Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Extrae & Paraver

Judit Giménez, Lau Mercadal, Germán Llort

tools@bsc.es

2024-09-05

Performance Analysis and Tools

Extrae features

- Platforms
  - Intel, ARM, RISC-V, POWER, Cray, BlueGene, Android, Fujitsu Sparc …
- Parallel programming models
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenACC, 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 anywhere in your program (Extrae API)

No need to recompile nor relink!

Extrae overhead

	Karolina
Punctual event	175 ns
Event + PAPI counters	319 ns
Event + 1-level callstack	1.471 us
Event + 6-levels callstack	3.461 us

How does Extrae work?

- Symbol substitution through LD_PRELOAD

```
export LD_PRELOAD=$EXTRAE_HOME/lib/libmpitrace.so
```

  - Specific libraries for each runtime and combinations
    - MPI
    - OpenMP
    - OpenMP+MPI
    - CUDA
    - …

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 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`

Step 1: Adapt the job script to load Extrae

```
hpc$ vi job.slurm
```

```
#!/bin/bash

#SBATCH --job-name=lulesh2.0_s65_27p
#SBATCH --output=%x_%j.out
#SBATCH --error=%x_%j.err
#SBATCH --ntasks=27
#SBATCH --time=00:10:00

export TRACE_NAME=lulesh2.0_s65_27p.prv
srun ./trace.sh ./lulesh2.0 -i 10 -s 65 -p
```

Run with Extrae

Step 1: Adapt the job script to load Extrae

```
hpc$ vi trace.sh
```

```
#!/bin/bash

#SBATCH --job-name=lulesh2.0_s65_27p
#SBATCH --output=%x_%j.out
#SBATCH --error=%x_%j.err
#SBATCH --ntasks=27
#SBATCH --time=00:10:00

export TRACE_NAME=lulesh2.0_s65_27p.prv
srun ./trace.sh ./lulesh2.0 -i 10 -s 65 -p
```

```
#!/bin/bash

# Configure Extrae
export EXTRAE_HOME=<installation_base_path>
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
$*
```

What to trace?

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

```
hpc$ sbatch job.slurm
```

- Easy!

Step 2: Extrae XML configuration

```
hpc$ vi extrae.xml
```

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

<openmp enabled="yes">
  <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)

```
hpc$ vi 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_VEC_SP, PAPI_SR_INS, 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)

```
hpc$ vi 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

All done! Check your resulting trace

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

```
hpc$ ls -l
    ...
    lulesh2.0_s65_27p.pcf
    lulesh2.0_s65_27p.prv
    lulesh2.0_s65_27p.row
```

- Now let’s look into it !

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Paraver Installation

Judit Giménez, Lau Mercadal, Germán Llort

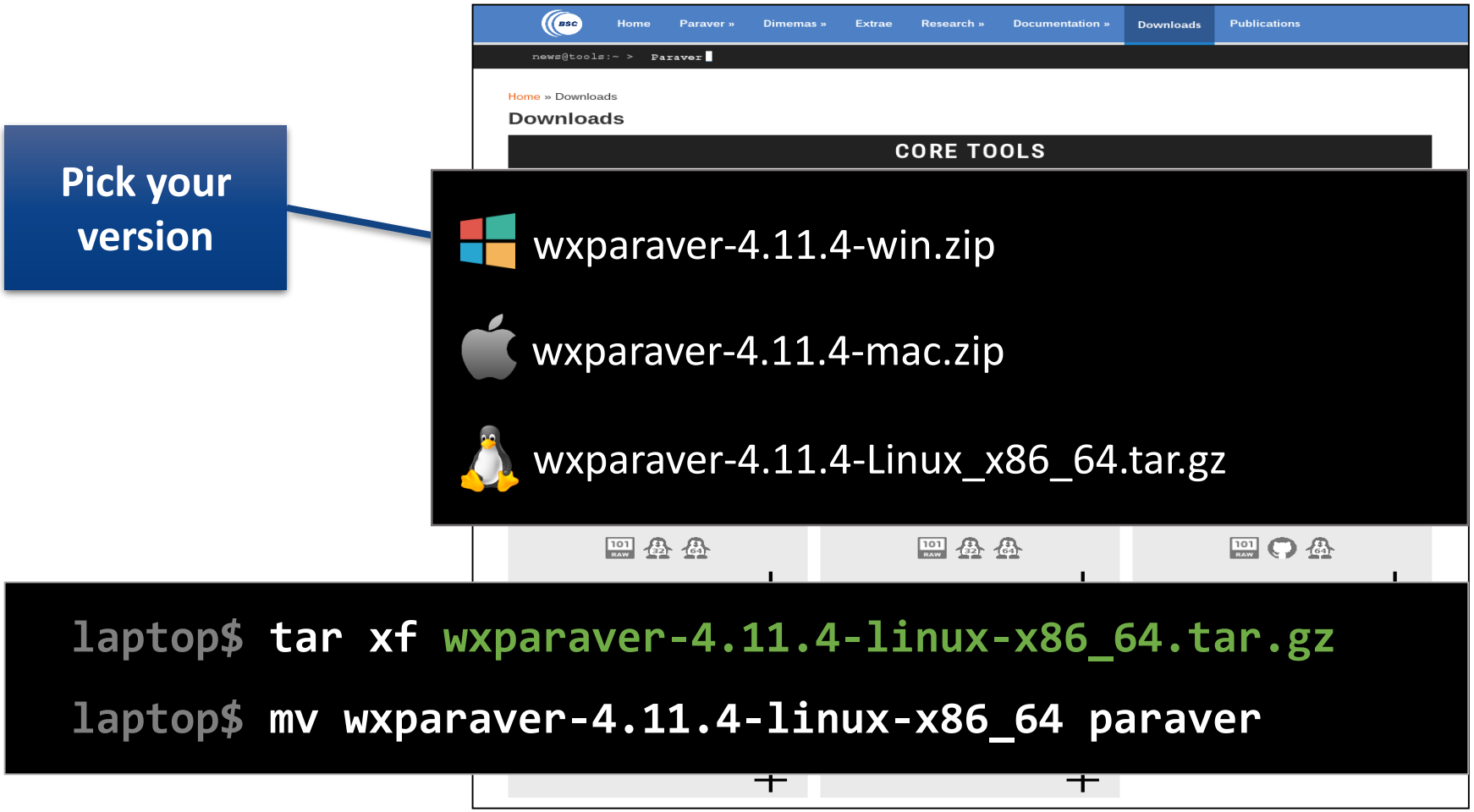
tools@bsc.es

2024-09-05

PATC: Performance Analysis and Tools

Install Paraver

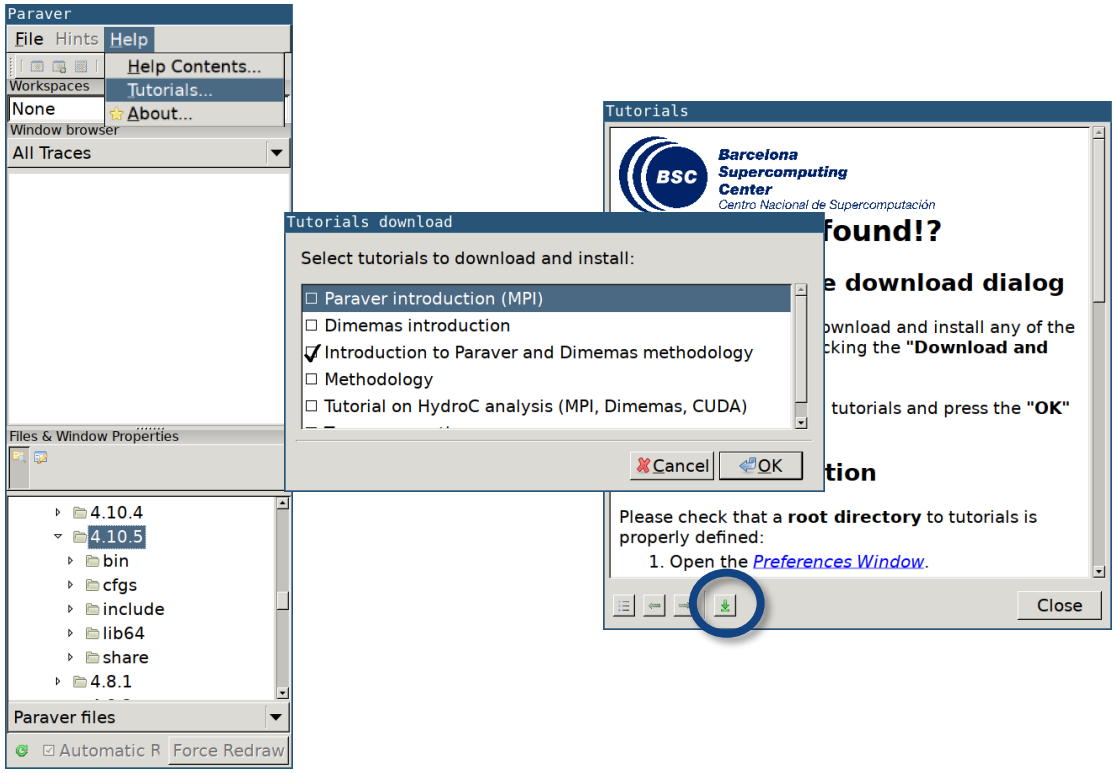
- Download from https://tools.bsc.es/downloads



Install Tutorials (Automatic)

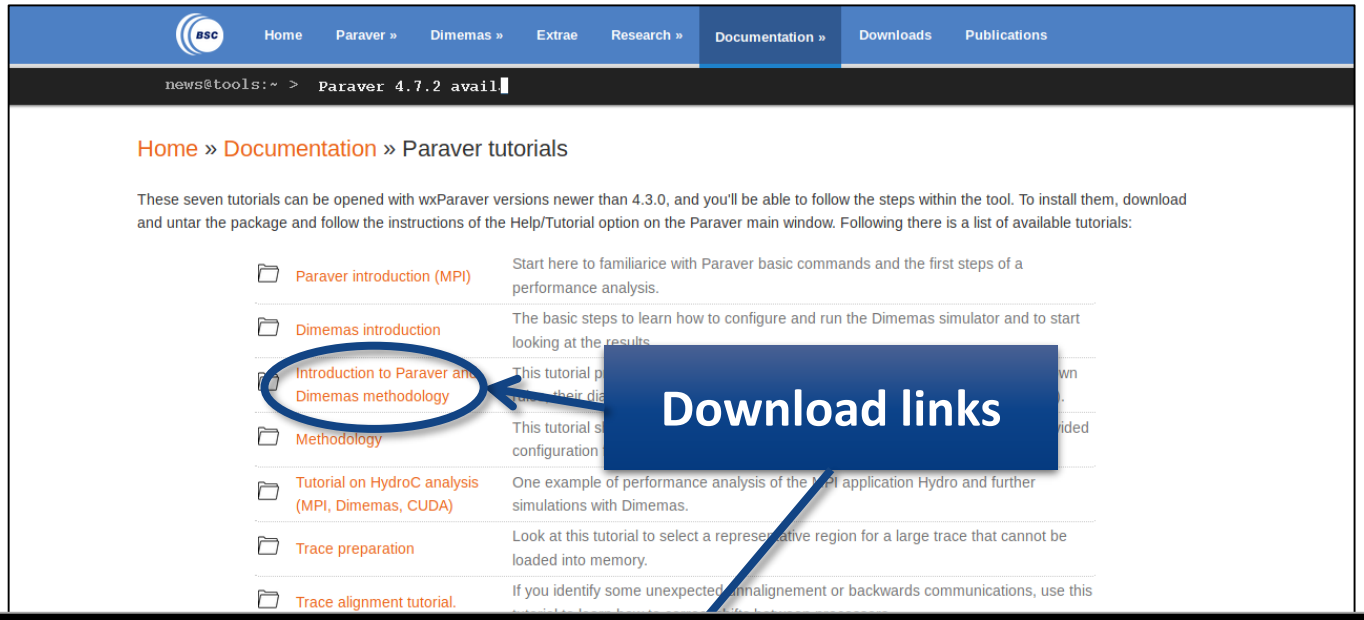
- Start Paraver

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



Install Tutorials (Manual)

- Download tutorials:
  - Documentation →Paraver Tutorials

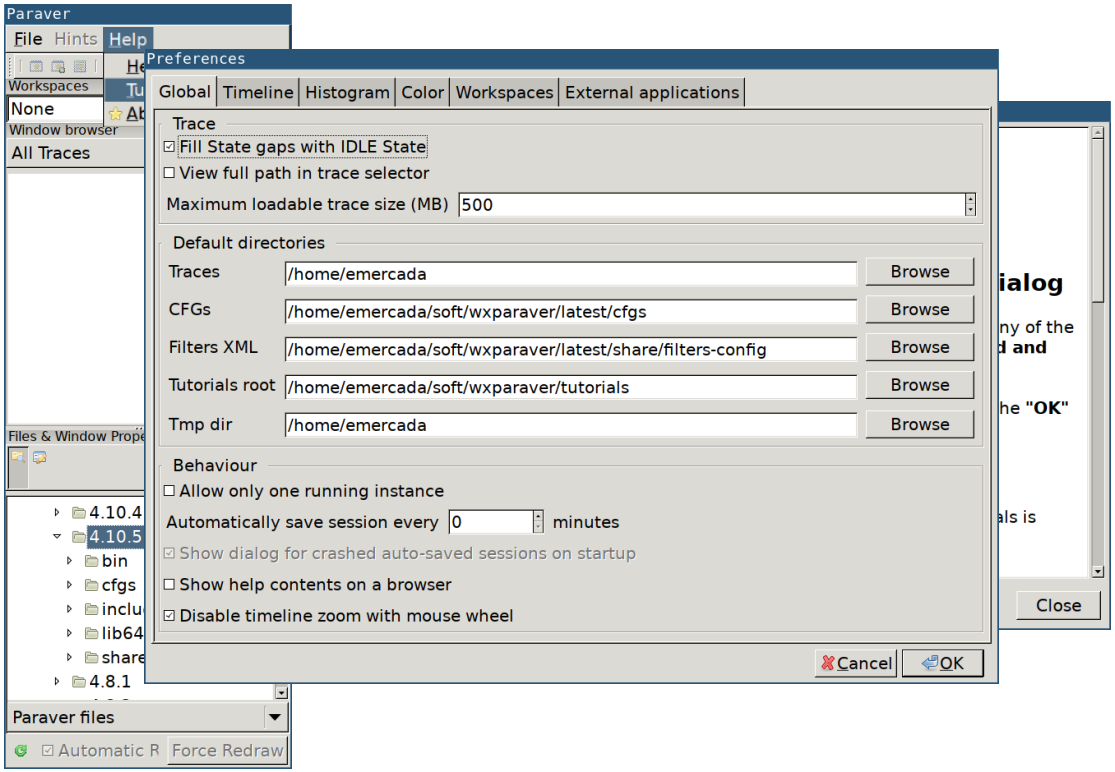



```
laptop$ mkdir paraver/tutorials
laptop$ tar xf 3.introduction* -C paraver/tutorials/
```

Install Tutorials (Manual)

- Start Paraver

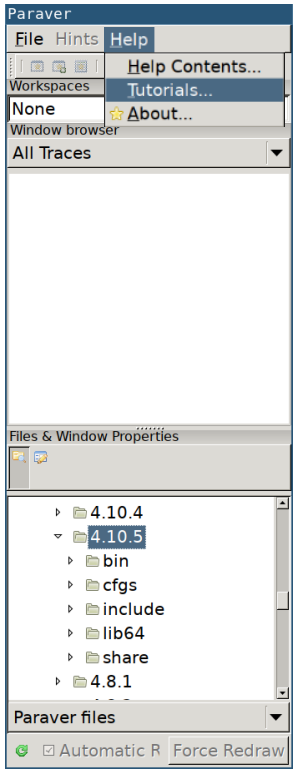
```
laptop$ paraver/bin/wxparaver
```

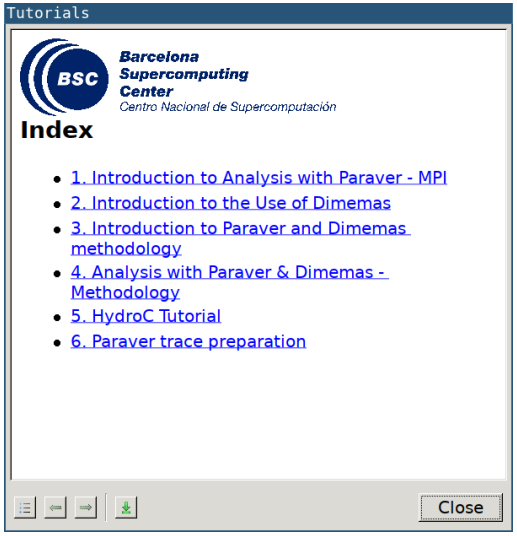


Install Paraver

- Start Paraver

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





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Paraver Hands-On

Judit Giménez, Lau Mercadal, Germán Llort

tools@bsc.es

2024-09-05

PATC: Performance Analysis and Tools

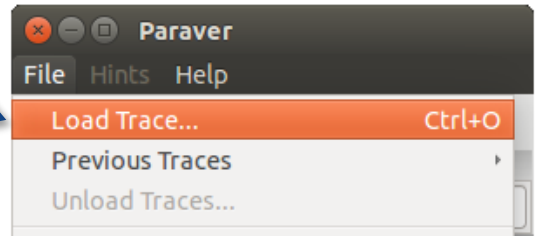
First steps of analysis

- Copy the trace to your laptop

```
laptop$ scp <USER>@hpc:lulesh2.0_s65_27p.* ./
```

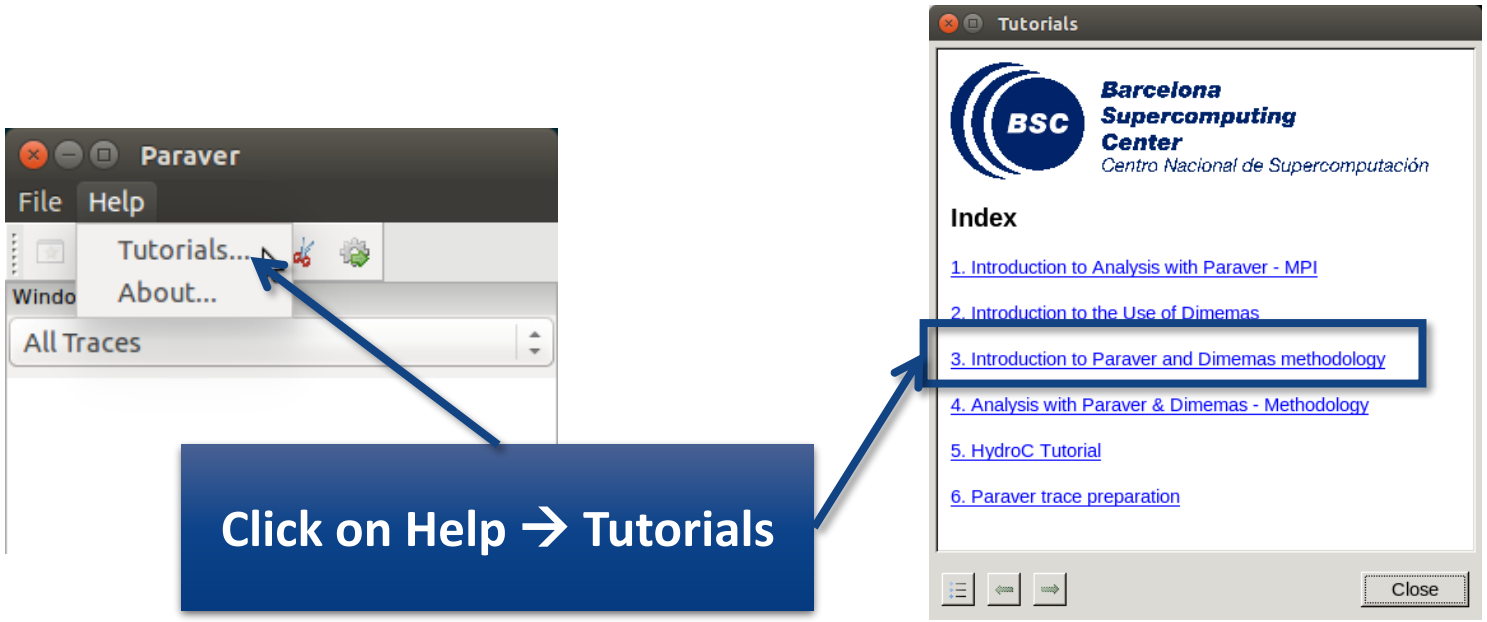
- Load the trace with Paraver

Click on File → Load Trace → Browse to … "lulesh2.0_s65_27p.prv"



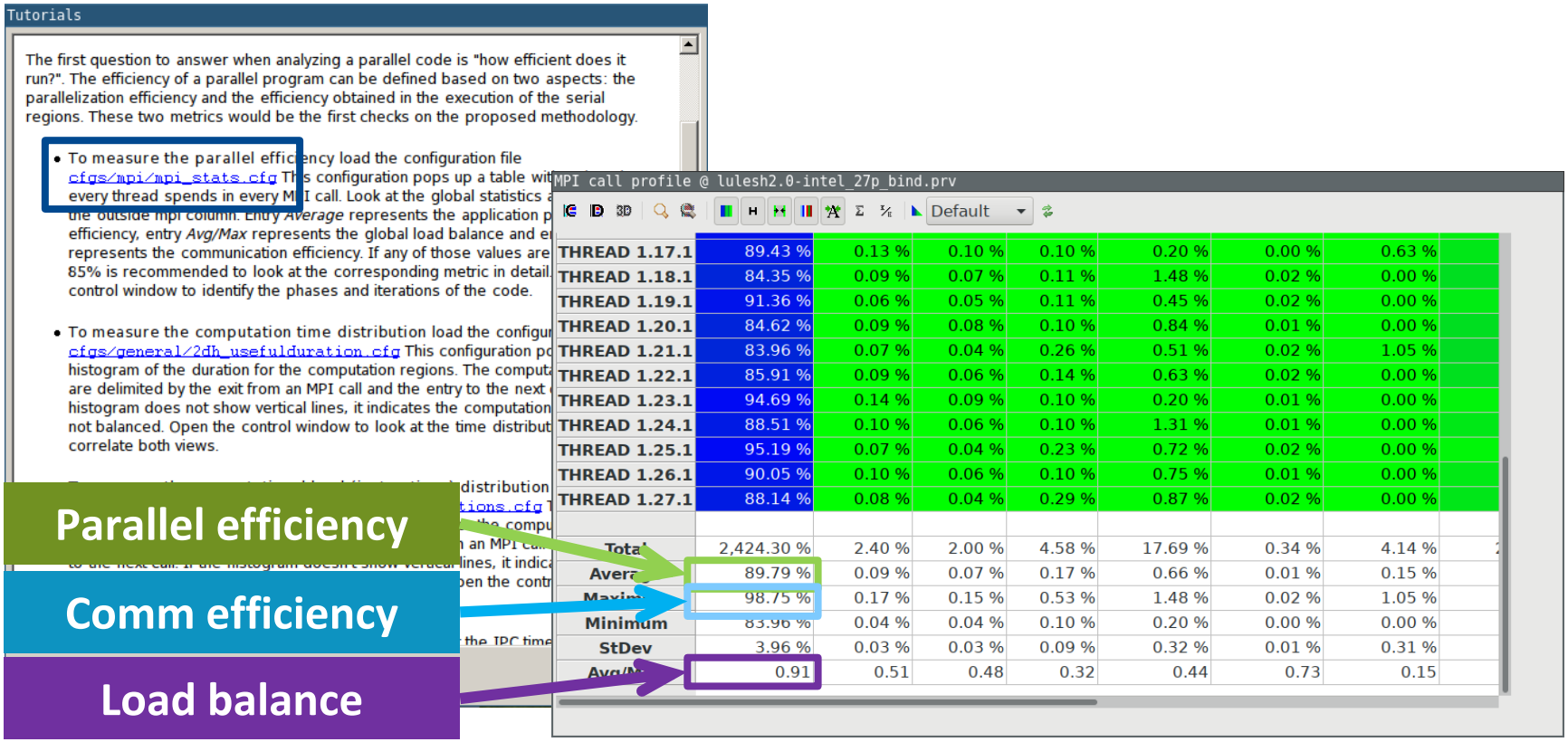
First steps of analysis

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

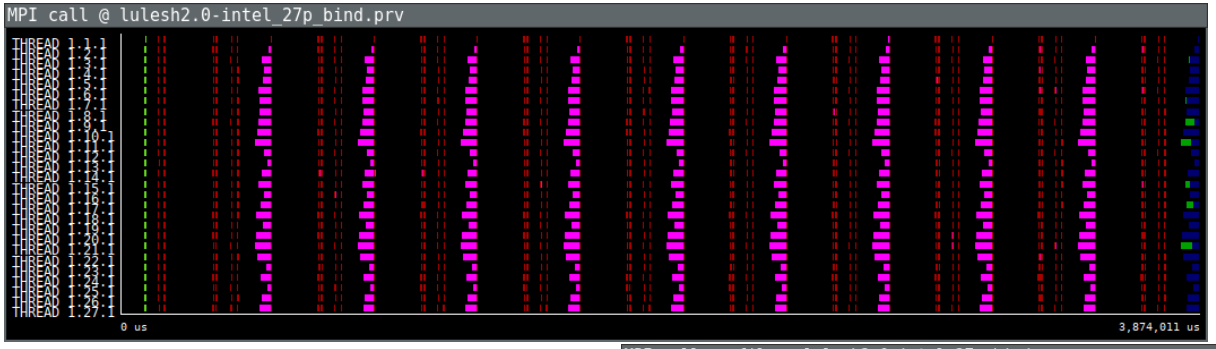


Measure the parallel efficiency

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



Focus on the iterative part



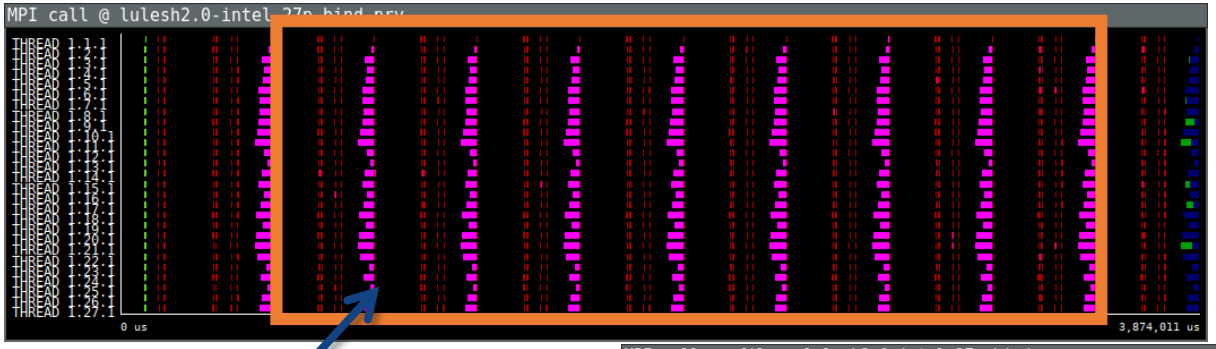
Click on Open Control Window

MPI call profile @ lulesh2.0-intel_27p_bind.prv

Default

THREAD 1.17.1	89.43 %	0.13 %	0.10 %	0.10 %	0.20 %	0.00 %	0.63 %
THREAD 1.18.1	84.35 %	0.09 %	0.07 %	0.11 %	1.48 %	0.02 %	0.00 %
THREAD 1.19.1	91.36 %	0.06 %	0.05 %	0.11 %	0.45 %	0.02 %	0.00 %
THREAD 1.20.1	84.62 %	0.09 %	0.08 %	0.10 %	0.84 %	0.01 %	0.00 %
THREAD 1.21.1	83.96 %	0.07 %	0.04 %	0.26 %	0.51 %	0.02 %	1.05 %
THREAD 1.22.1	85.91 %	0.09 %	0.06 %	0.14 %	0.63 %	0.02 %	0.00 %
THREAD 1.23.1	94.69 %	0.14 %	0.09 %	0.10 %	0.20 %	0.01 %	0.00 %
THREAD 1.24.1	88.51 %	0.10 %	0.06 %	0.10 %	1.31 %	0.01 %	0.00 %
THREAD 1.25.1	95.19 %	0.07 %	0.04 %	0.23 %	0.72 %	0.02 %	0.00 %
THREAD 1.26.1	90.05 %	0.10 %	0.06 %	0.10 %	0.75 %	0.01 %	0.00 %
THREAD 1.27.1	88.14 %	0.08 %	0.04 %	0.29 %	0.87 %	0.02 %	0.00 %
Total	2,424.30 %	2.40 %	2.00 %	4.58 %	17.69 %	0.34 %	4.14 %
Average	89.79 %	0.09 %	0.07 %	0.17 %	0.66 %	0.01 %	0.15 %
Maximum	98.75 %	0.17 %	0.15 %	0.53 %	1.48 %	0.02 %	1.05 %
Minimum	83.96 %	0.04 %	0.04 %	0.10 %	0.20 %	0.00 %	0.00 %
StDev	3.96 %	0.03 %	0.03 %	0.09 %	0.32 %	0.01 %	0.31 %
Avg/Max	0.91	0.51	0.48	0.32	0.44	0.73	0.15

Focus on the iterative part



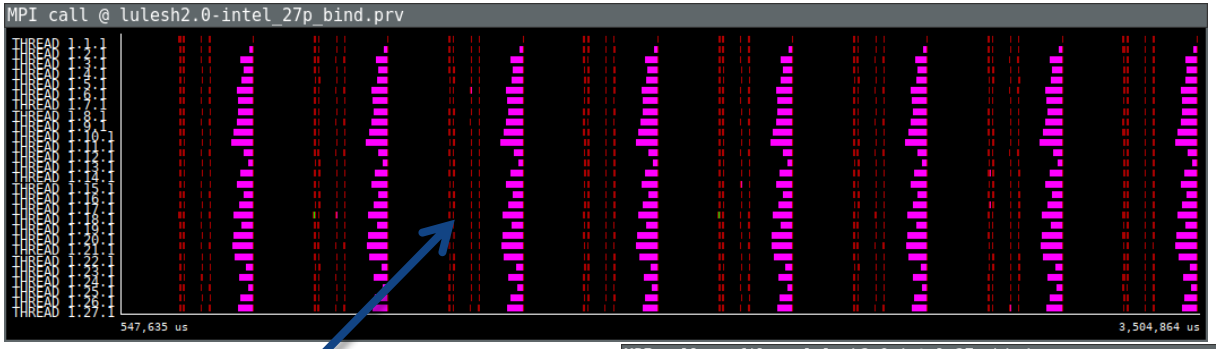
Drag & drop on this area to zoom on the iterative region

MPI call profile @ lulesh2.0-intel_27p_bind.prv

Default

THREAD 1.17.1	89.43 %	0.13 %	0.10 %	0.10 %	0.20 %	0.00 %	0.63 %
THREAD 1.18.1	84.35 %	0.09 %	0.07 %	0.11 %	1.48 %	0.02 %	0.00 %
THREAD 1.19.1	91.36 %	0.06 %	0.05 %	0.11 %	0.45 %	0.02 %	0.00 %
THREAD 1.20.1	84.62 %	0.09 %	0.08 %	0.10 %	0.84 %	0.01 %	0.00 %
THREAD 1.21.1	83.96 %	0.07 %	0.04 %	0.26 %	0.51 %	0.02 %	1.05 %
THREAD 1.22.1	85.91 %	0.09 %	0.06 %	0.14 %	0.63 %	0.02 %	0.00 %
THREAD 1.23.1	94.69 %	0.14 %	0.09 %	0.10 %	0.20 %	0.01 %	0.00 %
THREAD 1.24.1	88.51 %	0.10 %	0.06 %	0.10 %	1.31 %	0.01 %	0.00 %
THREAD 1.25.1	95.19 %	0.07 %	0.04 %	0.23 %	0.72 %	0.02 %	0.00 %
THREAD 1.26.1	90.05 %	0.10 %	0.06 %	0.10 %	0.75 %	0.01 %	0.00 %
THREAD 1.27.1	88.14 %	0.08 %	0.04 %	0.29 %	0.87 %	0.02 %	0.00 %
Total	2,424.30 %	2.40 %	2.00 %	4.58 %	17.69 %	0.34 %	4.14 %
Average	89.79 %	0.09 %	0.07 %	0.17 %	0.66 %	0.01 %	0.15 %
Maximum	98.75 %	0.17 %	0.15 %	0.53 %	1.48 %	0.02 %	1.05 %
Minimum	83.96 %	0.04 %	0.04 %	0.10 %	0.20 %	0.00 %	0.00 %
StDev	3.96 %	0.03 %	0.03 %	0.09 %	0.32 %	0.01 %	0.31 %
Avg/Max	0.91	0.51	0.48	0.32	0.44	0.73	0.15

Recalculate efficiency of iterative region



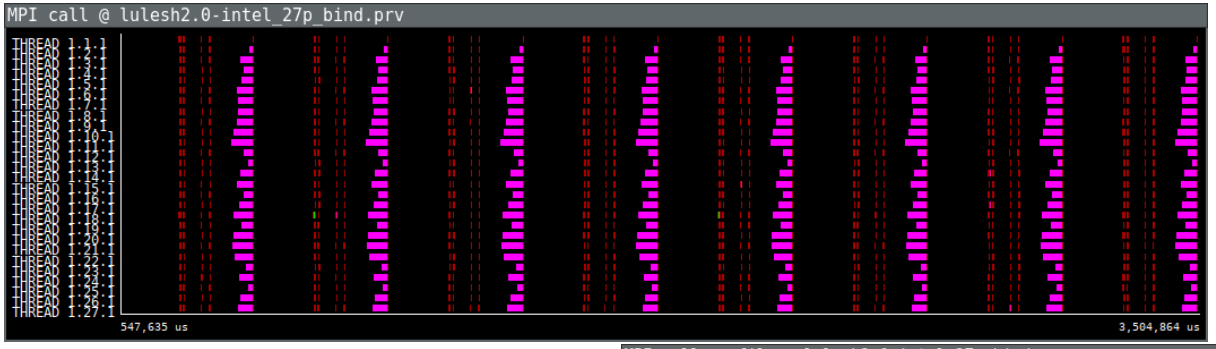
Right click
→ Copy

MPI call profile @ lulesh2.0-intel_27p_bind.prv

Default

THREAD 1.17.1	89.43 %	0.13 %	0.10 %	0.10 %	0.20 %	0.00 %	0.63 %
THREAD 1.18.1	84.35 %	0.09 %	0.07 %	0.11 %	1.48 %	0.02 %	0.00 %
THREAD 1.19.1	91.36 %	0.06 %	0.05 %	0.11 %	0.45 %	0.02 %	0.00 %
THREAD 1.20.1	84.62 %	0.09 %	0.08 %	0.10 %	0.84 %	0.01 %	0.00 %
THREAD 1.21.1	83.96 %	0.07 %	0.04 %	0.26 %	0.51 %	0.02 %	1.05 %
THREAD 1.22.1	85.91 %	0.09 %	0.06 %	0.14 %	0.63 %	0.02 %	0.00 %
THREAD 1.23.1	94.69 %	0.14 %	0.09 %	0.10 %	0.20 %	0.01 %	0.00 %
THREAD 1.24.1	88.51 %	0.10 %	0.06 %	0.10 %	1.31 %	0.01 %	0.00 %
THREAD 1.25.1	95.19 %	0.07 %	0.04 %	0.23 %	0.72 %	0.02 %	0.00 %
THREAD 1.26.1	90.05 %	0.10 %	0.06 %	0.10 %	0.75 %	0.01 %	0.00 %
THREAD 1.27.1	88.14 %	0.08 %	0.04 %	0.29 %	0.87 %	0.02 %	0.00 %
Total	2,424.30 %	2.40 %	2.00 %	4.58 %	17.69 %	0.34 %	4.14 %
Average	89.79 %	0.09 %	0.07 %	0.17 %	0.66 %	0.01 %	0.15 %
Maximum	98.75 %	0.17 %	0.15 %	0.53 %	1.48 %	0.02 %	1.05 %
Minimum	83.96 %	0.04 %	0.04 %	0.10 %	0.20 %	0.00 %	0.00 %
StDev	3.96 %	0.03 %	0.03 %	0.09 %	0.32 %	0.01 %	0.31 %
Avg/Max	0.91	0.51	0.48	0.32	0.44	0.73	0.15

Recalculate efficiency of iterative region



Right click → Paste → Time

MPI call profile @ lulesh2.0-intel_27p_bind.prv

Default

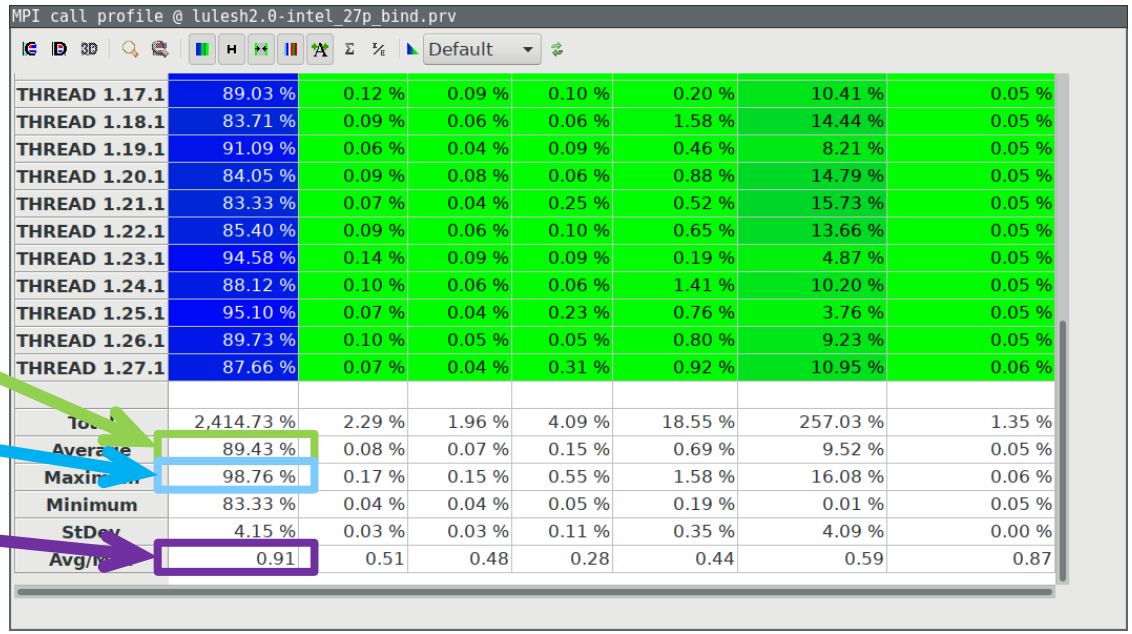
THREAD 1.17.1	89.03 %	0.12 %	0.09 %	0.10 %	0.20 %	10.41 %	0.05 %
THREAD 1.18.1	83.71 %	0.09 %	0.06 %	0.06 %	1.58 %	14.44 %	0.05 %
THREAD 1.19.1	91.09 %	0.06 %	0.04 %	0.09 %	0.46 %	8.21 %	0.05 %
THREAD 1.20.1	84.05 %	0.09 %	0.08 %	0.06 %	0.88 %	14.79 %	0.05 %
THREAD 1.21.1	83.33 %	0.07 %	0.04 %	0.25 %	0.52 %	15.73 %	0.05 %
THREAD 1.22.1	85.40 %	0.09 %	0.06 %	0.10 %	0.65 %	13.66 %	0.05 %
THREAD 1.23.1	94.58 %	0.14 %	0.09 %	0.09 %	0.19 %	4.87 %	0.05 %
THREAD 1.24.1	88.12 %	0.10 %	0.06 %	0.06 %	1.41 %	10.20 %	0.05 %
THREAD 1.25.1	95.10 %	0.07 %	0.04 %	0.23 %	0.76 %	3.76 %	0.05 %
THREAD 1.26.1	89.73 %	0.10 %	0.05 %	0.05 %	0.80 %	9.23 %	0.05 %
THREAD 1.27.1	87.66 %	[illegible] %	0.04 %	0.31 %	0.92 %	10.95 %	0.06 %
Total	2,414.73 %	2.29 %	1.96 %	4.09 %	18.55 %	257.03 %	1.35 %
Average	89.43 %	0.08 %	0.07 %	0.15 %	0.69 %	9.52 %	0.05 %
Maximum	98.76 %	0.17 %	0.15 %	0.55 %	1.58 %	16.08 %	0.06 %
Minimum	83.33 %	0.04 %	0.04 %	0.05 %	0.19 %	0.01 %	0.05 %
StDev	4.15 %	0.03 %	0.03 %	0.11 %	0.35 %	4.09 %	0.00 %
Avg/Max	0.91	0.51	0.48	0.28	0.44	0.59	0.87

Efficiency of iterative region

Parallel efficiency

Comm efficiency

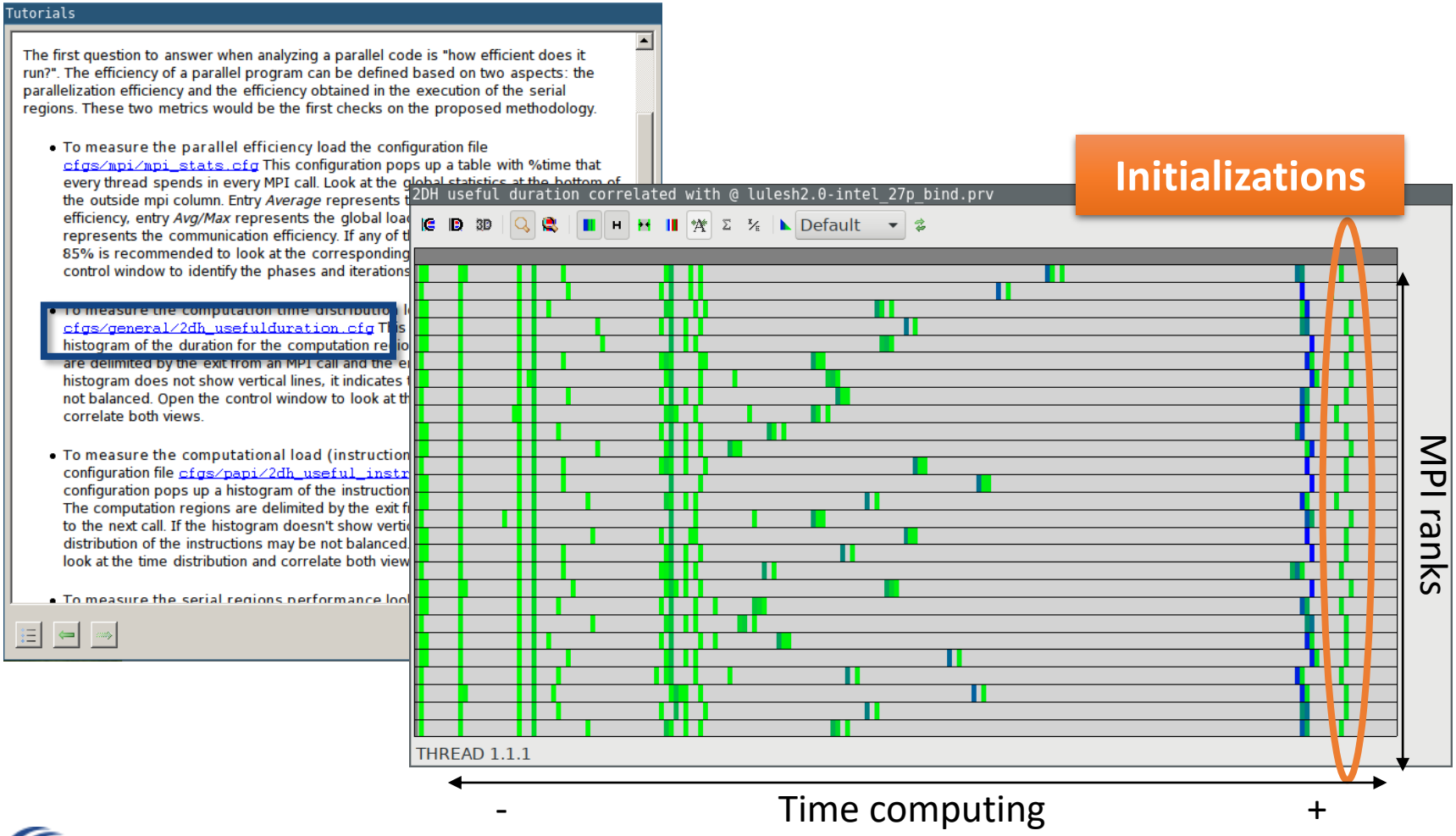
Load balance



THREAD 1.17.1	89.03 %	0.12 %	0.09 %	0.10 %	0.20 %	10.41 %	0.05 %
THREAD 1.18.1	83.71 %	0.09 %	0.06 %	0.06 %	1.58 %	14.44 %	0.05 %
THREAD 1.19.1	91.09 %	0.06 %	0.04 %	0.09 %	0.46 %	8.21 %	0.05 %
THREAD 1.20.1	84.05 %	0.09 %	0.08 %	0.06 %	0.88 %	14.79 %	0.05 %
THREAD 1.21.1	83.33 %	0.07 %	0.04 %	0.25 %	0.52 %	15.73 %	0.05 %
THREAD 1.22.1	85.40 %	0.09 %	0.06 %	0.10 %	0.65 %	13.66 %	0.05 %
THREAD 1.23.1	94.58 %	0.14 %	0.09 %	0.09 %	0.19 %	4.87 %	0.05 %
THREAD 1.24.1	88.12 %	0.10 %	0.06 %	0.06 %	1.41 %	10.20 %	0.05 %
THREAD 1.25.1	95.10 %	0.07 %	0.04 %	0.23 %	0.76 %	3.76 %	0.05 %
THREAD 1.26.1	89.73 %	0.10 %	0.05 %	0.05 %	0.80 %	9.23 %	0.05 %
THREAD 1.27.1	87.66 %	0.07 %	0.04 %	0.31 %	0.92 %	10.95 %	0.06 %
To…	2,414.73 %	2.29 %	1.96 %	4.09 %	18.55 %	257.03 %	1.35 %
Avera…	89.43 %	0.08 %	0.07 %	0.15 %	0.69 %	9.52 %	0.05 %
Maxi…	98.76 %	0.17 %	0.15 %	0.55 %	1.58 %	16.08 %	0.06 %
Minimum	83.33 %	0.04 %	0.04 %	0.05 %	0.19 %	0.01 %	0.05 %
StDev	4.15 %	0.03 %	0.03 %	0.11 %	0.35 %	4.09 %	0.00 %
Avg/…	0.91	0.51	0.48	0.28	0.44	0.59	0.87

Computation time distribution

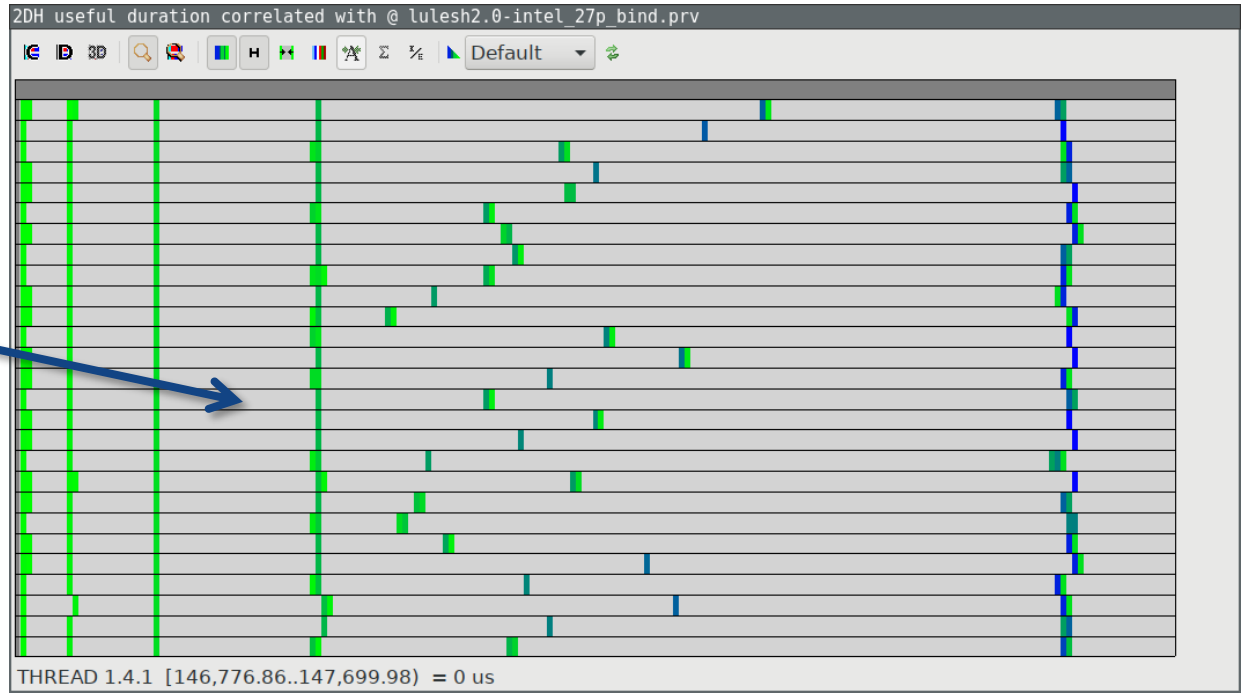
- Click on “2dh_usefulduration.cfg” (2nd link) → Shows **time computing**



Focus on the iterative part

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

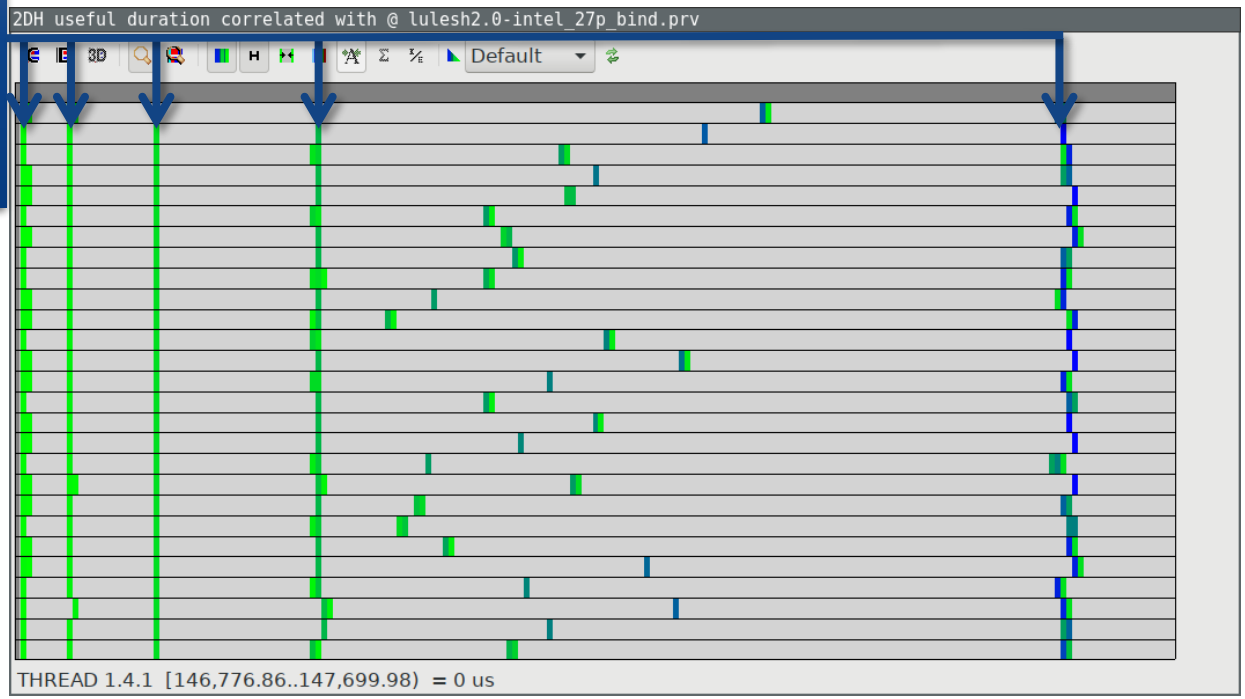
Right click → Paste → Time



Computation time distribution

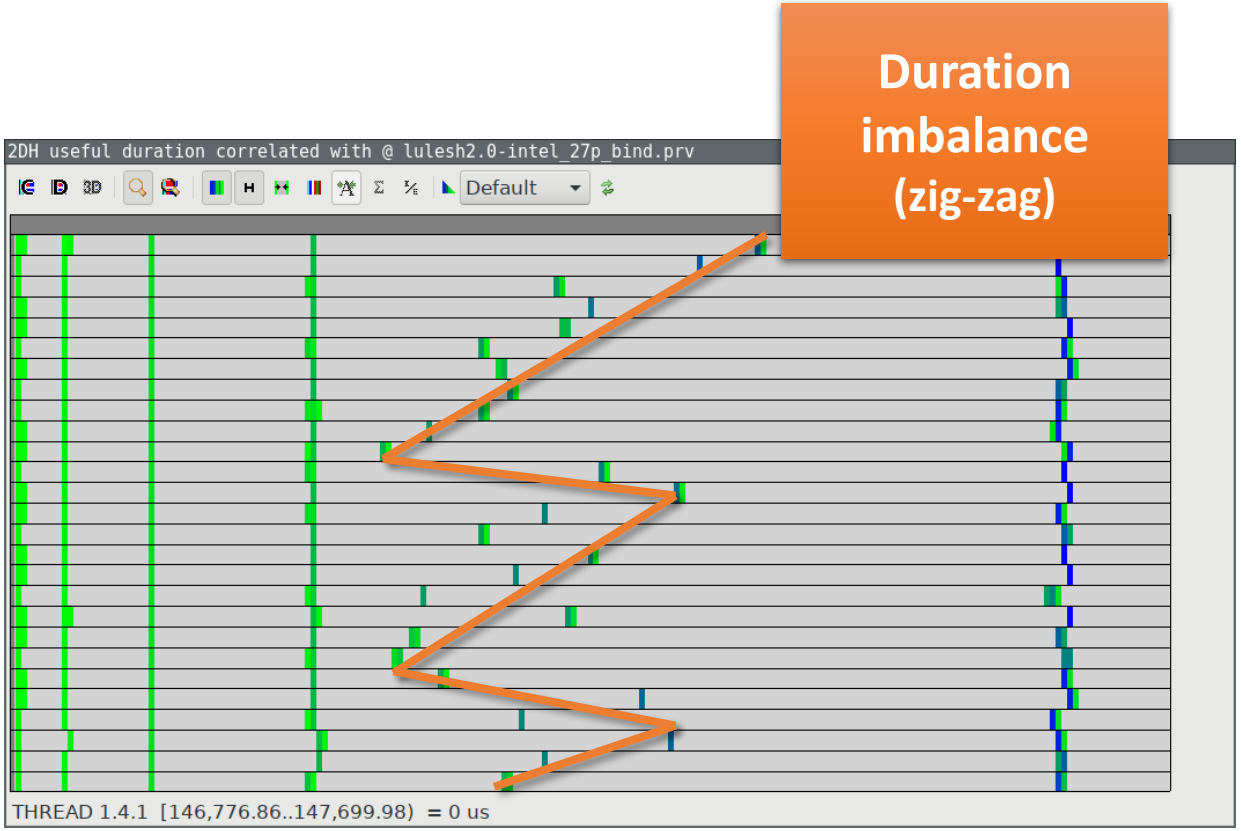
- Click on “2dh_usefulduration.cfg” (2nd link) → Shows **time computing**

Mostly straight vertical lines
=
All processes similar computing time



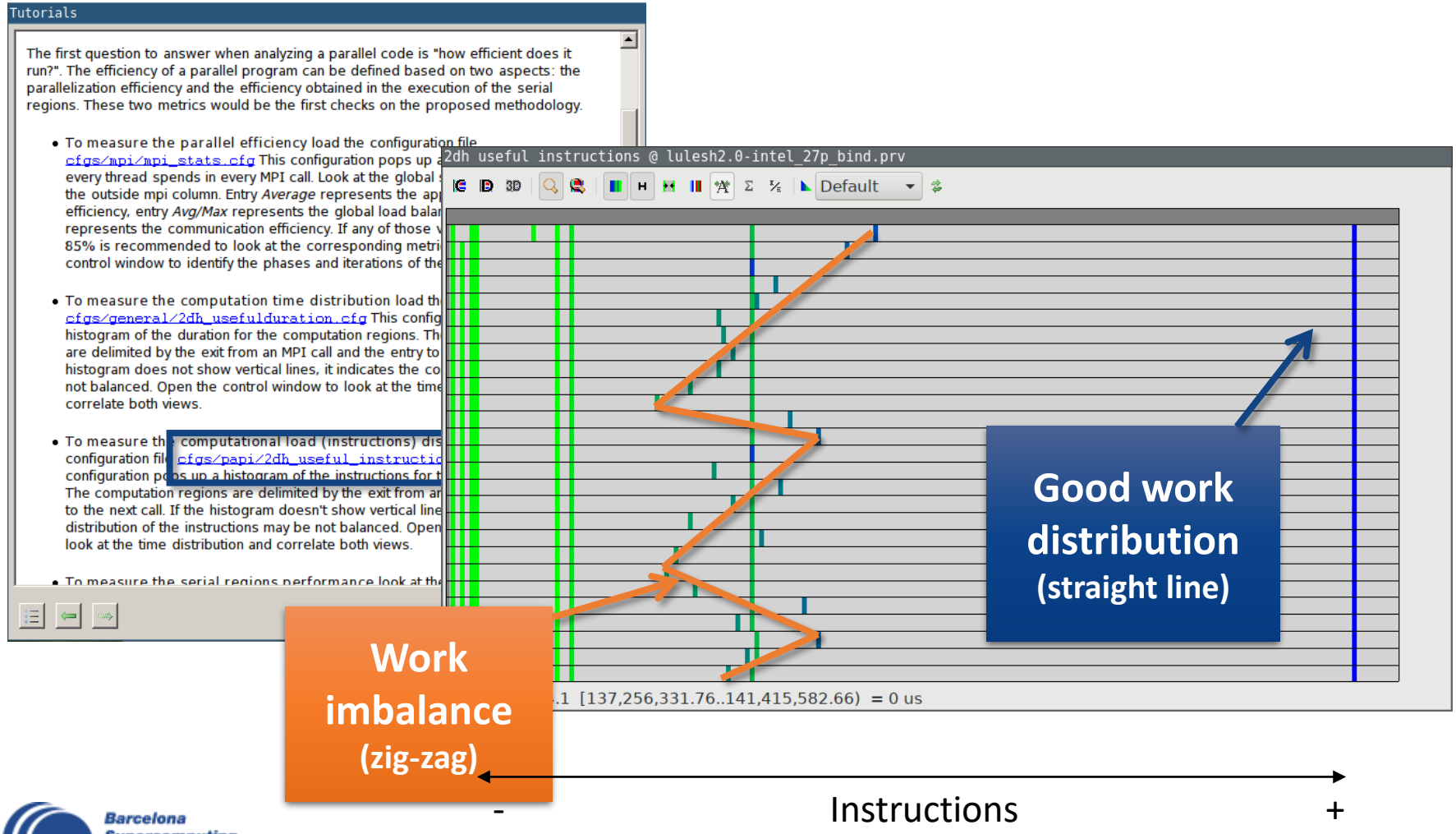
Computation time distribution

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



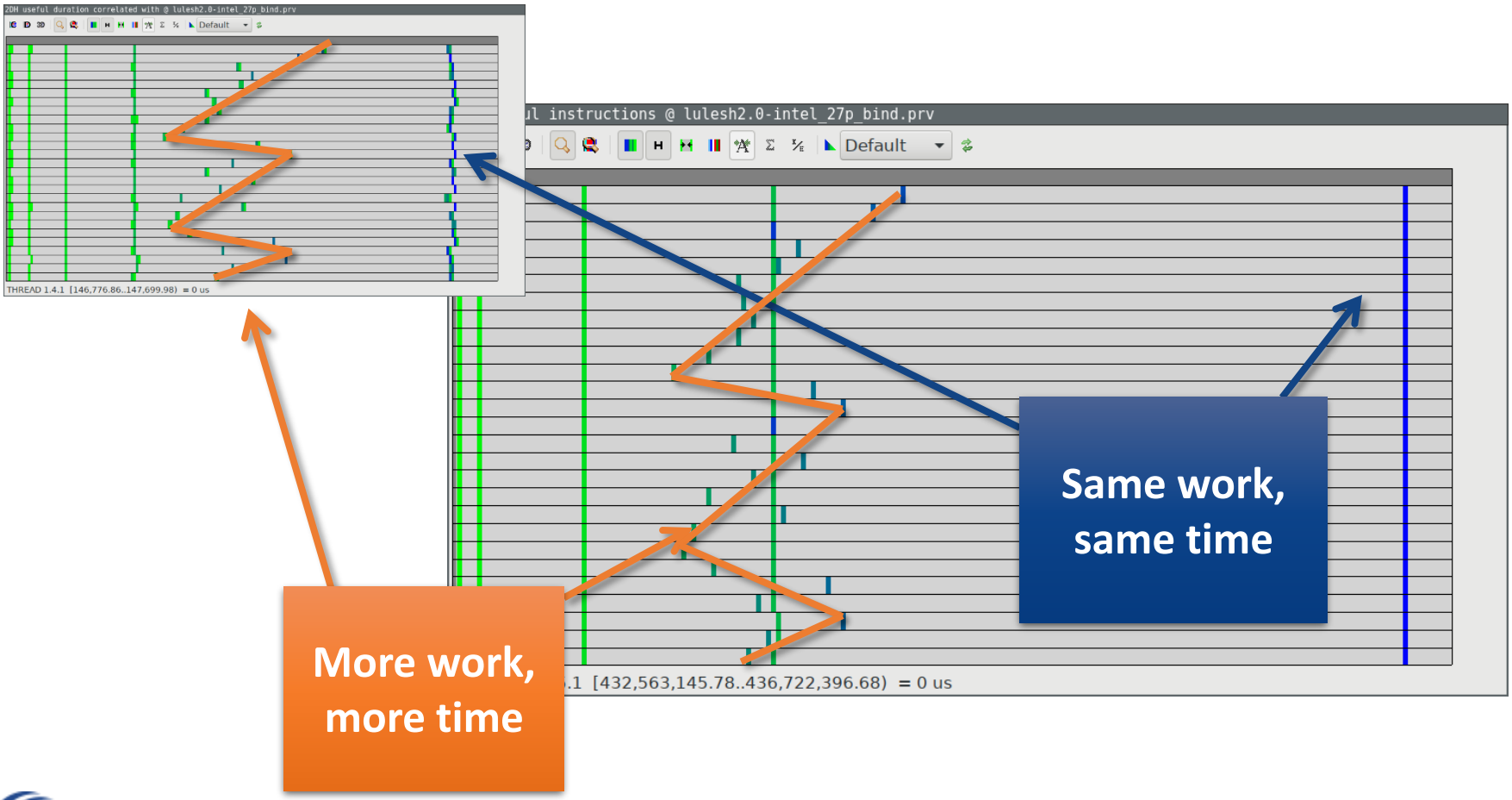
Computation load distribution

- Click on “2dh_useful_instructions.cfg” (3rd link) → Shows **amount of work**



Correlate two histograms

- Clear correlation between the **amount of work** and the **time computing**

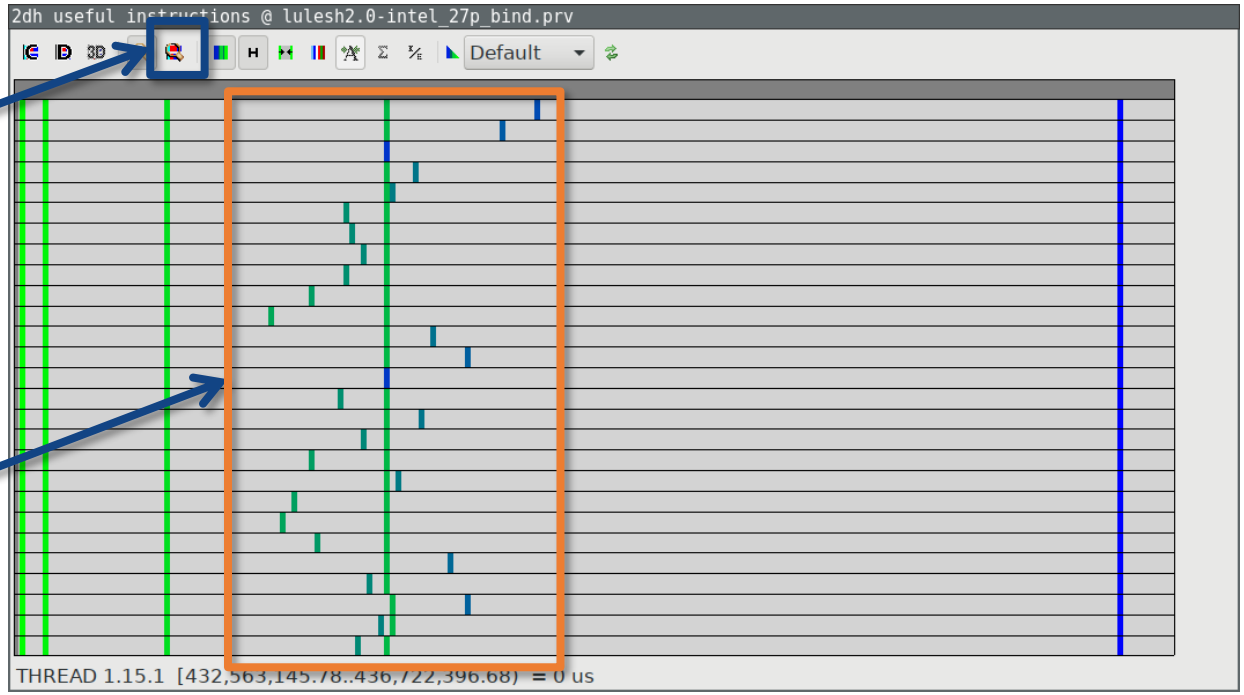


Where does this happen?

- Go from the table to the timeline

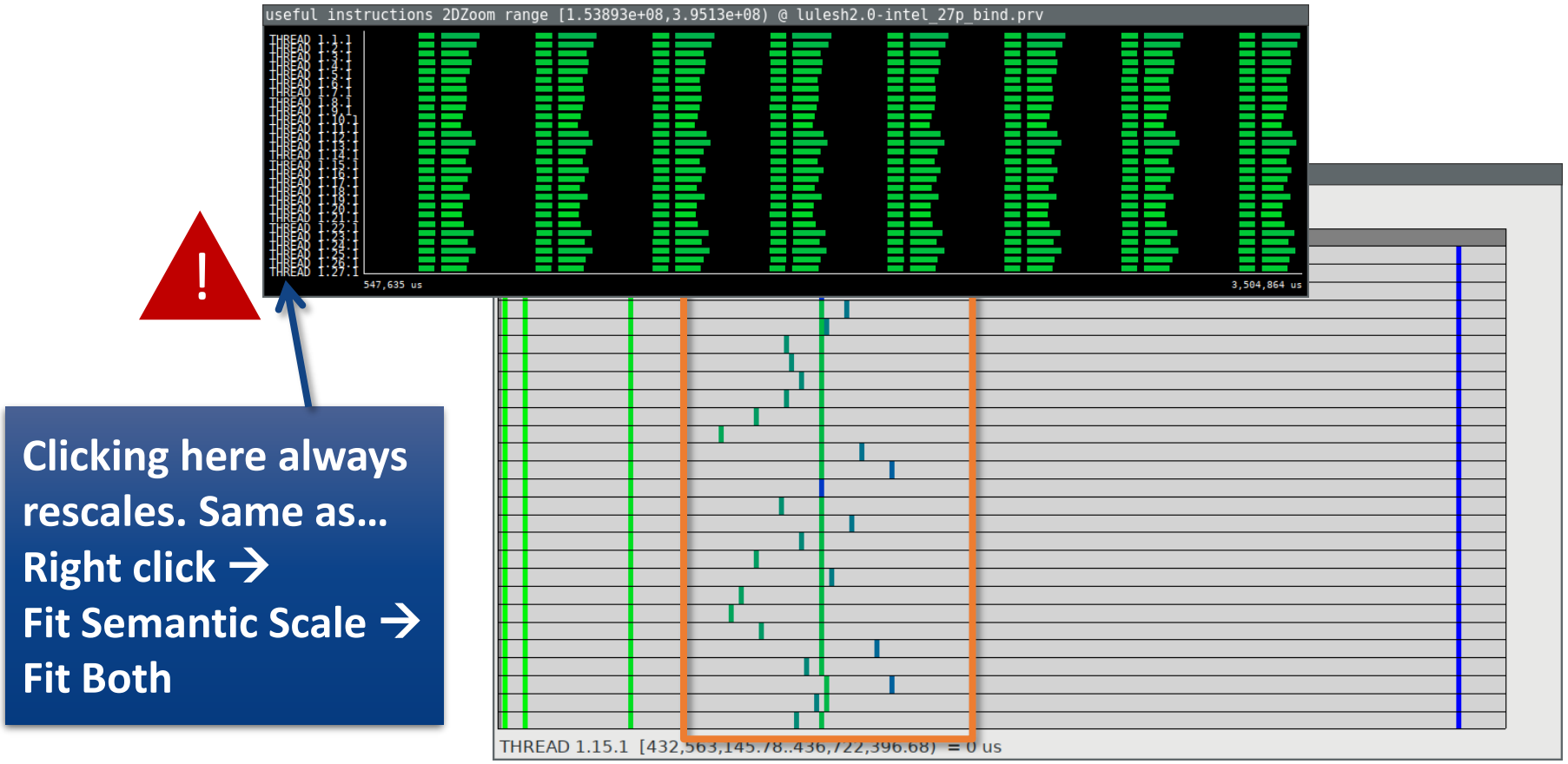
1. Click on "Open Filtered Control Window"

2. Select this area (drag-and-drop)



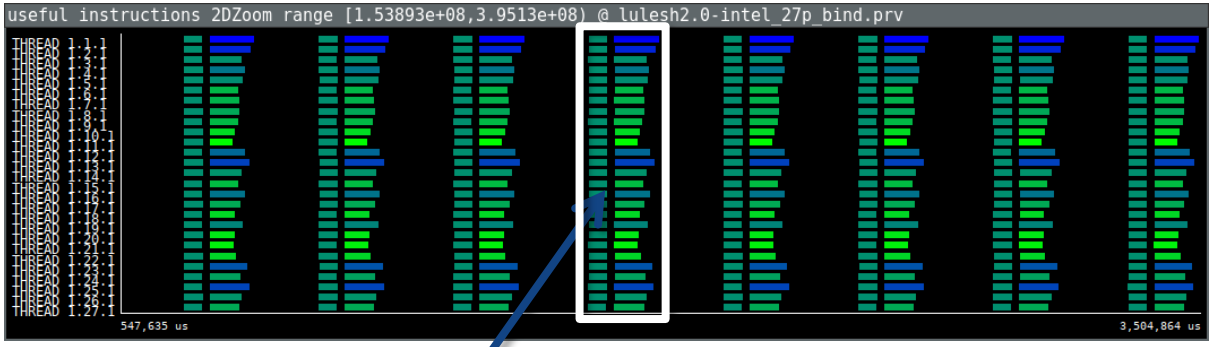
Where does this happen?

- Go from the table to the timeline



Where does this happen?

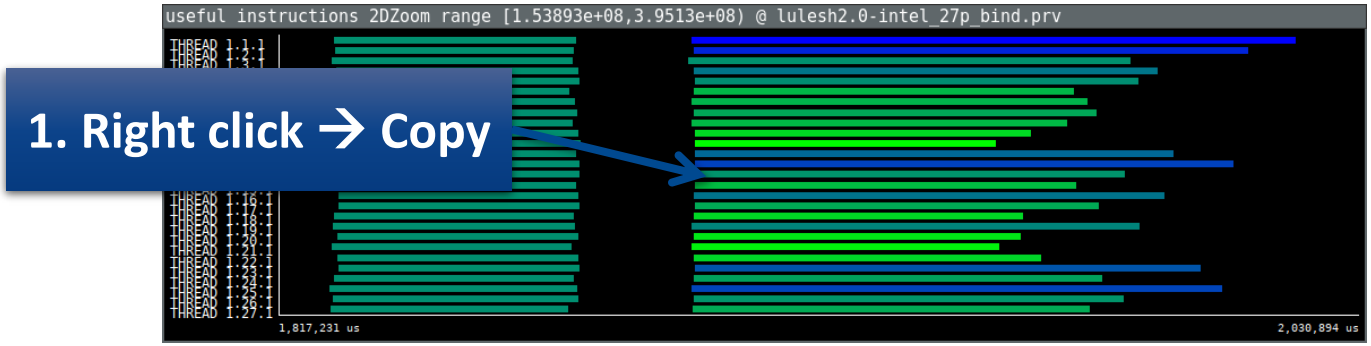
- **Slow** & **Fast** at the same time? → Imbalance



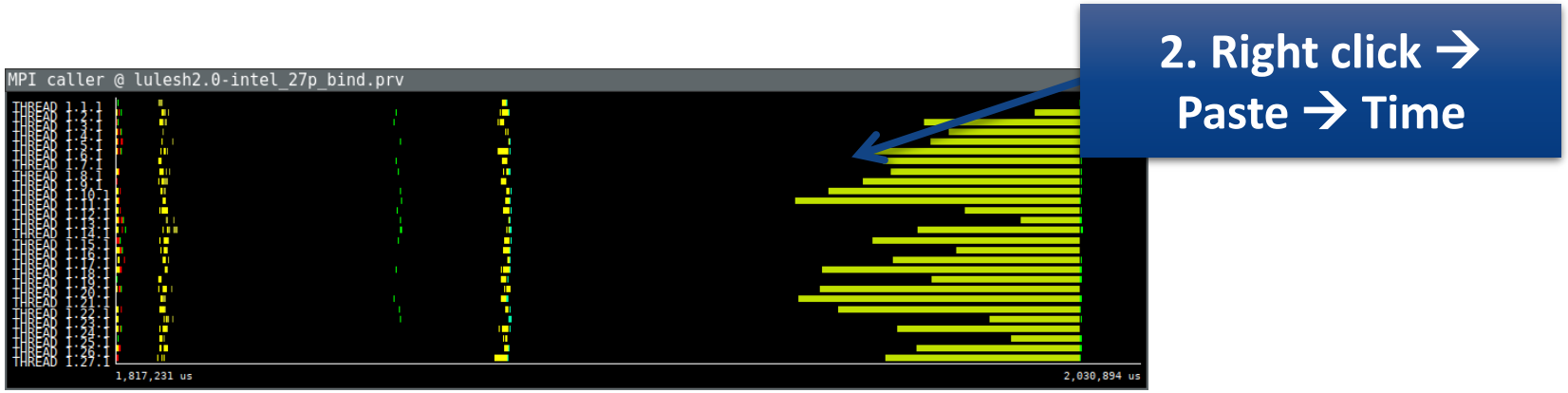
Zoom into
1 of the iterations
(by drag-and-dropping)

Where does this happen?

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

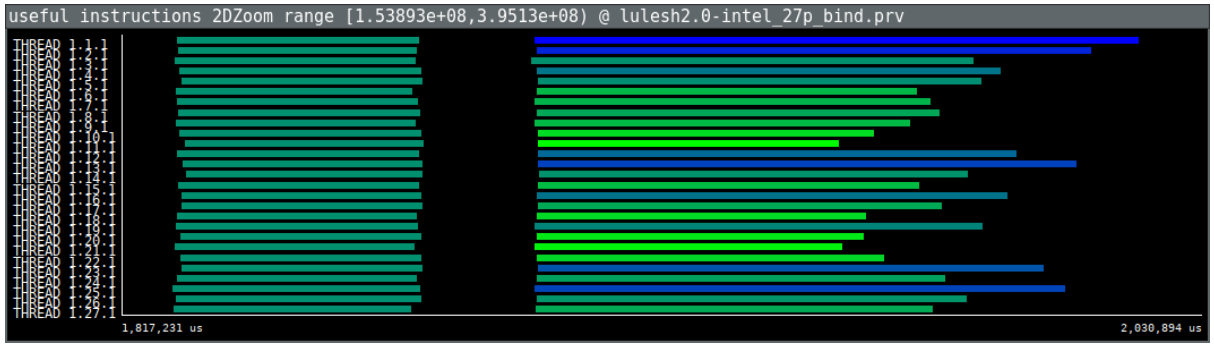


- **Hints → Call stack references → Caller function**

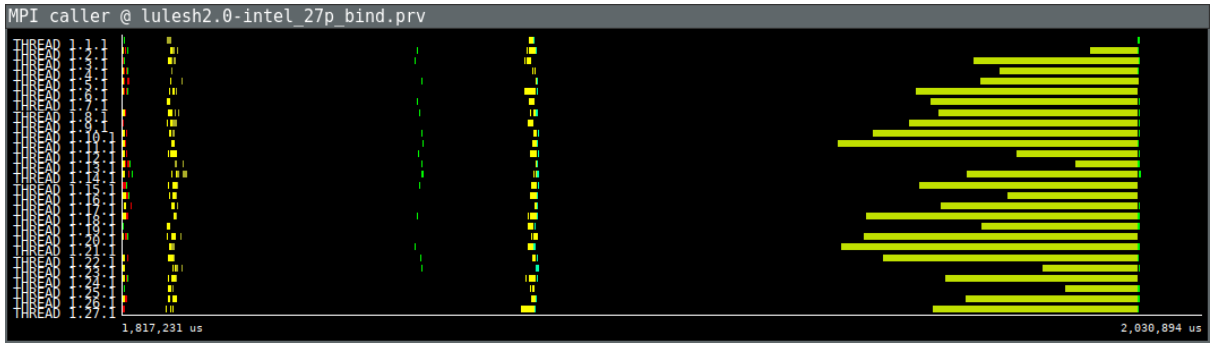


Where does this happen?

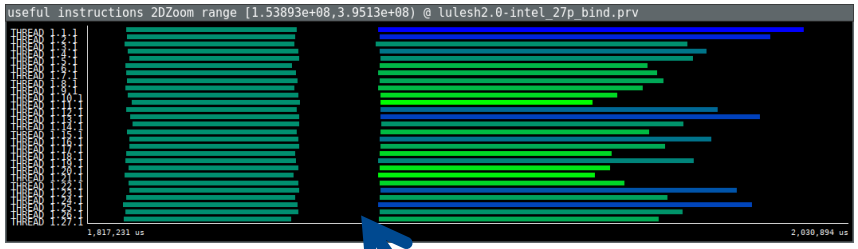
- **Slow** & **Fast** at the same time? → Imbalance



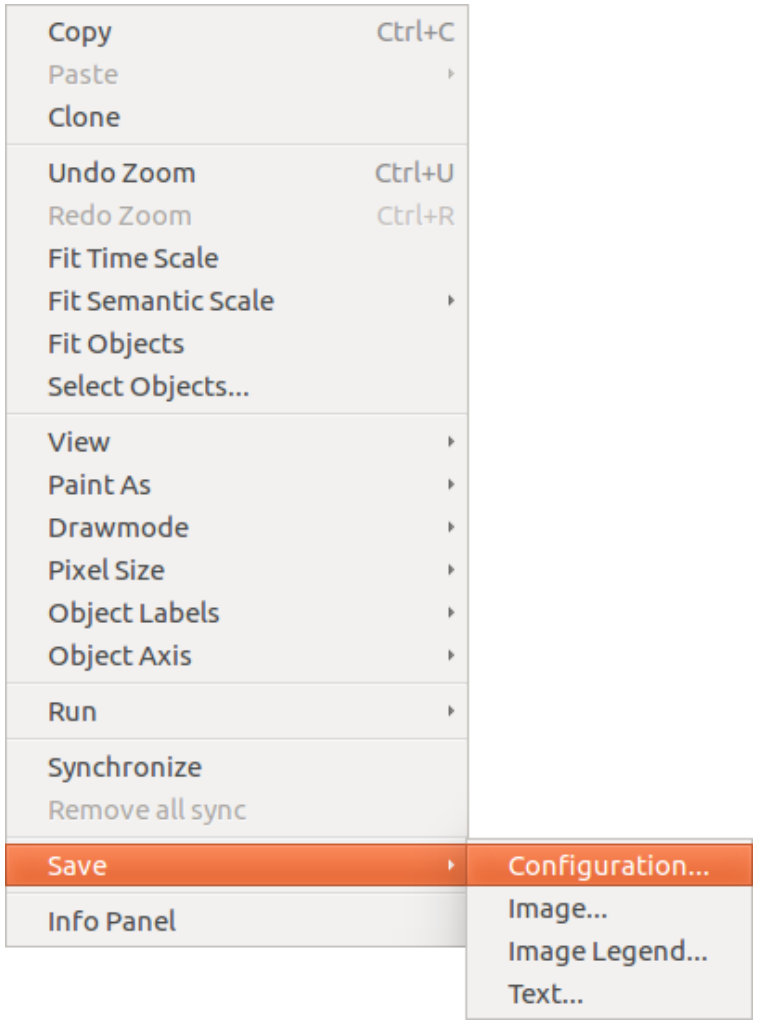
- **Hints → Call stack references → Caller function**



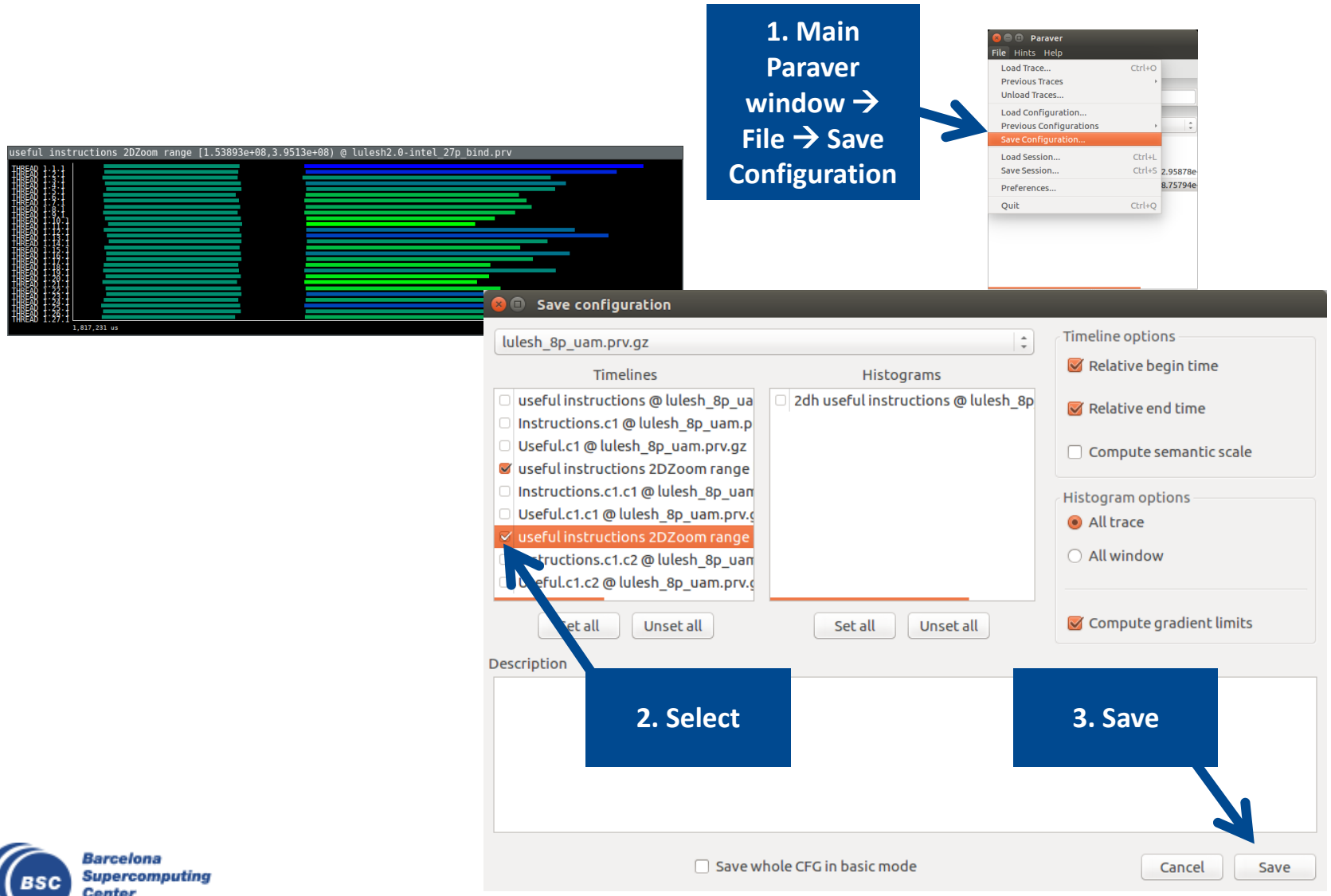
Save CFG's (method 1)



Right click on timeline

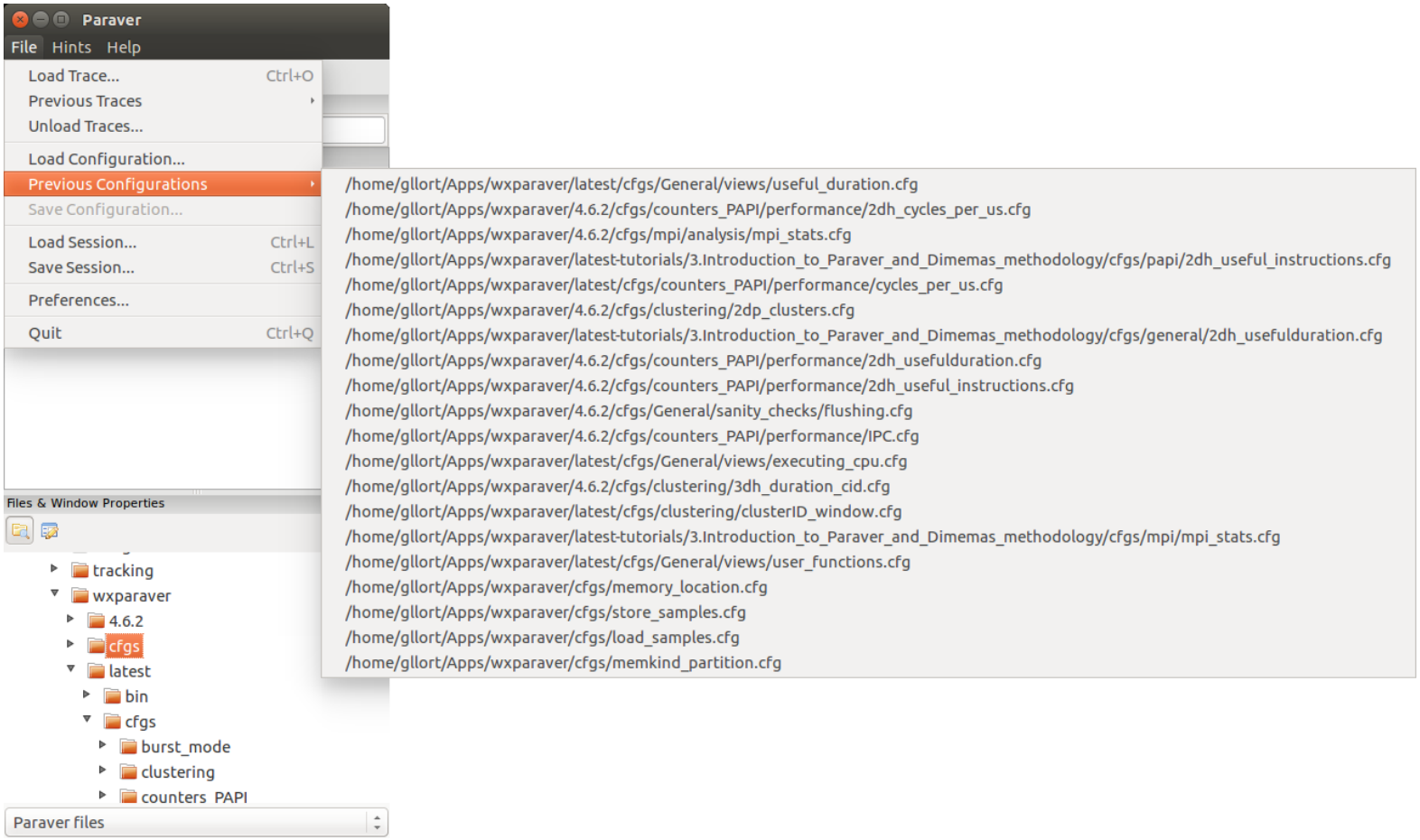


Save CFG's (method 2)



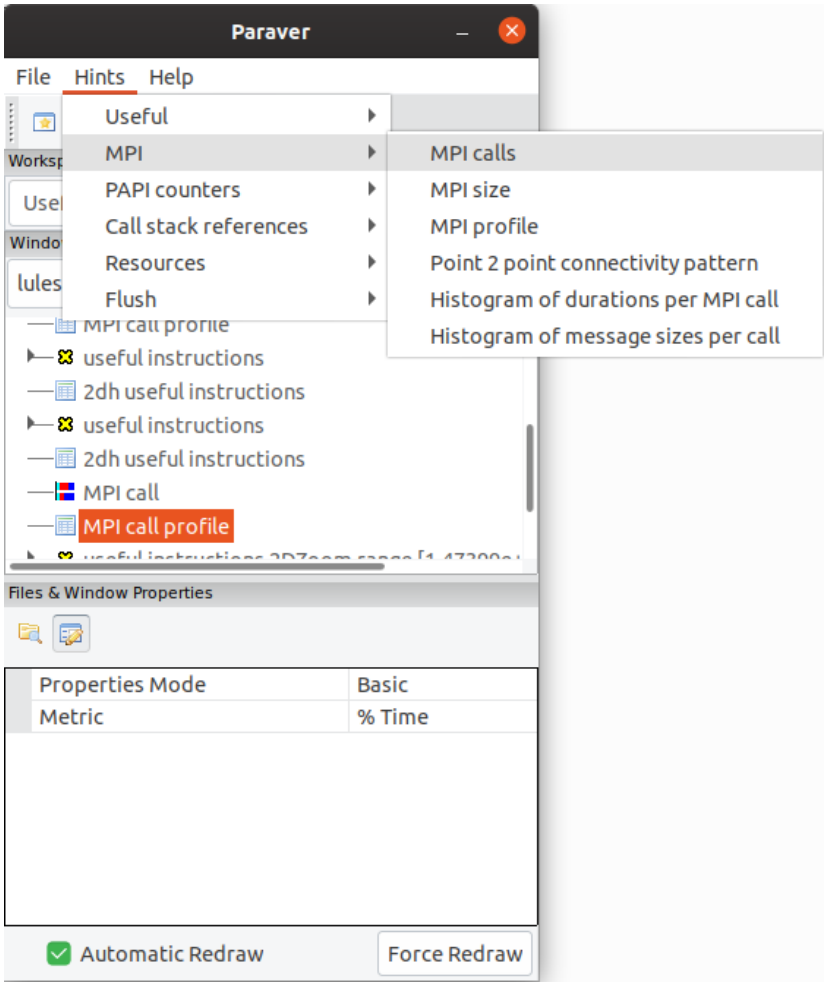
CFG's distribution

- Paraver comes with many more included CFG's



Hints: a good place to start!

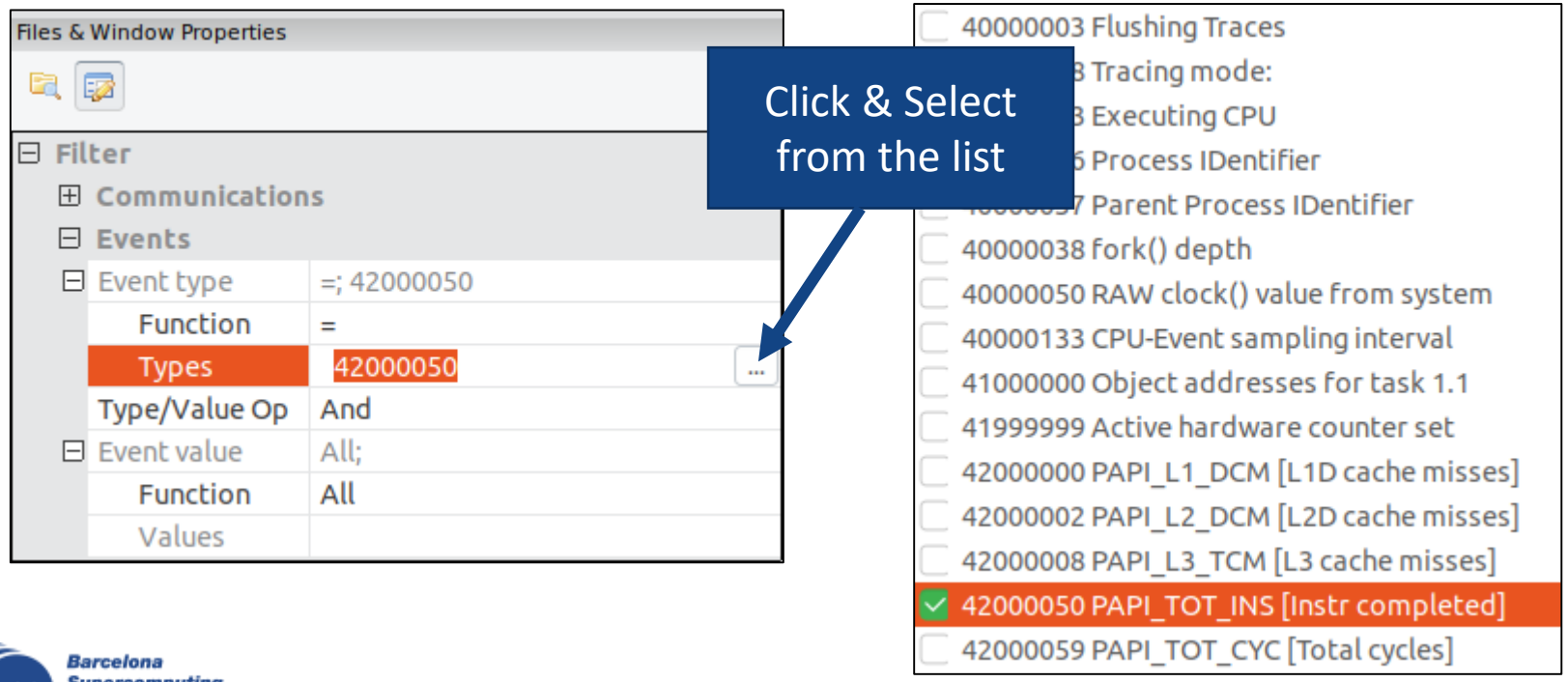
- Paraver suggests CFG's based on the contents of the trace



What is hiding behind CFG's…

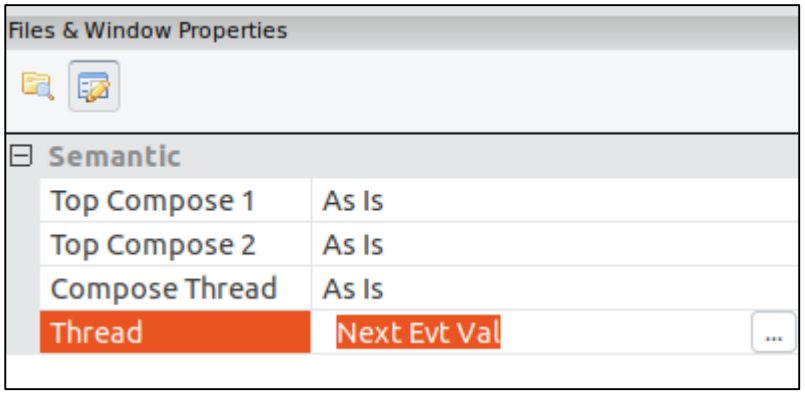
- Paraver is like an equation editor, and CFG's like a formula

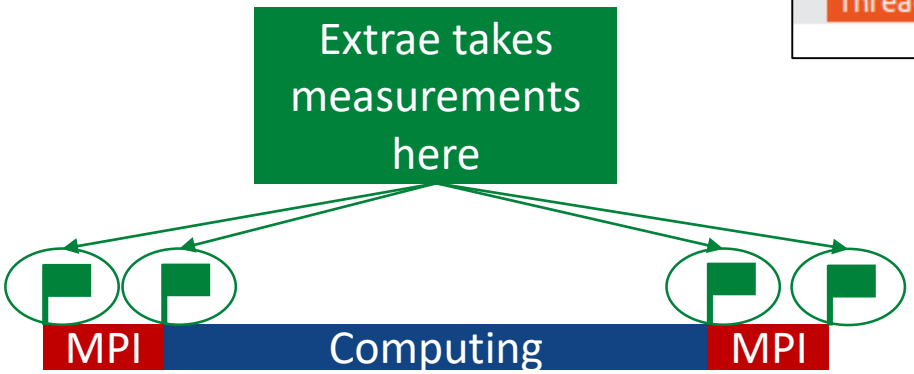
- Filter
  - The data I'm interested on seeing (Example: Instructions)



What is hiding behind CFG's…

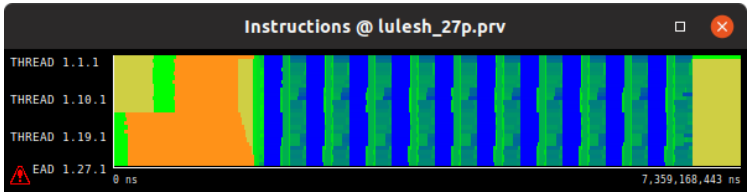
- Semantic
  - How to paint the filtered data
  - Next Evt Val?





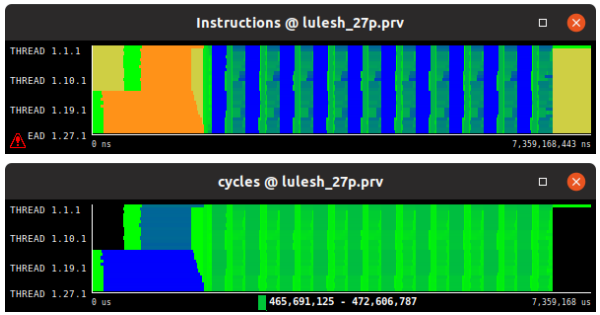
Each measurement counts how many Instructions were executed in preceding region

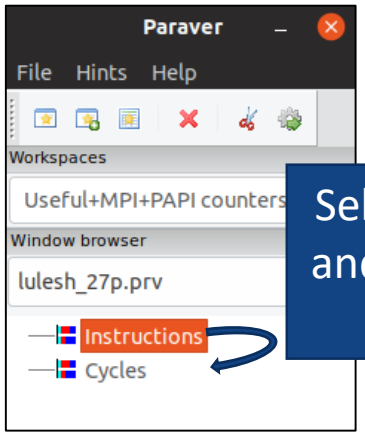
- Instructions count is found at the end of the region
- Next Evt Val = Color is determined by the next event value



What is hiding behind CFG's…

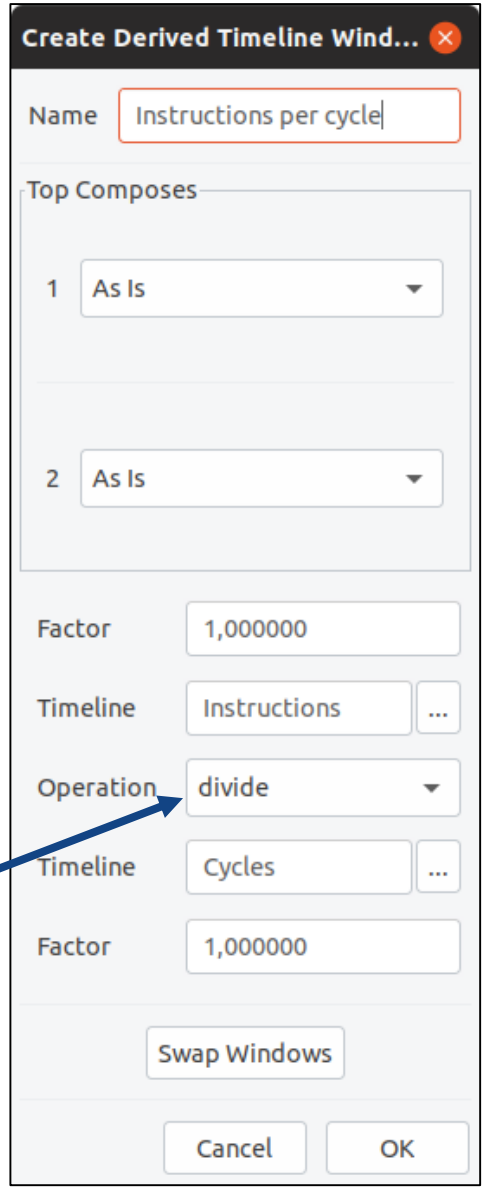
- Derived
  - Instructions
  - Cycles
  - Instructions / Cycles = IPC





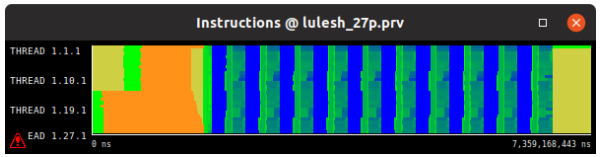
Select Instructions and drag-and-drop on Cycles

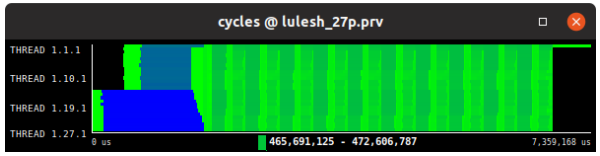
Operation: divide

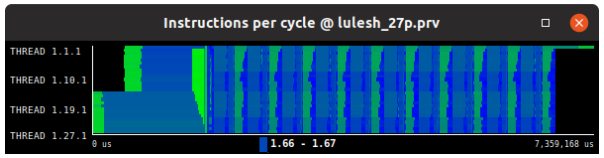


What is hiding behind CFG's…

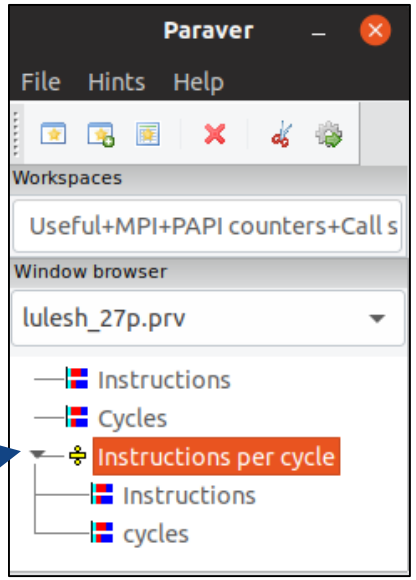
- Derived
  - Instructions
  - Cycles
  - Instructions / Cycles = IPC







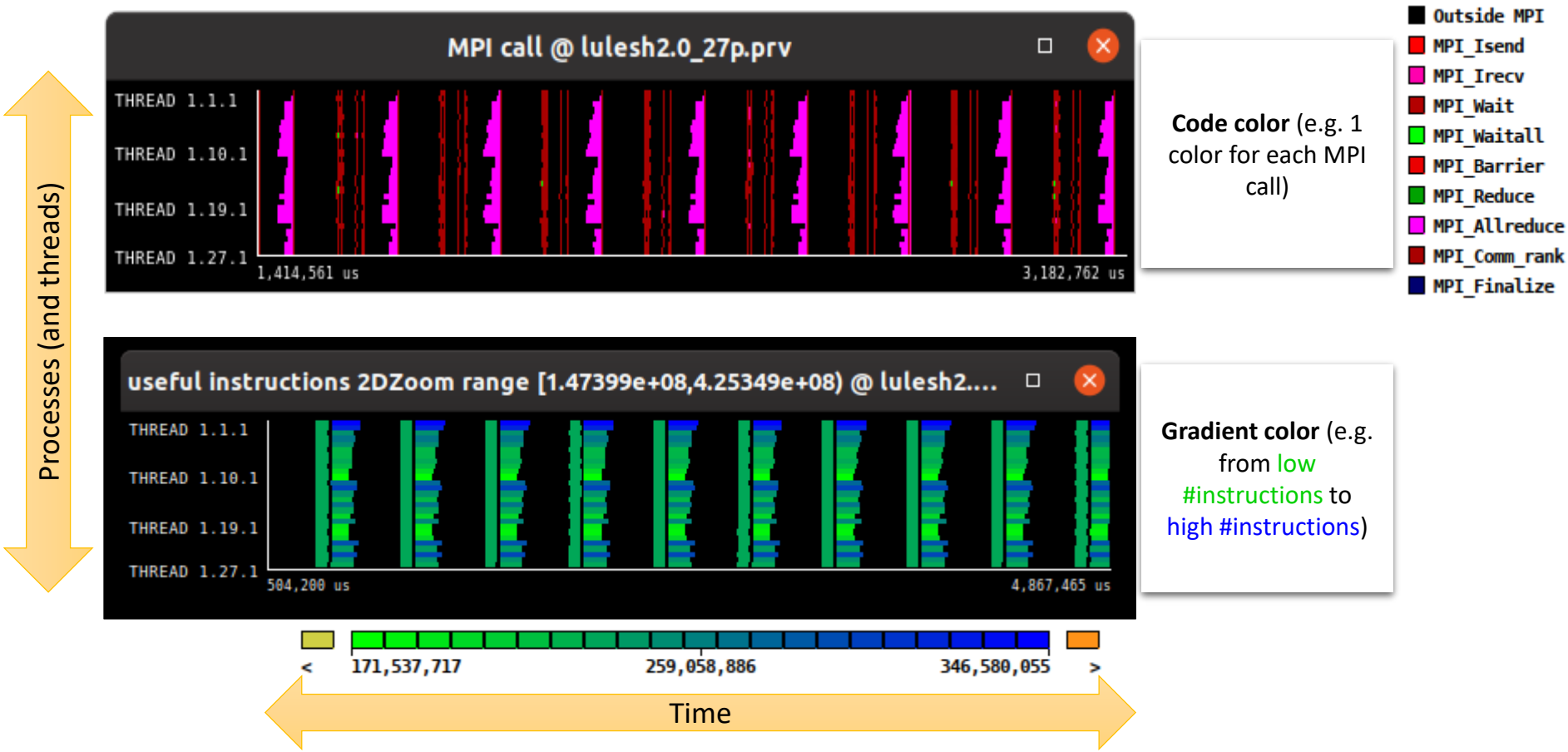
See how a CFG was constructed



- Build any formula with the data available and save a CFG!

Main views of Paraver (I)

- Timeline



Main views of Paraver (II)

- Table (Profile)

Categories (e.g. MPI calls)

Processes (and threads)

MPI call profile @ lulesh2.0_27p.prv

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Barrier	MPI_Reduce	MPI
THREAD 1.1.1	99.04 %	0.05 %	0.06 %	0.35 %	0.30 %	0.03 %	0.00 %	
THREAD 1.2.1	97.37 %	0.07 %	0.08 %	0.20 %	0.82 %	0.03 %	0.00 %	
THREAD 1.3.1	93.79 %	0.05 %	0.05 %	0.22 %	0.52 %	0.03 %	0.02 %	
THREAD 1.4.1	93.93 %	0.07 %	0.08 %	0.17 %	0.61 %	0.03 %	0.00 %	
THREAD 1.5.1	93.75 %	0.11 %	0.11 %	0.38 %	0.19 %	0.01 %	0.00 %	
THREAD 1.6.1	91.64 %	0.08 %	0.08 %	0.10 %	0.74 %	0.02 %	0.00 %	
THREAD 1.7.1	91.24 %	0.06 %	0.05 %	0.16 %	0.42 %	0.03 %	0.11 %	
THREAD 1.8.1	91.93 %	0.08 %	0.08 %	0.16 %	0.76 %	0.03 %	0.00 %	
THREAD 1.9.1	91.20 %	0.06 %	0.05 %	0.14 %	0.59 %	0.02 %	0.50 %	
THREAD 1.10.1	90.47 %	0.08 %	0.07 %	0.33 %	0.37 %	0.03 %	0.00 %	
THREAD 1.11.1	89.19 %	0.12 %	0.11 %	0.35 %	0.30 %	0.01 %	0.70 %	
THREAD 1.12.1	95.80 %	0.09 %	0.07 %	0.19 %	0.83 %	0.03 %	0.00 %	
THREAD 1.13.1	96.04 %	0.12 %	0.10 %	0.41 %	0.33 %	0.01 %	0.00 %	
THREAD 1.14.1	94.61 %	0.18 %	0.15 %	0.15 %	0.35 %	0.00 %	0.00 %	
THREAD 1.15.1	93.28 %	0.13 %	0.10 %	0.10 %	1.21 %	0.01 %	0.00 %	
THREAD 1.16.1	91.67 %	0.09 %	0.07 %	0.26 %	2.01 %	0.03 %	0.00 %	
THREAD 1.17.1	93.28 %	0.13 %	0.10 %	0.11 %	1.06 %	0.01 %	0.45 %	
THREAD 1.18.1	89.56 %	0.09 %	0.07 %	0.16 %	1.72 %	0.03 %	0.00 %	
THREAD 1.19.1	94.06 %	0.06 %	0.04 %	0.13 %	0.47 %	0.03 %	0.00 %	
THREAD 1.20.1	89.39 %	0.10 %	0.06 %	0.25 %	1.05 %	0.03 %	0.00 %	
THREAD 1.21.1	89.62 %	0.07 %	0.04 %	0.22 %	0.30 %	0.03 %	0.90 %	
THREAD 1.22.1	88.08 %	0.09 %	0.06 %	0.26 %	2.02 %	0.03 %	0.00 %	
THREAD 1.23.1	98.19 %	0.14 %	0.10 %	0.16 %	0.62 %	0.01 %	0.00 %	
THREAD 1.24.1	94.10 %	0.10 %	0.06 %	0.12 %	1.24 %	0.02 %	0.00 %	
THREAD 1.25.1	96.05 %	0.07 %	0.04 %	0.29 %	0.26 %	0.02 %	0.00 %	
THREAD 1.26.1	93.10 %	0.10 %	0.06 %	0.13 %	1.13 %	0.03 %	0.00 %	
THREAD 1.27.1	94.24 %	0.08 %	0.04 %	0.18 %	0.39 %	0.02 %	0.00 %	
Total	2,514.62 %	2.44 %	1.99 %	5.69 %	20.63 %	0.60 %	2.72 %	
Average	93.13 %	0.09 %	0.07 %	0.21 %	0.76 %	0.02 %	0.10 %	
Maximum	99.04 %	0.18 %	0.15 %	0.41 %	2.02 %	0.03 %	0.90 %	
Minimum	88.08 %	0.05 %	0.04 %	0.10 %	0.19 %	0.00 %	0.00 %	
StDev	2.79 %	0.03 %	0.03 %	0.09 %	0.51 %	0.01 %	0.24 %	
Avg/Max	0.94	0.50	0.49	0.52	0.38	0.72	0.11	

Summary

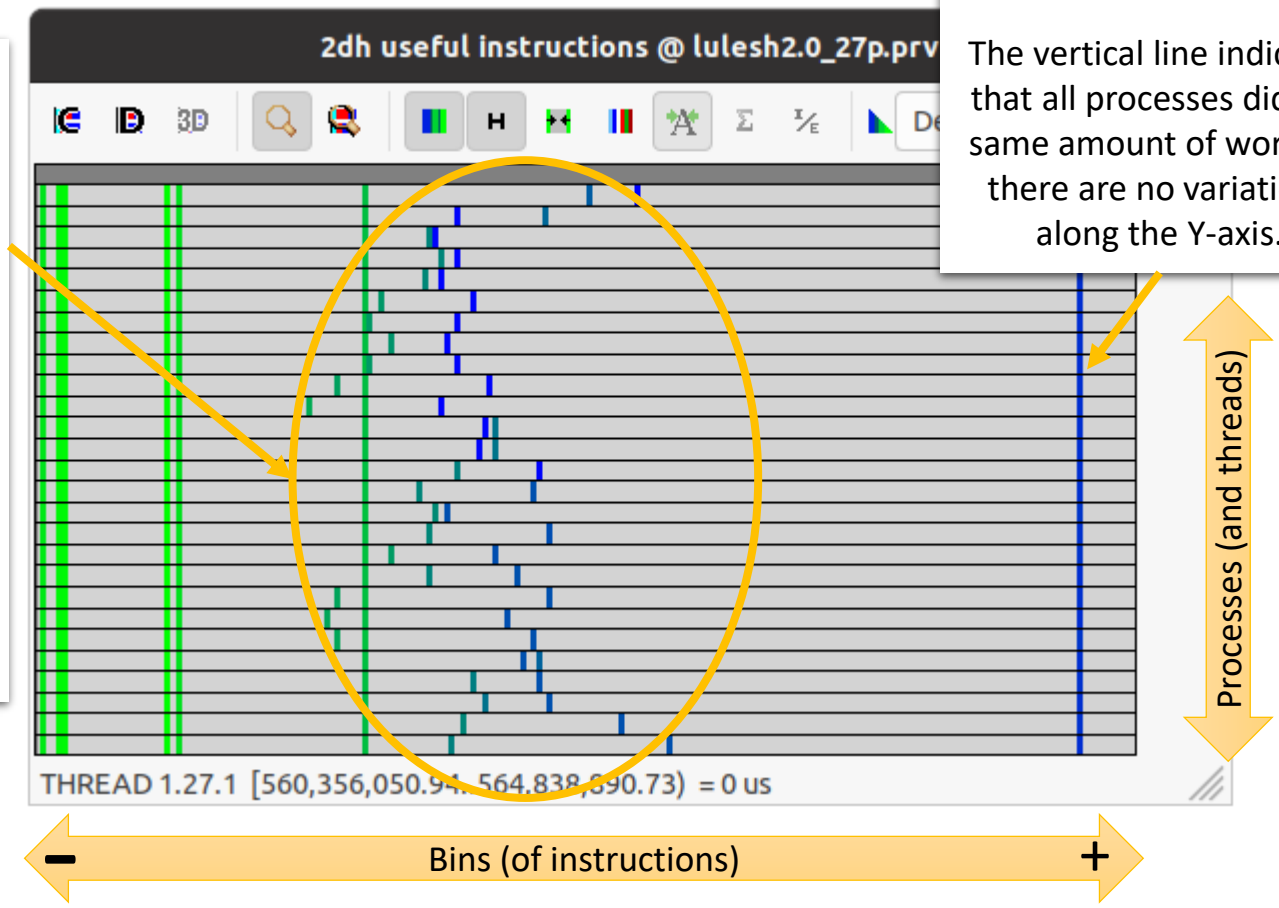
The table can display a variety of statistics (e.g. % of time, # of calls, etc.) with gradient coloring showing from low values to high values

Main views of Paraver (II

- Histograms

Having data in the middle of the histogram indicates that the program executes a medium-number of instructions.

Being the data scattered (to the left and right), indicates that different processes do different amounts of work (left→less; right→more).



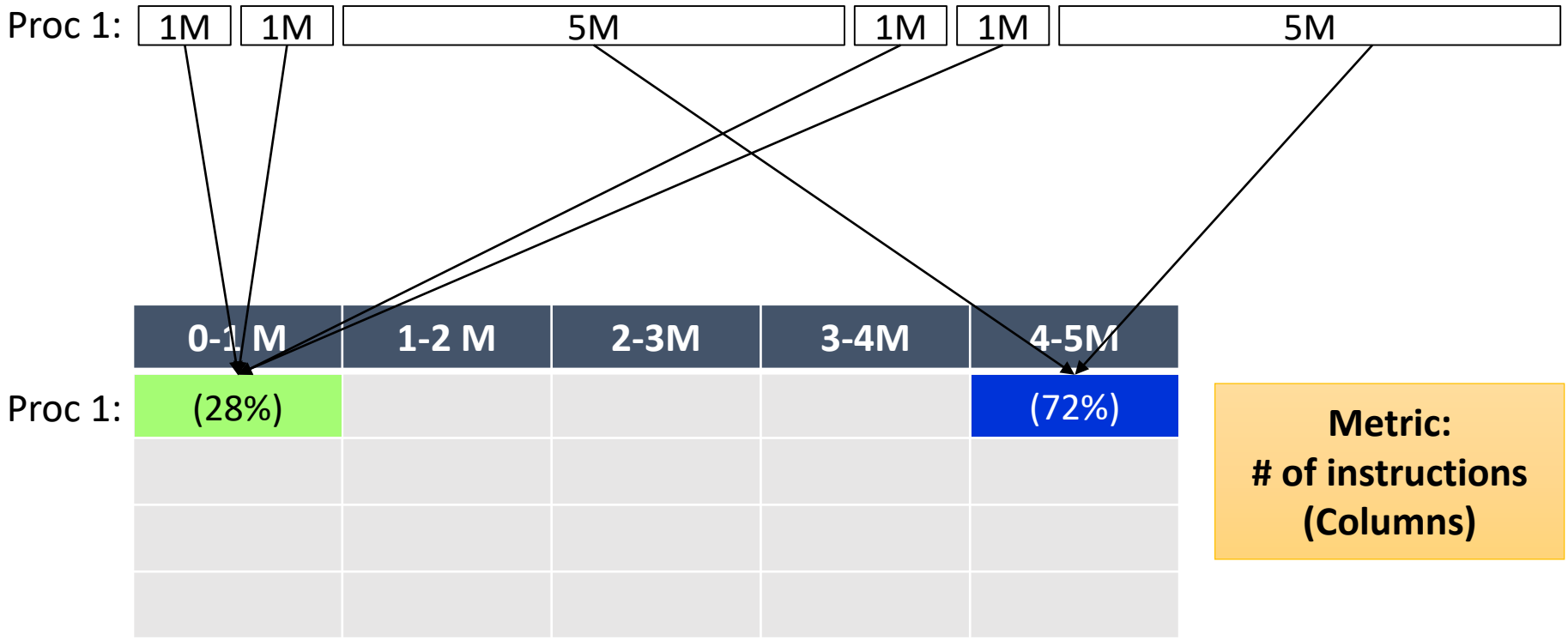
Having data in the right of the X-axis indicates that the program executed a very high number of instructions.

The vertical line indicates that all processes did the same amount of work, as there are no variations along the Y-axis.

Gradient from green to blue indicates low to high values of selected statistic (default is **Time**) :

- The more to the left → doing less instructions → greener → low % of exec time
- Data on the right → doing more instructions → bluer → high % of exec time

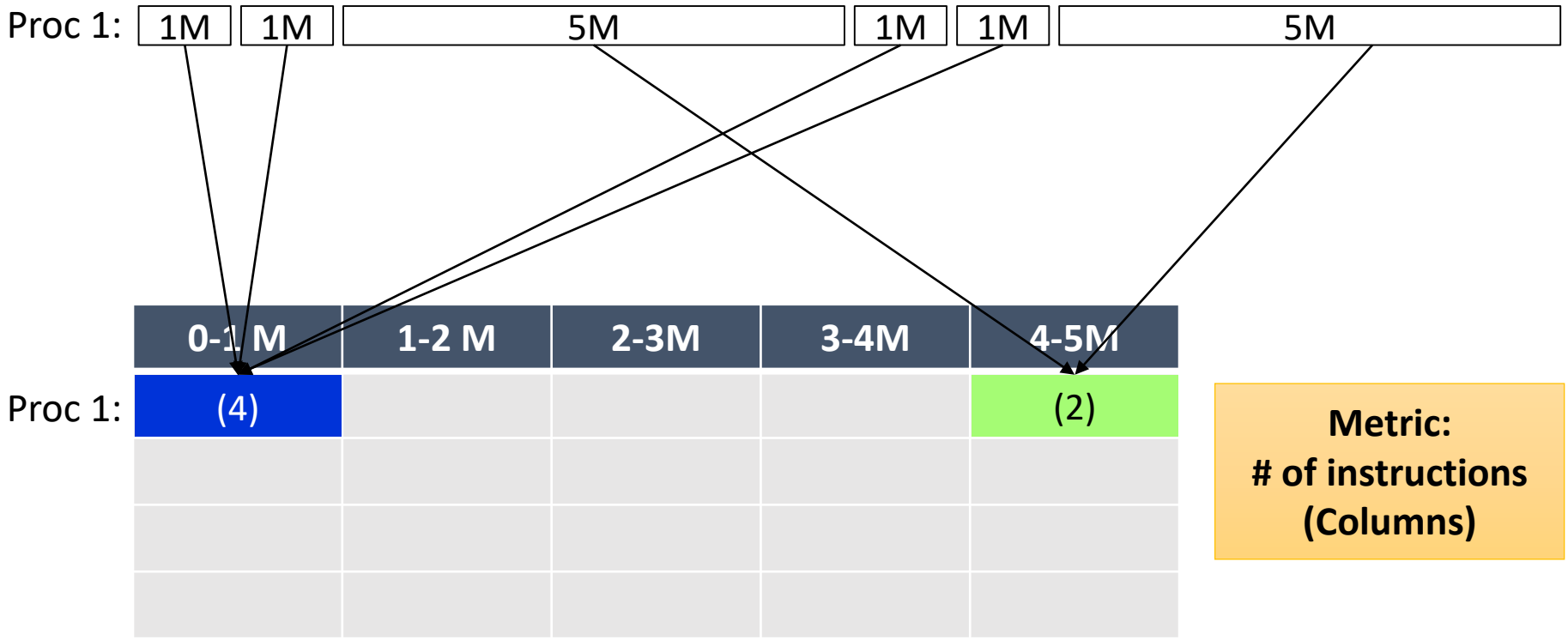
Histogram: position, metric, statistic… wait, what!?



Metric: # of instructions (Columns)

Statistic: % of time (Color)

Histogram: position, metric, statistic… wait, what!?



Metric: # of instructions (Columns)

Statistic: # of bursts (Color)

Histogram: position, metric, statistic… wait, what!?

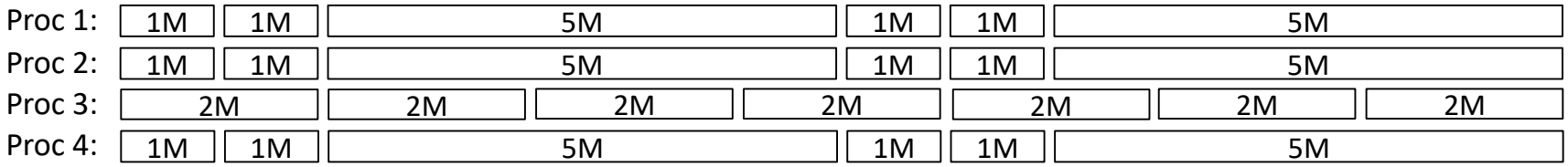
Proc 1:	1M	1M	5M	1M	1M	5M
Proc 2:	1M	1M	5M	1M	1M	5M
Proc 3:	1M	1M	5M	1M	1M	5M
Proc 4:	1M	1M	5M	1M	1M	5M

	0-1 M	1-2 M	2-3M	3-4M	4-5M
Proc 1:	(4)				(2)
Proc 2:	(4)				(2)
Proc 3:	(4)				(2)
Proc 4:	(4)				(2)

Metric: # of instructions (Columns)

Statistic: # of bursts (Color)

Histogram: position, metric, statistic… wait, what!?



	0-1 M	1-2 M	2-3M	3-4M	4-5M
Proc 1:	(4)				(2)
Proc 2:	(4)				(2)
Proc 3:			(7)		(2)
Proc 4:	(4)				(2)

Metric: # of instructions (Columns)

Statistic: # of bursts (Color)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Getting traces for different scales

Judit Giménez, Lau Mercadal, Germán Llort

tools@bsc.es

2024-09-05

PATC: Performance Analysis and Tools

Let's get more traces!

- What happens if we bind processes to cores?

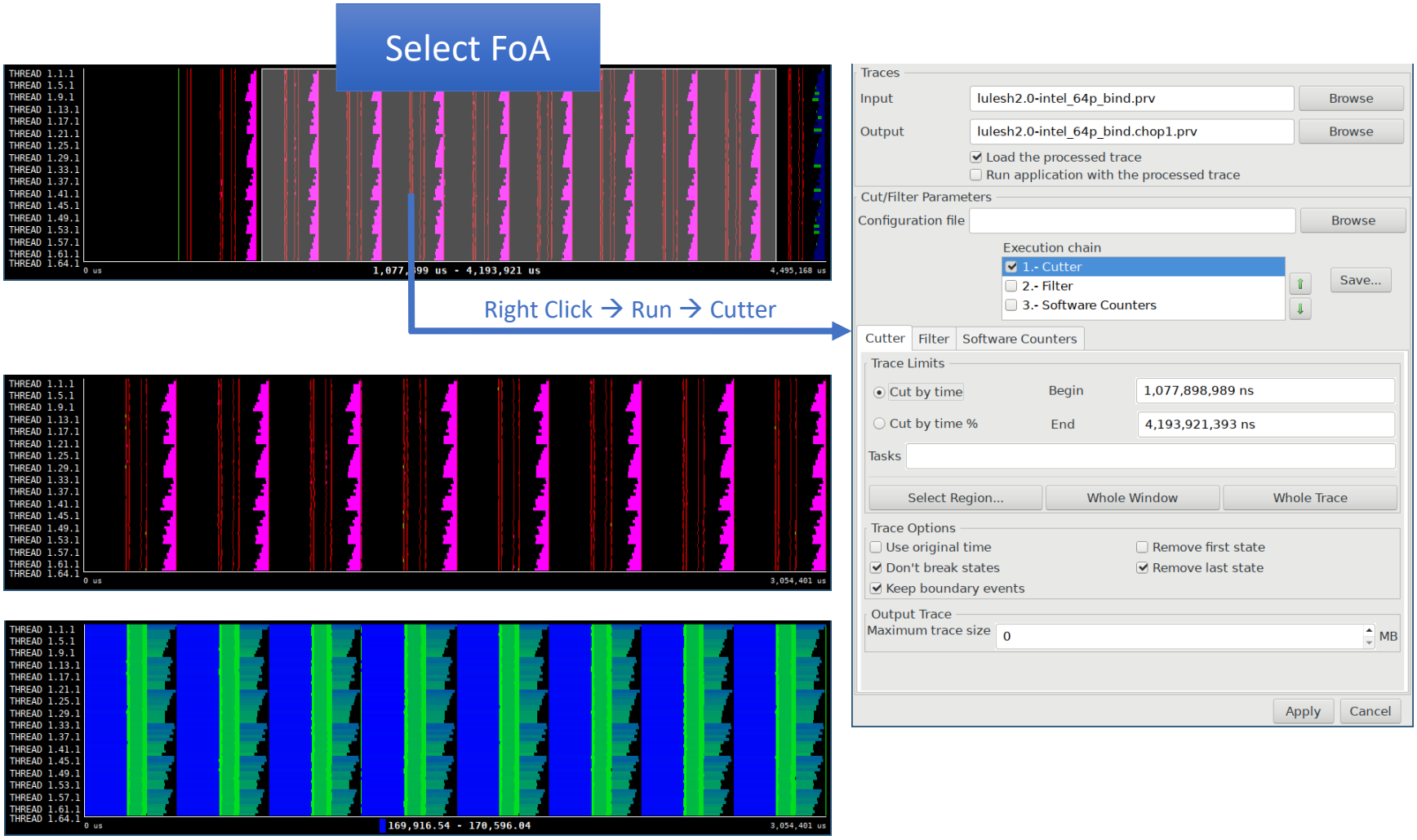
```
#!/bin/bash

#SBATCH --job-name=lulesh2.0_s65_27p_bound
#SBATCH --output=%x_%j.out
#SBATCH --error=%x_%j.err
#SBATCH --ntasks=27
#SBATCH --time=00:10:00

module load extrae

export TRACE_NAME=lulesh2.0_s65_27p_bound.prv

srun --cpu-bind=cores -m cyclic:cyclic \
     ./trace.sh lulesh2.0 -i 10 -s 65 -p
```

Let's get more traces!

- And if we run with 8 and 64 processes?

```
#!/bin/bash

#SBATCH --job-name=lulesh2.0s65_8p_bound
#SBATCH --output=%x_%j.out
#SBATCH --error=%x_%j.err
#SBATCH --ntasks=8
#SBATCH --time=00:10:00

module load extrae

export TRACE_NAME=lulesh2.0_s65_8p_bound.prv

srun --cpu_bind=cores -m cyclic:cyclic \
     ./trace.sh ./lulesh2.0 -i 10 -s 65 -p
```

And cut them to select the FoA



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Getting OpenMP traces

Judit Giménez, Lau Mercadal, Germán Llort

tools@bsc.es

2024-09-05

PATC: Performance Analysis and Tools

Let's get MPI+OpenMP traces!

- What changes do I need in my jobscript?

```
#!/bin/bash

#SBATCH --job-name=lulesh2.0_s65_8p_bound
#SBATCH --output=%x_%j.out
#SBATCH --error=%x_%j.err
#SBATCH --ntasks=8
#SBATCH --cpus-per-task=8
#SBATCH --time=00:10:00

export OMP_NUM_THREADS=8

export TRACE_NAME=lulesh2.0_s65_8p8omp_bound.prv
srun --cpu-bind=cores -m cyclic:cyclic \
     ./trace.sh lulesh2.0-omp -i 10 -s 65 -p
```

Let's get MPI+OpenMP traces!

- What changes do I need in my XML config file?

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

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

<pthread enabled="no">
    <locks enabled="no" />
    <counters enabled="yes" />
</pthread>
```

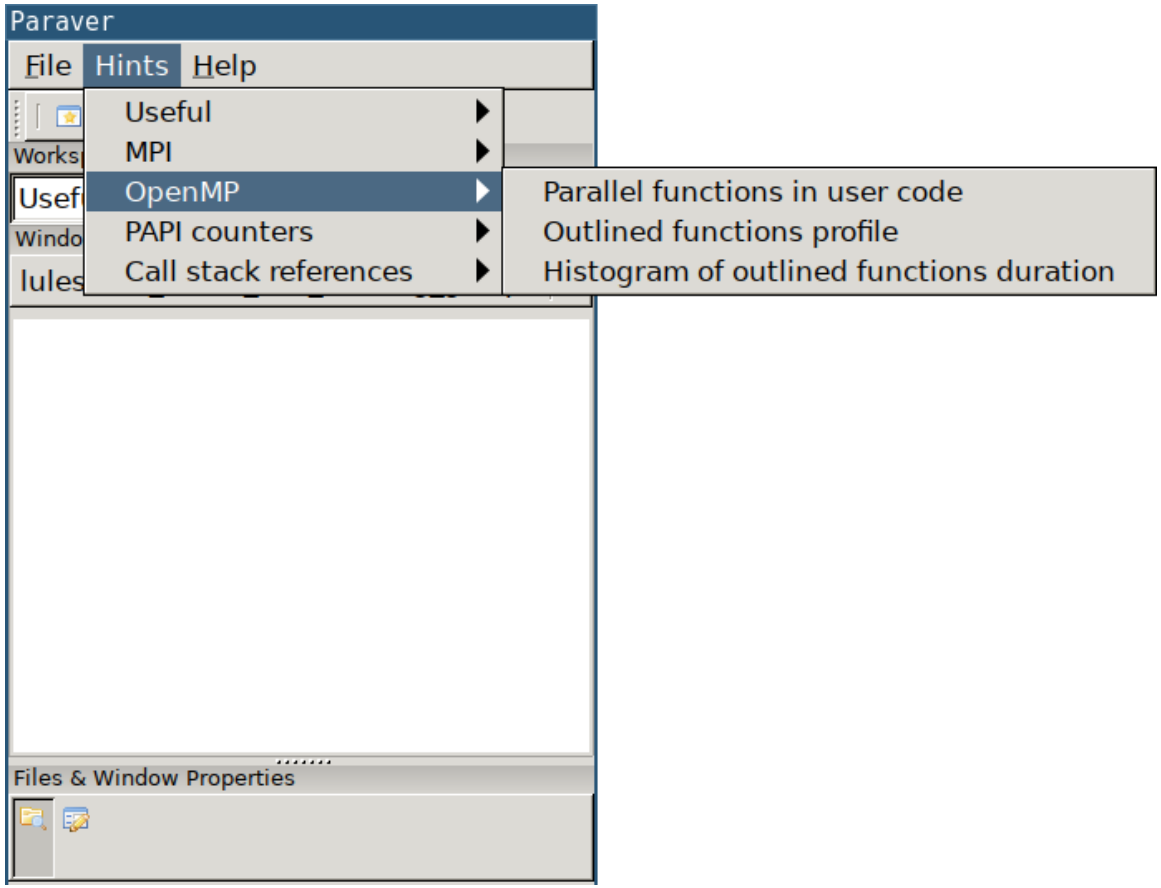
Let's get MPI+OpenMP traces!

- What changes do I need in my `trace.sh` file?

```
#!/bin/bash

# Configure Extrae
export EXTRAE_HOME=<installation_base_path>
export EXTRAE_CONFIG_FILE=./extrae.xml

# Load the tracing library (choose C/Fortran)
export LD_PRELOAD=$EXTRAE_HOME/lib/libompitrace.so
#export
LD_PRELOAD=$EXTRAE_HOME/lib/libmpitracef.so

# Run the program
$*
```

OpenMP Workspace

- Paraver detects OpenMP and suggests views through Hints



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thank you!

Judit Giménez, Lau Mercadal, Germán Llort

tools@bsc.es

2024-09-05

PATC: Performance Analysis and Tools