Extrae & Paraver Hands-On

Judit Giménez, Germán Llort

tools@bsc.es

16/09/19

POP-EoCoE

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 / relink!

Extrae overheads

	Average values	p2chpd
Event	150 – 200 ns	240 ns
Event + PAPI	750 ns – 1.5 us	5.8 us
Event + callstack (1 level)	1 us	814 ns
Event + callstack (6 levels)	2 us	2.7 us

How does Extrae work?

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

Recommended

- Dynamic instrumentation
  - Based on Dyninst (developed by U.Wisconsin / U.Maryland)
    - Instrumentation in memory
    - Binary rewriting

- Alternatives
  - 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/index.html
  - Also distributed with Extrae at $EXTRAE_HOME/share/doc

Log in

@ your laptop

```
> ssh -Y <USER>@p2chpd-login3.univ-lyon1.fr
```

- Copy material to your home folder:

@ p2chpd

```
> cp ~germain.llort/tools-material $HOME

> ls -l $HOME/tools-material
    … apps/
    … clustering/
    … extrae/
    … slides/
    … traces/
```

Here you have a copy of this slides.
Copy them to your laptop
or open remotely with:

```
> evince slides/Extrae-Paraver-Hands-On.pdf
```

Step 1: Adapt the job script to load Extrae (LD_PRELOAD)

@ p2chpd

```
> vi $HOME/tools-material/extrae/run_lulesh_27p.sh
```

```
#!/bin/bash
#SBATCH --partition=parallel
#SBATCH -t 00:10:00
#SBATCH -n 27
#SBATCH --nodes=2
#SBATCH --cpus-per-task 1
#SBATCH --output log_lulesh_27p.out
#SBATCH --error log_lulesh_27p.err
#SBATCH --exclusive
#SBATCH --reservation=BSC2019

module load mpi/openmpi/1.8.4/openmpi-gcc47



mpirun ../apps/lulesh2.0 -p -i 10 -s 65
```

Request resources

Set MPI version

Run the program

Step 1: Adapt the job script to load Extrae (LD_PRELOAD)

@ p2chpd

```
> vi $HOME/tools-material/extrae/run_lulesh_27p.sh
```

```
#!/bin/bash
#SBATCH --partition=parallel
#SBATCH -t 00:10:00
#SBATCH -n 27
#SBATCH --nodes=2
#SBATCH --cpus-per-task 1
#SBATCH --output log_lulesh_27p.out
#SBATCH --error log_lulesh_27p.err
#SBATCH --exclusive
#SBATCH --reservation=BSC2019

module load mpi/openmpi/1.8.4/openmpi-gcc47

export EXTRAE_HOME=/home_nfs/…/extrae/3.7.1
export TRACE_NAME=lulesh_27p.prv

mpirun ./trace.sh ../apps/lulesh2.0 -p -i 10 -s 65

$EXTRAE_HOME/bin/mpi2prv
  -f TRACE.mpits
  -o $TRACE_NAME
```

Where's the tool?

Activate Extrae in the execution

Generate the trace

Step 1: Adapt the job script to load Extrae (LD_PRELOAD)

@ mt1.bsc.es

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

```
#!/bin/bash
#SBATCH --partition=parallel
#SBATCH -t 00:10:00
#SBATCH -n 27
#SBATCH --nodes=2
#SBATCH --cpus-per-task 1
#SBATCH --output log_lulesh_27p.out
#SBATCH --error log_lulesh_27p.err
#SBATCH --exclusive
#SBATCH --reservation=BSC2019

module load mpi/openmpi/1.8.4/openmpi-gcc47

export EXTRAE_HOME=/home_nfs/…/extrae/3.7.1
export TRACE_NAME=lulesh_27p.prv

mpirun ./trace.sh ../apps/lulesh2.0 -p -i 10 -s 65

$EXTRAE_HOME/bin/mpi2prv
  -f TRACE.mpits
  -o $TRACE_NAME
```

```
#!/bin/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

@ p2chpd

```
> cd $HOME/tools-material/extrae

> sbatch run_lulesh_27p.sh
```

Step 2: Extrae XML configuration

@ p2chpd

```
> vi $HOME/tools-material/extrae/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)

@ p2chpd

```
> 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_L3_TCM, PAPI_L1_DCM,
      RESOURCE_STALLS:LB
    </set>
    <set enabled="yes" domain="all" changeat-time="500000us">
      PAPI_TOT_INS, PAPI_TOT_CYC, PAPI_L2_DCM,
      RESOURCE_STALLS:SB
    </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)

@ p2chpd

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

```
<buffer enabled="yes">
  <size enabled="yes">500000</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 you will have the trace (3 files):

@ p2chpd

```
> ls -l $HOME/tools-material/extrae
    ...
    lulesh_27p.pcf
    lulesh_27p.prv
    lulesh_27p.row
```

- Any trouble? Traces already generated here:

@ p2chpd

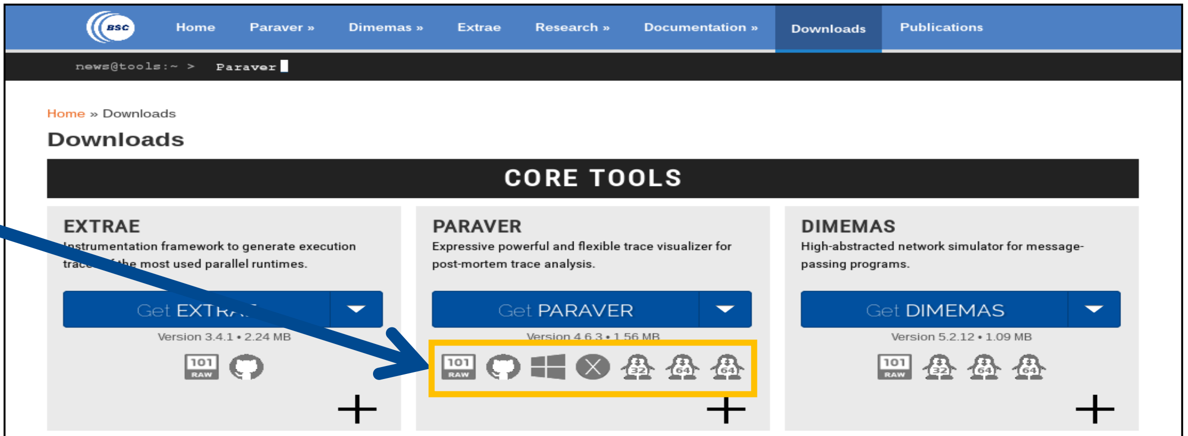
```
> ls $HOME/tools-material/traces
```

- Now let’s look into it !

Install Paraver

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

Pick your version



news@tools:~ > Paraver

Home » Downloads

Downloads

CORE TOOLS

EXTRAE
Instrumentation framework to generate execution traces of the most used parallel runtimes.
Get EXTRAE
Version 3.4.1 • 2.24 MB

PARAVER
Expressive powerful and flexible trace visualizer for post-mortem trace analysis.
Get PARAVER
Version 4.6.3 • 1.56 MB

DIMEMAS
High-abstracted network simulator for message-passing programs.
Get DIMEMAS
Version 5.2.12 • 1.09 MB

- Also @ p2chpd
  ~germain.llort/tools-packages

wxparaver-4.8.2-win.zip

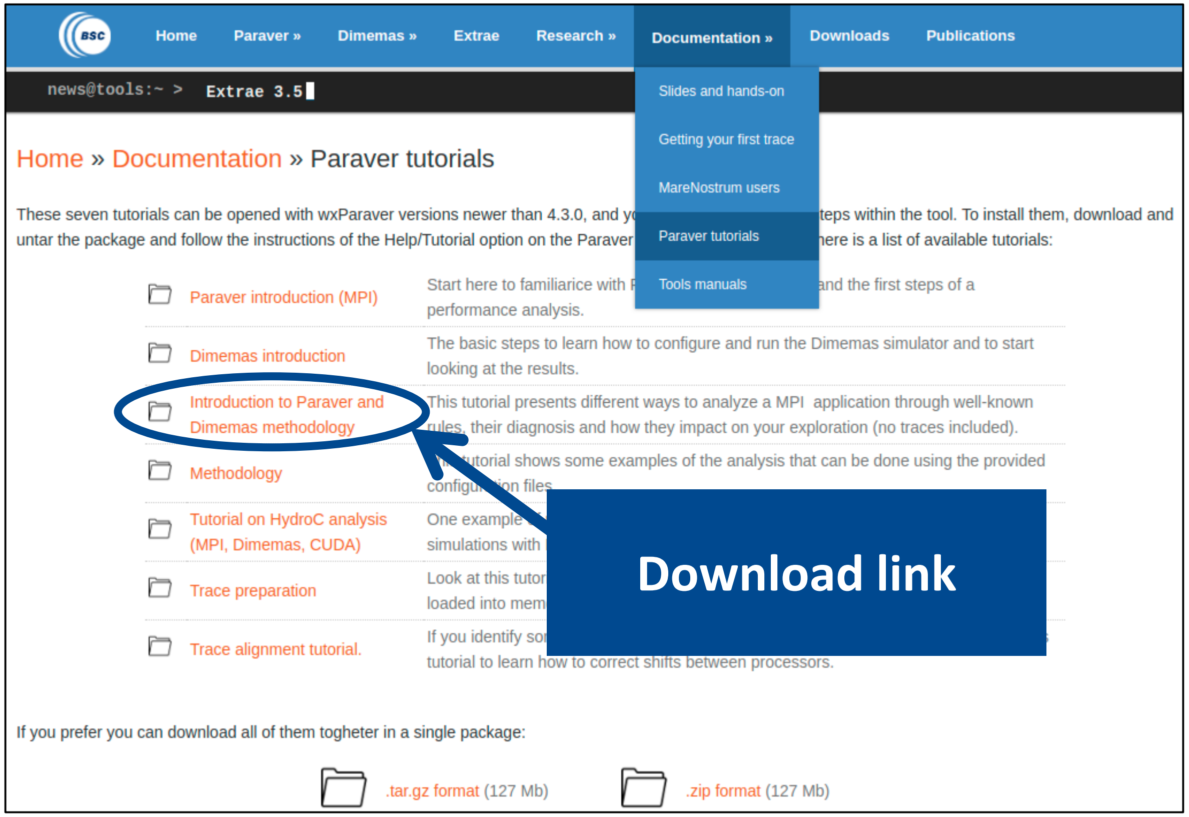
wxparaver-4.8.2-mac.zip

wxparaver-4.8.2-Linux_i686.tar.gz (32-bits)
wxparaver-4.8.2-Linux_x86_64.tar.gz (64-bits)

```
> scp <USER>@p2chpd-login3.univ-lyon1.fr:tools-
  packages/<PACKAGE> $HOME
```

Install Paraver (II)

- Download tutorials:
  - Documentation → Paraver tutorials



```
news@tools:~ >  Extrae 3.5
```

Home » Documentation » Paraver tutorials

These seven tutorials can be opened with wxParaver versions newer than 4.3.0, and y teps within the tool. To install them, download and untar the package and follow the instructions of the Help/Tutorial option on the Paraver here is a list of available tutorials:

Paraver introduction (MPI)	Start here to familiarice with P and the first steps of a performance analysis.
Dimemas introduction	The basic steps to learn how to configure and run the Dimemas simulator and to start looking at the results.
Introduction to Paraver and Dimemas methodology	This tutorial presents different ways to analyze a MPI application through well-known rules, their diagnosis and how they impact on your exploration (no traces included).
Methodology	utorial shows some examples of the analysis that can be done using the provided configu ion files.
Tutorial on HydroC analysis (MPI, Dimemas, CUDA)	One example simulations with
Trace preparation	Look at this tutor loaded into mem
Trace alignment tutorial.	If you identify so tutorial to learn how to correct shifts between processors.

If you prefer you can download all of them togheter in a single package:

.tar.gz format (127 Mb) .zip format (127 Mb)

Download link

- Also @ p2chpd
  ~germain.llort/tools-packages

```
> scp <USER>@p2chpd-login3.univ-lyon1.fr:~germain.llort/tools-packages/3.* $HOME
```

Uncompress, rename & move

- Paraver

@ your laptop

```
> tar xf wxparaver-4.8.2-linux-x86_64.tar.gz
> mv wxparaver-4.8.2-linux-x86_64 paraver
```

- Tutorials

@ your laptop

```
> mkdir paraver/tutorials
> tar xf 3.introduction_to_paraver_and_dimemas_methodology.tar.gz
> mv 3.I* paraver/tutorials
```

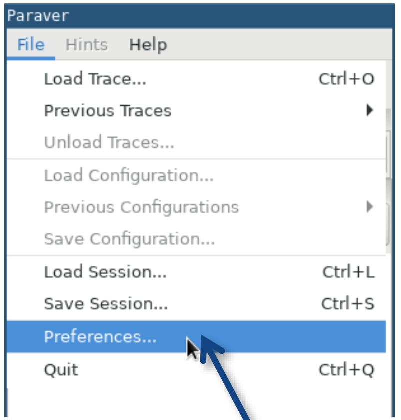
Check that everything works

- Start Paraver

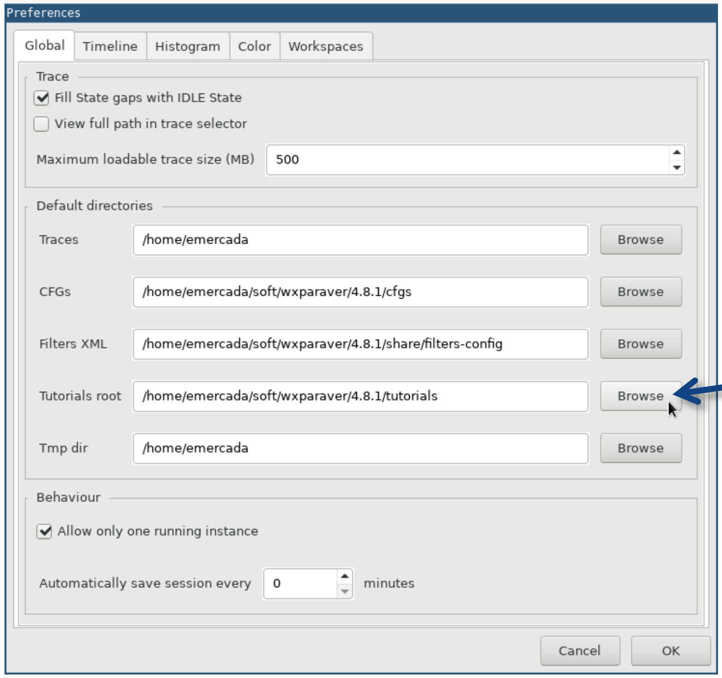
@ your laptop

```
> paraver/bin/wxparaver
```

- Tell Paraver where to find the tutorials



Click on File → Preferences



Trace
- [✓] Fill State gaps with IDLE State
- [ ] View full path in trace selector

Maximum loadable trace size (MB): 500

Default directories

Traces	/home/emercada	Browse
CFGs	/home/emercada/soft/wxparaver/4.8.1/cfgs	Browse
Filters XML	/home/emercada/soft/wxparaver/4.8.1/share/filters-config	Browse
Tutorials root	/home/emercada/soft/wxparaver/4.8.1/tutorials	Browse
Tmp dir	/home/emercada	Browse

Behaviour
- [✓] Allow only one running instance

Automatically save session every 0 minutes

Browse to path paraver/tutorials

Check that everything works

- Trouble installing locally? Remote open from p2chpd

@ p2chpd

```
> ~germain.llort/tools/wxparaver/bin/wxparaver &
```

First steps of analysis

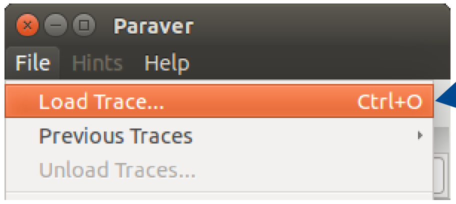
- Copy the trace to your laptop

@ your laptop

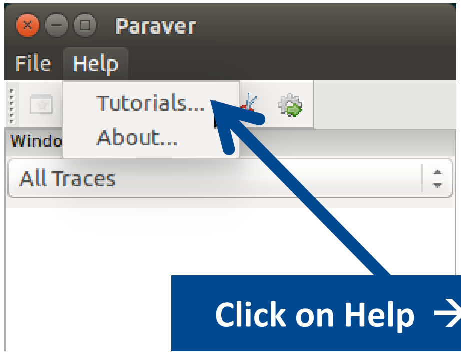
```
> scp <USER>@p2chpd-login3.univ-lyon1.fr:tools-
  material/extrae/lulesh_27p.* $HOME
```

- Load the trace with Paraver

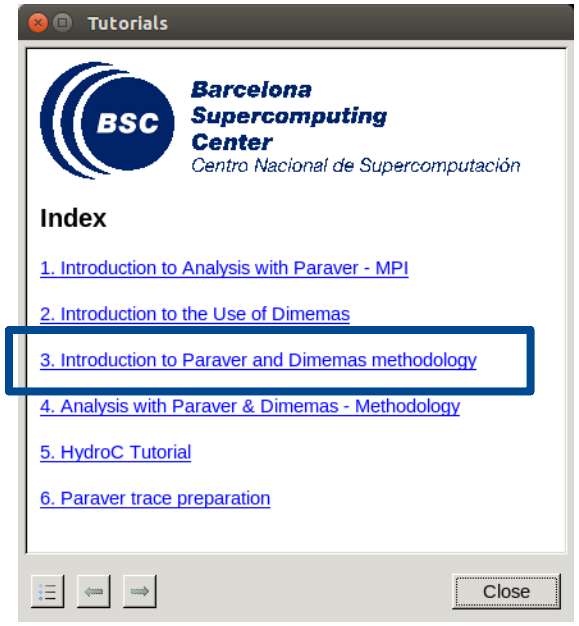
Click on File → Load Trace → Browse to “lulesh_27p.prv”



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



Click on Help → Tutorials

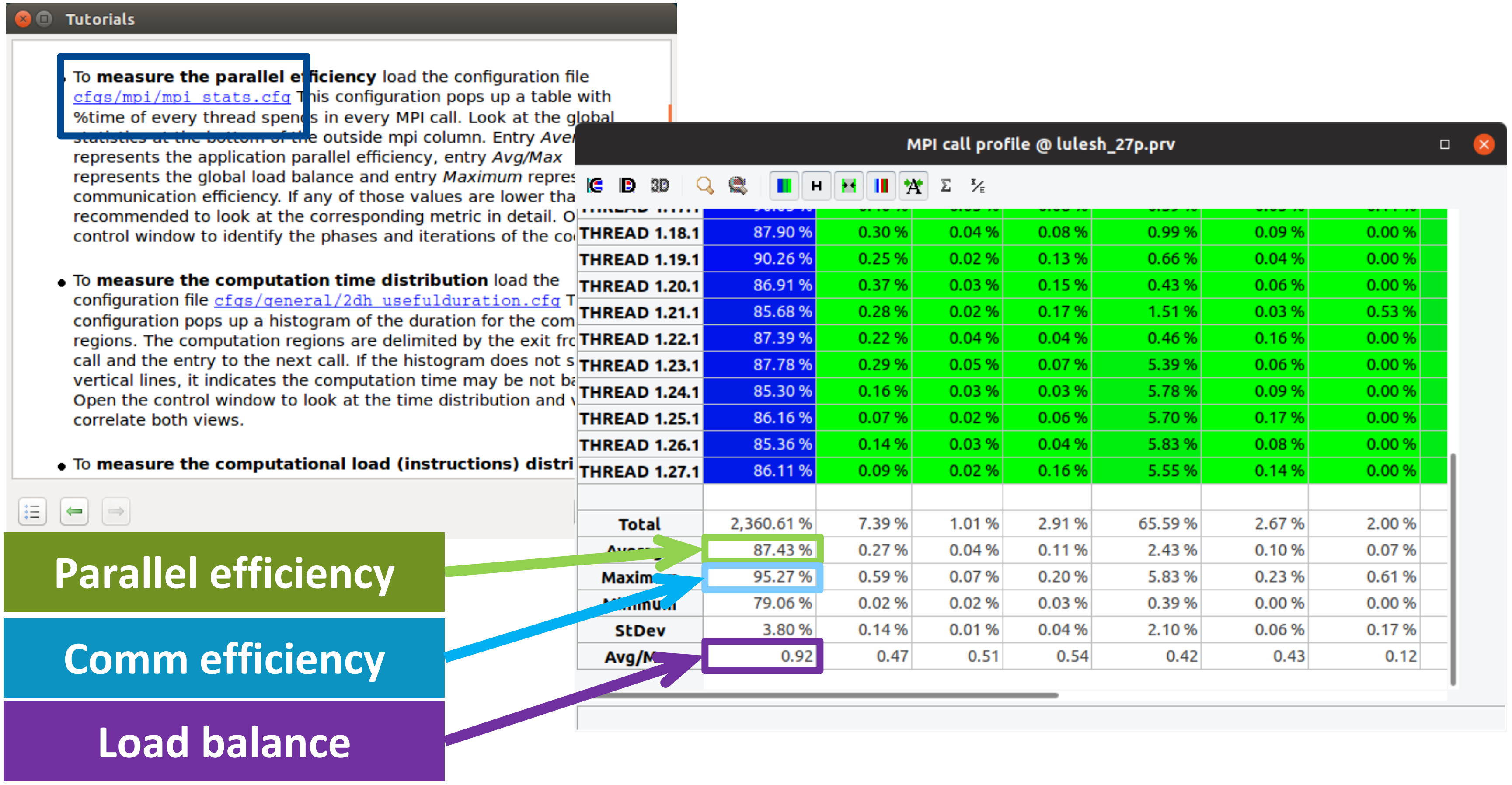

Tutorials
BSC
Barcelona Supercomputing Center
Centro Nacional de Supercomputación
Close

Index

1. Introduction to Analysis with Paraver - MPI
2. Introduction to the Use of Dimemas
3. Introduction to Paraver and Dimemas methodology
4. Analysis with Paraver & Dimemas - Methodology
5. HydroC Tutorial
6. Paraver trace preparation

Measure the parallel efficiency

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



- To **measure the parallel efficiency** load the configuration file cfgs/mpi/mpi_stats.cfg This configuration pops up a table with %time of every thread spends in every MPI call. Look at the global statistics at the bottom of the outside mpi column. Entry *Aver* represents the application parallel efficiency, entry *Avg/Max* represents the global load balance and entry *Maximum* repres communication efficiency. If any of those values are lower tha recommended to look at the corresponding metric in detail. O control window to identify the phases and iterations of the co

- To **measure the computation time distribution** load the configuration file cfgs/general/2dh_usefulduration.cfg T configuration pops up a histogram of the duration for the com regions. The computation regions are delimited by the exit fro call and the entry to the next call. If the histogram does not s vertical lines, it indicates the computation time may be not ba Open the control window to look at the time distribution and v correlate both views.

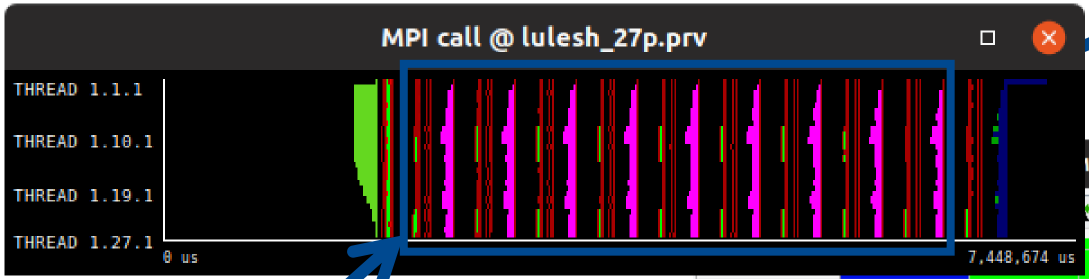
- To **measure the computational load (instructions) distri**

THREAD 1.18.1	87.90 %	0.30 %	0.04 %	0.08 %	0.99 %	0.09 %	0.00 %
THREAD 1.19.1	90.26 %	0.25 %	0.02 %	0.13 %	0.66 %	0.04 %	0.00 %
THREAD 1.20.1	86.91 %	0.37 %	0.03 %	0.15 %	0.43 %	0.06 %	0.00 %
THREAD 1.21.1	85.68 %	0.28 %	0.02 %	0.17 %	1.51 %	0.03 %	0.53 %
THREAD 1.22.1	87.39 %	0.22 %	0.04 %	0.04 %	0.46 %	0.16 %	0.00 %
THREAD 1.23.1	87.78 %	0.29 %	0.05 %	0.07 %	5.39 %	0.06 %	0.00 %
THREAD 1.24.1	85.30 %	0.16 %	0.03 %	0.03 %	5.78 %	0.09 %	0.00 %
THREAD 1.25.1	86.16 %	0.07 %	0.02 %	0.06 %	5.70 %	0.17 %	0.00 %
THREAD 1.26.1	85.36 %	0.14 %	0.03 %	0.04 %	5.83 %	0.08 %	0.00 %
THREAD 1.27.1	86.11 %	0.09 %	0.02 %	0.16 %	5.55 %	0.14 %	0.00 %
Total	2,360.61 %	7.39 %	1.01 %	2.91 %	65.59 %	2.67 %	2.00 %
Avera	87.43 %	0.27 %	0.04 %	0.11 %	2.43 %	0.10 %	0.07 %
Maxim	95.27 %	0.59 %	0.07 %	0.20 %	5.83 %	0.23 %	0.61 %
Minimu	79.06 %	0.02 %	0.02 %	0.03 %	0.39 %	0.00 %	0.00 %
StDev	3.80 %	0.14 %	0.01 %	0.04 %	2.10 %	0.06 %	0.17 %
Avg/M	0.92	0.47	0.51	0.54	0.42	0.43	0.12

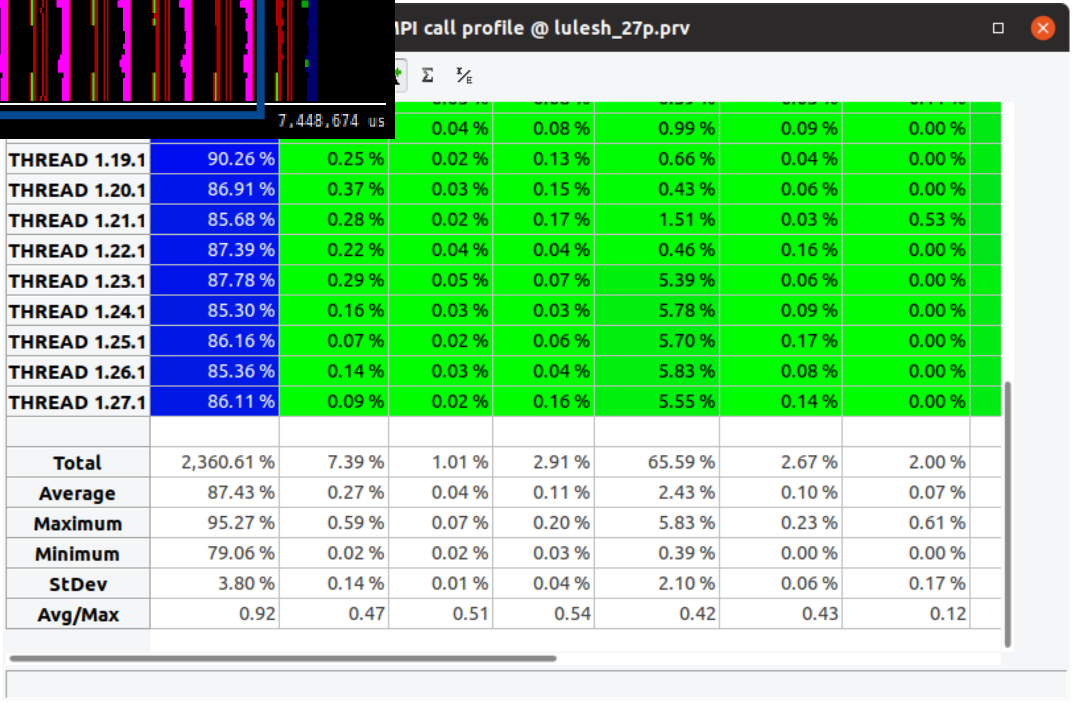
Measure the parallel efficiency

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

Click on “Open Control Window”



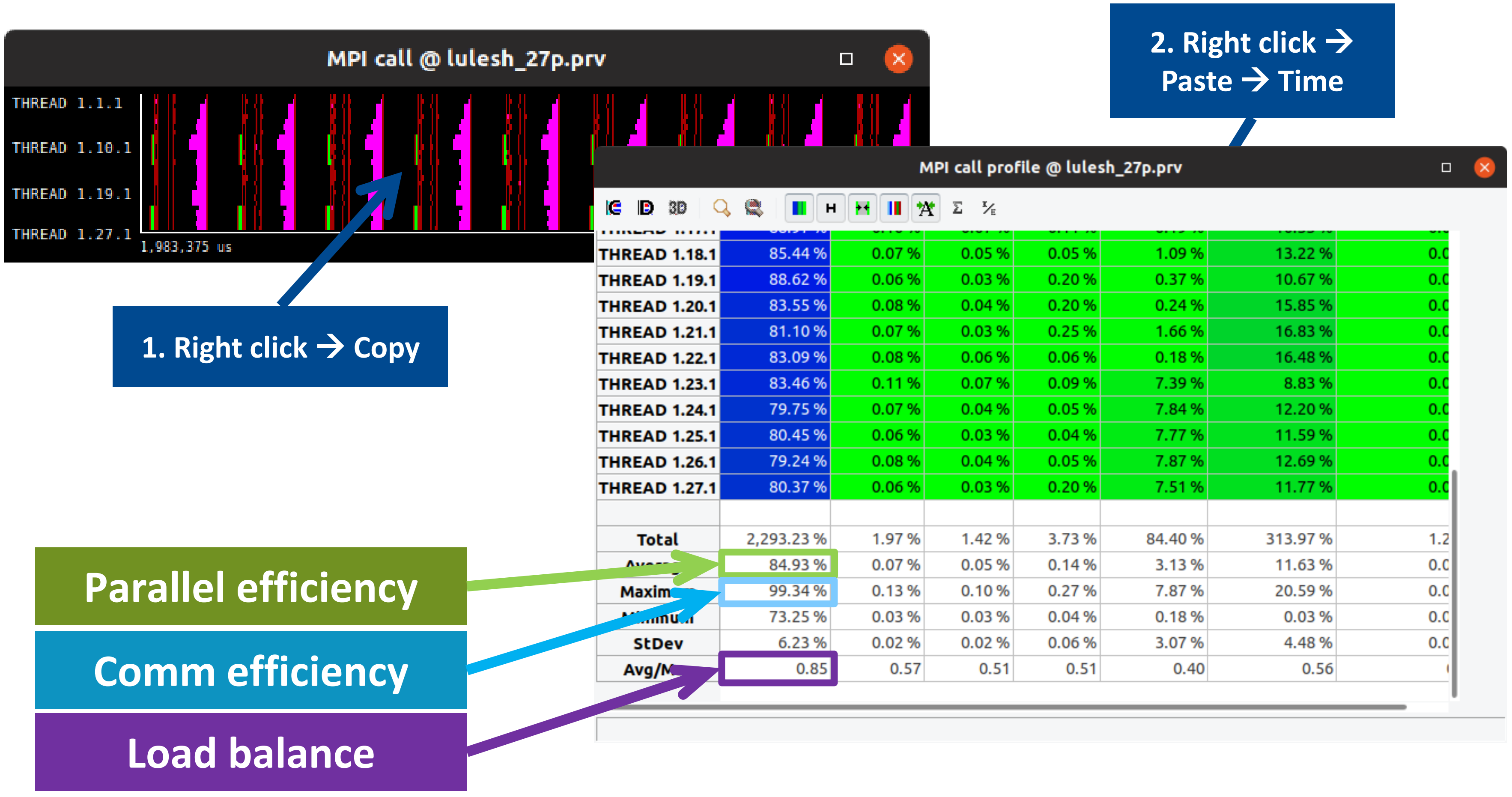
Zoom on the iterative región (drag-and-drop)



			0.04 %	0.08 %	0.99 %	0.09 %	0.00 %
THREAD 1.19.1	90.26 %	0.25 %	0.02 %	0.13 %	0.66 %	0.04 %	0.00 %
THREAD 1.20.1	86.91 %	0.37 %	0.03 %	0.15 %	0.43 %	0.06 %	0.00 %
THREAD 1.21.1	85.68 %	0.28 %	0.02 %	0.17 %	1.51 %	0.03 %	0.53 %
THREAD 1.22.1	87.39 %	0.22 %	0.04 %	0.04 %	0.46 %	0.16 %	0.00 %
THREAD 1.23.1	87.78 %	0.29 %	0.05 %	0.07 %	5.39 %	0.06 %	0.00 %
THREAD 1.24.1	85.30 %	0.16 %	0.03 %	0.03 %	5.78 %	0.09 %	0.00 %
THREAD 1.25.1	86.16 %	0.07 %	0.02 %	0.06 %	5.70 %	0.17 %	0.00 %
THREAD 1.26.1	85.36 %	0.14 %	0.03 %	0.04 %	5.83 %	0.08 %	0.00 %
THREAD 1.27.1	86.11 %	0.09 %	0.02 %	0.16 %	5.55 %	0.14 %	0.00 %
Total	2,360.61 %	7.39 %	1.01 %	2.91 %	65.59 %	2.67 %	2.00 %
Average	87.43 %	0.27 %	0.04 %	0.11 %	2.43 %	0.10 %	0.07 %
Maximum	95.27 %	0.59 %	0.07 %	0.20 %	5.83 %	0.23 %	0.61 %
Minimum	79.06 %	0.02 %	0.02 %	0.03 %	0.39 %	0.00 %	0.00 %
StDev	3.80 %	0.14 %	0.01 %	0.04 %	2.10 %	0.06 %	0.17 %
Avg/Max	0.92	0.47	0.51	0.54	0.42	0.43	0.12

Measure the parallel efficiency

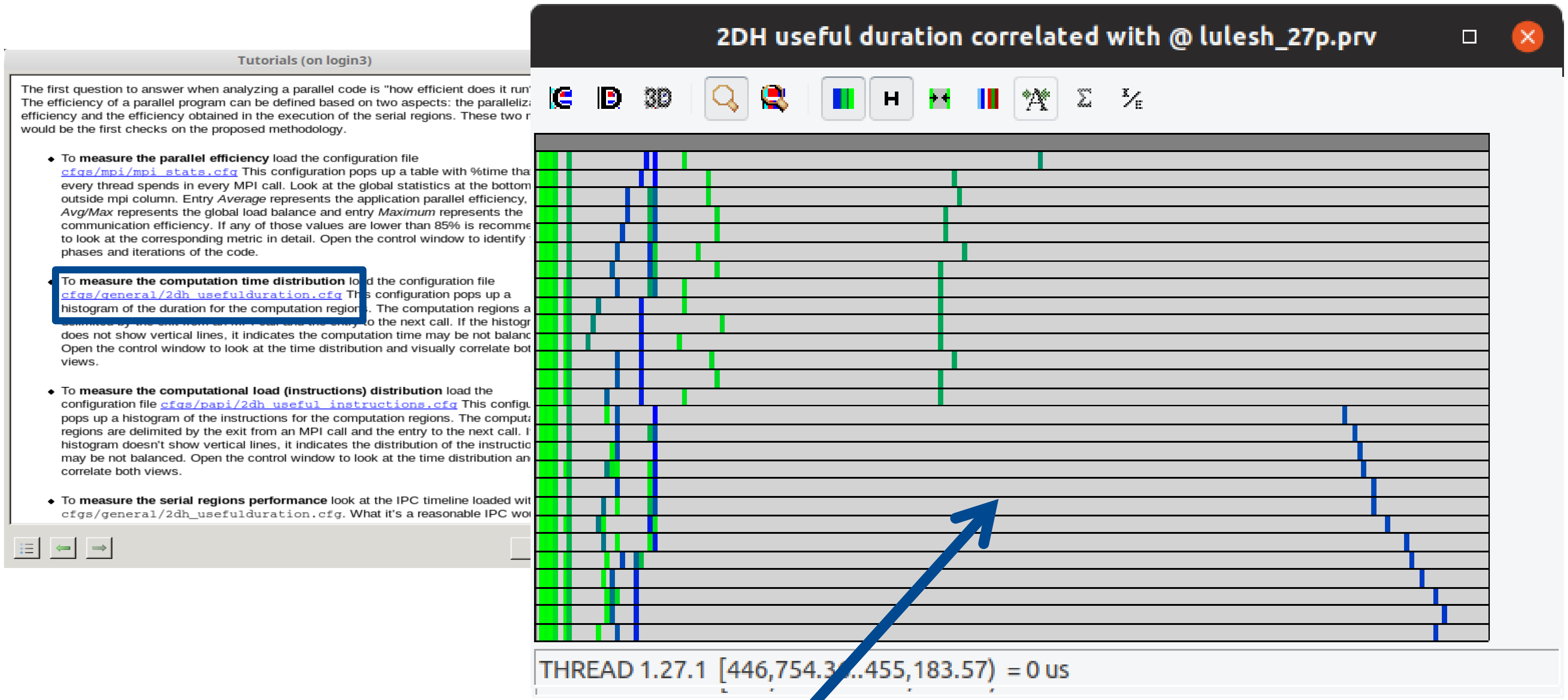
- Click on “mpi_stats.cfg”
  - Check the **Average** for the column labeled “**Outside MPI**”



THREAD 1.18.1	85.44 %	0.07 %	0.05 %	0.05 %	1.09 %	13.22 %	0.0
THREAD 1.19.1	88.62 %	0.06 %	0.03 %	0.20 %	0.37 %	10.67 %	0.0
THREAD 1.20.1	83.55 %	0.08 %	0.04 %	0.20 %	0.24 %	15.85 %	0.0
THREAD 1.21.1	81.10 %	0.07 %	0.03 %	0.25 %	1.66 %	16.83 %	0.0
THREAD 1.22.1	83.09 %	0.08 %	0.06 %	0.06 %	0.18 %	16.48 %	0.0
THREAD 1.23.1	83.46 %	0.11 %	0.07 %	0.09 %	7.39 %	8.83 %	0.0
THREAD 1.24.1	79.75 %	0.07 %	0.04 %	0.05 %	7.84 %	12.20 %	0.0
THREAD 1.25.1	80.45 %	0.06 %	0.03 %	0.04 %	7.77 %	11.59 %	0.0
THREAD 1.26.1	79.24 %	0.08 %	0.04 %	0.05 %	7.87 %	12.69 %	0.0
THREAD 1.27.1	80.37 %	0.06 %	0.03 %	0.20 %	7.51 %	11.77 %	0.0
Total	2,293.23 %	1.97 %	1.42 %	3.73 %	84.40 %	313.97 %	1.2
Avera	84.93 %	0.07 %	0.05 %	0.14 %	3.13 %	11.63 %	0.0
Maxim	99.34 %	0.13 %	0.10 %	0.27 %	7.87 %	20.59 %	0.0
inimu	73.25 %	0.03 %	0.03 %	0.04 %	0.18 %	0.03 %	0.0
StDev	6.23 %	0.02 %	0.02 %	0.06 %	3.07 %	4.48 %	0.0
Avg/M	0.85	0.57	0.51	0.51	0.40	0.56	

Computation load & time distribution

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



Tutorials (on login3)

The first question to answer when analyzing a parallel code is "how efficient does it run
The efficiency of a parallel program can be defined based on two aspects: the paralleliz
efficiency and the efficiency obtained in the execution of the serial regions. These two r
would be the first checks on the proposed methodology.

- To **measure the parallel efficiency** load the configuration file cfgs/mpi/mpi_stats.cfg This configuration pops up a table with %time tha every thread spends in every MPI call. Look at the global statistics at the bottom outside mpi column. Entry *Average* represents the application parallel efficiency, *Avg/Max* represents the global load balance and entry *Maximum* represents the communication efficiency. If any of those values are lower than 85% is recomme to look at the corresponding metric in detail. Open the control window to identify phases and iterations of the code.
- To **measure the computation time distribution** load the configuration file cfgs/general/2dh_usefulduration.cfg This configuration pops up a histogram of the duration for the computation regions. The computation regions a delimited by the exit from an MPI call and the entry to the next call. If the histogr does not show vertical lines, it indicates the computation time may be not balanc Open the control window to look at the time distribution and visually correlate bot views.
- To **measure the computational load (instructions) distribution** load the configuration file cfgs/papi/2dh_useful_instructions.cfg This configu pops up a histogram of the instructions for the computation regions. The computa regions are delimited by the exit from an MPI call and the entry to the next call. I histogram doesn't show vertical lines, it indicates the distribution of the instructio may be not balanced. Open the control window to look at the time distribution an correlate both views.
- To **measure the serial regions performance** look at the IPC timeline loaded wit cfgs/general/2dh_usefulduration.cfg. What it's a reasonable IPC wo

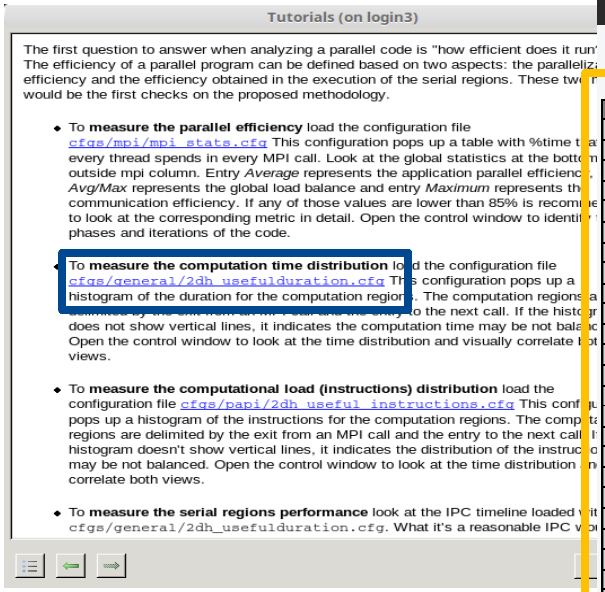
2DH useful duration correlated with @ lulesh_27p.prv

THREAD 1.27.1 [446,754.3...455,183.57) = 0 us

Right click → Paste → Time

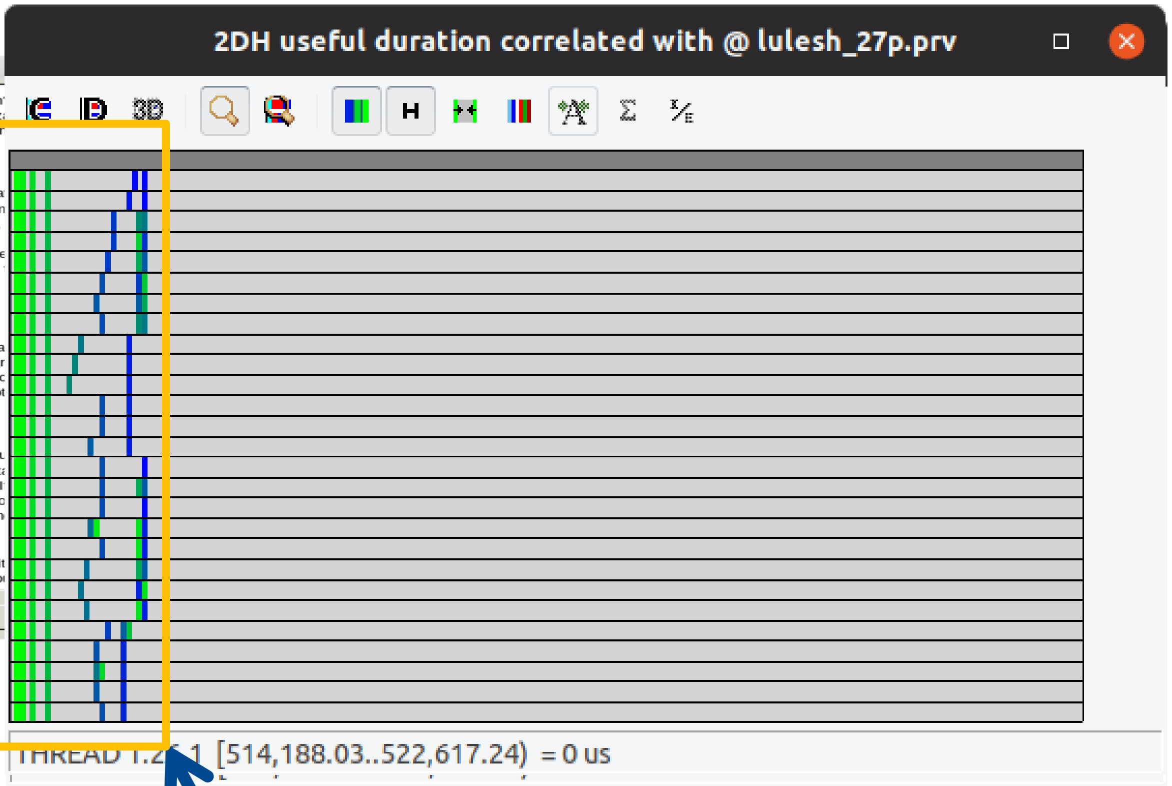
Computation load & time distribution

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



The first question to answer when analyzing a parallel code is "how efficient does it run
The efficiency of a parallel program can be defined based on two aspects: the paralleliz
efficiency and the efficiency obtained in the execution of the serial regions. These two r
would be the first checks on the proposed methodology.

- To **measure the parallel efficiency** load the configuration file cfgs/mpi/mpi_stats.cfg This configuration pops up a table with %time t every thread spends in every MPI call. Look at the global statistics at the botto outside mpi column. Entry *Average* represents the application parallel efficienc *Avg/Max* represents the global load balance and entry *Maximum* represents th communication efficiency. If any of those values are lower than 85% is recomme to look at the corresponding metric in detail. Open the control window to identif phases and iterations of the code.
- To **measure the computation time distribution** load the configuration file cfgs/general/2dh_usefulduration.cfg This configuration pops up a histogram of the duration for the computation regions. The computation regions a delimited by the exit from an MPI call and the entry to the next call. If the histogr does not show vertical lines, it indicates the computation time may be not balanc Open the control window to look at the time distribution and visually correlate bot views.
- To **measure the computational load (instructions) distribution** load the configuration file cfgs/papi/2dh_useful_instructions.cfg This configu pops up a histogram of the instructions for the computation regions. The compta regions are delimited by the exit from an MPI call and the entry to the next call. I histogram doesn't show vertical lines, it indicates the distribution of the instructio may be not balanced. Open the control window to look at the time distribution an correlate both views.
- To **measure the serial regions performance** look at the IPC timeline loaded wit cfgs/general/2dh_usefulduration.cfg. What it's a reasonable IPC wo

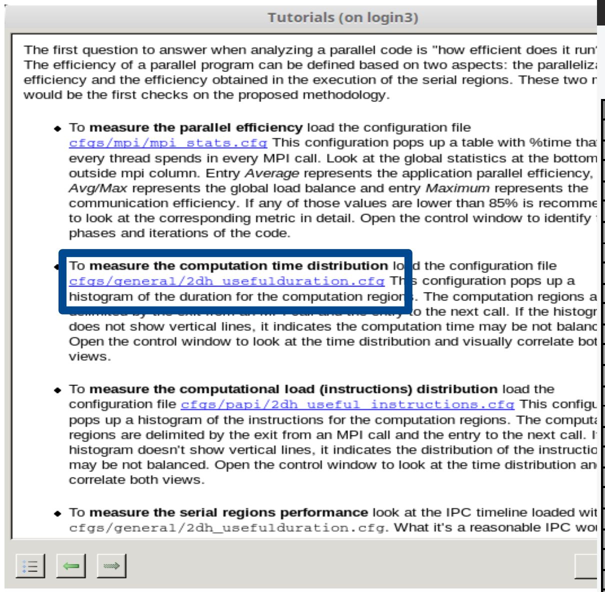


THREAD 1.2 1 [514,188.03..522,617.24) = 0 us

Zoom on data (drag-and-drop)

Computation load & time distribution

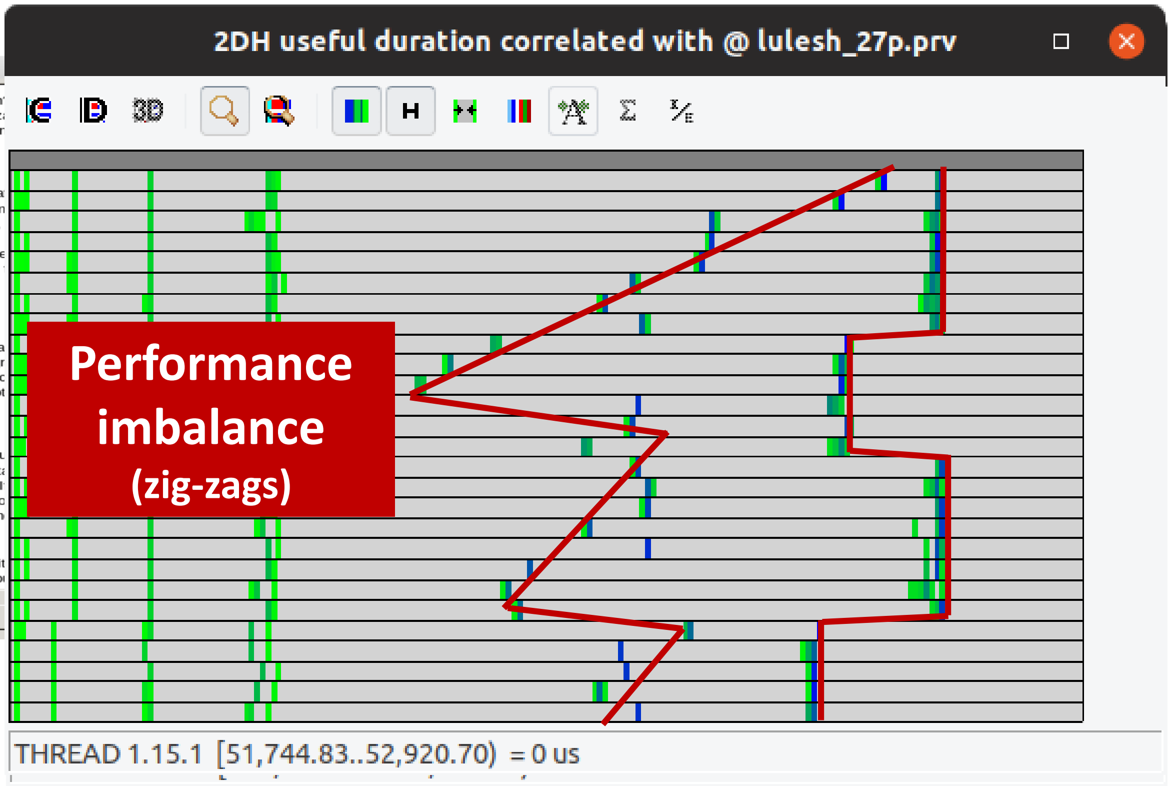
- Click on “2dh_usefulduration.cfg” (2nd link) ➔ Shows **time computing**



Tutorials (on login3)

The first question to answer when analyzing a parallel code is "how efficient does it run
The efficiency of a parallel program can be defined based on two aspects: the paralleliz
efficiency and the efficiency obtained in the execution of the serial regions. These two r
would be the first checks on the proposed methodology.

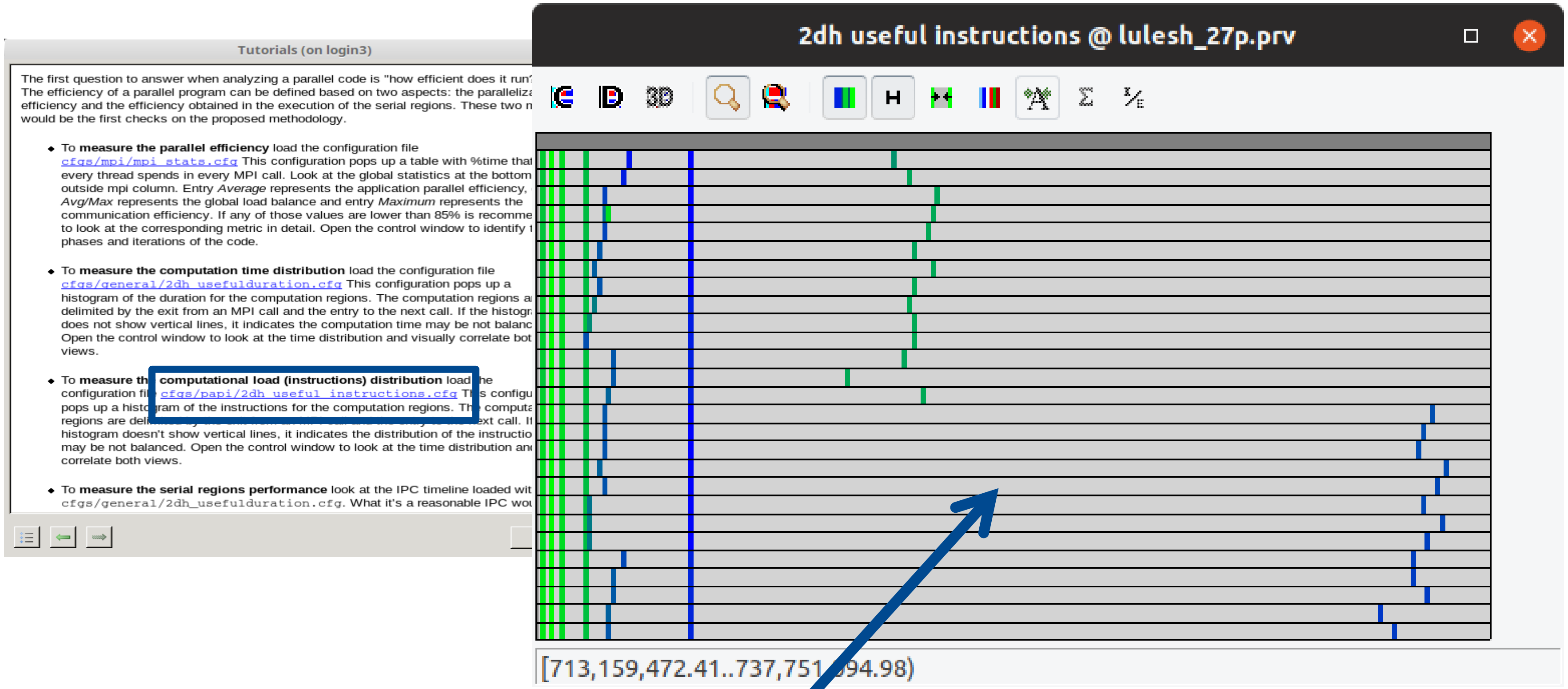
- To **measure the parallel efficiency** load the configuration file cfgs/mpi/mpi_stats.cfg This configuration pops up a table with %time tha every thread spends in every MPI call. Look at the global statistics at the bottom outside mpi column. Entry *Average* represents the application parallel efficiency, *Avg/Max* represents the global load balance and entry *Maximum* represents the communication efficiency. If any of those values are lower than 85% is recomme to look at the corresponding metric in detail. Open the control window to identify phases and iterations of the code.
- To **measure the computation time distribution** load the configuration file cfgs/general/2dh_usefulduration.cfg This configuration pops up a histogram of the duration for the computation regions. The computation regions a delimited by the exit from an MPI call and the entry to the next call. If the histogr does not show vertical lines, it indicates the computation time may be not balanc Open the control window to look at the time distribution and visually correlate bot views.
- To **measure the computational load (instructions) distribution** load the configuration file cfgs/papi/2dh_useful_instructions.cfg This configu pops up a histogram of the instructions for the computation regions. The computa regions are delimited by the exit from an MPI call and the entry to the next call. I histogram doesn't show vertical lines, it indicates the distribution of the instructio may be not balanced. Open the control window to look at the time distribution an correlate both views.
- To **measure the serial regions performance** look at the IPC timeline loaded wit cfgs/general/2dh_usefulduration.cfg. What it's a reasonable IPC wo


2DH useful duration correlated with @ lulesh_27p.prv
Performance imbalance (zig-zags)

```
THREAD 1.15.1  [51,744.83..52,920.70)  = 0 us
```

Computation load & time distribution

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



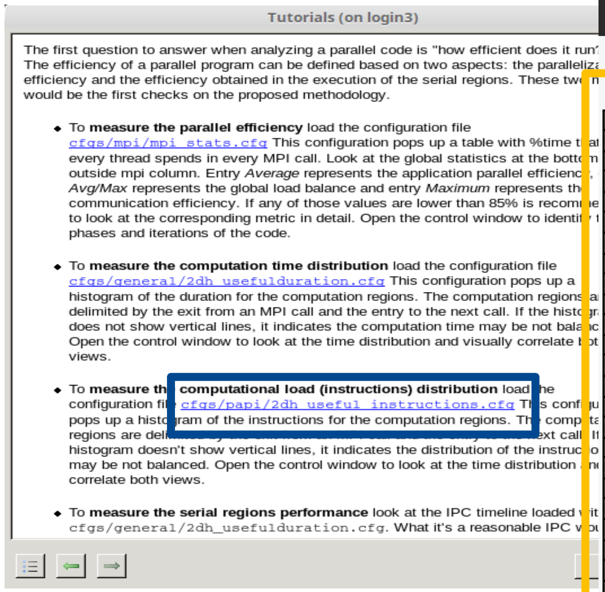
The first question to answer when analyzing a parallel code is "how efficient does it run
The efficiency of a parallel program can be defined based on two aspects: the paralleliza
efficiency and the efficiency obtained in the execution of the serial regions. These two m
would be the first checks on the proposed methodology.

- To **measure the parallel efficiency** load the configuration file cfgs/mpi/mpi_stats.cfg This configuration pops up a table with %time that every thread spends in every MPI call. Look at the global statistics at the bottom outside mpi column. Entry *Average* represents the application parallel efficiency, *Avg/Max* represents the global load balance and entry *Maximum* represents the communication efficiency. If any of those values are lower than 85% is recomme to look at the corresponding metric in detail. Open the control window to identify t phases and iterations of the code.
- To **measure the computation time distribution** load the configuration file cfgs/general/2dh_usefulduration.cfg This configuration pops up a histogram of the duration for the computation regions. The computation regions a delimited by the exit from an MPI call and the entry to the next call. If the histogr does not show vertical lines, it indicates the computation time may be not balanc Open the control window to look at the time distribution and visually correlate bot views.
- To **measure th computational load (instructions) distribution** load he configuration fil cfgs/papi/2dh_useful_instructions.cfg Th s configu pops up a histo gram of the instructions for the computation regions. Th computa regions are deli [illegible] ext call. If histogram doesn't show vertical lines, it indicates the distribution of the instructio may be not balanced. Open the control window to look at the time distribution an correlate both views.
- To **measure the serial regions performance** look at the IPC timeline loaded wit cfgs/general/2dh_usefulduration.cfg. What it's a reasonable IPC wou

Right click → Paste → Time

Computation load & time distribution

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



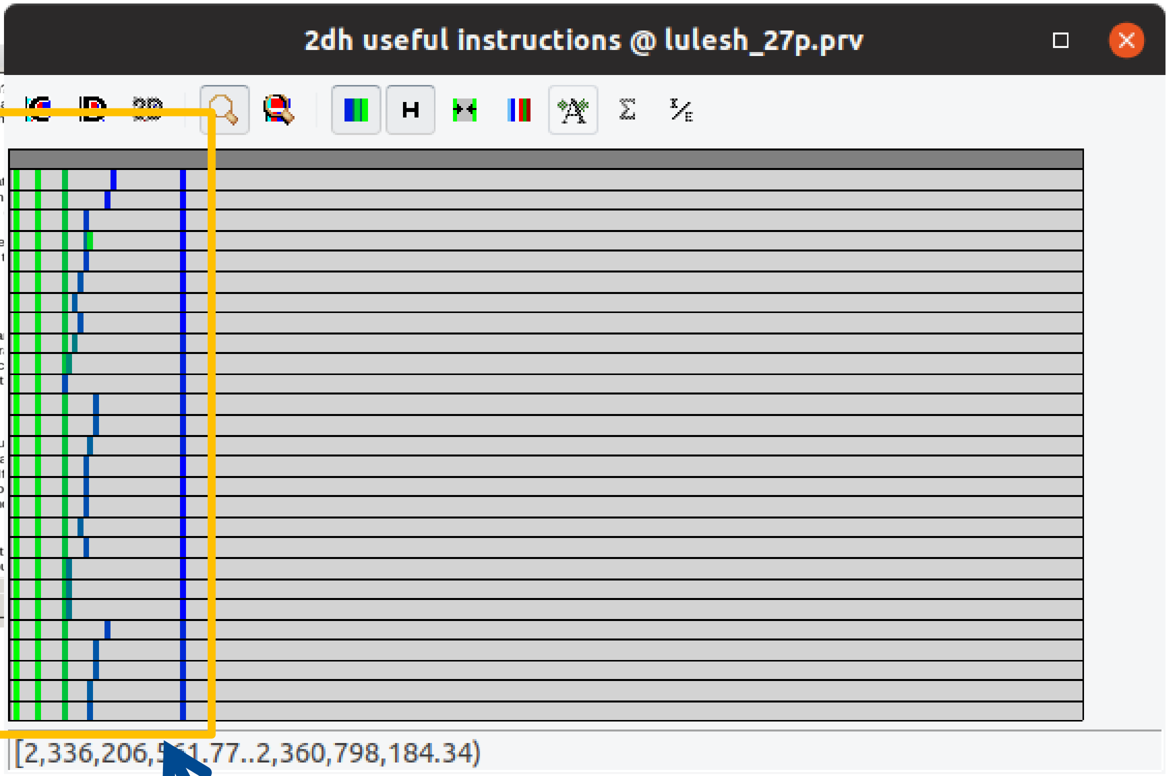
The first question to answer when analyzing a parallel code is "how efficient does it run
The efficiency of a parallel program can be defined based on two aspects: the paralleliz
efficiency and the efficiency obtained in the execution of the serial regions. These tw
would be the first checks on the proposed methodology.

- To **measure the parallel efficiency** load the configuration file cfgs/mpi/mpi_stats.cfg This configuration pops up a table with %time t
every thread spends in every MPI call. Look at the global statistics at the botto
outside mpi column. Entry *Average* represents the application parallel efficienc
Avg/Max represents the global load balance and entry *Maximum* represents th
communication efficiency. If any of those values are lower than 85% is recomm
to look at the corresponding metric in detail. Open the control window to identif
phases and iterations of the code.

- To **measure the computation time distribution** load the configuration file cfgs/general/2dh_usefulduration.cfg This configuration pops up a
histogram of the duration for the computation regions. The computation regions
delimited by the exit from an MPI call and the entry to the next call. If the histo
does not show vertical lines, it indicates the computation time may be not bala
Open the control window to look at the time distribution and visually correlate
views.

- To **measure the computational load (instructions) distribution** load the
configuration file cfgs/papi/2dh_useful_instructions.cfg This configu
pops up a histogram of the instructions for the computation regions. The compu
regions are deli ext call. If
histogram doesn't show vertical lines, it indicates the distribution of the instruc
may be not balanced. Open the control window to look at the time distribution
correlate both views.

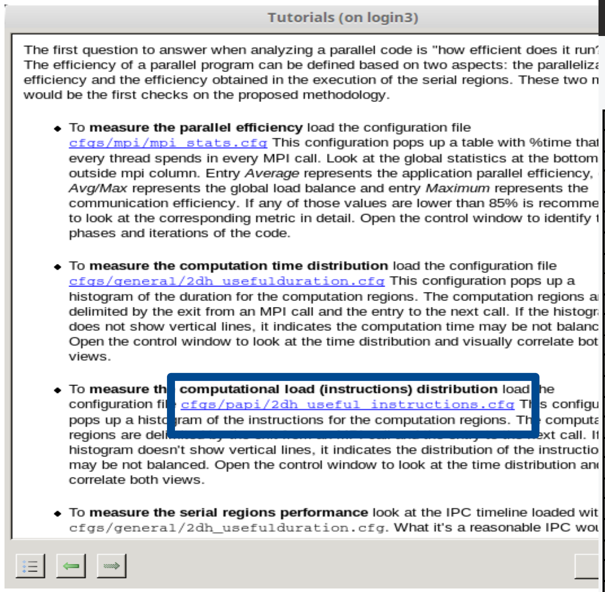
- To **measure the serial regions performance** look at the IPC timeline loaded
`cfgs/general/2dh_usefulduration.cfg`. What it's a reasonable IPC



Zoom on data (drag-and-drop)

Computation load & time distribution

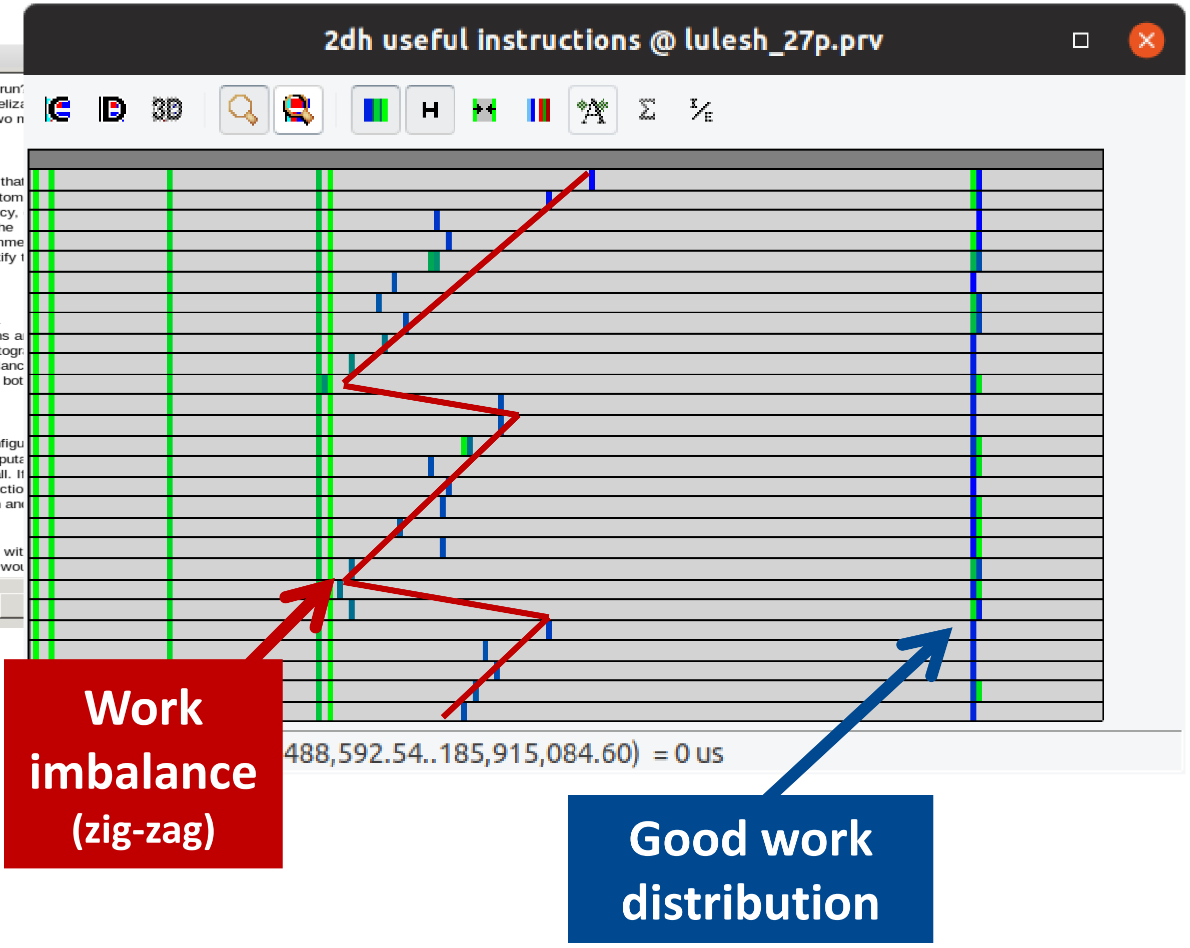
- Click on “2dh_useful_instructions.cfg” (3rd link) ➔ Shows **amount of work**



Tutorials (on login3)

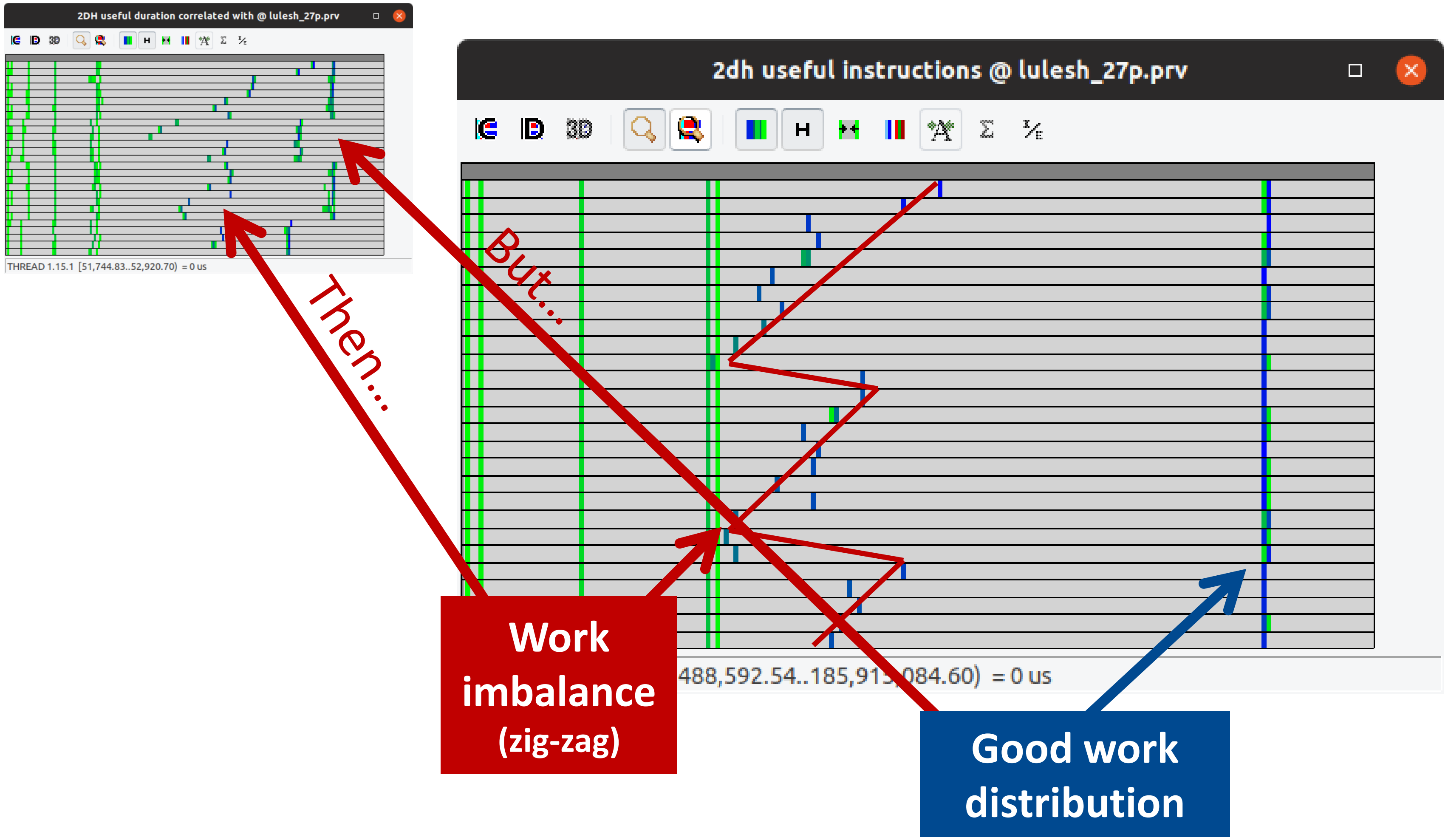
The first question to answer when analyzing a parallel code is "how efficient does it run
The efficiency of a parallel program can be defined based on two aspects: the paralleliza
efficiency and the efficiency obtained in the execution of the serial regions. These two m
would be the first checks on the proposed methodology.

- To **measure the parallel efficiency** load the configuration file cfgs/mpi/mpi_stats.cfg This configuration pops up a table with %time that every thread spends in every MPI call. Look at the global statistics at the bottom outside mpi column. Entry *Average* represents the application parallel efficiency, *Avg/Max* represents the global load balance and entry *Maximum* represents the communication efficiency. If any of those values are lower than 85% is recomme to look at the corresponding metric in detail. Open the control window to identify t phases and iterations of the code.
- To **measure the computation time distribution** load the configuration file cfgs/general/2dh_usefulduration.cfg This configuration pops up a histogram of the duration for the computation regions. The computation regions a delimited by the exit from an MPI call and the entry to the next call. If the histogr does not show vertical lines, it indicates the computation time may be not balanc Open the control window to look at the time distribution and visually correlate bot views.
- To **measure th computational load (instructions) distribution** load he configuration fil cfgs/papi/2dh_useful_instructions.cfg Th s configu pops up a histo gram of the instructions for the computation regions. Th computa regions are deli[illegible]ext call. If histogram doesn't show vertical lines, it indicates the distribution of the instructio may be not balanced. Open the control window to look at the time distribution an correlate both views.
- To **measure the serial regions performance** look at the IPC timeline loaded wit cfgs/general/2dh_usefulduration.cfg. What it's a reasonable IPC wou


2dh useful instructions @ lulesh_27p.prv
488,592.54..185,915,084.60) = 0 us
Work imbalance (zig-zag)
Good work distribution

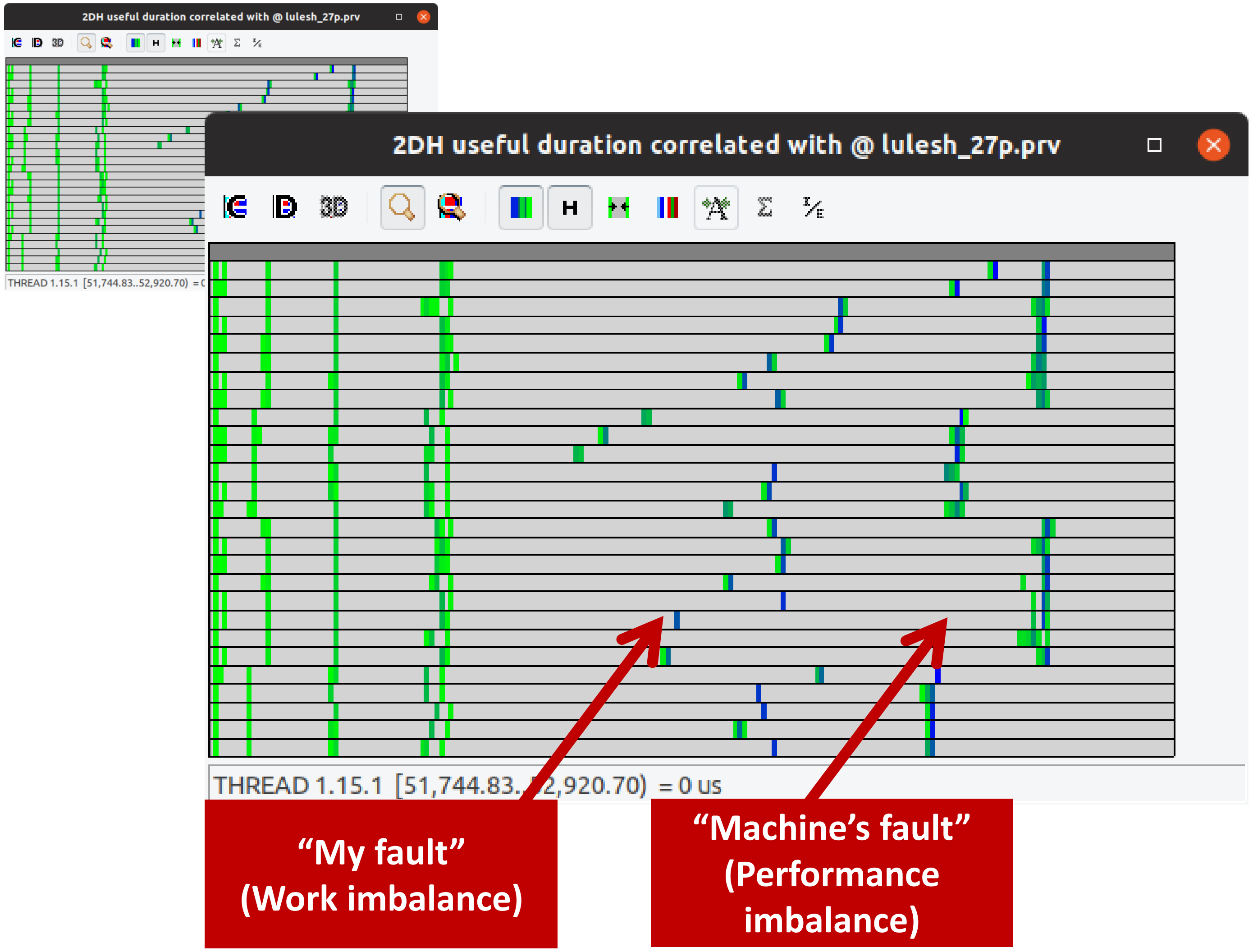
Computation load & time distribution

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

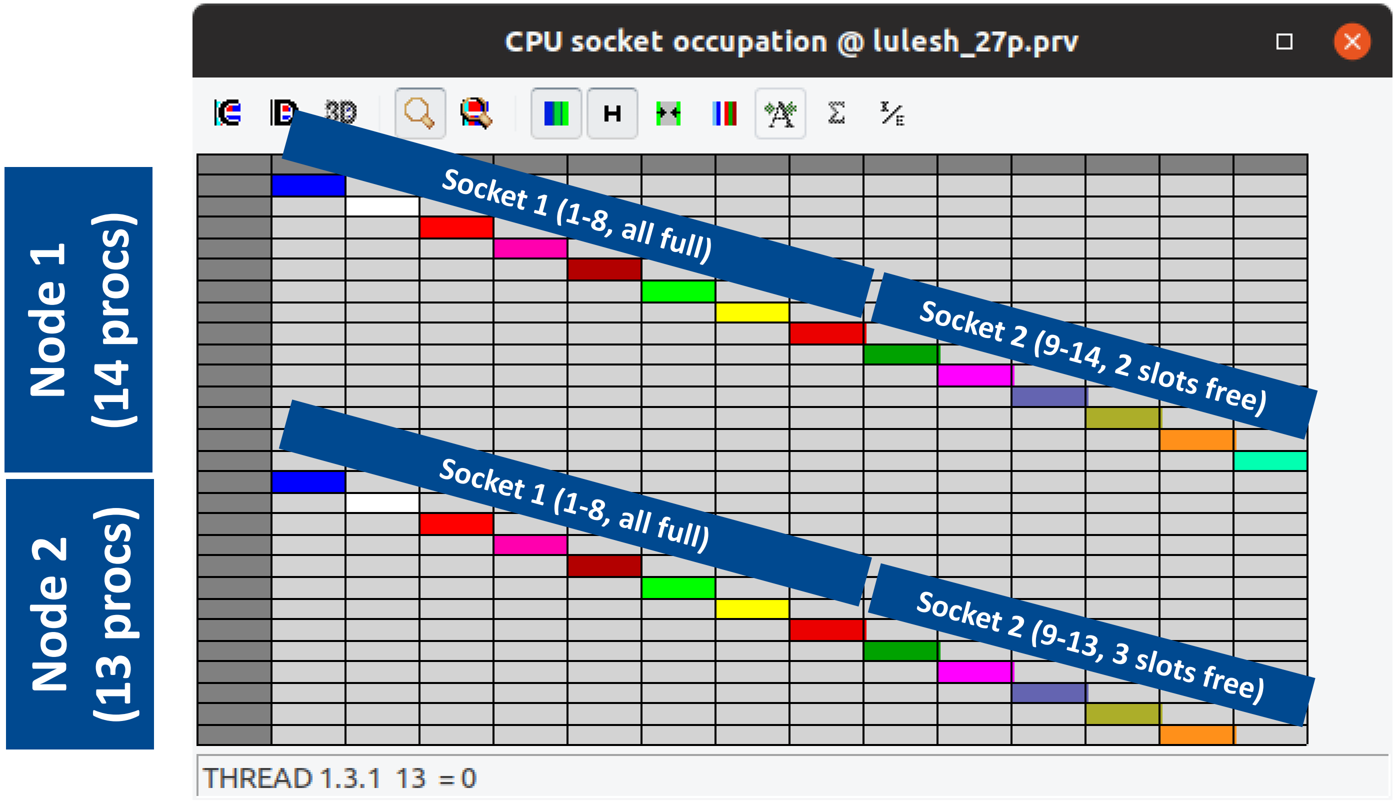

2DH useful duration correlated with @ lulesh_27p.prv
THREAD 1.15.1 [51,744.83..52,920.70) = 0 us
2dh useful instructions @ lulesh_27p.prv
488,592.54..185,913,084.60) = 0 us
Then...
But...
Work imbalance (zig-zag)
Good work distribution

Computation load & time distribution

- Unbalanced sockets impact performance

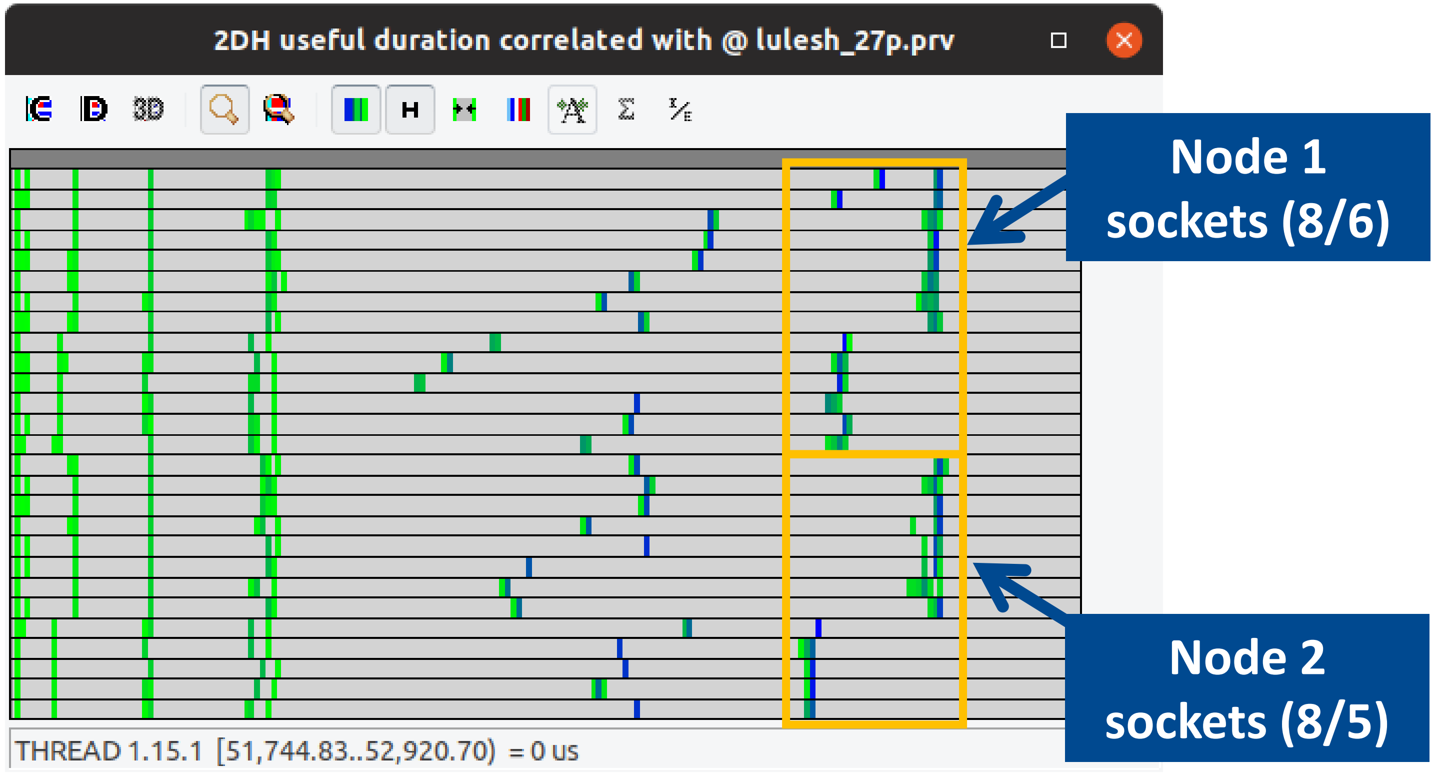

2DH useful duration correlated with @ lulesh_27p.prv
THREAD 1.15.1 [51,744.83..52,920.70) = 0 us
"My fault" (Work imbalance)
"Machine's fault" (Performance imbalance)

Check the process mapping



Computation load & time distribution

- Unbalanced sockets impact performance



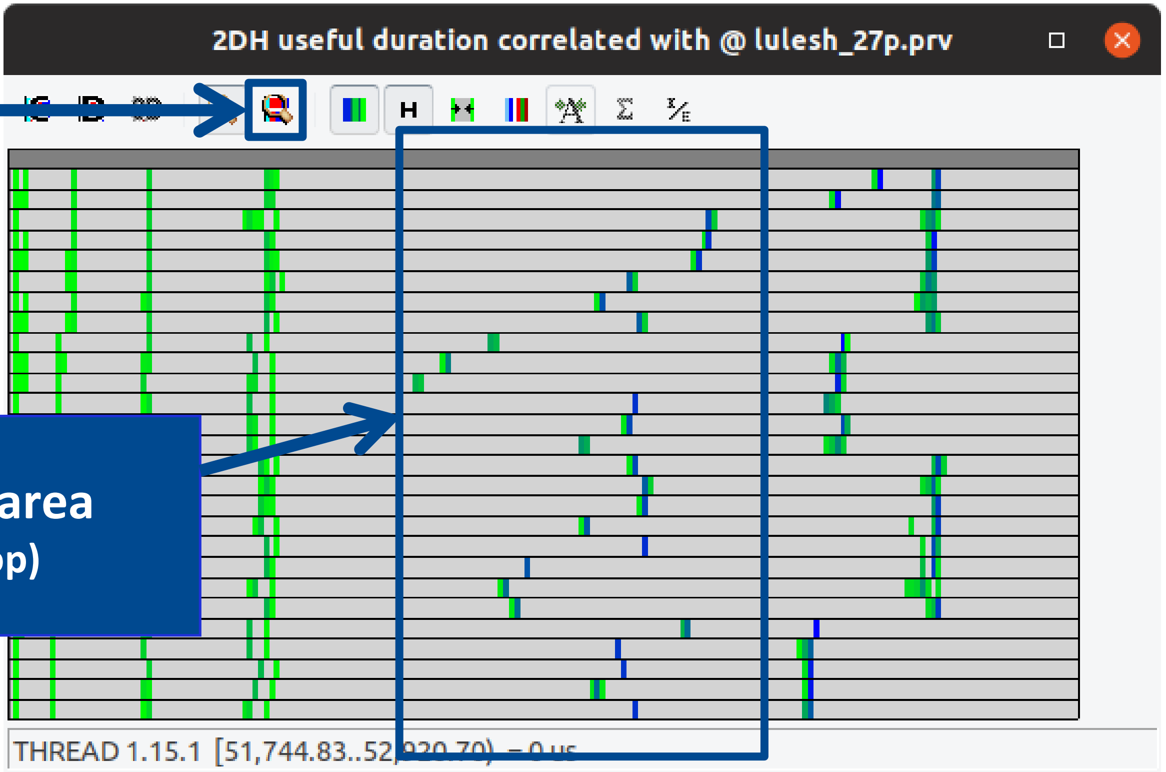
THREAD 1.15.1 [51,744.83..52,920.70) = 0 us

Where do things happen?

- Go from the table to the timeline

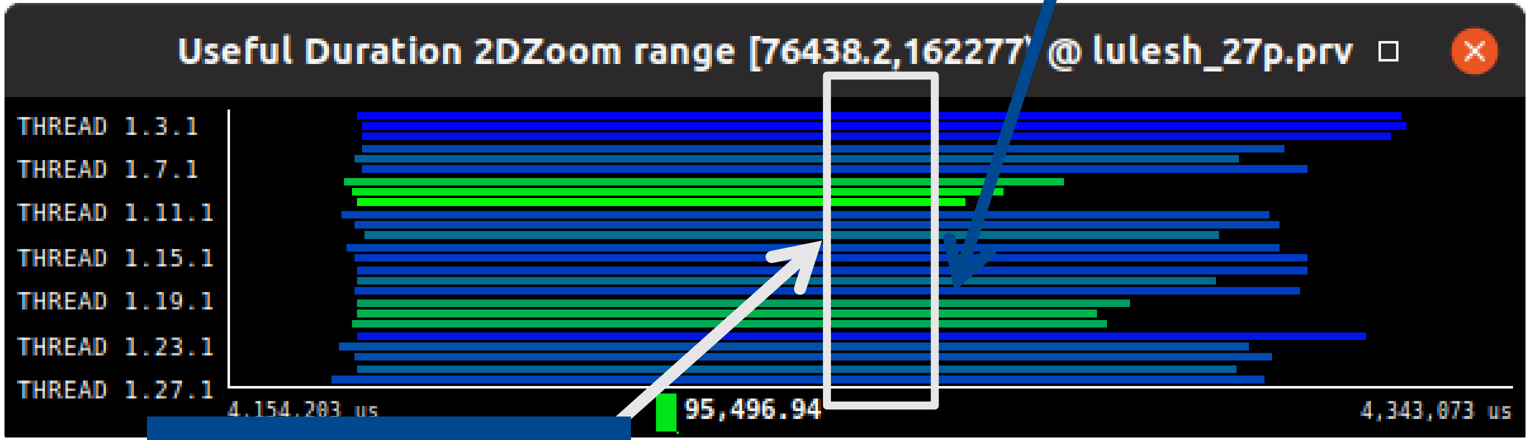
1. Click on "Open Filtered Control Window"

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


2DH useful duration correlated with @ lulesh_27p.prv
H
THREAD 1.15.1 [51,744.83..52,920.70) – 0 us

Where do things happen?

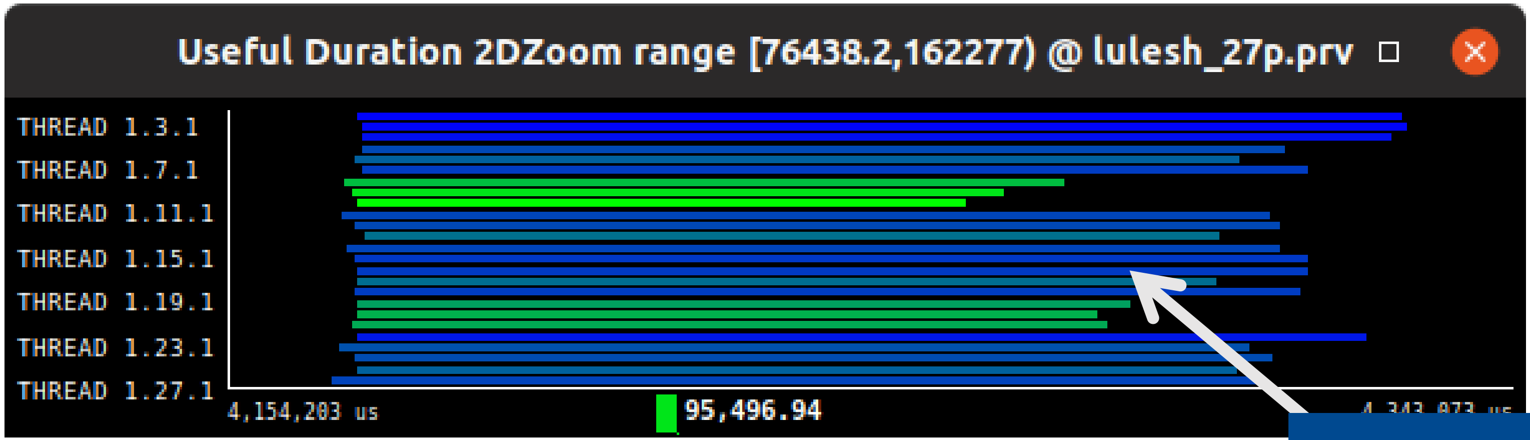
Right click → Fit Semantic Scale → Fit both



Zoom 1 iteration
(drag-and-drop)

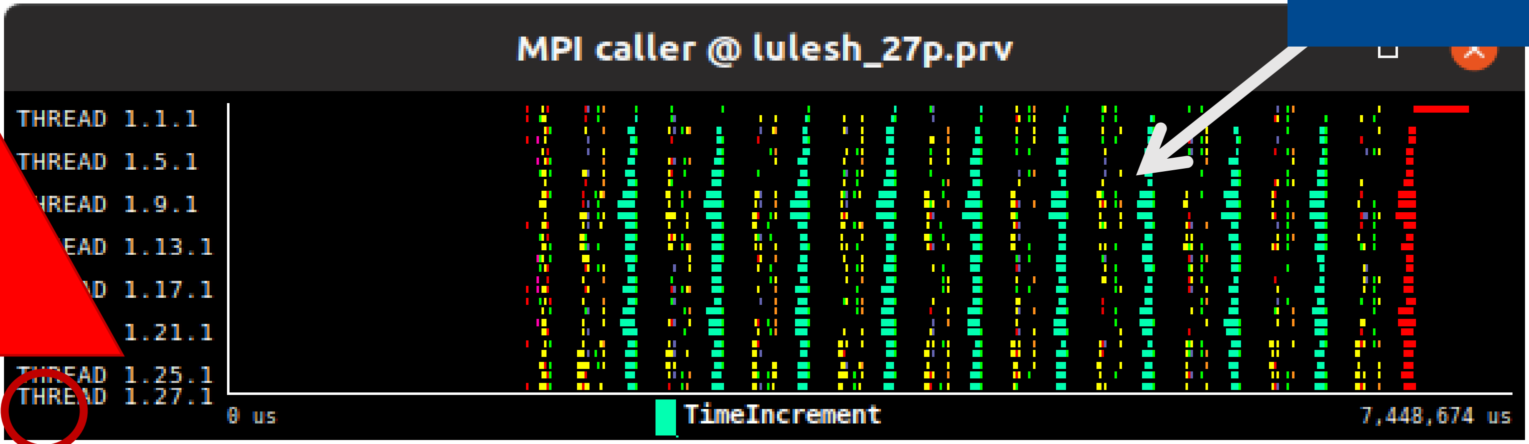
Where do things happen?

- Slow & Fast at the same time? ➔ Imbalance



- Hints → Callers → Caller function

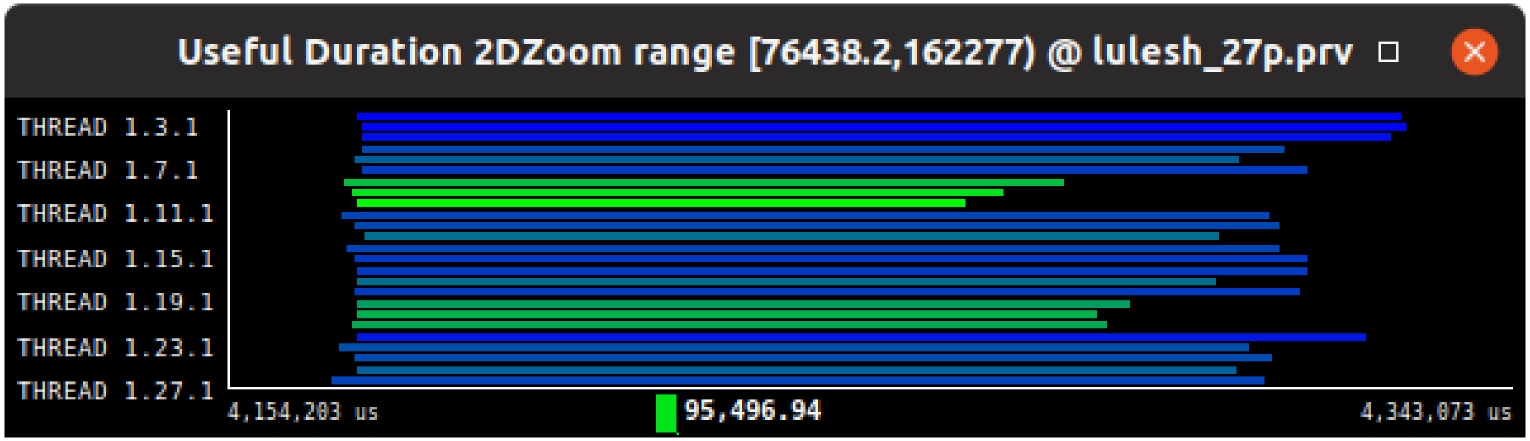
Copy & Paste → Time



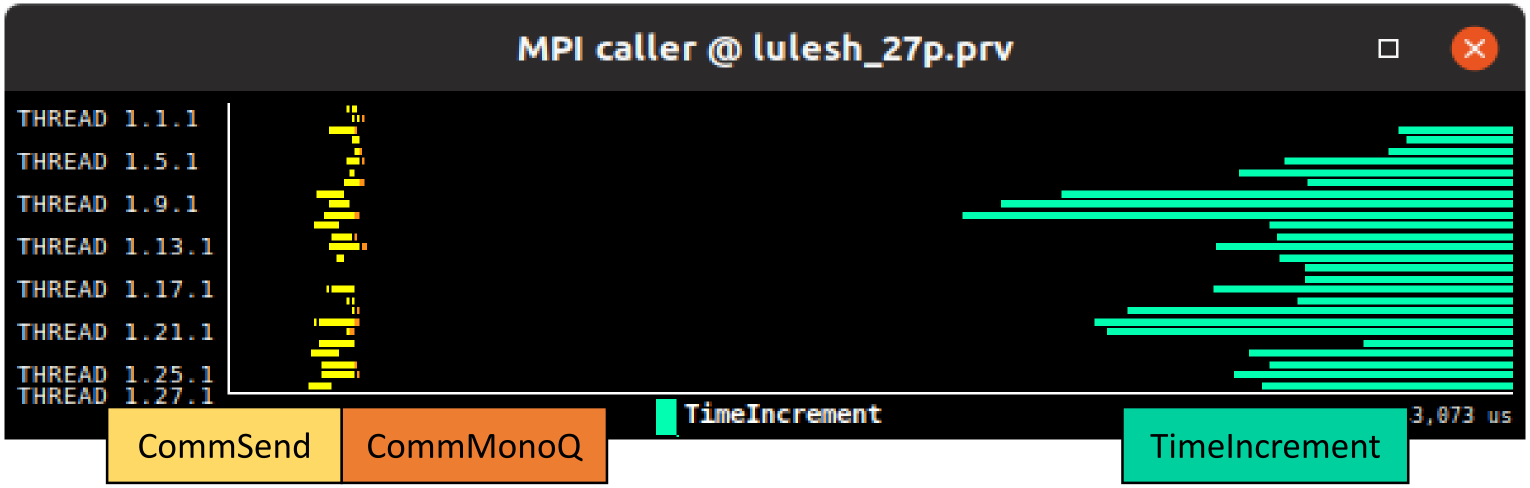
There's more values currently not shown!

Where do things happen?

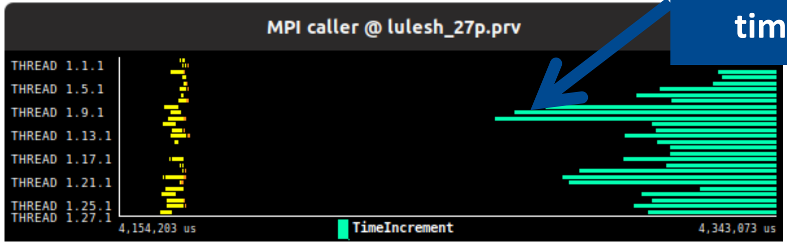
- Slow & Fast at the same time? → Imbalance



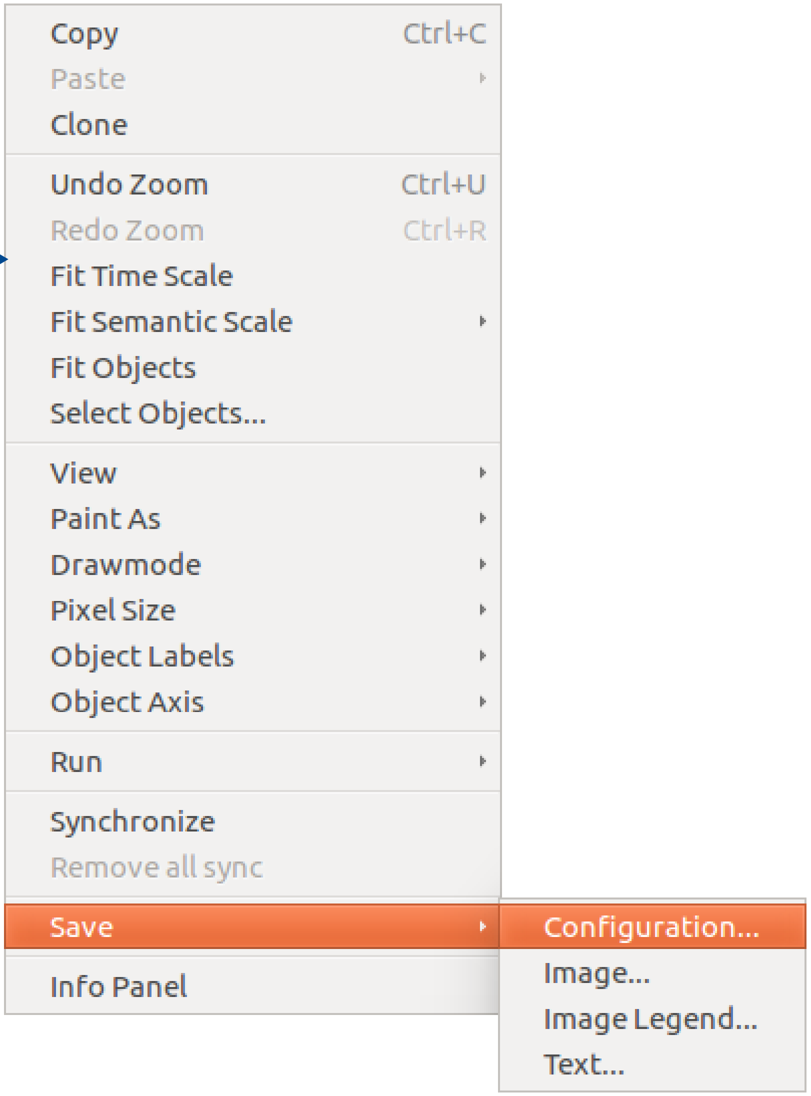
- Hints → Callers → Caller function



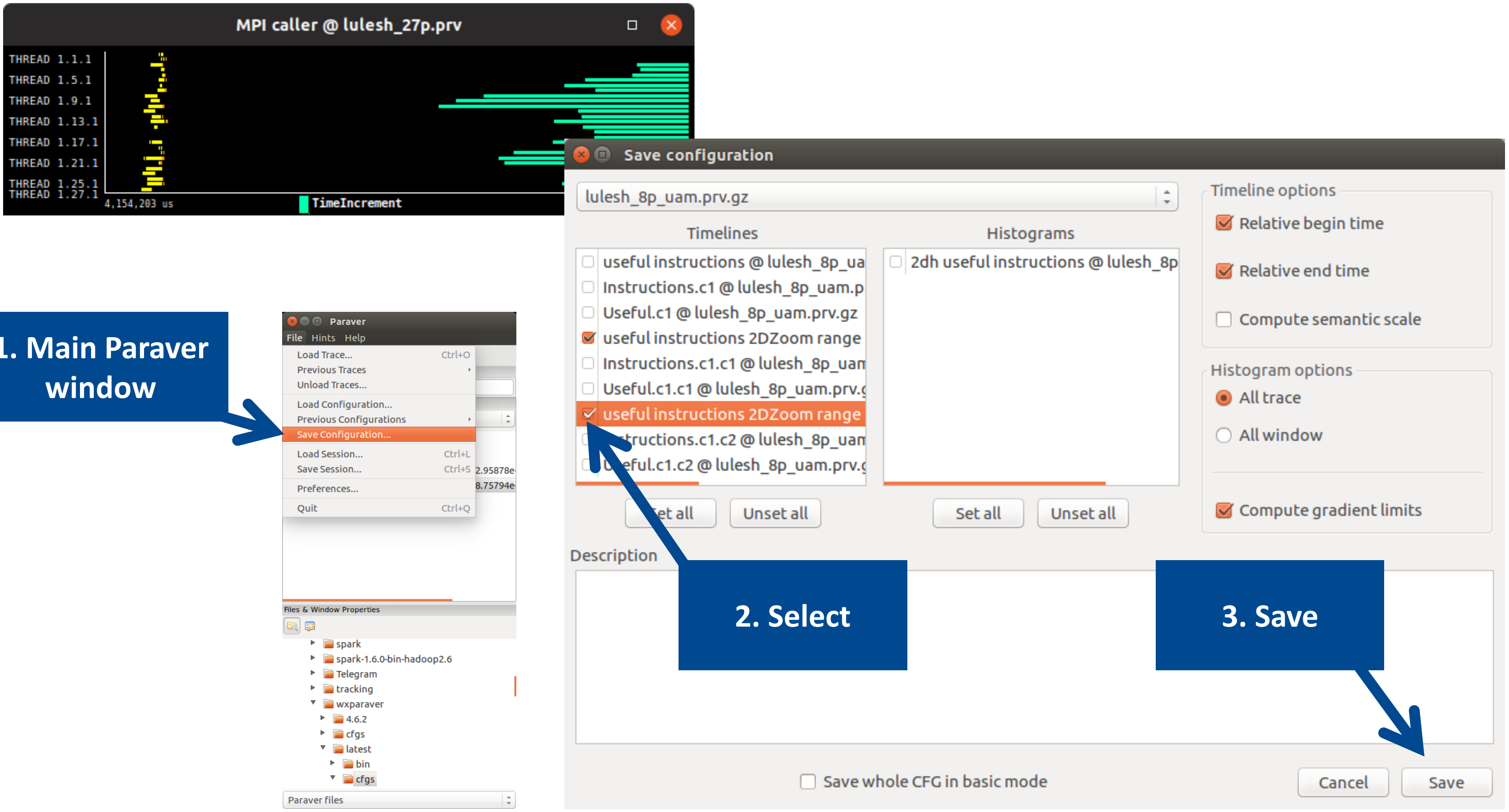
Save CFG's (2 methods)



Right click on timeline



Save CFG's (2 methods)



1. Main Paraver window

2. Select

3. Save

MPI caller @ lulesh_27p.prv

```
THREAD 1.1.1
THREAD 1.5.1
THREAD 1.9.1
THREAD 1.13.1
THREAD 1.17.1
THREAD 1.21.1
THREAD 1.25.1
THREAD 1.27.1
4,154,203 us
```

TimeIncrement

Paraver

File Hints Help

- Load Trace... Ctrl+O
- Previous Traces
- Unload Traces...
- Load Configuration...
- Previous Configurations
- Save Configuration...
- Load Session... Ctrl+L
- Save Session... Ctrl+S
- Preferences...
- Quit Ctrl+Q

Files & Window Properties

- spark
- spark-1.6.0-bin-hadoop2.6
- Telegram
- tracking
- wxparaver
  - 4.6.2
  - cfgs
  - latest
    - bin
    - cfgs

Paraver files

Save configuration

lulesh_8p_uam.prv.gz

Timelines
- [ ] useful instructions @ lulesh_8p_ua
- [ ] Instructions.c1 @ lulesh_8p_uam.p
- [ ] Useful.c1 @ lulesh_8p_uam.prv.gz
- [✓] useful instructions 2DZoom range
- [ ] Instructions.c1.c1 @ lulesh_8p_uam
- [ ] Useful.c1.c1 @ lulesh_8p_uam.prv.g
- [✓] useful instructions 2DZoom range
- [ ] Instructions.c1.c2 @ lulesh_8p_uam
- [ ] Useful.c1.c2 @ lulesh_8p_uam.prv.g

Set all | Unset all

Histograms
- [ ] 2dh useful instructions @ lulesh_8p

Set all | Unset all

Timeline options
- [✓] Relative begin time
- [✓] Relative end time
- [ ] Compute semantic scale

Histogram options
- ● All trace
- ○ All window
- [✓] Compute gradient limits

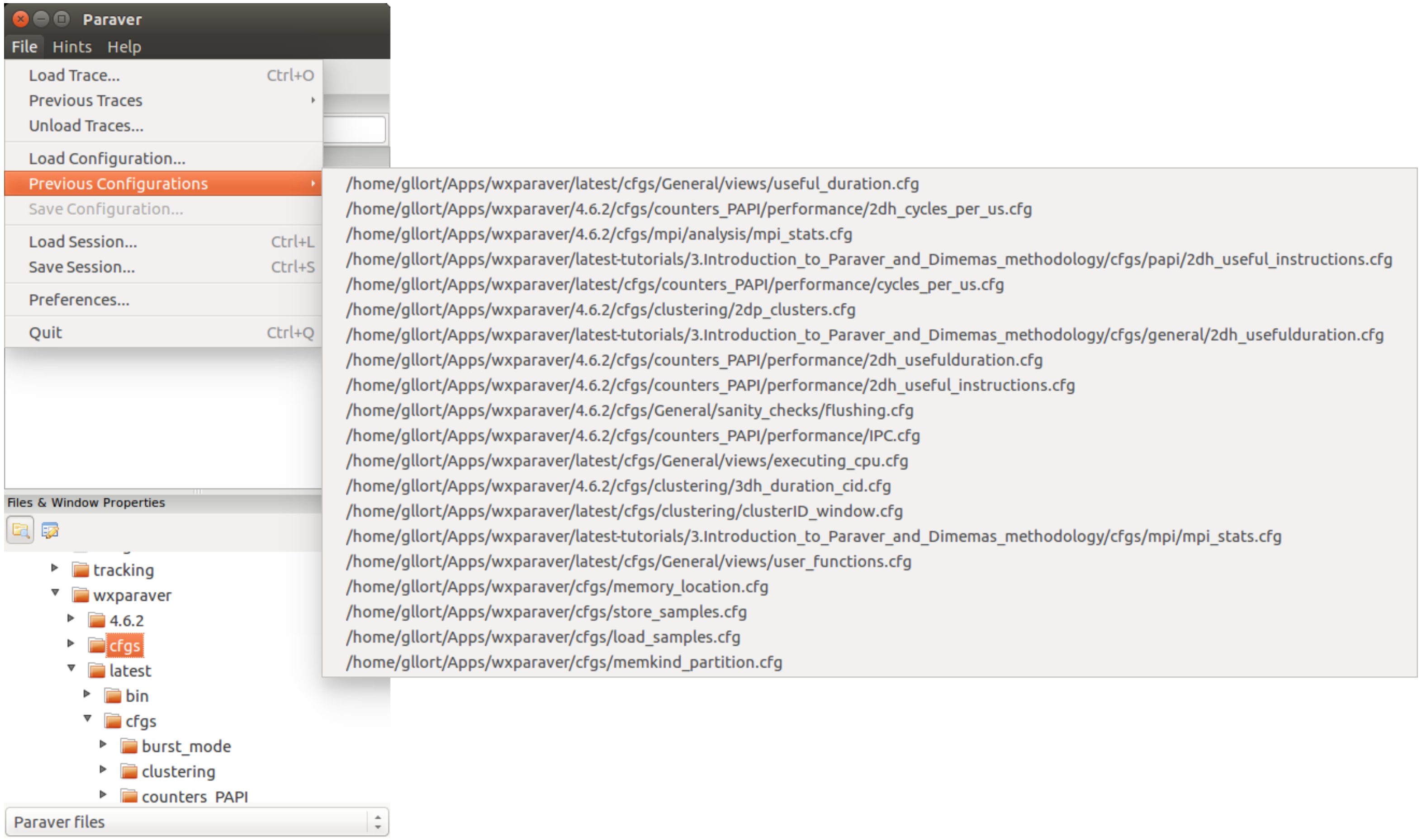
Description

- [ ] Save whole CFG in basic mode

Cancel | Save

CFG's distribution

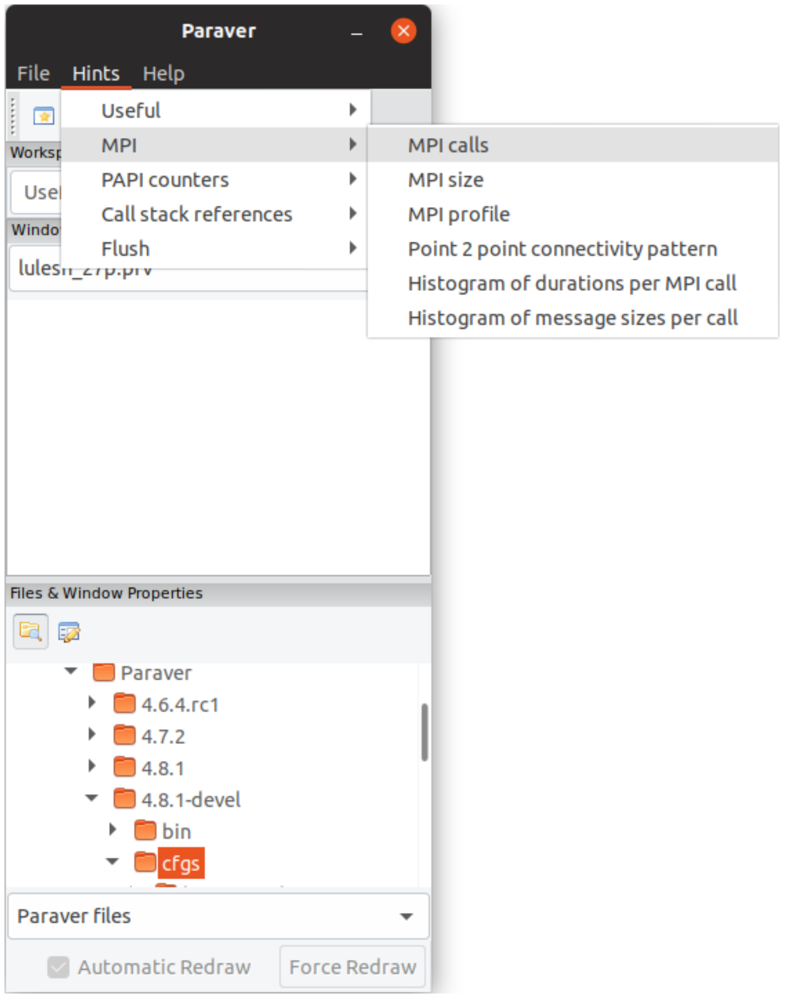
- Paraver comes with many more included CFG's



```
/home/gllort/Apps/wxparaver/latest/cfgs/General/views/useful_duration.cfg
/home/gllort/Apps/wxparaver/4.6.2/cfgs/counters_PAPI/performance/2dh_cycles_per_us.cfg
/home/gllort/Apps/wxparaver/4.6.2/cfgs/mpi/analysis/mpi_stats.cfg
/home/gllort/Apps/wxparaver/latest-tutorials/3.Introduction_to_Paraver_and_Dimemas_methodology/cfgs/papi/2dh_useful_instructions.cfg
/home/gllort/Apps/wxparaver/latest/cfgs/counters_PAPI/performance/cycles_per_us.cfg
/home/gllort/Apps/wxparaver/4.6.2/cfgs/clustering/2dp_clusters.cfg
/home/gllort/Apps/wxparaver/latest-tutorials/3.Introduction_to_Paraver_and_Dimemas_methodology/cfgs/general/2dh_usefulduration.cfg
/home/gllort/Apps/wxparaver/4.6.2/cfgs/counters_PAPI/performance/2dh_usefulduration.cfg
/home/gllort/Apps/wxparaver/4.6.2/cfgs/counters_PAPI/performance/2dh_useful_instructions.cfg
/home/gllort/Apps/wxparaver/4.6.2/cfgs/General/sanity_checks/flushing.cfg
/home/gllort/Apps/wxparaver/4.6.2/cfgs/counters_PAPI/performance/IPC.cfg
/home/gllort/Apps/wxparaver/latest/cfgs/General/views/executing_cpu.cfg
/home/gllort/Apps/wxparaver/4.6.2/cfgs/clustering/3dh_duration_cid.cfg
/home/gllort/Apps/wxparaver/latest/cfgs/clustering/clusterID_window.cfg
/home/gllort/Apps/wxparaver/latest-tutorials/3.Introduction_to_Paraver_and_Dimemas_methodology/cfgs/mpi/mpi_stats.cfg
/home/gllort/Apps/wxparaver/latest/cfgs/General/views/user_functions.cfg
/home/gllort/Apps/wxparaver/cfgs/memory_location.cfg
/home/gllort/Apps/wxparaver/cfgs/store_samples.cfg
/home/gllort/Apps/wxparaver/cfgs/load_samples.cfg
/home/gllort/Apps/wxparaver/cfgs/memkind_partition.cfg
```

Hints: a good place to start!

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



Cluster-based analysis

Judit Giménez, Germán Llort

tools@bsc.es

16/09/19

POP-EoCoE

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 P2CHPD

```
p2chpd> ls ~germain.llort/tools/ClusteringSuite
```

Use clustering analysis

- Run clustering:

```
p2chpd> cd $HOME/tools-material/clustering
p2chpd> ~germain.llort/tools/ClusteringSuite/bin/BurstClustering
        -d cluster.xml
        -i ../extrae/lulesh_27p.prv
        -o lulesh_27p_clustered.prv
```

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

```
p2chpd> ls $HOME/tools-material/traces/lulesh_27p.prv
```

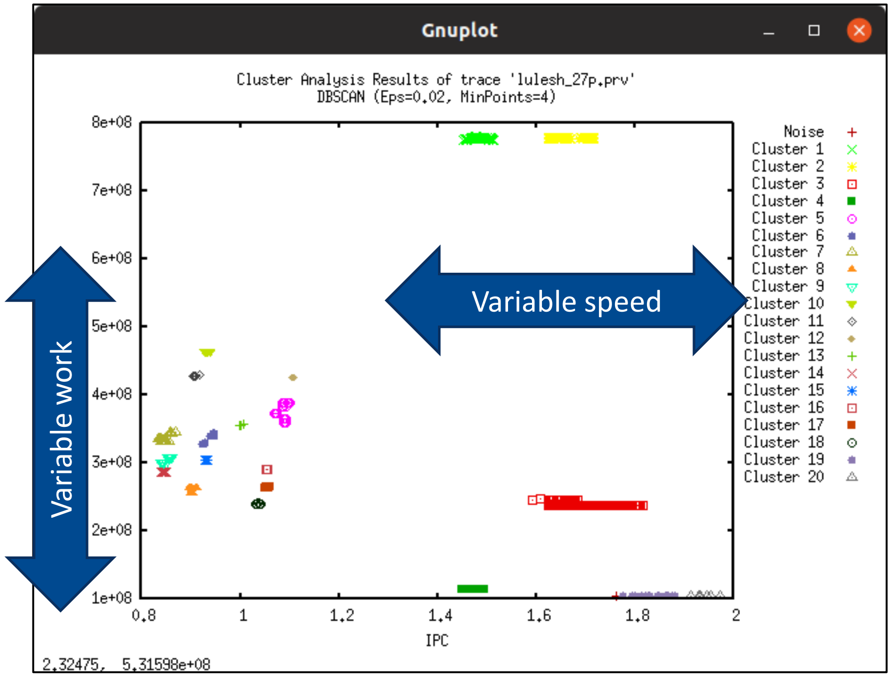
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Cluster-based analysis

- Check the resulting scatter plot

```
p2chpd> gnuplot lulesh_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

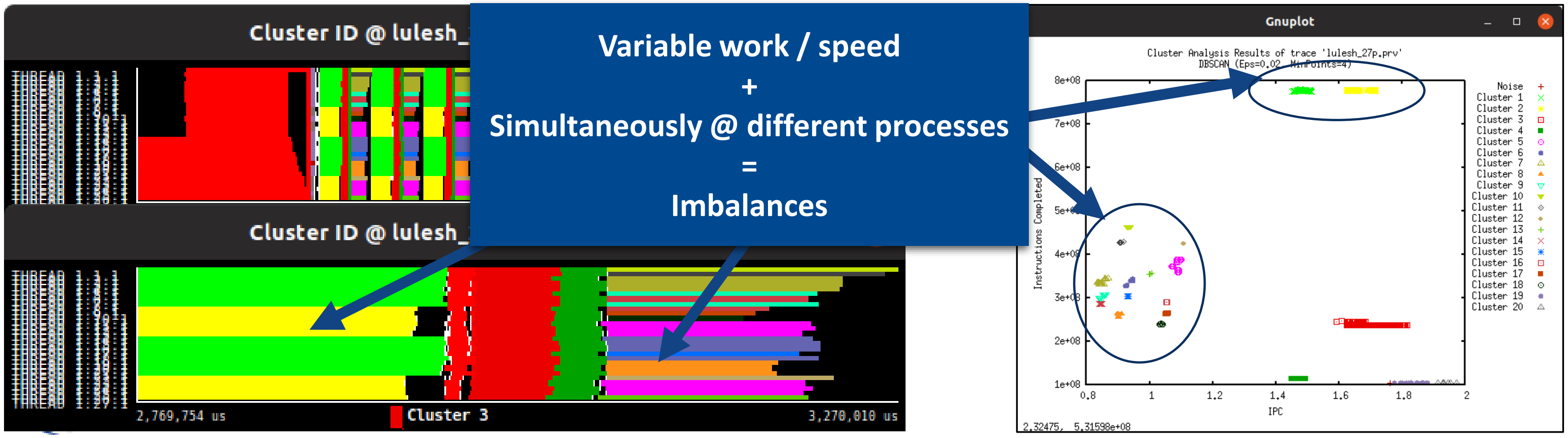


Correlating scatter plot and time distribution

- Open the clustered trace with Paraver and look at it

```
laptop> scp <USER>@p2chpd-login3.univ-lyon1.fr:tools-
        material/clustering/*clustered* $HOME
laptop> $HOME/paraver/bin/wxparaver $HOME/lulesh_27p_clustered.prv
```

- Display the distribution of clusters over time
  - Hints → Clustering → Profile of clusters → Open Control Window



Variable work / speed
+
Simultaneously @ different processes
=
Imbalances

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

EXCELENCIA SEVERO OCHOA

Thank you!

Judit Giménez, Germán Llort

tools@bsc.es