BSC Tools
Hands-On

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

Getting a trace with Extrae

Extrae features

- Platforms
  - Intel, Cray, BlueGene, MIC, ARM, Android, Fujitsu Sparc …
- Parallel programming models
  - 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 (Dyninst)
- Periodic sampling

- User events (Extrae API)

No need to recompile or relink!

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

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

Extrae on MEGGIE (I)

- Log into MEGGIE:

```
laptop$ ssh -Y <USER>@cshpc.rrze.fau.de
laptop$ ssh meggie
```

- Extrae is available via modules for both for Intel and GNU toolchains… choose yours!
  - Intel compiler with IntelMPI

```
meggie$ module use $HOME/VIHPS/modulefiles
meggie$ module load intel64 intelmpi
meggie$ module load extrae/3.8.3
```

  - GCC compiler with OpenMPI

```
meggie$ module use $HOME/VIHPS/modulefiles
meggie$ module load gcc openmpi/1.10.7-gcc
meggie$ module load extrae/3.8.3
```

Getting your first trace

- Provided folder **`tools-material`** in `$HOME/VIHPS/bsctools` contains:
  - Application compiled for the Intel and GNU toolchains (`lulesh2.0-intel, lulesh2.0-gnu`)
  - Jobscripts to execute and trace (`job-intel.sh, job-gnu.sh, trace.sh`)
  - Configuration of the tracing tool (`extrae.xml`)
  - Already generated tracefiles (`traces/*.{pcf,prv,row}`)
  - Clustering analysis configuration file (`cluster.xml`)

- Copy this folder to your `$HOME` and you are ready to follow this hands-on tutorial

```
meggie$ cp -R $HOME/VIHPS/bsctools/tools-material $HOME
```

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

- Example of a standard jobscript (without tracing)

```
!/bin/bash -l

#SBATCH --job-name=lulesh2.0-intel_27p
#SBATCH --nodes=2
#SBATCH --ntasks=27
#SBATCH --time=00:10:00
#SBATCH --output=%x.out
#SBATCH --error=%x.err
#SBATCH --partition=devel
#SBATCH --export=NONE

unset SLURM_EXPORT_ENV
module load intel64 intelmpi

# Run on 27 processes
mpirun -n 27 ../apps/lulesh2.0-intel -i 10 -s 65 -p
```

Request resources

Run the program

Step 1: Adapt the job script to load Extrae

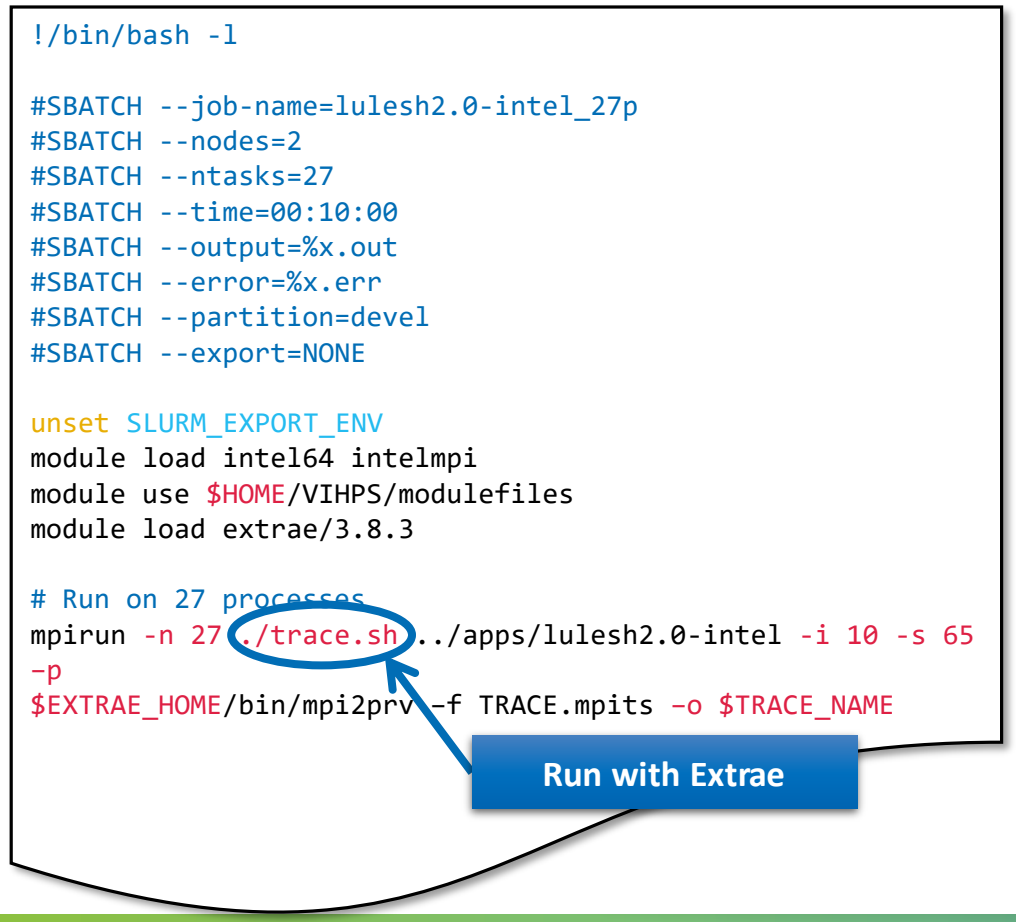
- Jobscript modified to load Extrae (`extrae/job-intel.sh`)

```
!/bin/bash -l

#SBATCH --job-name=lulesh2.0-intel_27p
#SBATCH --nodes=2
#SBATCH --ntasks=27
#SBATCH --time=00:10:00
#SBATCH --output=%x.out
#SBATCH --error=%x.err
#SBATCH --partition=devel
#SBATCH --export=NONE

unset SLURM_EXPORT_ENV
module load intel64 intelmpi
module use $HOME/VIHPS/modulefiles
module load extrae/3.8.3

# Run on 27 processes
mpirun -n 27 ./trace.sh ../apps/lulesh2.0-intel -i 10 -s 65
-p
$EXTRAE_HOME/bin/mpi2prv -f TRACE.mpits -o $TRACE_NAME
```



Step 1: Adapt the job script to load Extrae

- Tracing launcher helper script (`extrae/trace.sh`)

```
!/bin/bash -l

#SBATCH --job-name=lulesh2.0-intel_27p
#SBATCH --nodes=2
#SBATCH --ntasks=27
#SBATCH --time=00:10:00
#SBATCH --output=%x.out
#SBATCH --error=%x.err
#SBATCH --partition=devel
#SBATCH --export=NONE

unset SLURM_EXPORT_ENV
module load intel64 intelmpi
module use $HOME/VIHPS/modulefiles
module load extrae/3.8.3

# Run on 27 processes
mpirun -n 27 ./trace.sh ../apps/lulesh2.0-intel -i 10 -s 65
-p
$EXTRAE_HOME/bin/mpi2prv -f TRACE.mpits -o $TRACE_NAME
```

```
#!/bin/bash

export EXTRAE_SKIP_AUTO_LIBRARY_INITIALIZE=1
export EXTRAE_CONFIG_FILE=./extrae.xml
# For C apps
export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitrace.so
# For Fortran apps
#export LD_PRELOAD=${EXTRAE_HOME}/lib/libmpitracef.so

## Run the desired program
$*
```

What to trace?

Choose a tracing library depending on the app type (see next slide)

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] add suffix "f" in Fortran codes

Step 2: Extrae XML configuration

```
<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>
```

Instrument the MPI calls
(What's the program doing?)

Instrument the call-stack
(Where in my code?)

Step 2: Extrae XML configuration (II)

```
<counters enabled="yes">
  <cpu enabled="yes" starting-set-distribution="1">
    <set enabled="yes" domain="all" changeat-time="0">
      PAPI_TOT_INS,PAPI_TOT_CYC
    </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?)

```
meggie$ papi_best_set \
omnipresent <list-of-counters> \
<list-of-counters>
```

```
<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>
```

Extrae buffer size
(Flush/memory trade-off)

Additional sampling
(Want more details?)

Automatic post-processing to generate the Paraver trace

Step 3: Run it!

- Submit your job as usual

```
meggie$ sbatch --reservation=VIHPS3 job-intel.sh
```

All done! Check your resulting trace

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

```
meggie$ ls -l
  ...
  lulesh2.0-intel_27p.pcf
  lulesh2.0-intel_27p.prv
  lulesh2.0-intel_27p.row
```

- Any trouble? There's a trace already generated under the "`traces`" folder

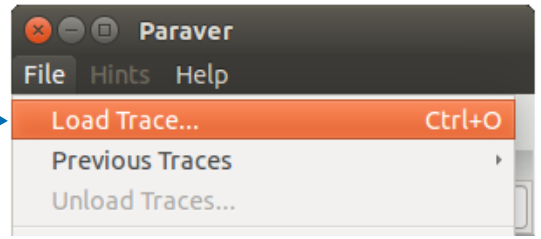
- Now let's look into it!

Analysing a trace with Extrae

First steps of analysis

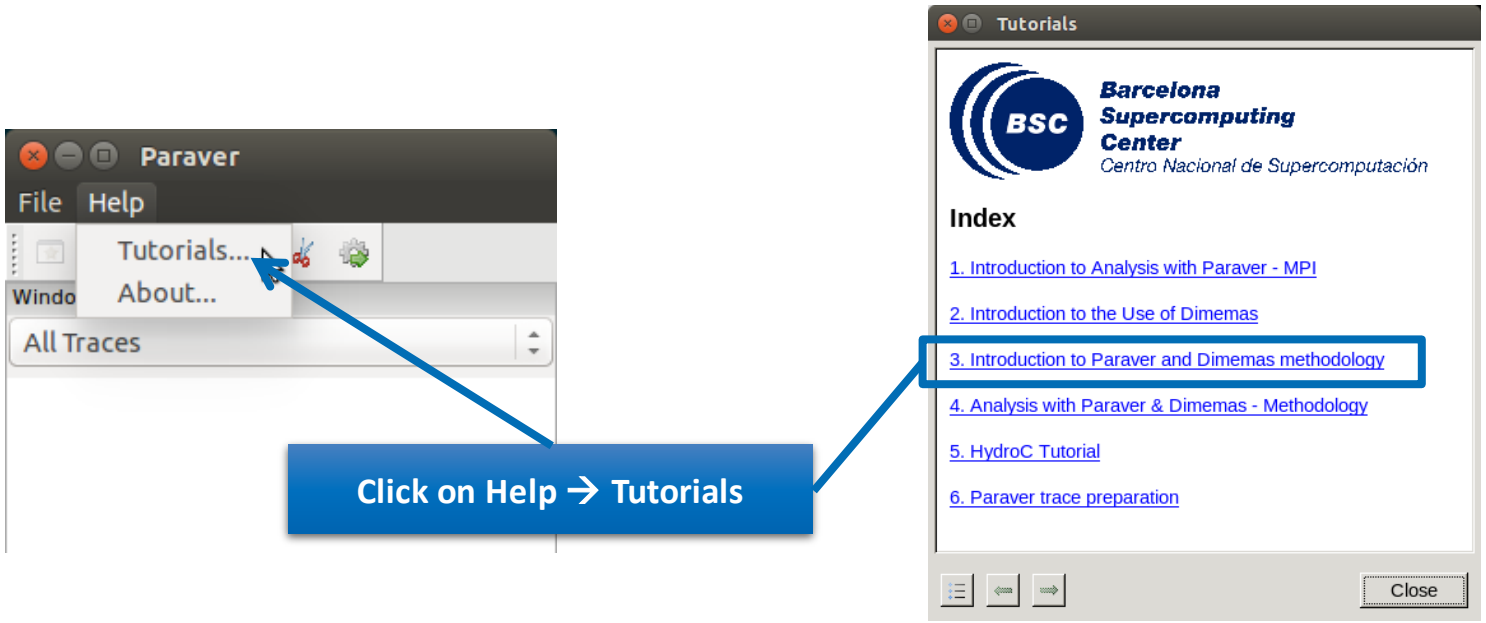
- Copy the trace to your computer
- Load the trace with Paraver

Click on File → Load Trace
→ Browse to "lulesh2.0-intel_27p.prv"



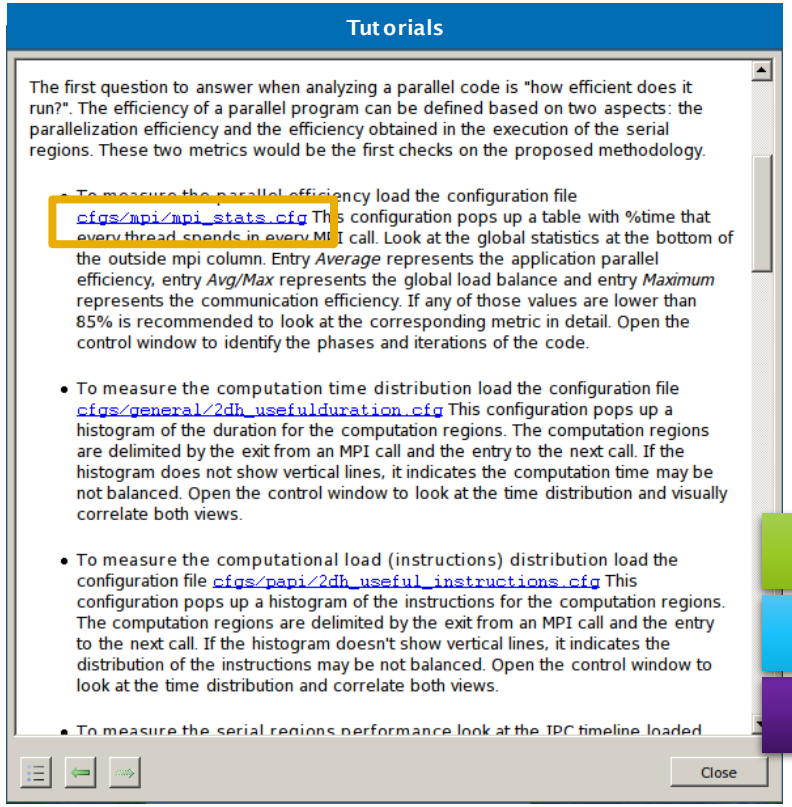
First steps of analysis

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



Measure the parallel efficiency

- Click on "mpi_stats.cfg"
  - Check the **Average** for the column labelled "**Outside MPI**"



Parallel efficiency (Avg)

Comm efficiency (Max)

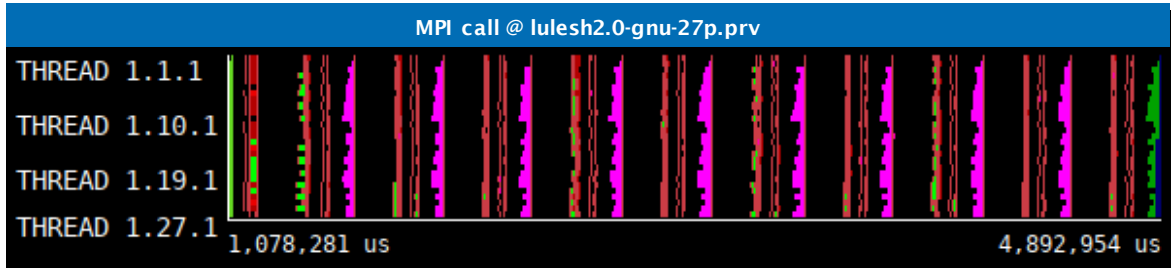
Load balance (Avg/Max)

MPI call profile @ lulesh2.0-gnu-27p.prv

Default

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Barrier	MPI_Reduce
THREAD 1.15.1	86.19 %	0.33 %	0.27 %	0.29 %	2.50 %	0.01 %	0.68 %
THREAD 1.16.1	89.34 %	0.23 %	0.17 %	0.48 %	2.79 %	0.77 %	0.11 %
THREAD 1.17.1	88.21 %	0.32 %	0.26 %	0.97 %	1.76 %	0.05 %	0.45 %
THREAD 1.18.1	83.07 %	0.23 %	0.17 %	0.31 %	4.35 %	0.03 %	0.82 %
THREAD 1.19.1	88.96 %	0.15 %	0.12 %	0.13 %	3.32 %	0.02 %	0.23 %
THREAD 1.20.1	83.68 %	0.22 %	0.17 %	0.19 %	3.17 %	0.01 %	0.70 %
THREAD 1.21.1	82.59 %	0.16 %	0.11 %	0.31 %	3.86 %	0.02 %	0.62 %
THREAD 1.22.1	79.46 %	0.22 %	0.16 %	1.49 %	5.78 %	0.04 %	0.82 %
THREAD 1.23.1	87.28 %	0.32 %	0.22 %	0.61 %	5.92 %	0.01 %	0.00 %
THREAD 1.24.1	82.04 %	0.22 %	0.15 %	1.36 %	5.92 %	0.03 %	0.60 %
THREAD 1.25.1	87.67 %	0.15 %	0.09 %	0.77 %	5.53 %	0.78 %	0.06 %
THREAD 1.26.1	82.51 %	0.23 %	0.14 %	1.36 %	6.27 %	0.11 %	0.48 %
THREAD 1.27.1	83.15 %	0.16 %	0.09 %	0.60 %	4.95 %	0.09 %	0.69 %
Total	2,334.32 %	5.82 %	5.11 %	28.70 %	91.30 %	3.26 %	17.66 %
Average	86.46 %	0.22 %	0.19 %	1.06 %	3.38 %	0.12 %	0.65 %
Maximum	95.90 %	0.45 %	0.41 %	2.57 %	6.27 %	0.78 %	1.36 %
Minimum	79.46 %	0.10 %	0.09 %	0.13 %	1.02 %	0.00 %	0.00 %
StDev	3.81 %	0.08 %	0.07 %	0.60 %	1.48 %	0.23 %	0.39 %
Avg/Max	0.90	0.48	0.46	0.41	0.54	0.15	0.48

Focus on the iterative part



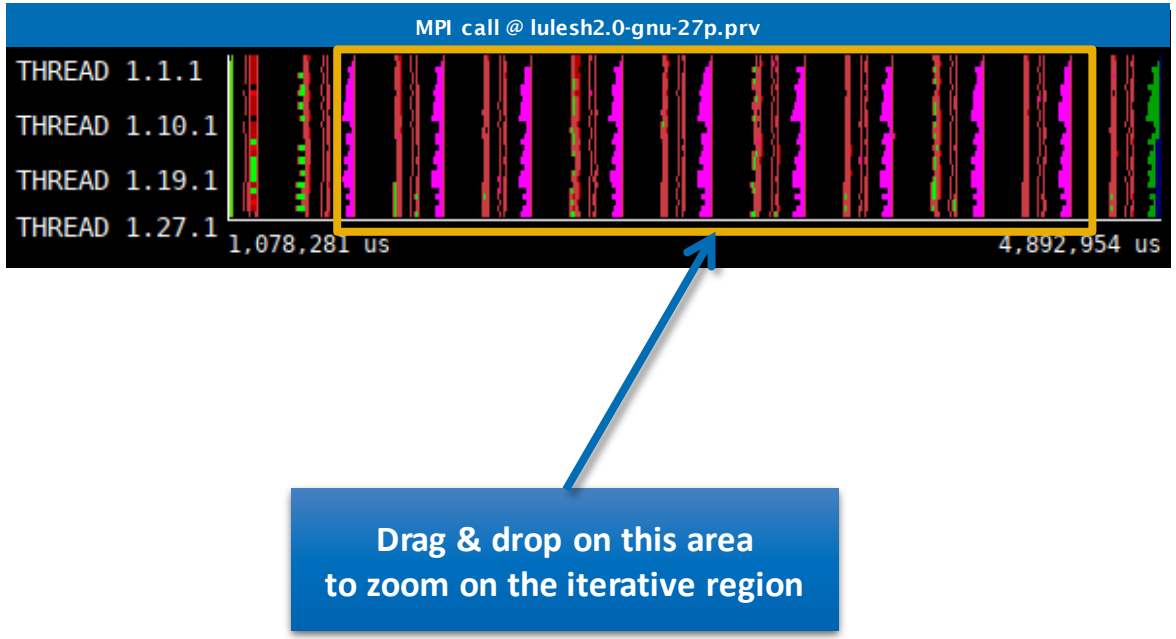
Click on
Open Control Window

MPI call profile @ lulesh2.0-gnu-27p.prv

Default

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Barrier	MPI_Reduce
THREAD 1.15.1	86.19 %	0.33 %	0.27 %	0.29 %	2.50 %	0.01 %	0.68 %
THREAD 1.16.1	89.34 %	0.23 %	0.17 %	0.48 %	2.79 %	0.77 %	0.11 %
THREAD 1.17.1	88.21 %	0.32 %	0.26 %	0.97 %	1.76 %	0.05 %	0.45 %
THREAD 1.18.1	83.07 %	0.23 %	0.17 %	0.31 %	4.35 %	0.03 %	0.82 %
THREAD 1.19.1	88.96 %	0.15 %	0.12 %	0.13 %	3.32 %	0.02 %	0.23 %
THREAD 1.20.1	83.68 %	0.22 %	0.17 %	0.19 %	3.17 %	0.01 %	0.70 %
THREAD 1.21.1	82.59 %	0.16 %	0.11 %	0.31 %	3.86 %	0.02 %	0.62 %
THREAD 1.22.1	79.46 %	0.22 %	0.16 %	1.49 %	5.78 %	0.04 %	0.82 %
THREAD 1.23.1	87.28 %	0.32 %	0.22 %	0.61 %	5.92 %	0.01 %	0.00 %
THREAD 1.24.1	82.04 %	0.22 %	0.15 %	1.36 %	5.92 %	0.03 %	0.60 %
THREAD 1.25.1	87.67 %	0.15 %	0.09 %	0.77 %	5.53 %	0.78 %	0.06 %
THREAD 1.26.1	82.51 %	0.23 %	0.14 %	1.36 %	6.27 %	0.11 %	0.48 %
THREAD 1.27.1	83.15 %	0.16 %	0.09 %	0.60 %	4.95 %	0.09 %	0.69 %
Total	2,334.32 %	5.82 %	5.11 %	28.70 %	91.30 %	3.26 %	17.66 %
Average	86.46 %	0.22 %	0.19 %	1.06 %	3.38 %	0.12 %	0.65 %
Maximum	95.90 %	0.45 %	0.41 %	2.57 %	6.27 %	0.78 %	1.36 %
Minimum	79.46 %	0.10 %	0.09 %	0.13 %	1.02 %	0.00 %	0.00 %
StDev	3.81 %	0.08 %	0.07 %	0.60 %	1.48 %	0.23 %	0.39 %
Avg/Max	0.90	0.48	0.46	0.41	0.54	0.15	0.48

Focus on the iterative part

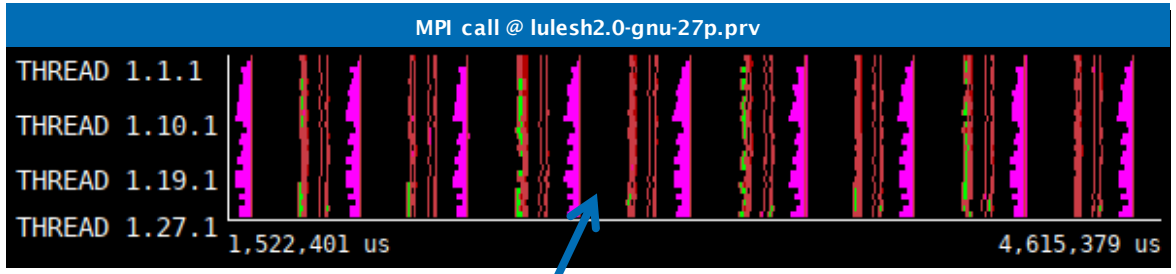


MPI call profile @ lulesh2.0-gnu-27p.prv

Default

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Barrier	MPI_Reduce
THREAD 1.15.1	86.19 %	0.33 %	0.27 %	0.29 %	2.50 %	0.01 %	0.68 %
THREAD 1.16.1	89.34 %	0.23 %	0.17 %	0.48 %	2.79 %	0.77 %	0.11 %
THREAD 1.17.1	88.21 %	0.32 %	0.26 %	0.97 %	1.76 %	0.05 %	0.45 %
THREAD 1.18.1	83.07 %	0.23 %	0.17 %	0.31 %	4.35 %	0.03 %	0.82 %
THREAD 1.19.1	88.96 %	0.15 %	0.12 %	0.13 %	3.32 %	0.02 %	0.23 %
THREAD 1.20.1	83.68 %	0.22 %	0.17 %	0.19 %	3.17 %	0.01 %	0.70 %
THREAD 1.21.1	82.59 %	0.16 %	0.11 %	0.31 %	3.86 %	0.02 %	0.62 %
THREAD 1.22.1	79.46 %	0.22 %	0.16 %	1.49 %	5.78 %	0.04 %	0.82 %
THREAD 1.23.1	87.28 %	0.32 %	0.22 %	0.61 %	5.92 %	0.01 %	0.00 %
THREAD 1.24.1	82.04 %	0.22 %	0.15 %	1.36 %	5.92 %	0.03 %	0.60 %
THREAD 1.25.1	87.67 %	0.15 %	0.09 %	0.77 %	5.53 %	0.78 %	0.06 %
THREAD 1.26.1	82.51 %	0.23 %	0.14 %	1.36 %	6.27 %	0.11 %	0.48 %
THREAD 1.27.1	83.15 %	0.16 %	0.09 %	0.60 %	4.95 %	0.09 %	0.69 %
Total	2,334.32 %	5.82 %	5.11 %	28.70 %	91.30 %	3.26 %	17.66 %
Average	86.46 %	0.22 %	0.19 %	1.06 %	3.38 %	0.12 %	0.65 %
Maximum	95.90 %	0.45 %	0.41 %	2.57 %	6.27 %	0.78 %	1.36 %
Minimum	79.46 %	0.10 %	0.09 %	0.13 %	1.02 %	0.00 %	0.00 %
StDev	3.81 %	0.08 %	0.07 %	0.60 %	1.48 %	0.23 %	0.39 %
Avg/Max	0.90	0.48	0.46	0.41	0.54	0.15	0.48

Recalculate efficiency of iterative region



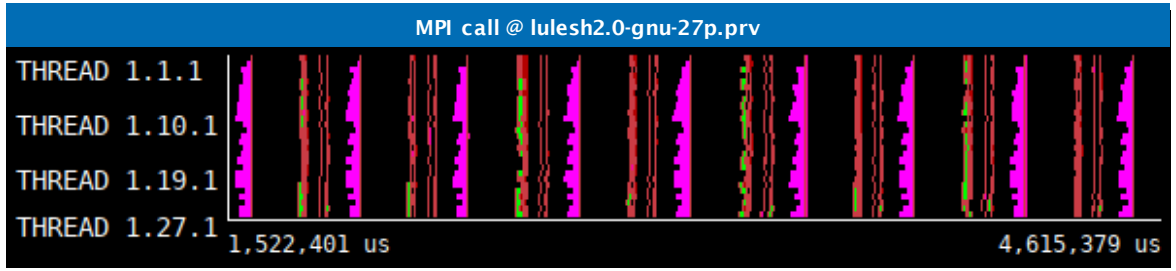
Right click → Copy

MPI call profile @ lulesh2.0-gnu-27p.prv

Default

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Barrier	MPI_Reduce
THREAD 1.15.1	86.19 %	0.33 %	0.27 %	0.29 %	2.50 %	0.01 %	0.68 %
THREAD 1.16.1	89.34 %	0.23 %	0.17 %	0.48 %	2.79 %	0.77 %	0.11 %
THREAD 1.17.1	88.21 %	0.32 %	0.26 %	0.97 %	1.76 %	0.05 %	0.45 %
THREAD 1.18.1	83.07 %	0.23 %	0.17 %	0.31 %	4.35 %	0.03 %	0.82 %
THREAD 1.19.1	88.96 %	0.15 %	0.12 %	0.13 %	3.32 %	0.02 %	0.23 %
THREAD 1.20.1	83.68 %	0.22 %	0.17 %	0.19 %	3.17 %	0.01 %	0.70 %
THREAD 1.21.1	82.59 %	0.16 %	0.11 %	0.31 %	3.86 %	0.02 %	0.62 %
THREAD 1.22.1	79.46 %	0.22 %	0.16 %	1.49 %	5.78 %	0.04 %	0.82 %
THREAD 1.23.1	87.28 %	0.32 %	0.22 %	0.61 %	5.92 %	0.01 %	0.00 %
THREAD 1.24.1	82.04 %	0.22 %	0.15 %	1.36 %	5.92 %	0.03 %	0.60 %
THREAD 1.25.1	87.67 %	0.15 %	0.09 %	0.77 %	5.53 %	0.78 %	0.06 %
THREAD 1.26.1	82.51 %	0.23 %	0.14 %	1.36 %	6.27 %	0.11 %	0.48 %
THREAD 1.27.1	83.15 %	0.16 %	0.09 %	0.60 %	4.95 %	0.09 %	0.69 %
Total	2,334.32 %	5.82 %	5.11 %	28.70 %	91.30 %	3.26 %	17.66 %
Average	86.46 %	0.22 %	0.19 %	1.06 %	3.38 %	0.12 %	0.65 %
Maximum	95.90 %	0.45 %	0.41 %	2.57 %	6.27 %	0.78 %	1.36 %
Minimum	79.46 %	0.10 %	0.09 %	0.13 %	1.02 %	0.00 %	0.00 %
StDev	3.81 %	0.08 %	0.07 %	0.60 %	1.48 %	0.23 %	0.39 %
Avg/Max	0.90	0.48	0.46	0.41	0.54	0.15	0.48

Recalculate efficiency of iterative region



Right click → Paste → Time

MPI call profile @ lulesh2.0-gnu-27p.prv

Default

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm
THREAD 1.15.1	86.02 %	0.30 %	0.26 %	0.28 %	2.00 %	11.05 %	
THREAD 1.16.1	90.24 %	0.21 %	0.17 %	0.53 %	2.15 %	6.60 %	
THREAD 1.17.1	88.26 %	0.30 %	0.26 %	0.26 %	1.94 %	8.90 %	
THREAD 1.18.1	83.45 %	0.21 %	0.17 %	0.33 %	3.12 %	12.62 %	
THREAD 1.19.1	89.30 %	0.14 %	0.11 %	0.12 %	2.47 %	7.75 %	
THREAD 1.20.1	83.76 %	0.20 %	0.16 %	0.17 %	1.91 %	13.70 %	
THREAD 1.21.1	81.91 %	0.15 %	0.11 %	0.33 %	3.17 %	14.21 %	
THREAD 1.22.1	80.08 %	0.21 %	0.15 %	0.73 %	4.85 %	13.88 %	
THREAD 1.23.1	87.98 %	0.29 %	0.22 %	0.45 %	4.97 %	6.00 %	
THREAD 1.24.1	82.75 %	0.21 %	0.14 %	0.72 %	5.09 %	10.99 %	
THREAD 1.25.1	88.36 %	0.14 %	0.09 %	0.54 %	5.61 %	5.15 %	
THREAD 1.26.1	83.41 %	0.21 %	0.13 %	0.81 %	5.29 %	10.05 %	
THREAD 1.27.1	82.12 %	0.15 %	0.09 %	0.61 %	5.24 %	11.70 %	
Total	2,341.75 %	5.30 %	5.03 %	18.19 %	80.27 %	246.82 %	
Average	86.73 %	0.20 %	0.19 %	0.67 %	2.97 %	9.14 %	
Maximum	96.71 %	0.42 %	0.41 %	1.81 %	5.61 %	14.63 %	
Minimum	80.08 %	0.09 %	0.09 %	0.12 %	0.78 %	0.03 %	
StDev	4.06 %	0.07 %	0.07 %	0.37 %	1.38 %	3.73 %	
Avg/Max	0.90	0.47	0.45	0.37	0.53	0.63	

Efficiency of iterative region

- 3 numbers to quickly describe the efficiency of your code
  - Parallel efficiency → % of time my program is computing (100% is perfect)
  - Comm efficiency → At least 1 process can finish all communications in 100 - Maximum % of the program's time (100% is perfect)
  - Load balance → Ratio of slow/fast processes (1 is perfectly balanced)
  - Any value below 85% (0.85)?
    - Pay attention there

MPI call profile @ lulesh2.0-gnu-27p.prv

Default

	Outside MPI	MPI_Isend	MPI_Irecv	MPI_Wait	MPI_Waitall	MPI_Allreduce	MPI_Comm
THREAD 1.15.1	86.02 %	0.30 %	0.26 %	0.28 %	2.00 %	11.05 %	
THREAD 1.16.1	90.24 %	0.21 %	0.17 %	0.53 %	2.15 %	6.60 %	
THREAD 1.17.1	88.26 %	0.30 %	0.26 %	0.26 %	1.94 %	8.90 %	
THREAD 1.18.1	83.45 %	0.21 %	0.17 %	0.33 %	3.12 %	12.62 %	
THREAD 1.19.1	89.30 %	0.14 %	0.11 %	0.12 %	2.47 %	7.75 %	
THREAD 1.20.1	83.76 %	0.20 %	0.16 %	0.17 %	1.91 %	13.70 %	
THREAD 1.21.1	81.91 %	0.15 %	0.11 %	0.33 %	3.17 %	14.21 %	
THREAD 1.22.1	80.08 %	0.21 %	0.15 %	0.73 %	4.85 %	13.88 %	
THREAD 1.23.1	87.98 %	0.29 %	0.22 %	0.45 %	4.97 %	6.00 %	
THREAD 1.24.1	82.75 %	0.21 %	0.14 %	0.72 %	5.09 %	10.99 %	
THREAD 1.25.1	88.36 %	0.14 %	0.09 %	0.54 %	5.61 %	5.15 %	
THREAD 1.26.1	83.41 %	0.21 %	0.13 %	0.81 %	5.29 %	10.05 %	
THREAD 1.27.1	82.12 %	0.15 %	0.09 %	0.61 %	5.24 %	11.70 %	
Total	2,341.75 %	5.30 %	5.03 %	18.19 %	80.27 %	246.82 %	
Average	86.73 %	0.20 %	0.19 %	0.67 %	2.97 %	9.14 %	
Maximum	96.71 %	0.42 %	0.41 %	1.81 %	5.61 %	14.63 %	
Minimum	80.08 %	0.09 %	0.09 %	0.12 %	0.78 %	0.03 %	
StDev	4.06 %	0.07 %	0.07 %	0.37 %	1.38 %	3.73 %	
Avg/Max	0.90	0.47	0.45	0.37	0.53	0.63	

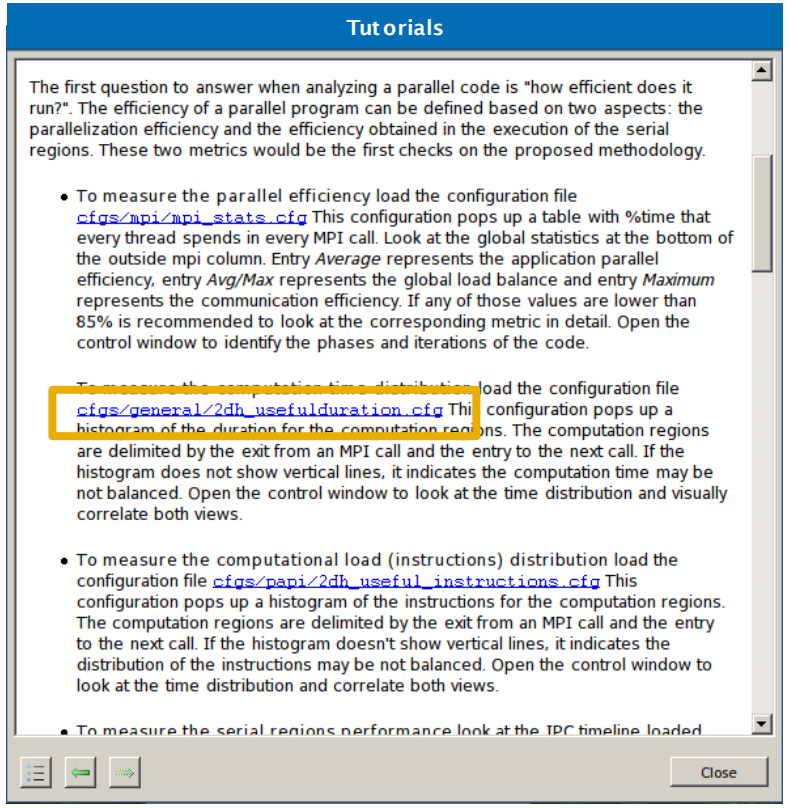
Parallel efficiency (Avg)

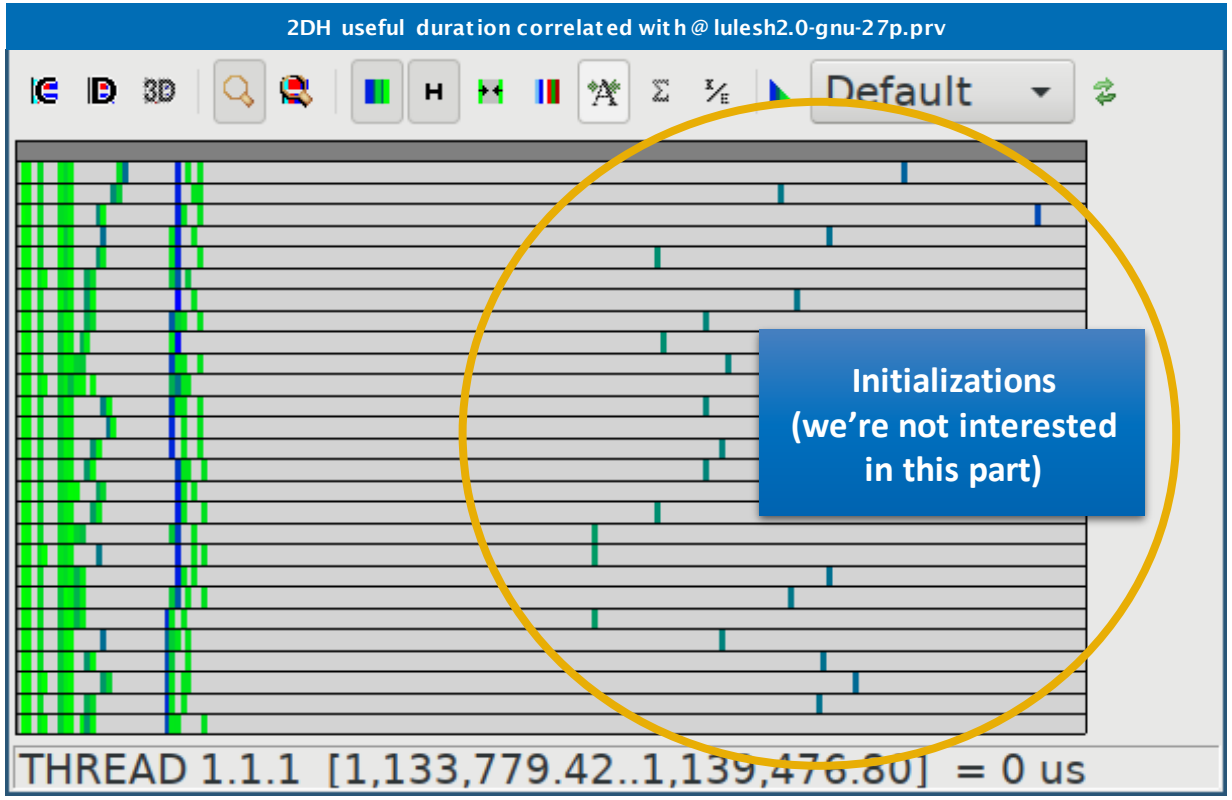
Comm efficiency (Max)

Load balance (Avg/Max)

Computation time distribution

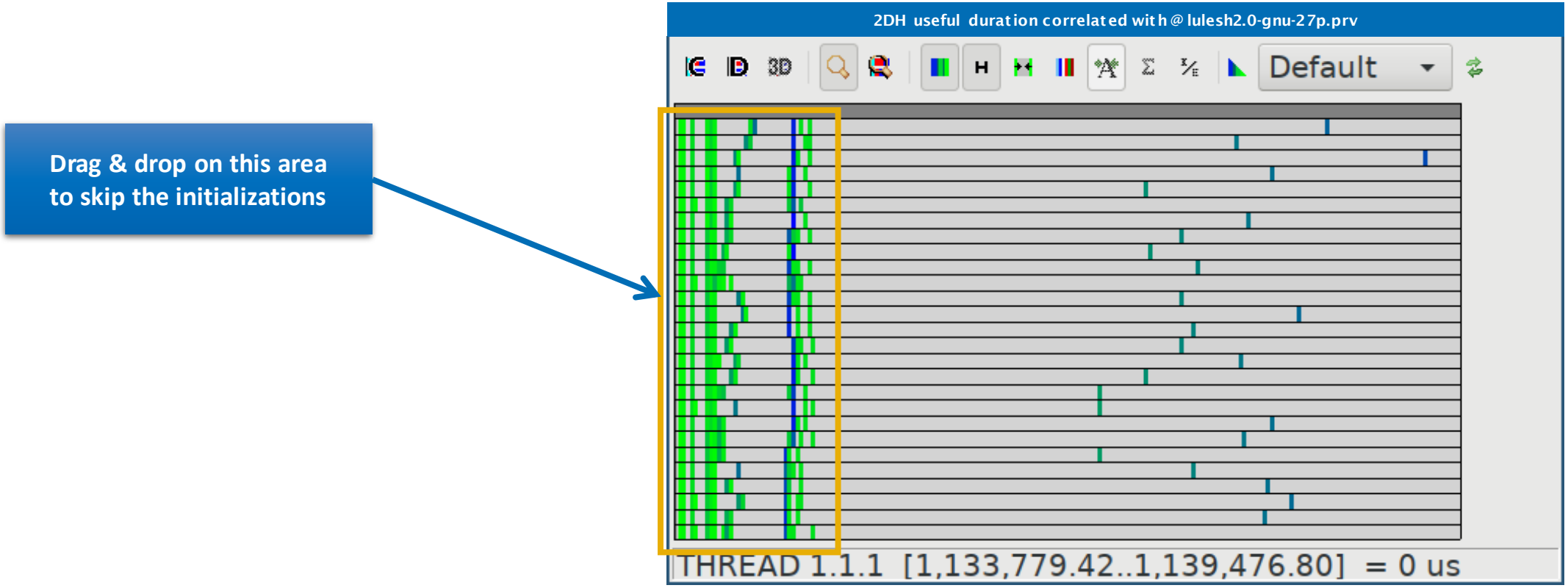
- Click on "2dh_usefulduration.cfg" (2nd link) → Shows **time computing**





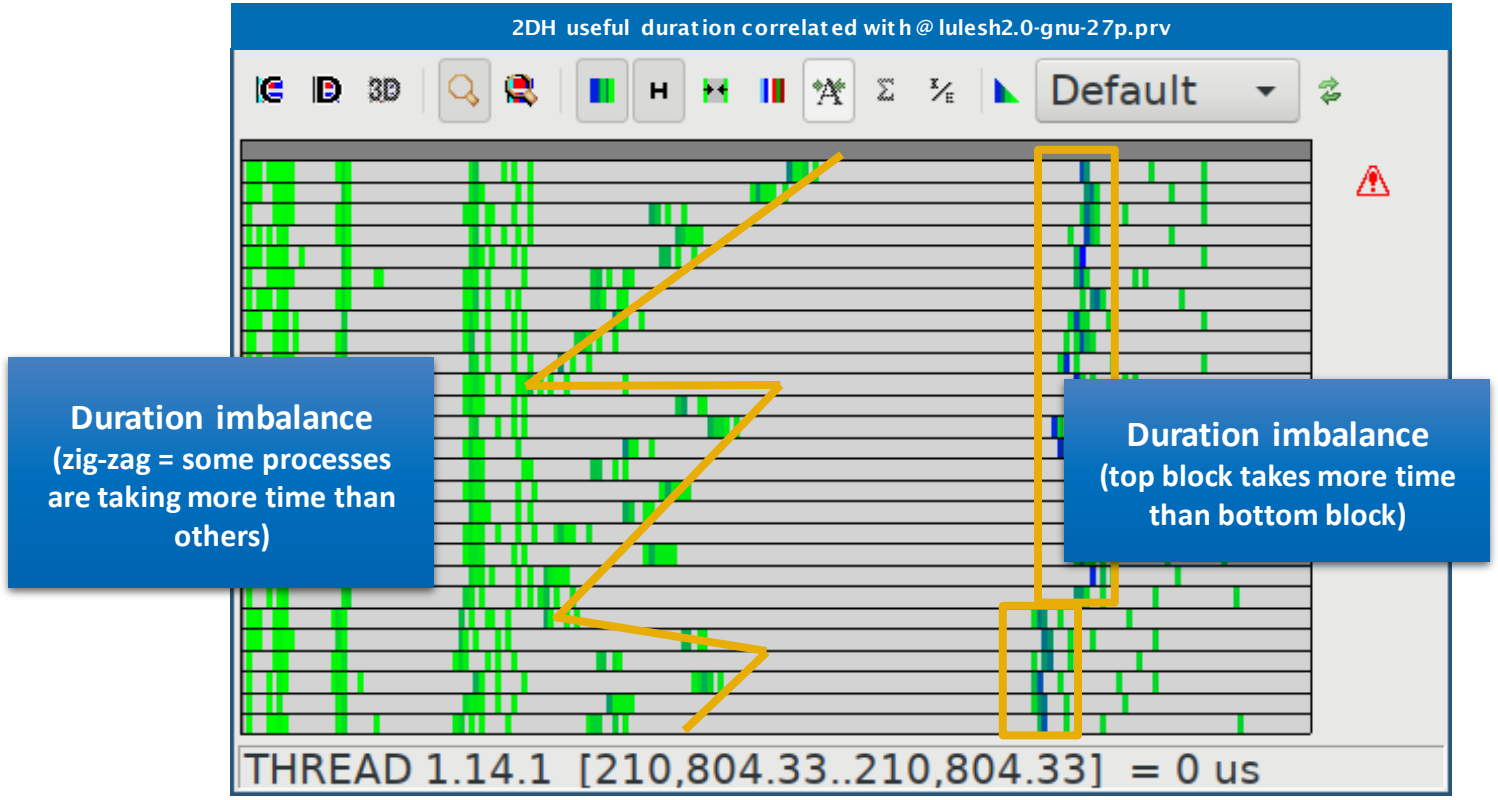
Focus on the iterative part

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



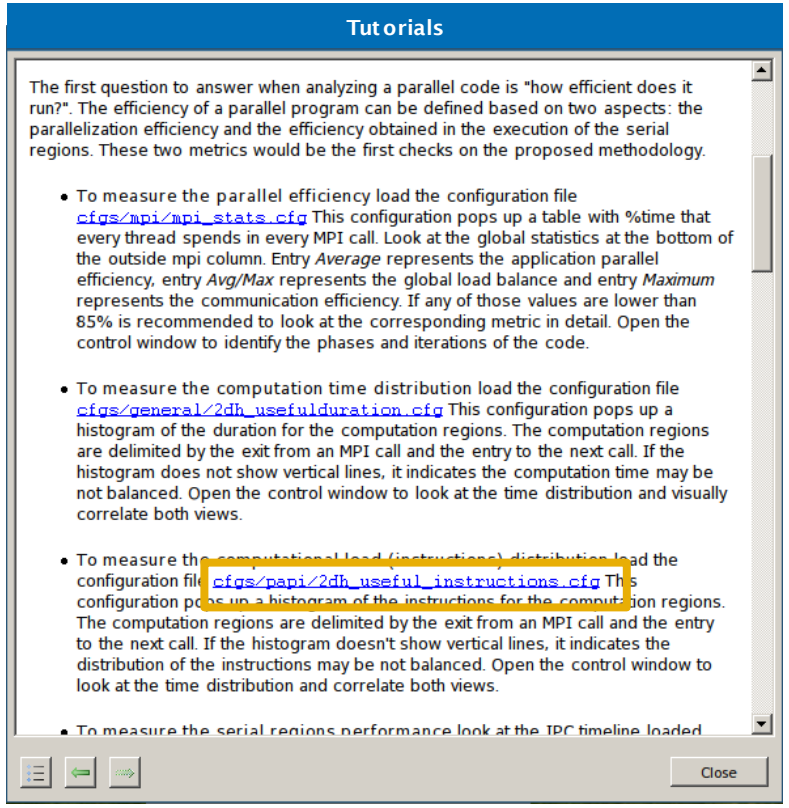
Computation time distribution

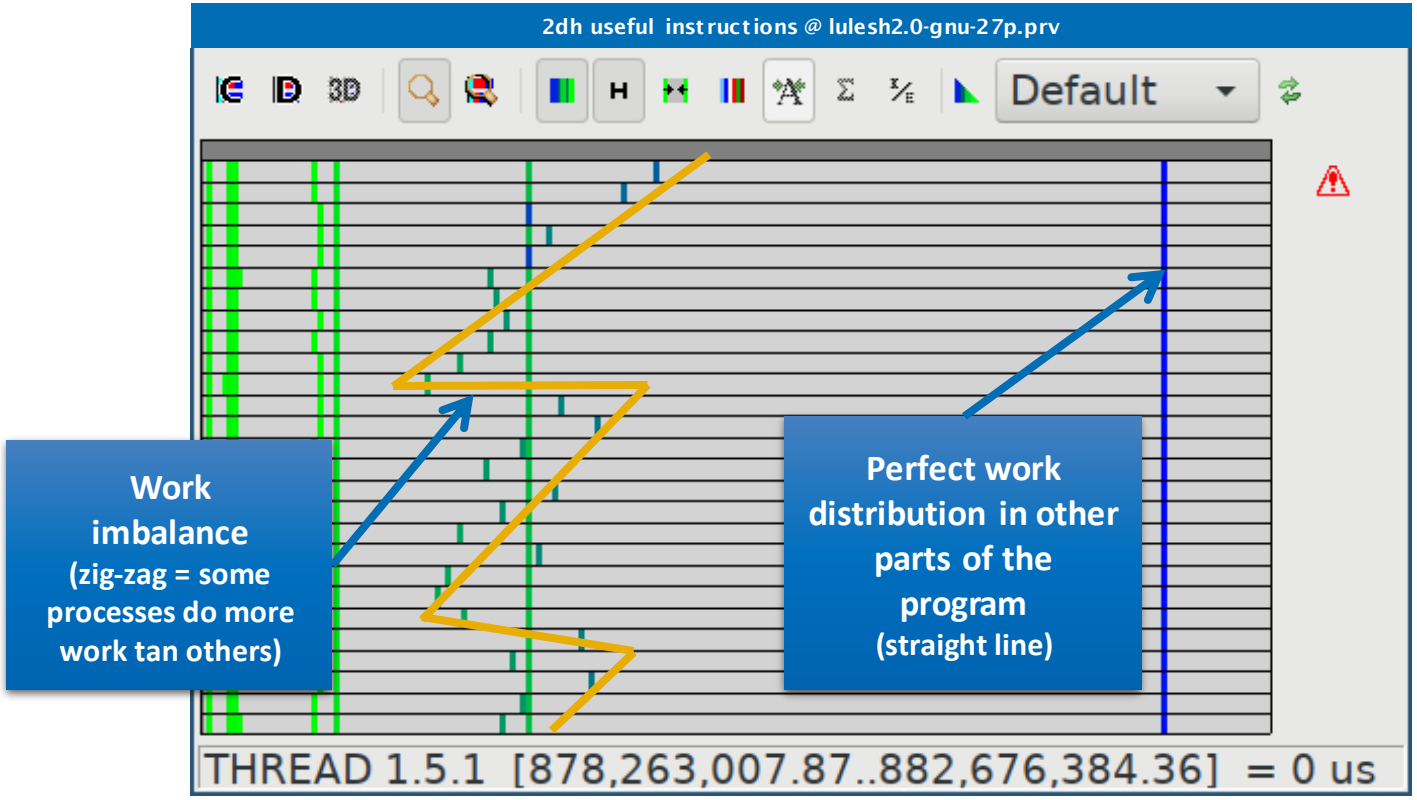
- Click on "2dh_usefulduration.cfg" (2nd link) → Shows **time computing**



Computation load distribution

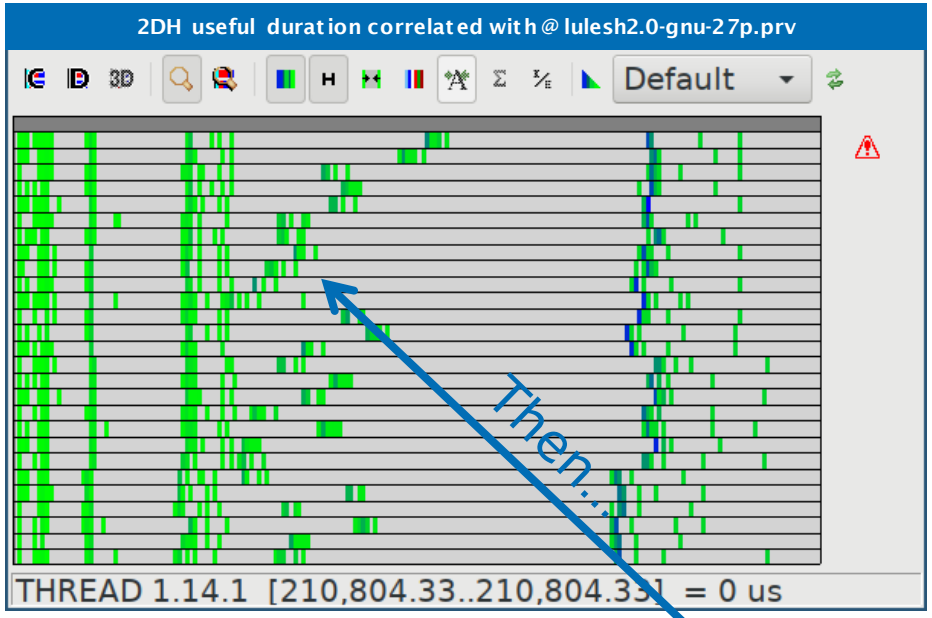
- Click on "2dh_useful_instructions.cfg" (3rd link) → Shows **amount of work**





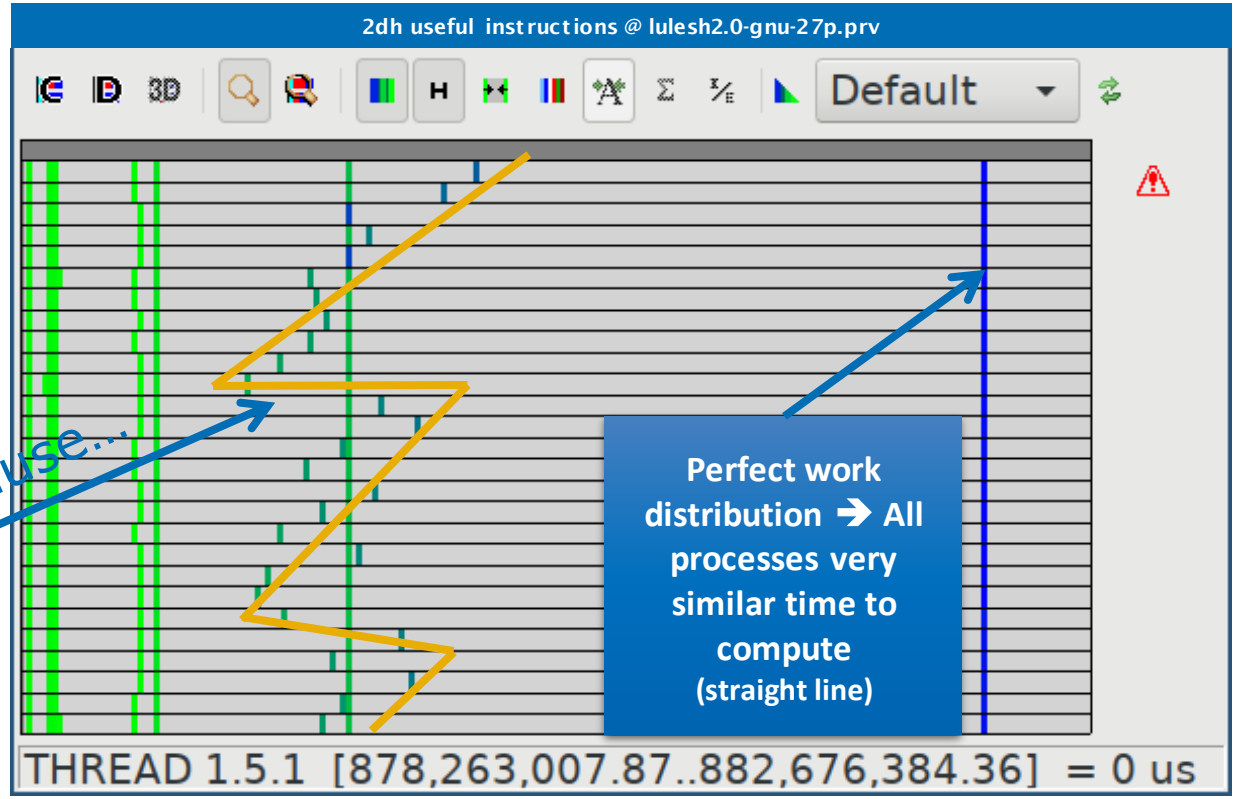
Computation load distribution

- Comparing the two histograms → **Similar shapes** → Work distribution determines time computing



Then...

Because...

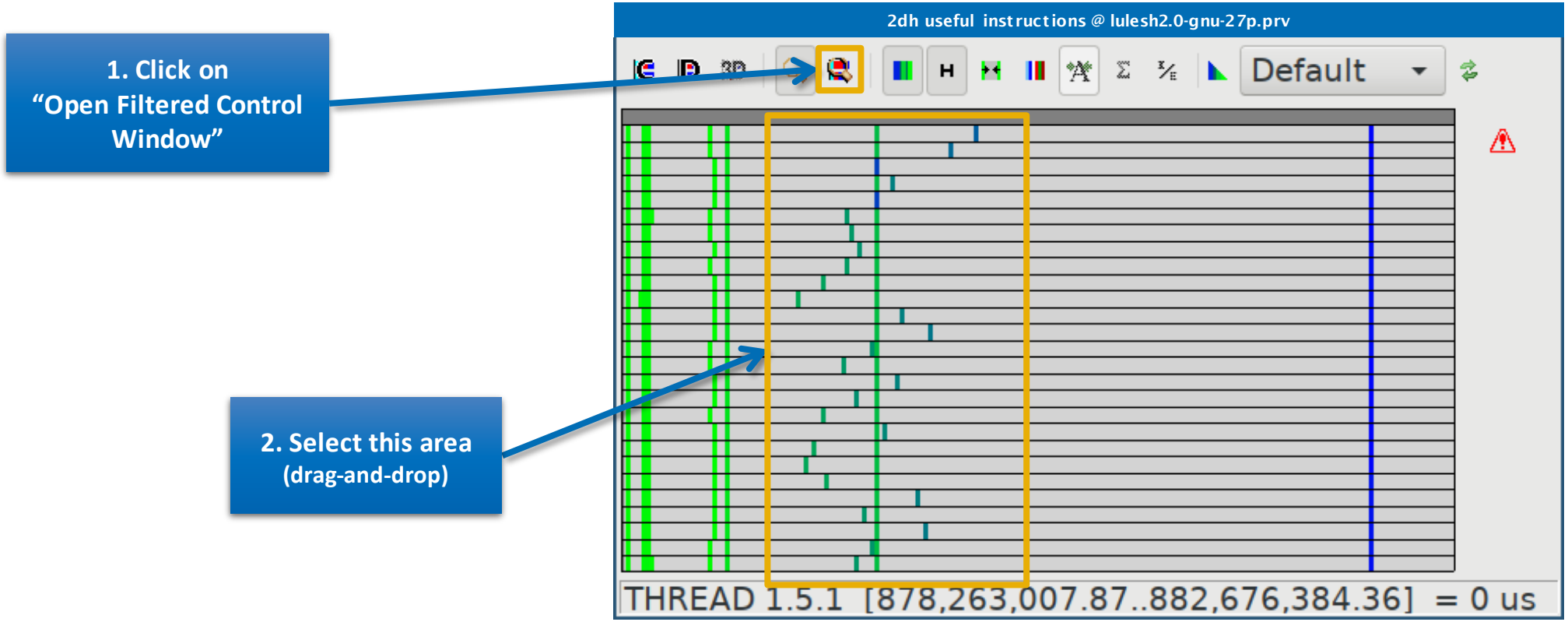


Work imbalance → Slow/fast processes (zig-zag)

Perfect work distribution → All processes very similar time to compute (straight line)

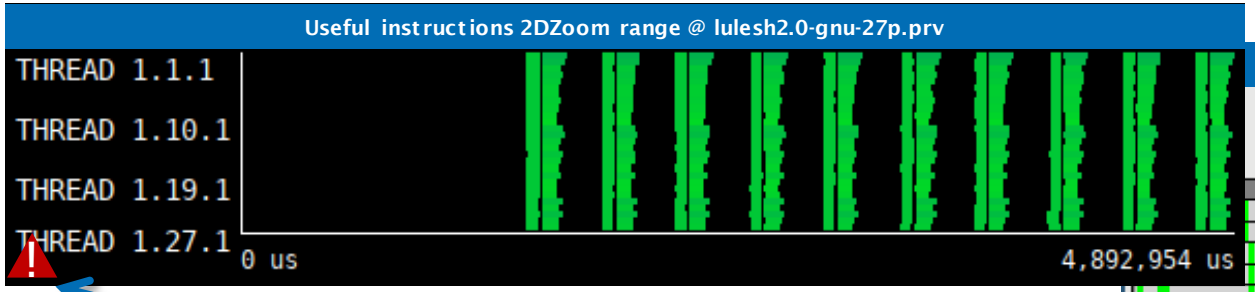
Where does this happen?

- Go from the table to the timeline

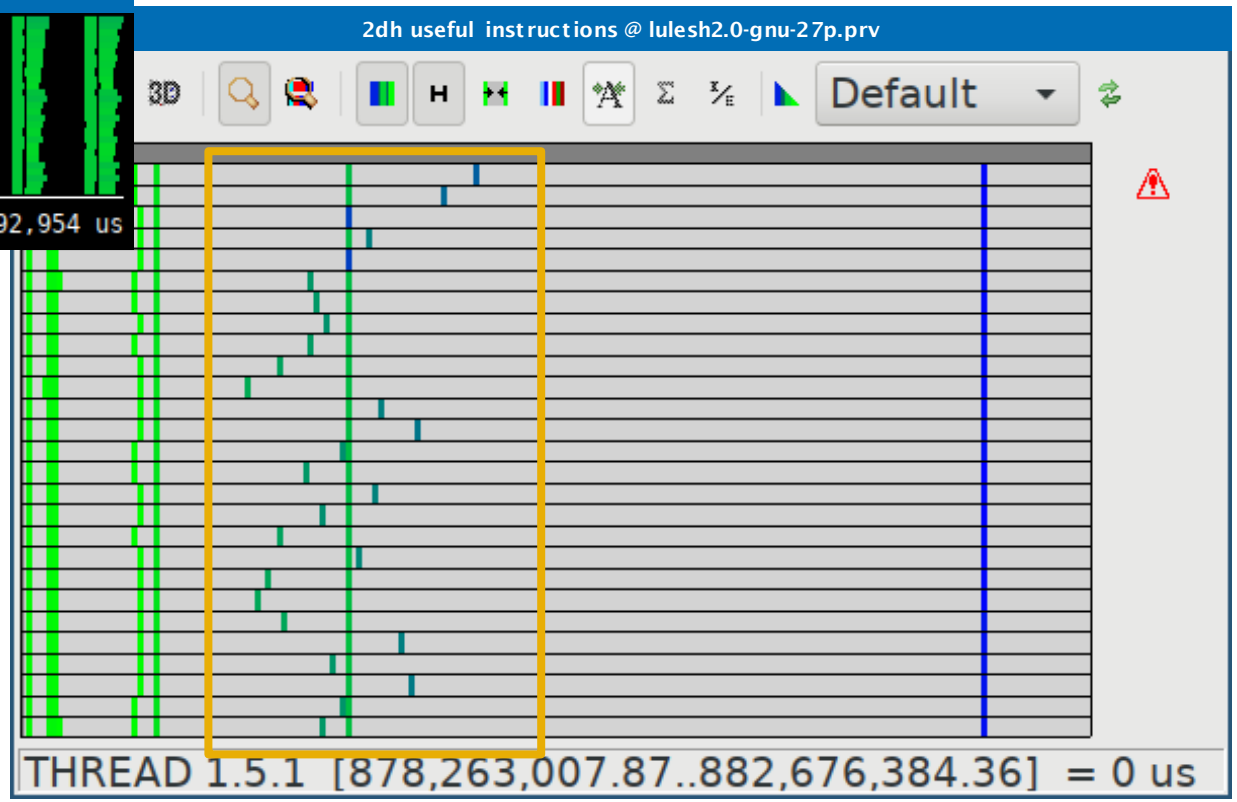


Where does this happen?

- Go from the table to the timeline

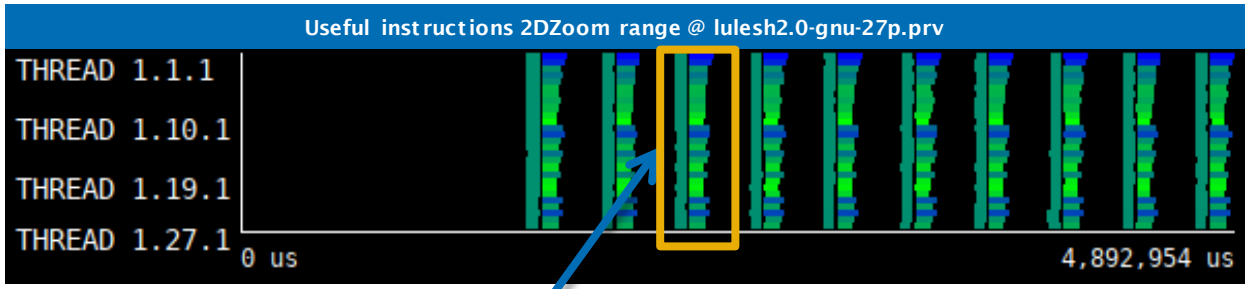


Clicking here always rescales.
Same as…
Right click →
Fit Semantic Scale → Fit Both



Where does this happen?

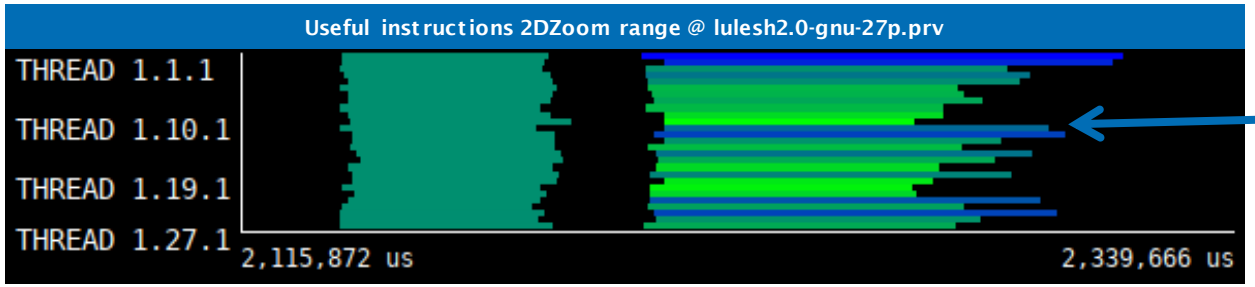
- **Slow** & **Fast** at the same time? → Imbalance



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

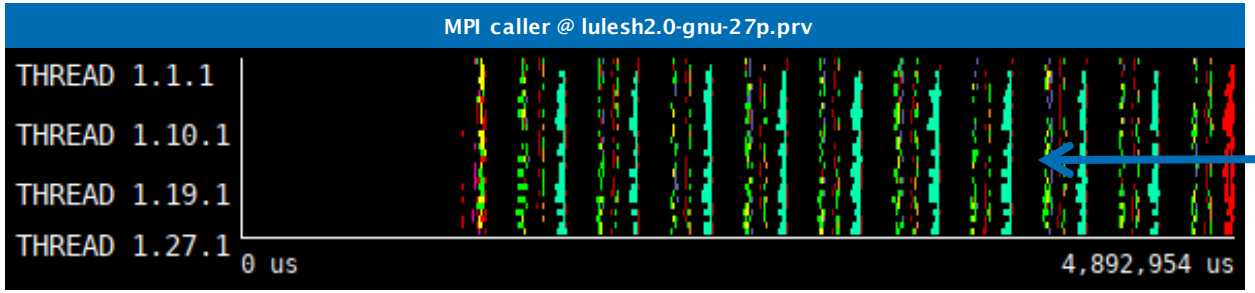
Where does this happen?

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



1. Right click → Copy

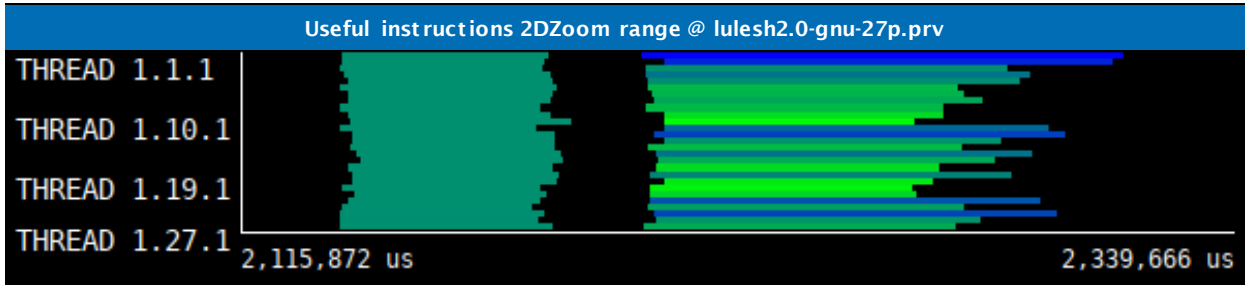
- **Hints → Call stack references → Caller function**



2. Right click → Paste → Time

Where does this happen?

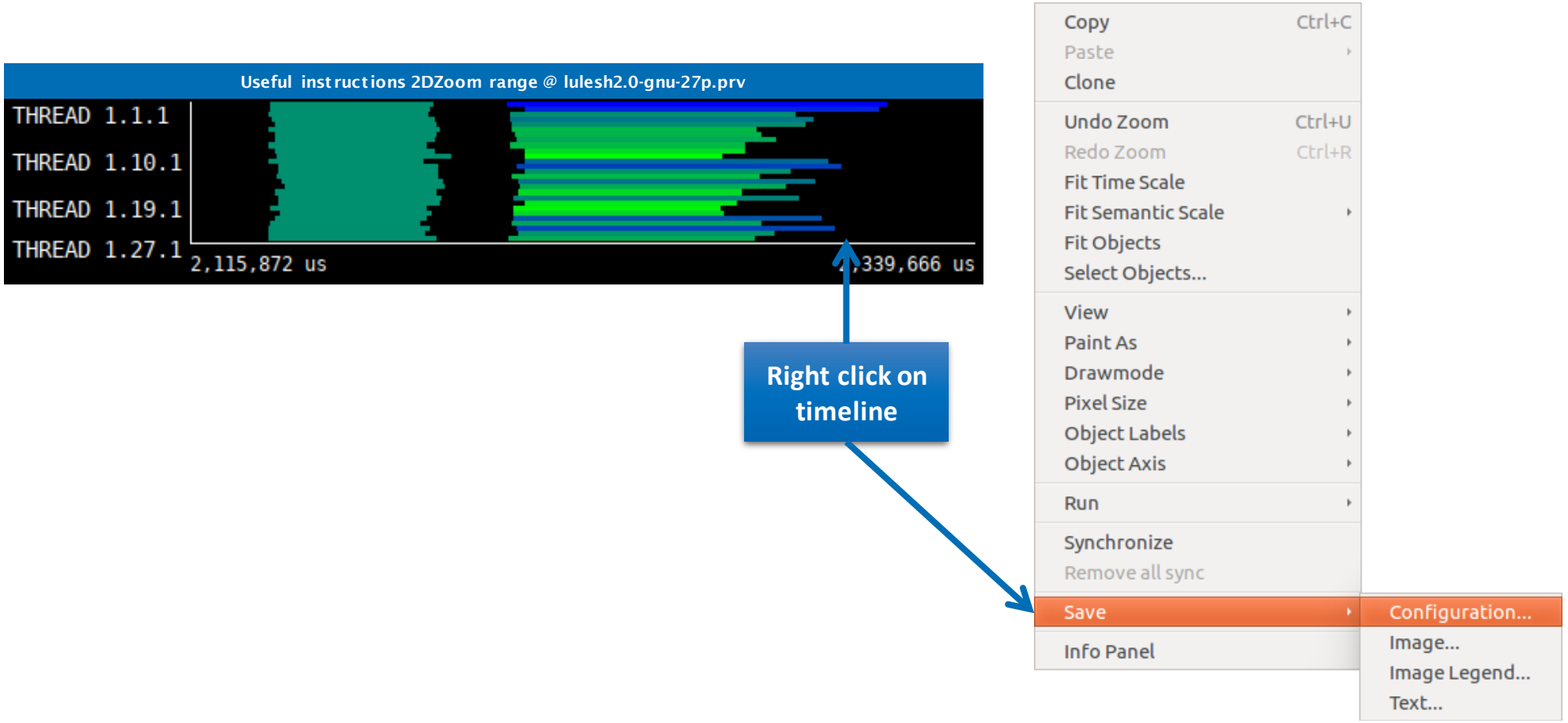
- **Slow** & **Fast** at the same time? → Imbalance



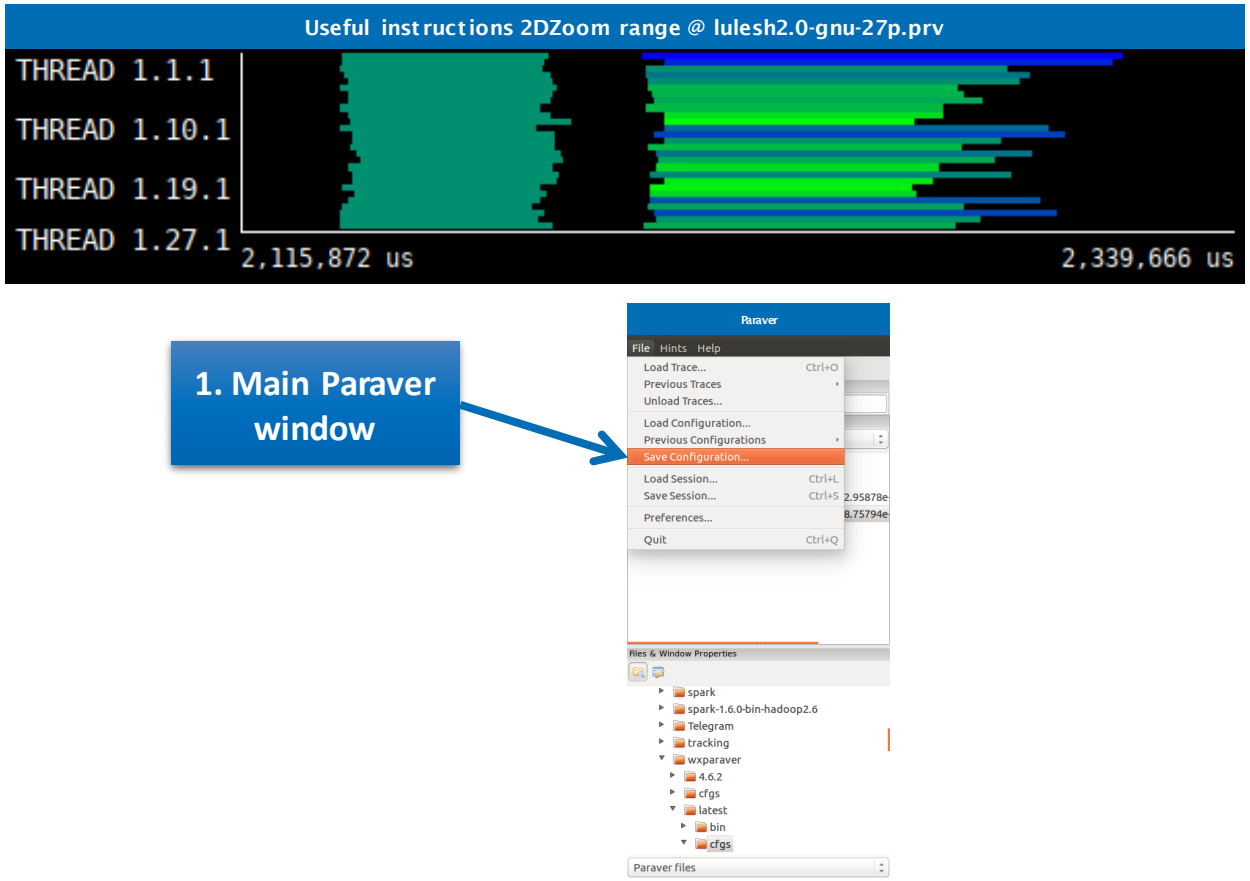
- **Hints → Call stack references → Caller function**

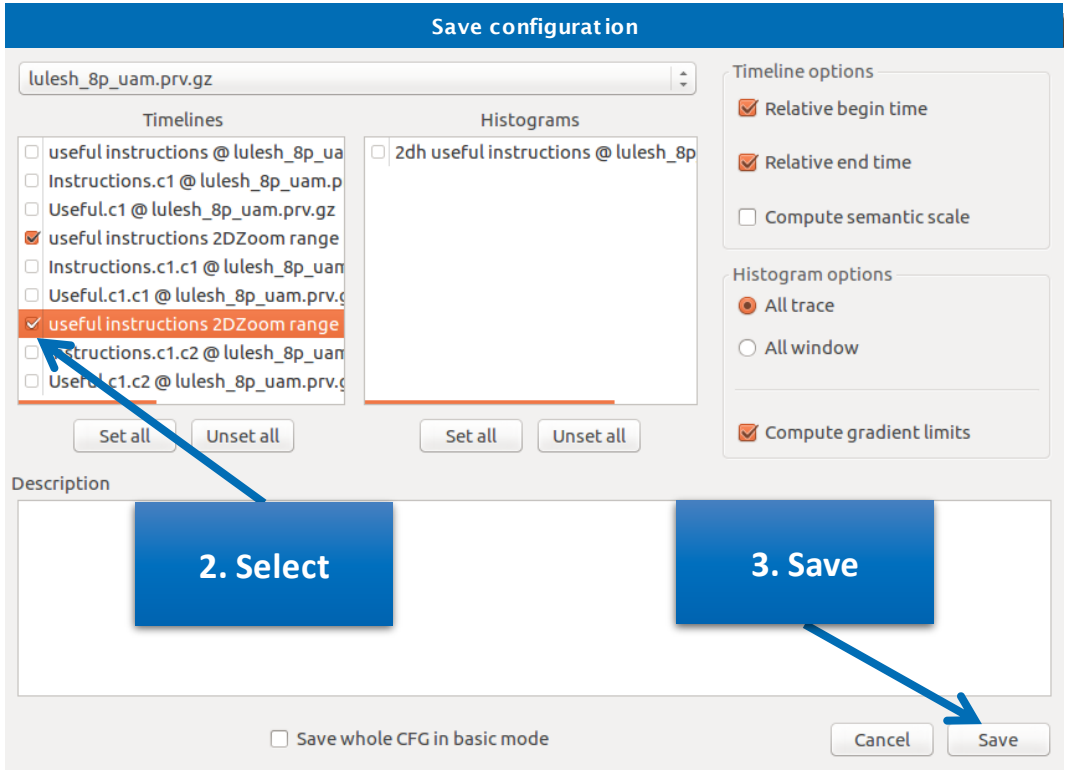
CommSend | CommMonoQ | TimeIncrement

Save CFG's (method 1)



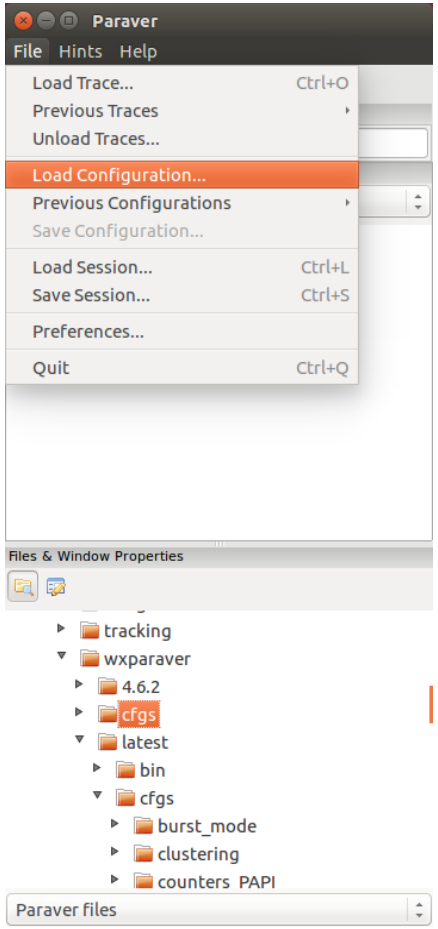
Save CFG's (method 2)

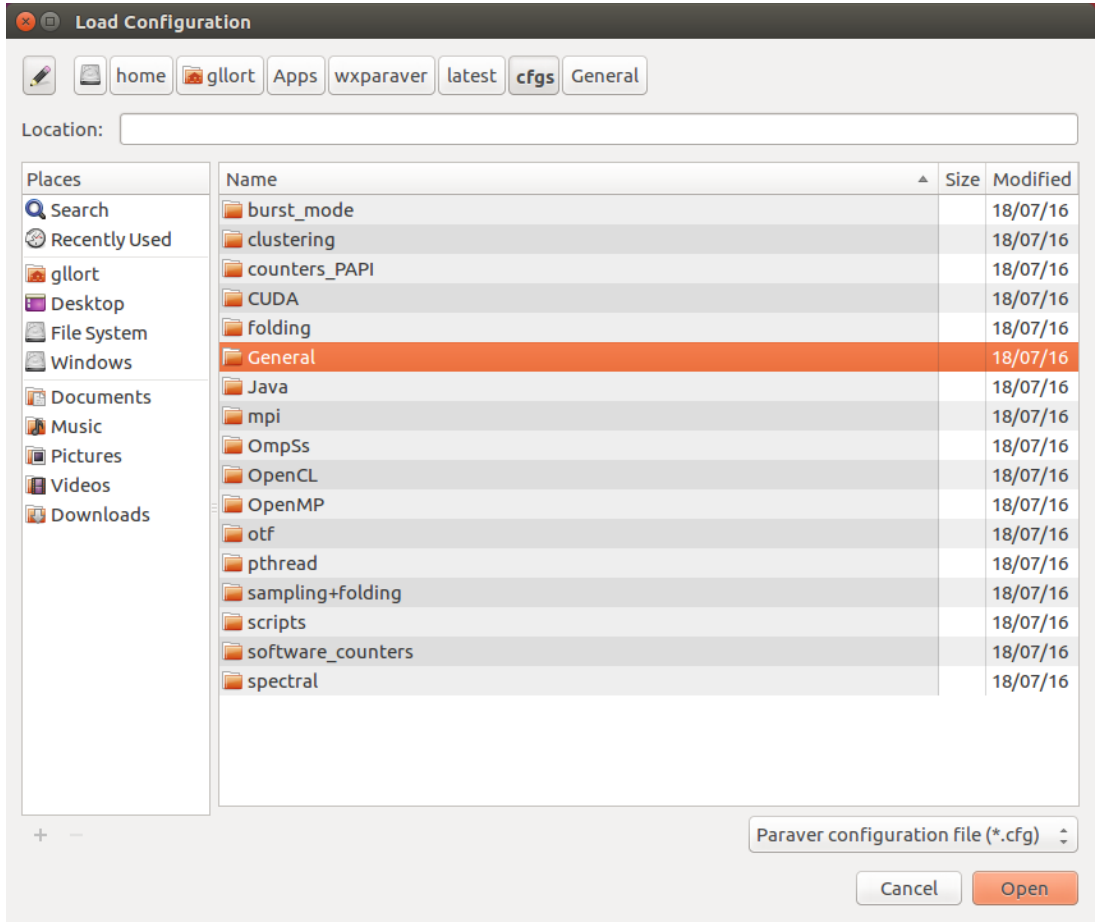




CFG's distribution

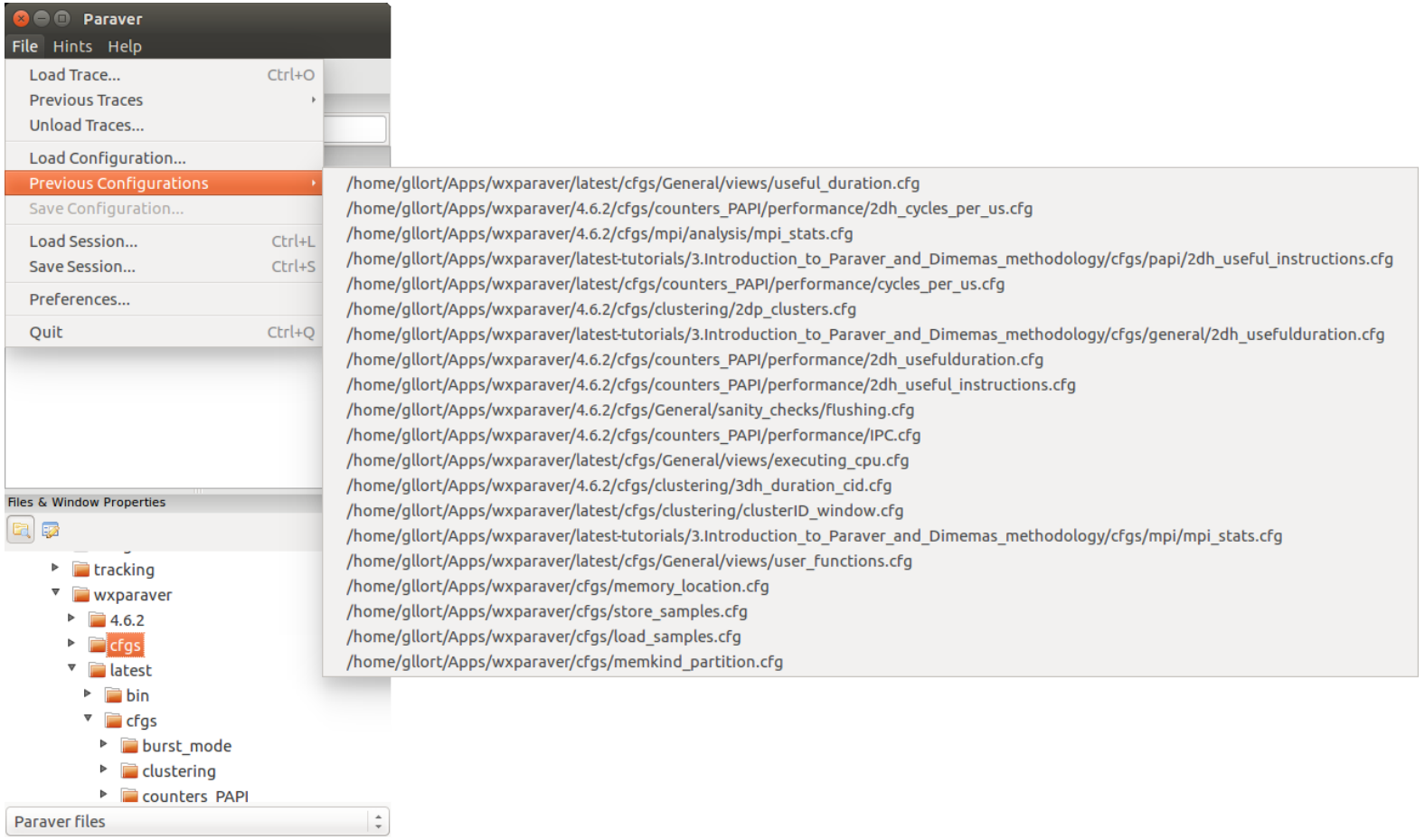
- Paraver comes with many included CFG's ➔ Apply any CFG to any trace!





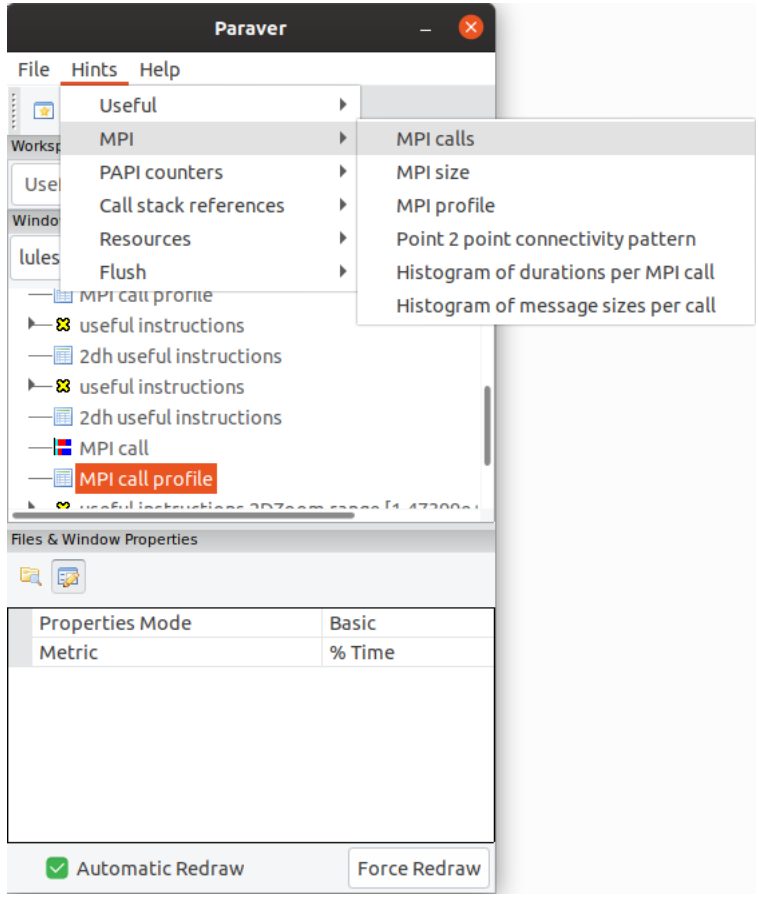
CFG's distribution

- Paraver comes with many included CFG's ➔ Apply any CFG to any trace!



Hints: a good place to start!

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



Do it on your code!

- Follow guidelines from slides 7-16 to your own code to get a trace
  - There are more examples of tracing scripts for different programming models under `$EXTRAE_HOME/share/examples`
- Follow guidelines from slides 17-34 to conduct an initial analysis
  - The usual suspects:
    - Parallel Efficiency is low? Load balance issues?
    - Imbalances in the durations of computations?
    - Are these caused by work imbalance?

Cluster-based analysis

Use clustering analysis

- Run clustering

```
meggie> cd $HOME/tools-material/clustering
meggie> module load clustering
meggie> BurstClustering -d cluster.xml \
        -i ../extrae/lulesh2.0-intel_27p.prv \
        -o lulesh2.0-intel_27p_clustered.prv
```

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

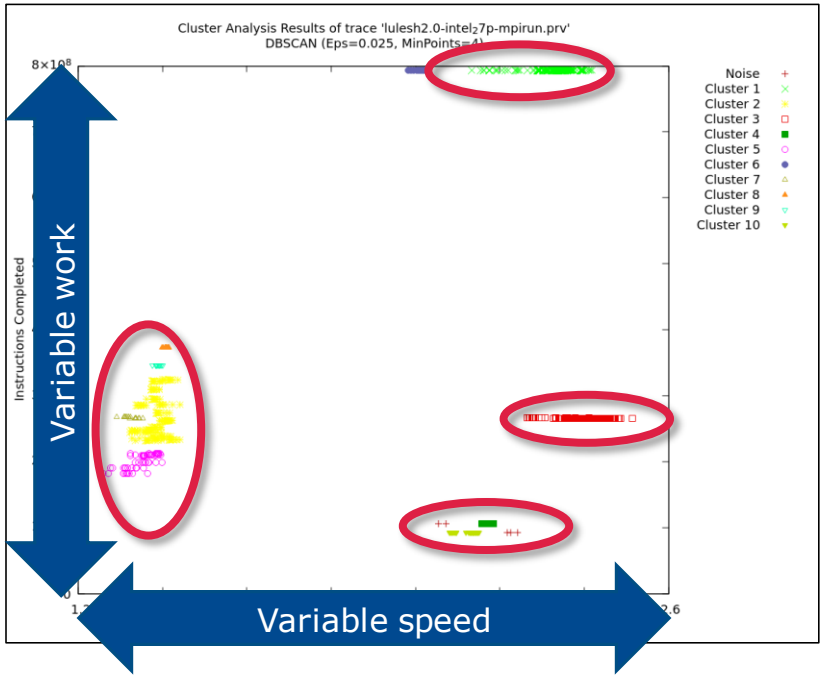
```
meggie> ls $HOME/tools-material/traces/lulesh2.0-intel_27p-mpirun.prv
```

Cluster-based analysis

- Check the resulting scatter plot

```
laptop> gnuplot lulesh2.0-intel_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**

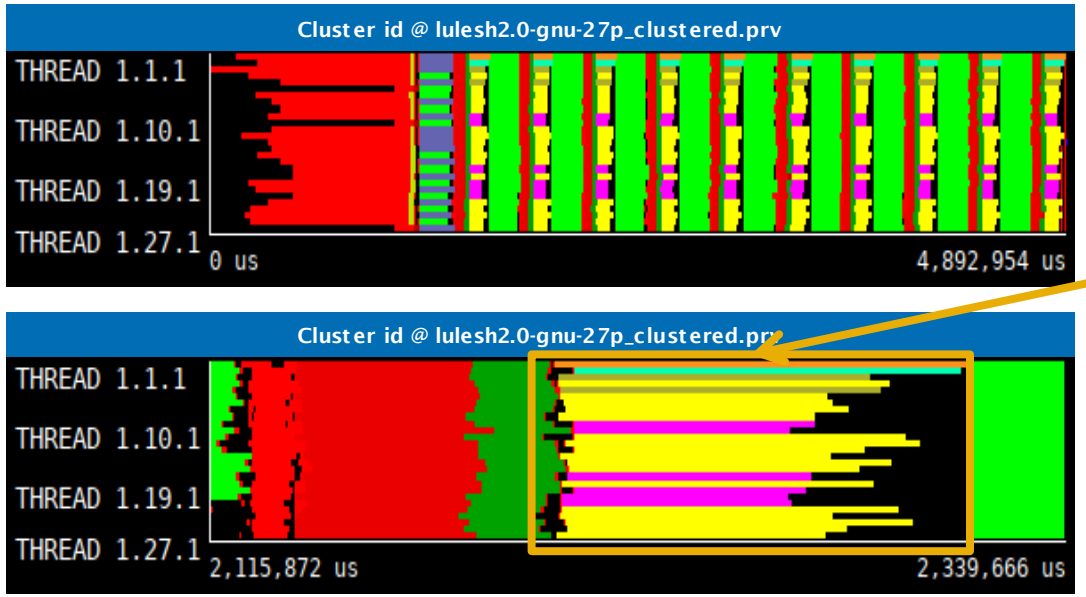


Correlating scatter plot and time distribution

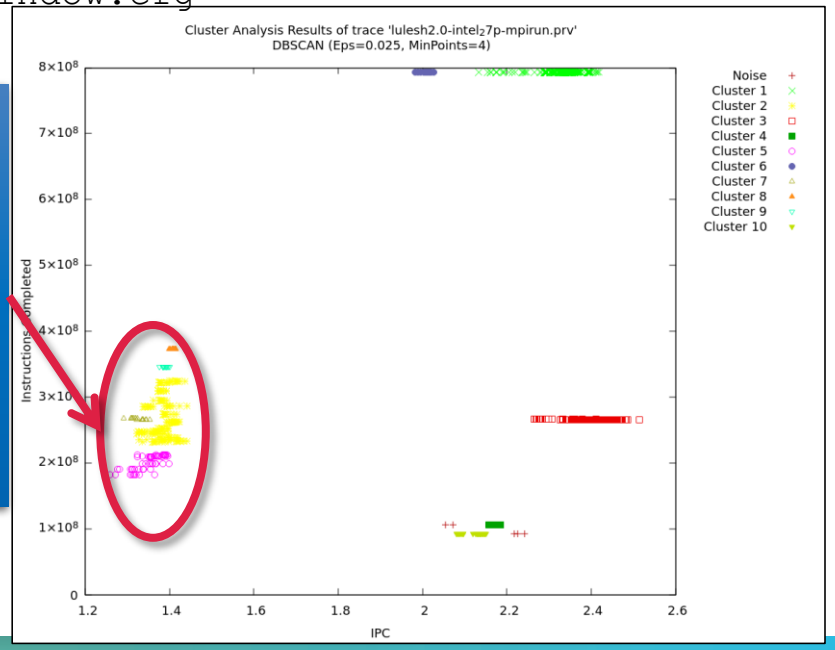
- Open the clustered trace with Paraver and look at it

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