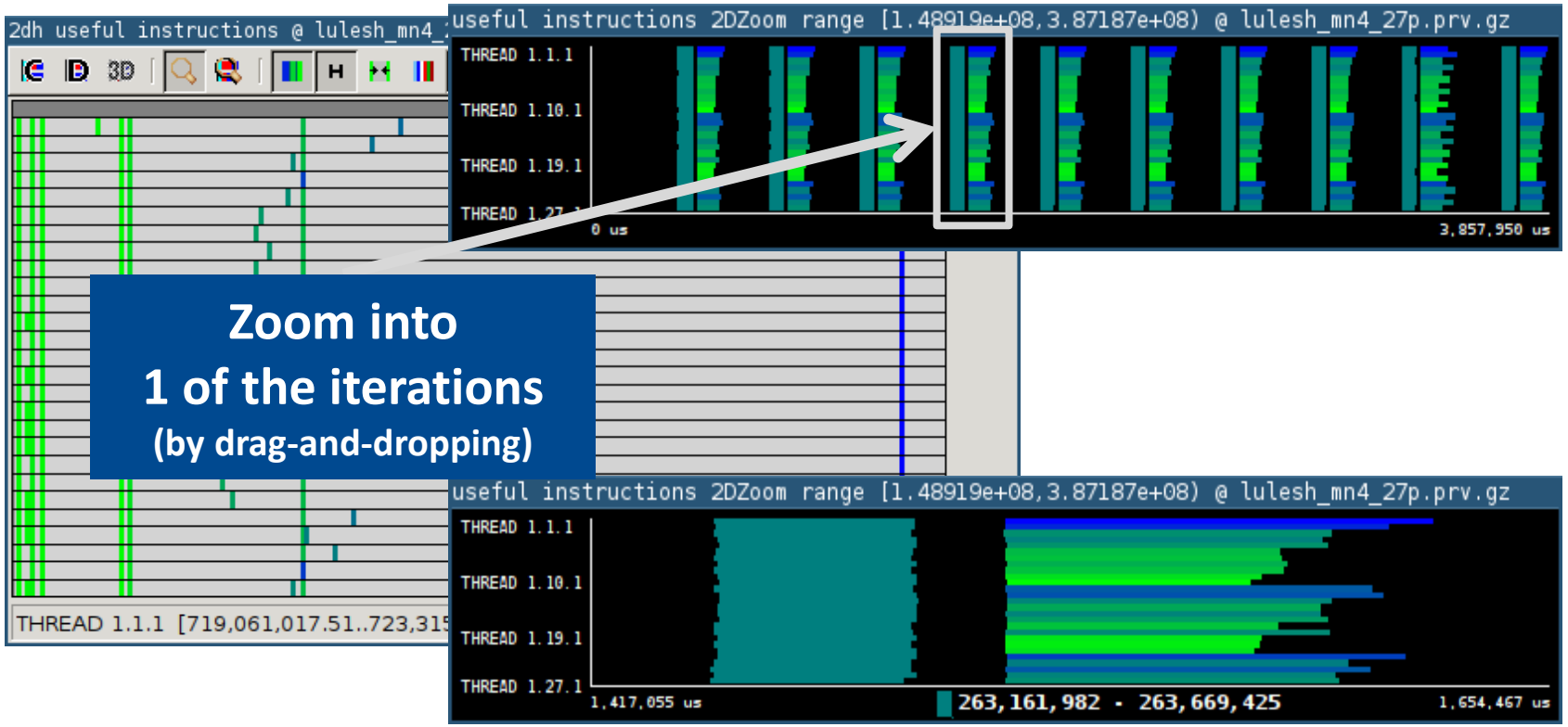
Where does this happen?

- Go from the table to the timeline



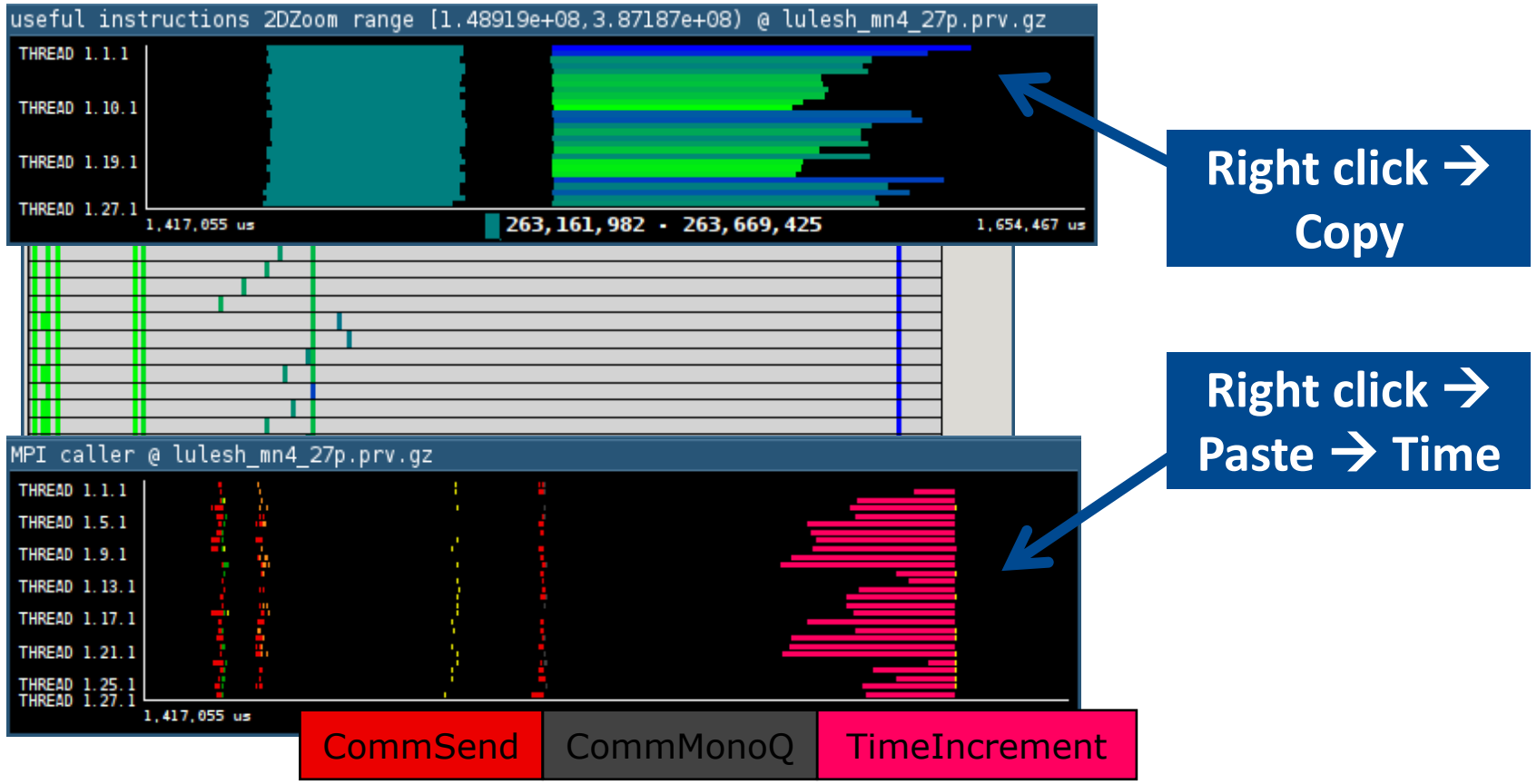
Where does this happen?

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

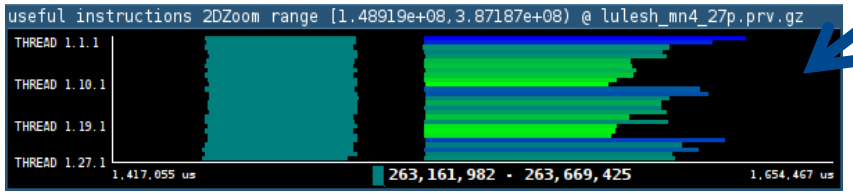


Where does this happen?

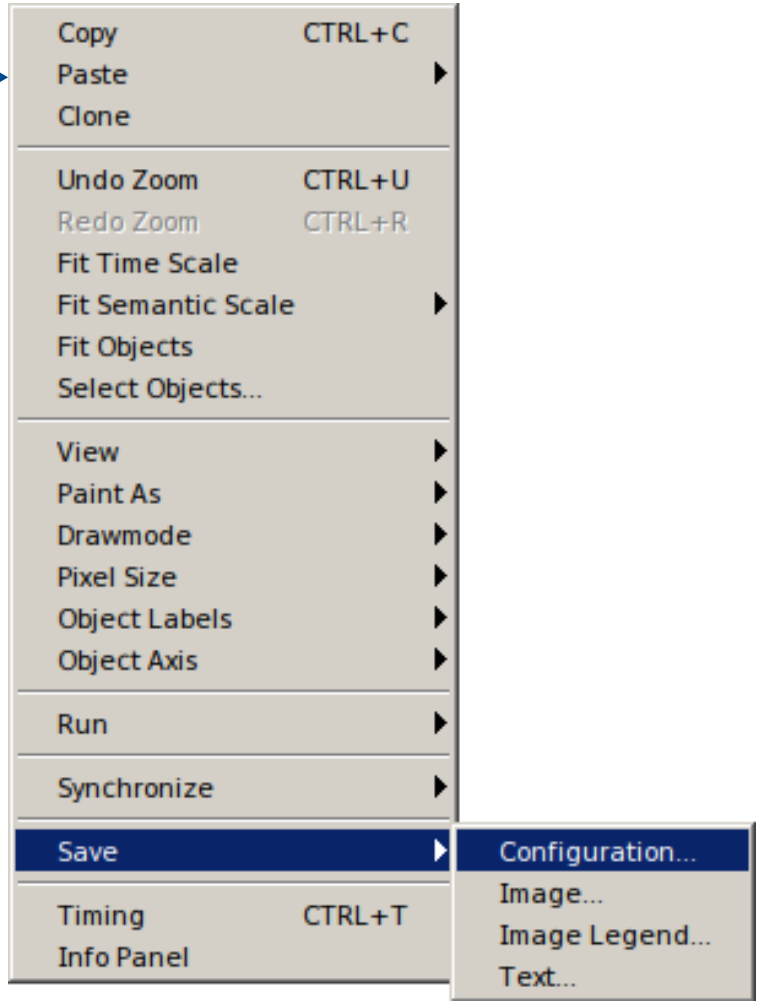
- Hints → Callers → Caller function



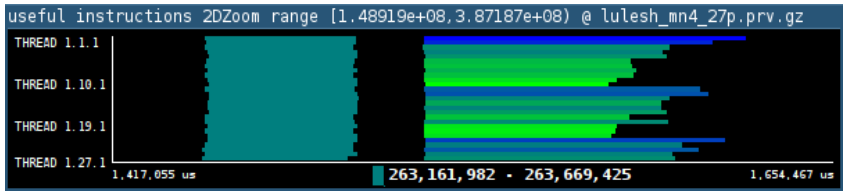
Save CFG's (2 methods)



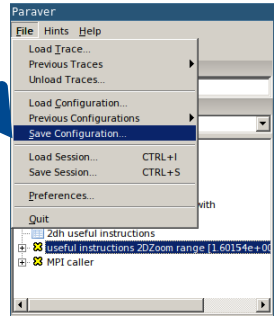
Right click on timeline

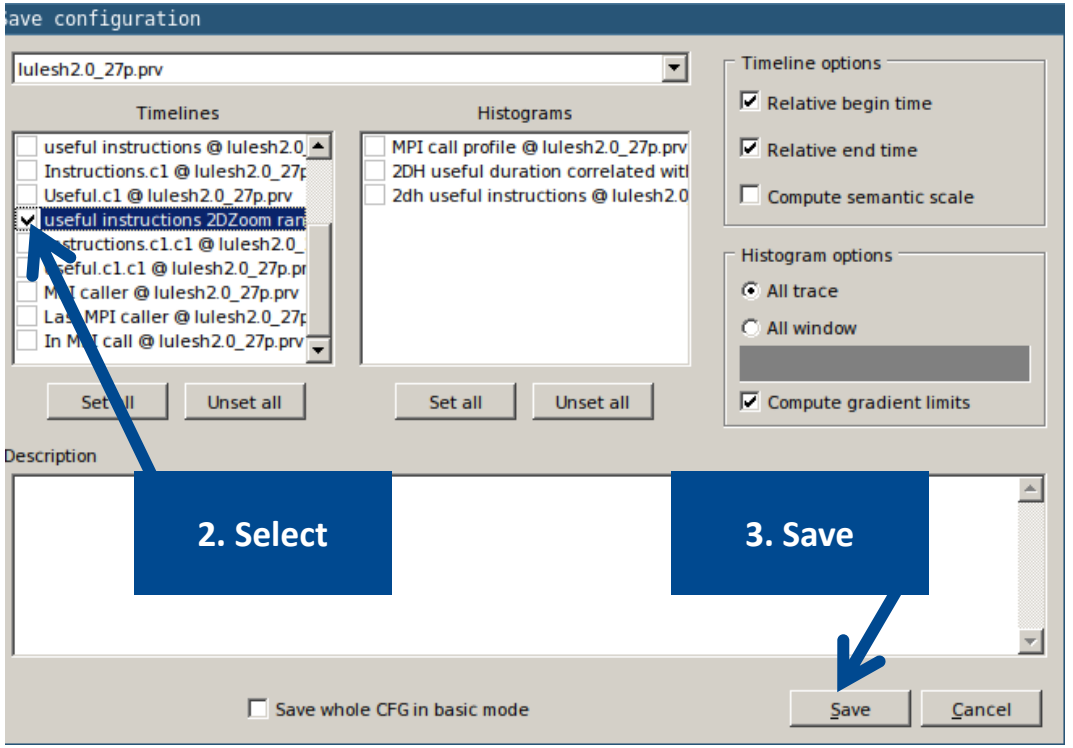


Save CFG's (2 methods)



1. Main Paraver window



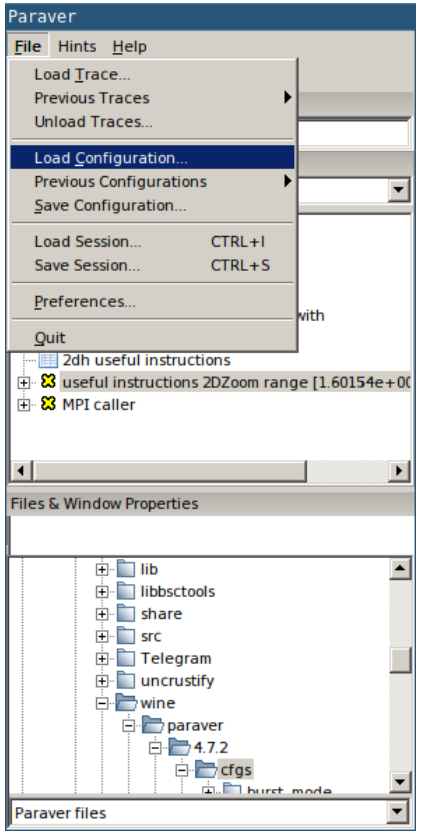


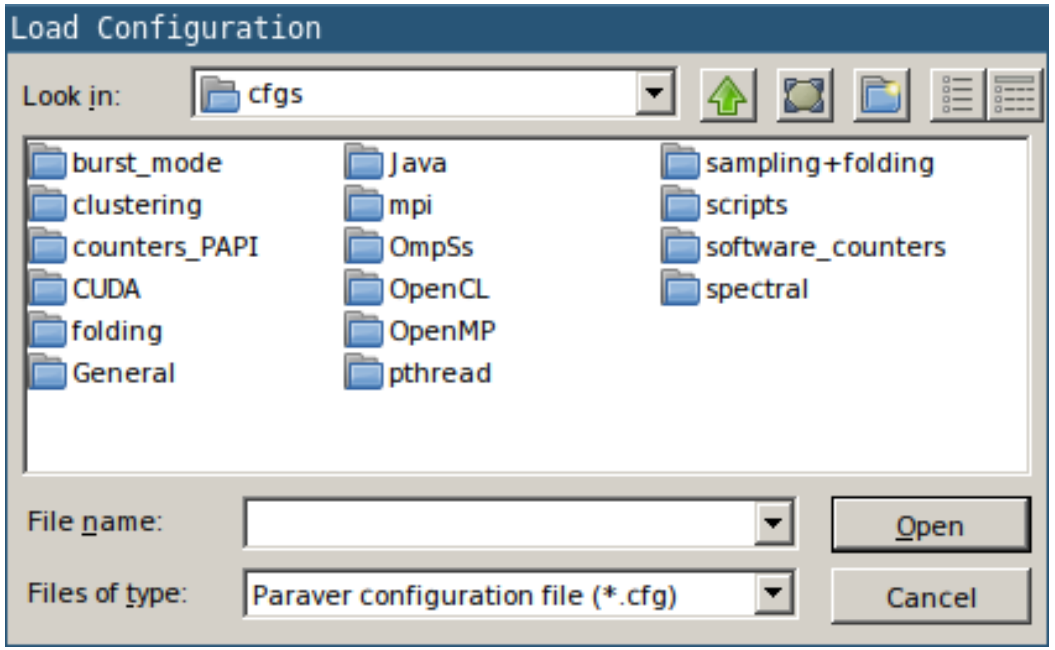
2. Select

3. Save

CFG's distribution

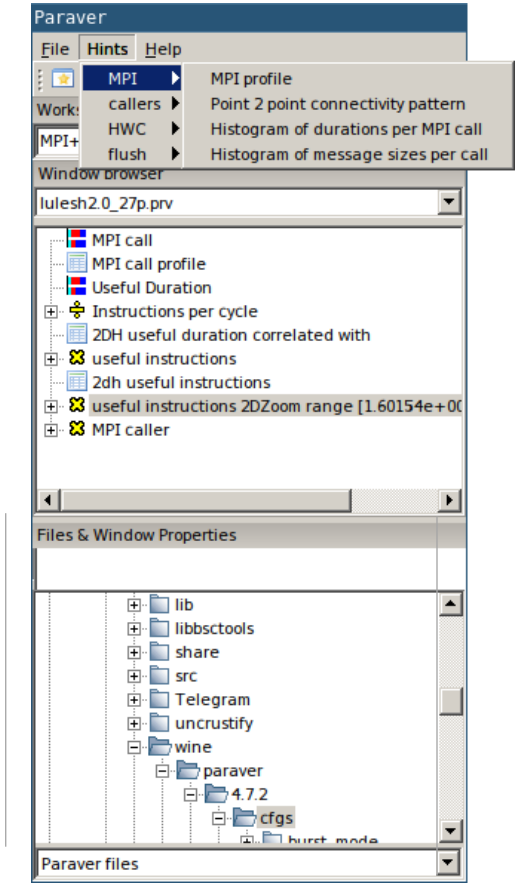
- Paraver comes with many more included CFG's





Hints: a good place to start!

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



Cluster-based analysis

Use clustering analysis

- Run clustering

```
login1> module load clustering_suite/2.6.8
login1> cd $HOME/tools-material/clustering
login1> BurstClustering -d cluster.xml -i ../extrae_mn4/lulesh_mn4_27p.prv \
        -o lulesh_mn4_27p_clustered.prv
```

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

```
login1> ls $HOME/tools-material/traces/lulesh_mn4_27p.prv
```

- Copy the results to your computer

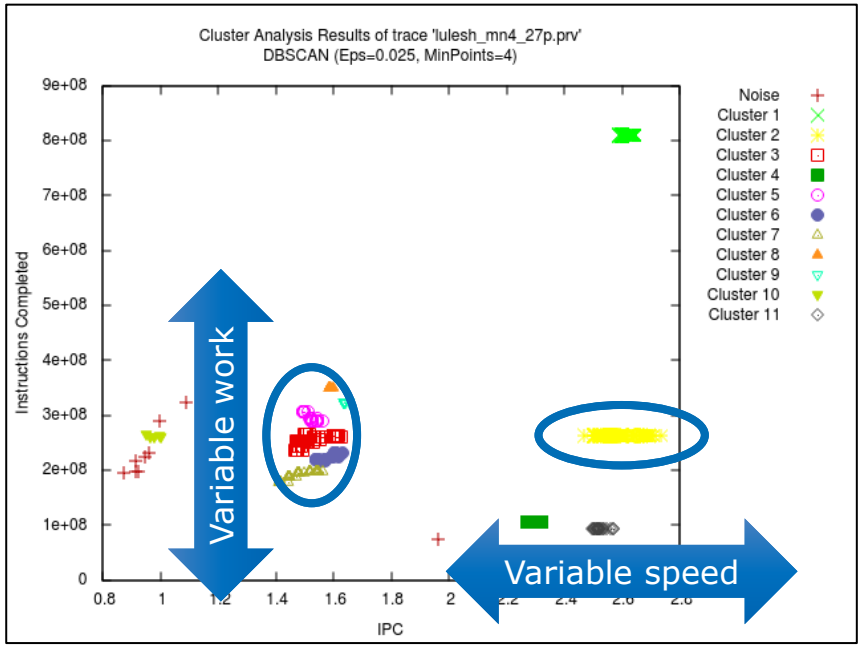
```
laptop> scp <USER>@mn1:$HOME/tools-material/clustering/* ./
```

Cluster-based analysis

- Check the resulting scatter plot

```
laptop> gnuplot lulesh_mn4_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**

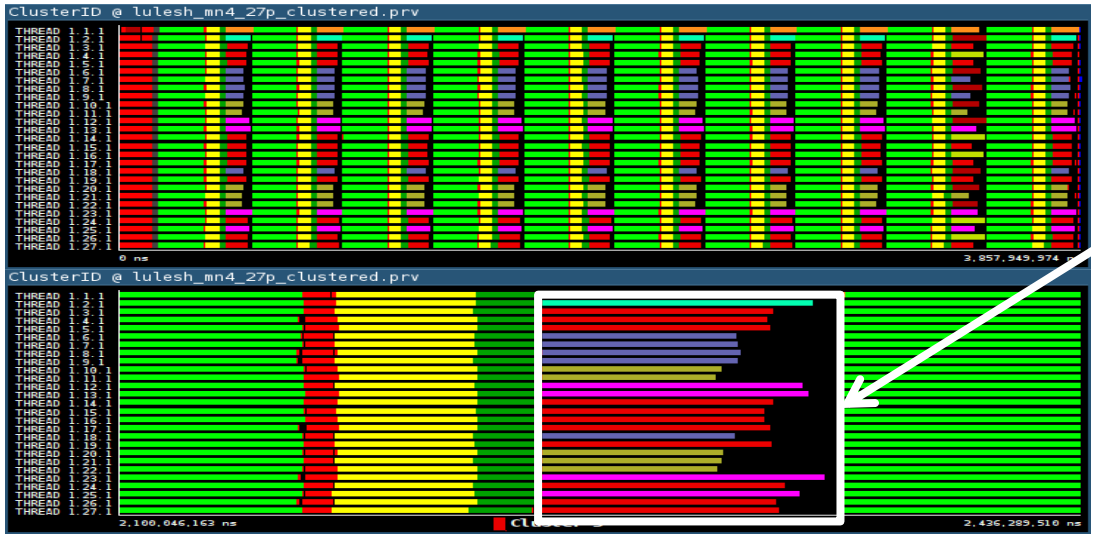


Correlating scatter plot and time distribution

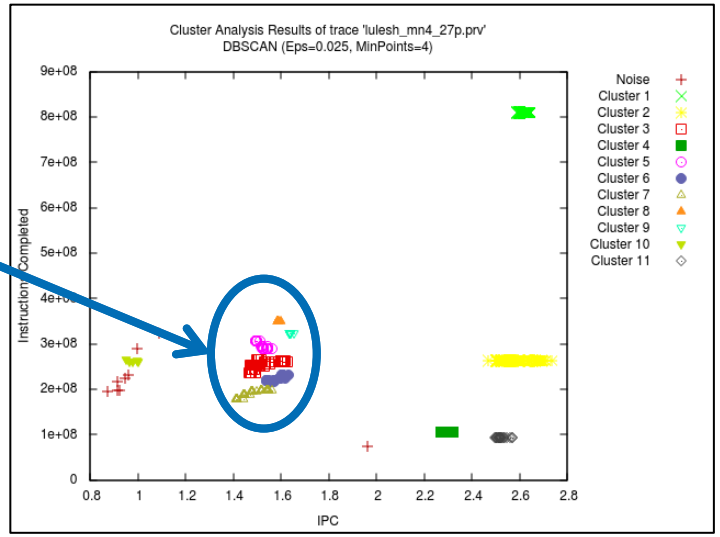
- Open the clustered trace with Paraver and look at it

```
laptop> $HOME/paraver/bin/wxparaver lulesh_mn4_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 Melià
(tools@bsc.es)
Barcelona Supercomputing Center