



# HANDS-ON: PERFORMANCE ANALYSIS USING POP METHODOLOGY

Radim VAVŘÍK, Tomáš PANOC  
Infrastructure Research Lab, IT4I

5. 4. 2022

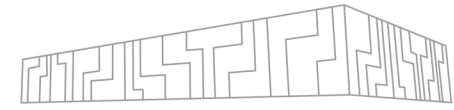


EUROPEAN UNION  
European Structural and Investment Funds  
Operational Programme Research,  
Development and Education



MINISTRY OF EDUCATION,  
YOUTH AND SPORTS

# PERFORMANCE ANALYSIS



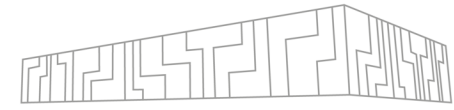
## Who has any experience with a performance analysis tool?

- What was the tool?

## Objectives today?

- Not to reach an incredible performance improvement of the example code
- Rather to learn the procedures and best practices with tools

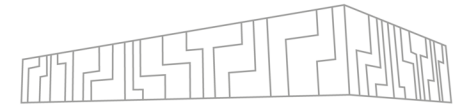
# PERFORMANCE ANALYSIS



## General steps:

- Installation of the application (and the tools!)
- Execute the application as is
  - Test if it works as expected
  - Measure runtime of the application with no extra tool overhead
- Perform tracing – configuration and collection of performance data
  - Execute the tracing runs typically for strong or weak scaling
- Perform analysis
  - Preprocess traces
  - Identify the structure of the application
  - Select regions of interest
  - Evaluate the basic metrics
  - Dig down as necessary

# PERFORMANCE ANALYSIS



**Barcelona  
Supercomputing  
Center**

*Centro Nacional de Supercomputación*

## Paraver

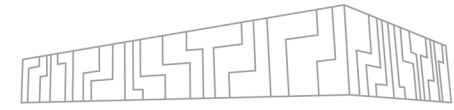
- Performance analysing tool
- Visual analysis using timelines and 2D/3D tables (profiles, histograms)
- Comparative analysis of multiple traces
- Trace manipulation support (cutting, filtering)
- Additional applications (Stats, Dimemas, Clustering, ...)
- Predefined + custom derived metrics

## Extrae

- Package devoted to collect performance data and generate Paraver profiles
- Instrumentation, tracing, sampling, burst mode tracing, User events API
- Linker preload – no need of source codes
- `libseqtrace`, `libmpitrace[f]`, `libomptrace`, `libompitrace[f]`, `libpttrace`, `libptmpitrace[f]`, `libcudatrace`, `libcudampitrace[f]`, `libcudaompitrace[f]`, `libocltrace`, `liboclpitrace[f]`, ...

**Not going through all the features but trial and error approach ;)**

# PERFORMANCE ANALYSIS



## How to control the features?

- **extrae.xml** to control the features – application specific

## Extrae documentation!

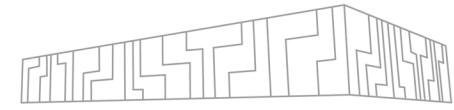
<https://tools.bsc.es/doc/html/extrae/index.html>

## Why so much configuration?

- Size of traced data grows enormously with scale and walltime
- We have to limit the amount of information obtained
- Controlled by the configuration of the tool
- Tradeoff between size of trace files and detail of information

Can be challenging, you'll see ;)

# GET READY



## Submit an interactive job

- Open your VNC session or login to one of the Barbora login nodes with X-Window systém enabled

```
| qsub -q R603003 -l select=1:mpiprocs=36 -IX
```

```
| export TMPDIR=/scratch/project/dd-22-26/tmp
```

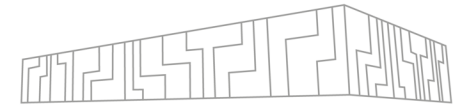
**#Barbora specific**

## Obtain the hands on

```
| cp -r /mnt/proj1/dd-22-26/perf-handson/ ~
```

```
| cd ~/perf-handson/false-cc
```



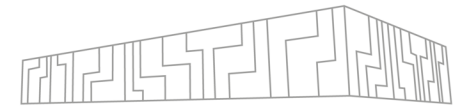


## FalseCC - mock-up application

- Simple C code
- Pure MPI
- Implements the following pattern:
  1. Data packing
  2. Set of non-blocking sends/receives between neighbors that may overlap each other
  3. Wait for communication
  4. Computation
- Simulates bad **Transfer efficiency** due to long waiting for messages
- Observed in many real-world applications

```
| vim false-cc.c
```

# FALSE-CC



## Built and run the application

- To check it works and get a baseline timing

```
| ml OpenMPI/4.1.1-GCC-11.2.0
```

```
| make
```

```
| mpirun -n 8 ./false-cc.exe
```

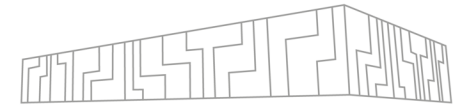
```
| mpirun -n 16 ./false-cc.exe
```

```
| mpirun -n 32 ./false-cc.exe
```

- We can observe the weak scaling
- The runtime should be constant in an ideal case

MPI Processes	Time [s]
8	0.5
16	0.7
32	1.2

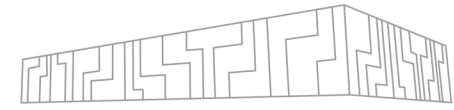




## Prepare and run tracing

```
| module use /mnt/proj1/dd-22-26/perf-tools/modules/all/  
| ml EXTRAЕ/3.8.3-OpenMPI-4.1.1  
| cp $EXTRAЕ_HOME/share/example/MPI/extrae_explained.xml .  
| cp $EXTRAЕ_HOME/share/example/MPI/ld-preload/trace.sh .  
  ▪ Edit the path to extrae_explained.xml  
  ▪ Enable libmpitrace.so library for C apps  
| vim Makefile  
  ▪ Add -finstrument-functions to CFLAGS (Optional. Be careful with real codes, can be too intrusive!)  
| make clean && make  
| mpirun -n 2 ./trace.sh ./false-cc.exe      # Ouch :(  
  ▪ .prv, .pcf, .row trace files generated, but...
```

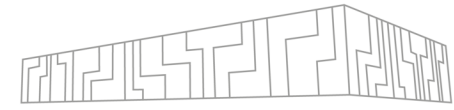
# FALSE-CC - EXTRA



## Learn to fix all the errors and warnings!

```
| unset OMP_NUM_THREADS      # May not be needed, depends on terminal
| ./get-uf.sh false-cc.exe   # generate user functions
| vim extrae_explained.xml
  ▪ Disable OpenMP tracing
  ▪ Edit the absolute path to user-functions list
  ▪ Tune the CPU counters sets using the following utility (tip: always keep INS and CYC)
| papi_best_set              # tip: omnipresent INS and CYC
| papi_avail -a              # might be helpful
| mpirun -n 2 ./trace.sh ./false-cc.exe
```

## Did you expected a second trace?



## Extrae tips & tricks

- Disable trace overwriting by merge attribute

```
| <merge ... overwrite="no" />
```

- Customize the trace file name enclosing the string by merge tags

```
| <merge ... >my_custom_name.prv</merge>
```

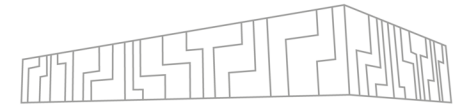
- Or use a variable

```
| export TRACE_FILE=my_custom_name.prv
```

```
| <merge ... >${TRACE_FILE}$</merge>
```

- Save some (a lot of) disk space removing the raw data

```
| <merge ... keep-mpits ="no" />
```



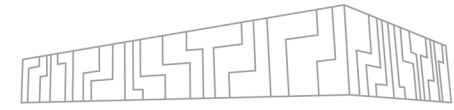
## Execute the traced scalability runs

- To obtain the data for analysis

```
| TRACE_FILE=false-cc-8p.prv mpirun -n 8 ./trace.sh ./false-cc.exe  
| TRACE_FILE=false-cc-16p.prv mpirun -n 16 ./trace.sh ./false-cc.exe  
| TRACE_FILE=false-cc-32p.prv mpirun -n 32 ./trace.sh ./false-cc.exe 5000
```

- Note sizes of the trace files

# FALSE-CC - PARAVER



## Start analysing with Paraver

- ```
| ml use /mnt/proj1/dd-22-26/perf-  
tools/modules/all/  
| ml Paraver/4.9.2-foss-2021a  
| wxparaver&
```
- Load the trace **false-cc-16p.prv**
  - Explore the Main window

| Name             | L1 cache misses |
|------------------|-----------------|
| Begin time       | 0.00 us         |
| End time         | 1,036,737.19 us |
| Semantic Minimum | 0               |
| Semantic Maximum | 8.07918e+07     |
| Level            | Thread          |
| Time unit        | Microseconds    |

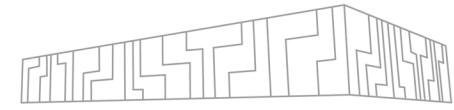
Filter

- Communications
- Events
  - Event type: =; 42000000
  - Function: =
  - Types: 42000000
  - Type/Value Op: And
  - Event value: All;
- Semantic


PARAVER

- bin
- cfgs
- burst\_mode
- clustering
- counters\_PAPI
  - architecture
    - 2dh\_preemption\_time.cfg
    - 3dh\_percentpreempted\_useful.cfg
    - 3dh\_preempted\_time\_useful.cfg
    - 3dh\_preempted\_useful.cfg
    - L1\_Load\_misses.cfg
    - L1\_store\_misses.cfg
    - L1D\_misses.cfg
    - L1D\_missratio.cfg

# FALSE-CC - PARAVR

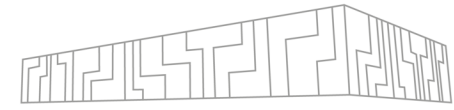


## Timeline window

- Click on **New single timeline window** icon 
- Explore the controls
  - Zoom to selection - Drag&Drop
  - Zoom to selected threads - Ctrl+D&D
  - Zoom in – Scroll up
  - Zoom out – Scroll down
  - Move in time - Shift+D&D
  - Info Panel – Double click
  - Undo Zoom – Ctrl+U
  - Redo Zoom – Ctrl+R
  - Fit Time Scale

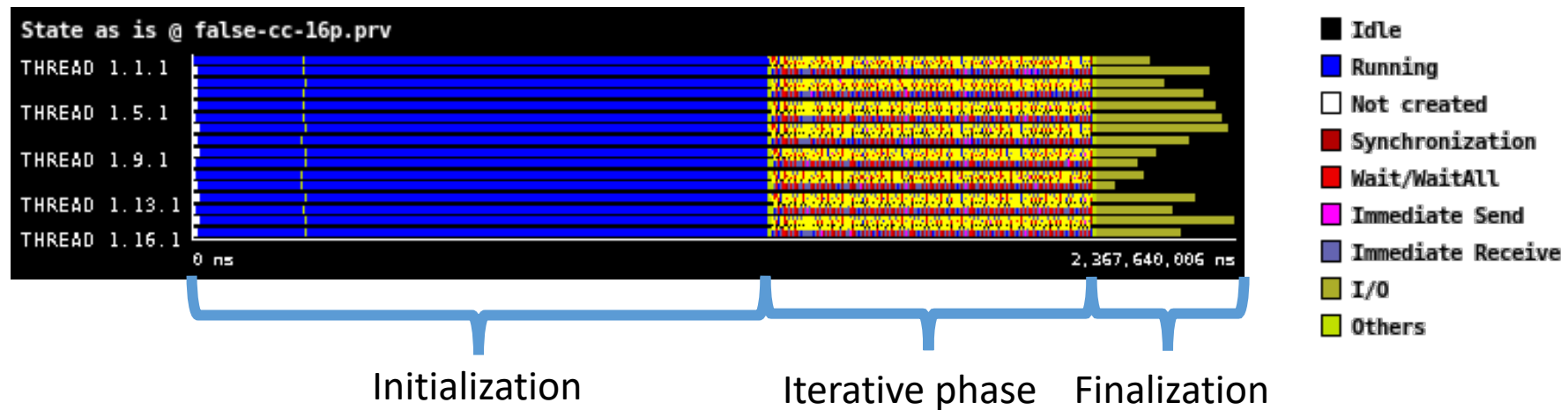


# FALSE-CC - PARAVR



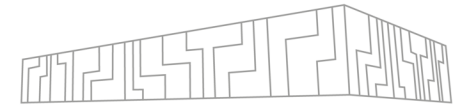
## Identify structure

- To understand the main characteristics and for easier navigation during analysis
- Typical structure:
  - Initialization – sometimes important to analyze, very often can be omitted
  - Iterative phase – usually the most interesting part for analysis
  - Finalization – can be omitted in most cases



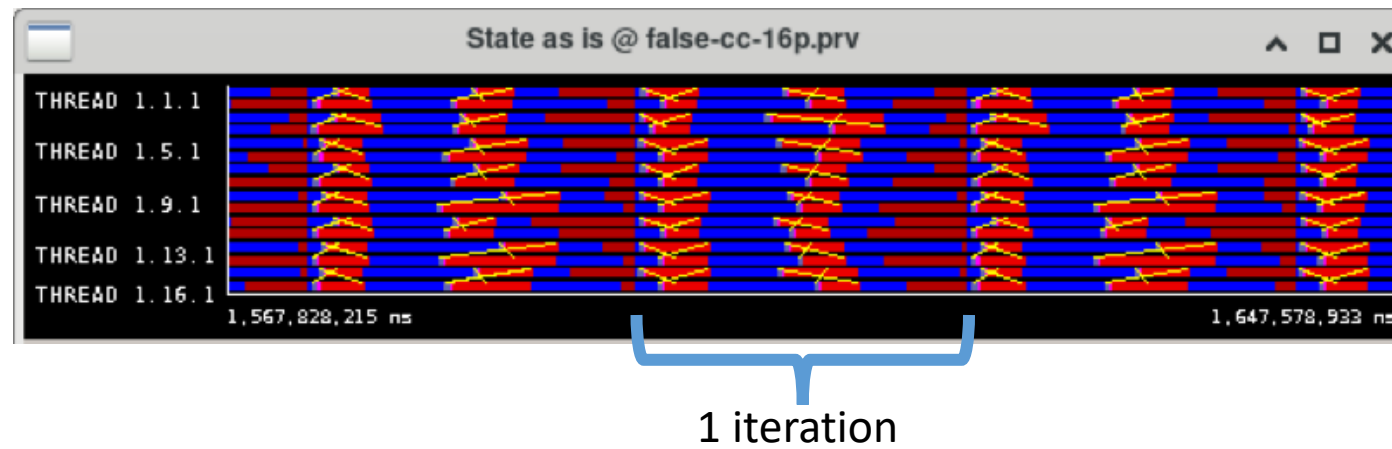


# FALSE-CC - PARAVR

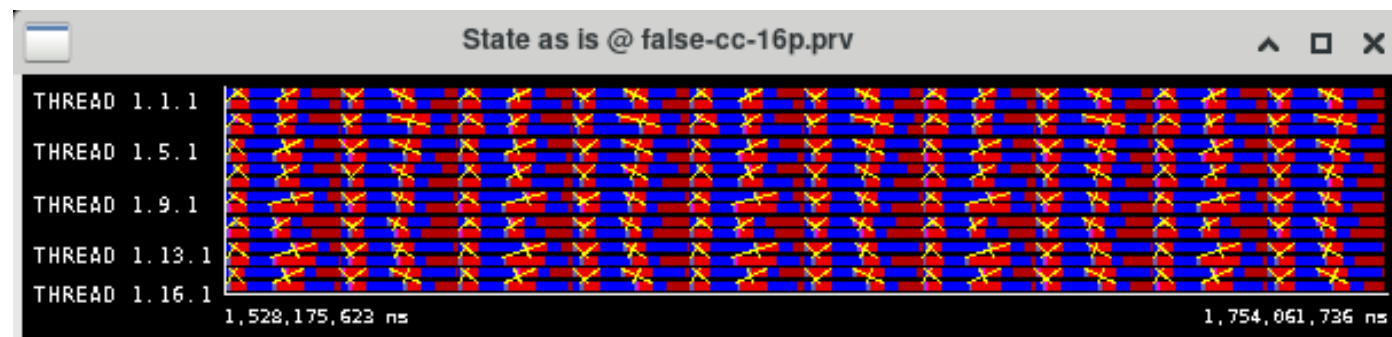


## Select a region of interest (RoI)

- To focus on important part and eliminate disruptions
- Identify one iteration – the repetitive pattern



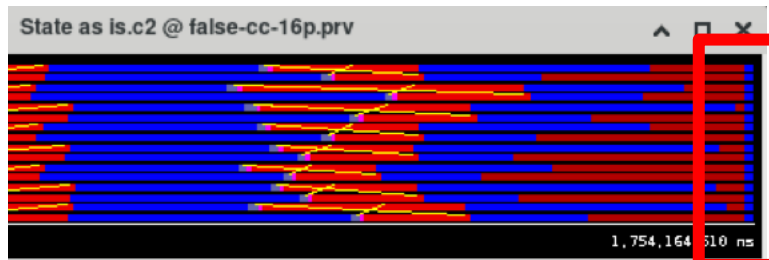
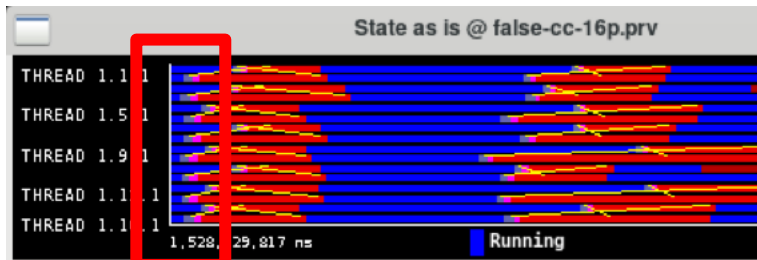
- Find 10 representative iterations using zooming features
- Recommended to omit first and last iterations of the iterative phase



# FALSE-CC - PARAVR

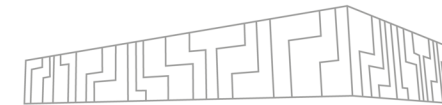
## Make a cut

- Let the left and right borders be in the **Running state** (blue) for all the threads
- Cloning timelines might be helpful
- Right click inside the timeline window and select the option **Clone**
- In the **original** window, find the appropriate **beginning** of an iteration
- In the **cloned** window, find the appropriate **end** of the 10th iteration
- hint: the first Running state after Synchronization




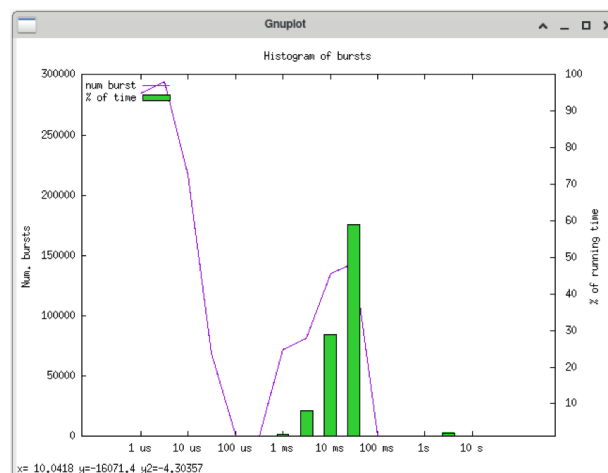
- Copy the **End time** from the Properties of cloned window and paste it into the **End time** of the original window
- Right click inside the original window and select **Run -> Cutter -> Apply**
- Repeat for **false-cc-8p.prv**

# FALSE-CC - PARAVR



## Run Stats

- To get an initial statistics of the (large) trace
- Click on **Run Application** icon 
- Select **Stats** application
- Select the **false-cc-32p.prv** trace
- Run
- Open **Histogram of bursts** via the generated \*.gnuplot link



Run Application (on cn13.barbora.it4i.cz)

Application: Stats

Trace: false-cc-32p.prv Browse

Parameters:

Output Prefix: /home/vav0038/perf-handson/false-cc/false-cc-32p.prv

☒ Generate bursts histogram ☐ Generate communications histogram

☐ Only generate .dat file ☐ Exclusive instead of inclusive times

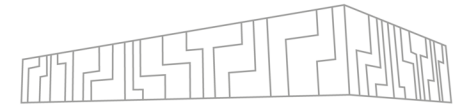
Preview: stats-wrapper.sh "/home/vav0038/perf-handson/false-cc/false-cc-32p.prv" -o "/home/vav0038/perf-handson/false-cc/false-cc-32p.prv" -bursts\_histo

Run Kill Clear Log

Processing trace 8/% Processing trace 88% Processing trace 89%  
Processing trace 90% Processing trace 91% Processing trace 92%  
Processing trace 93% Processing trace 94% Processing trace 95%  
Processing trace 96% Processing trace 97% Processing trace 98%  
Processing trace 99% Processing trace 100% Processing trace 100%  
Generating file  
</home/vav0038/perf-handson/false-cc/false-cc-32p.bursts.dat>  
Generating file  
</home/vav0038/perf-handson/false-cc/false-cc-32p.bursts.gnuplot>

Exit

# FALSE-CC - PARAVR

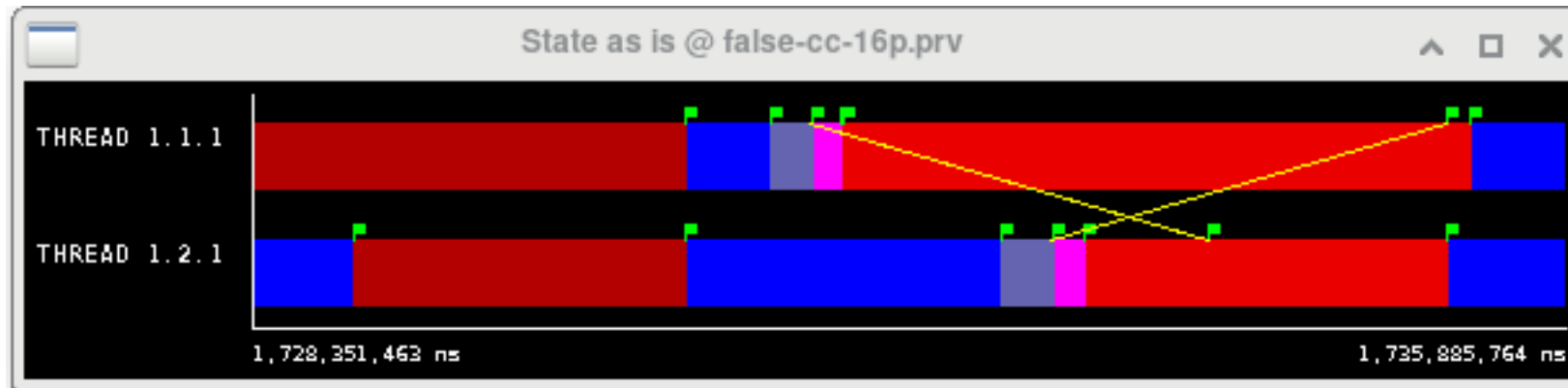


## Bursts?

- Sequence of **useful instructions** inside the Running state
- e.g. between 2 MPI calls, inside OpenMP a parallel section, etc.

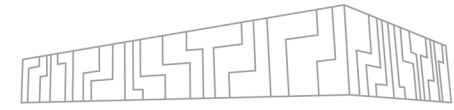
## Useful?

- Useful means user code outside the MPI or OpenMP runtime



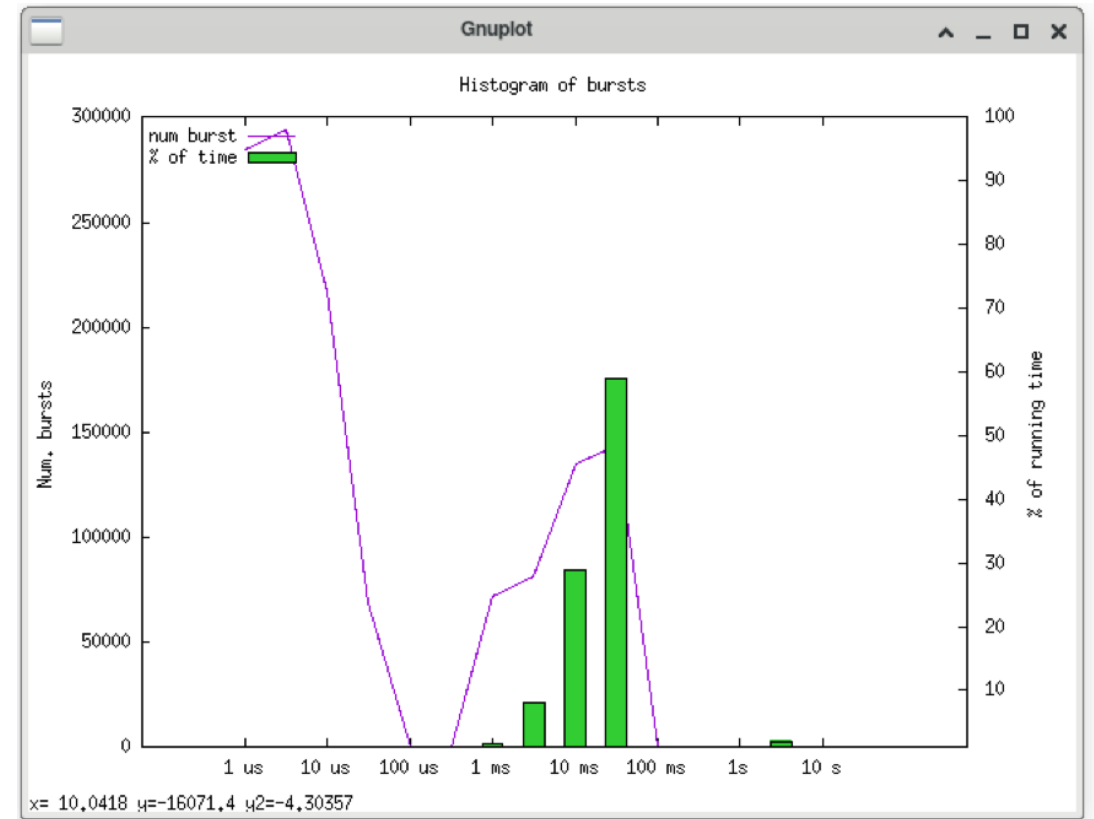
- Tip: Events bounding the states can be displayed via right click -> View -> Event Flags

# FALSE-CC - PARAVÉR

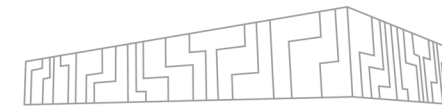


## Histogram of bursts

- The green bars are the total amount of time incurred by computation bursts of the different durations
- 60% of the total time in ~50ms bursts
- 30% of the total time in ~10ms bursts
- The purple line shows the total amount of bursts of a given duration
- Most of the bursts shorter than **10us**
- We can infer a duration such that the number of bursts above is reasonably small though they represent the most of the runtime – will be used in the next step!

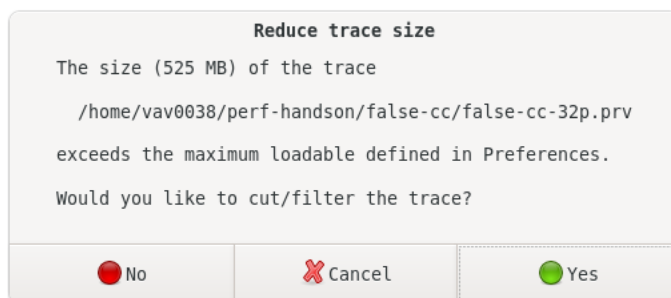


# FALSE-CC - PARAVR

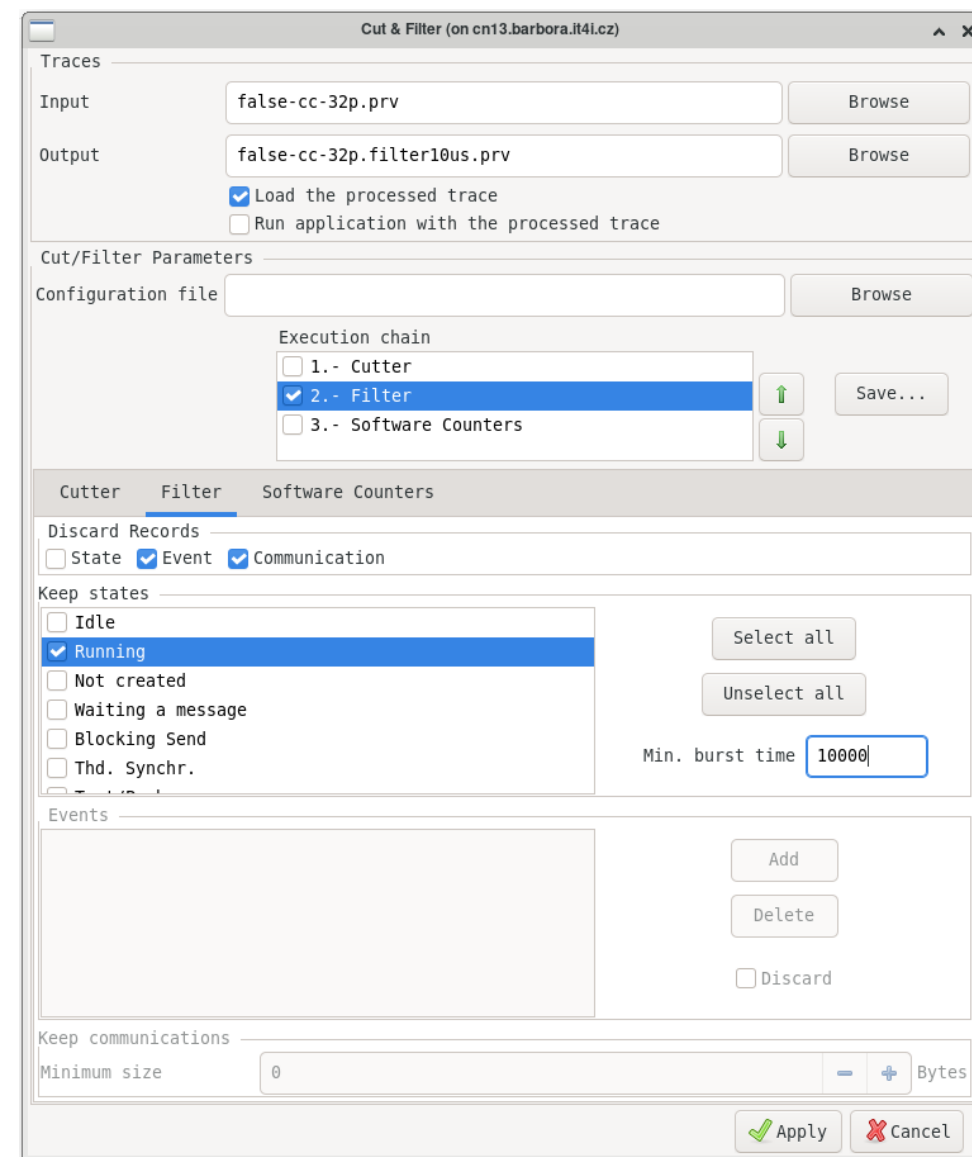


## Perform filtering

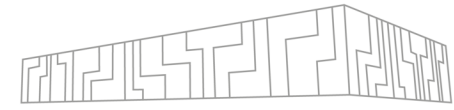
- To get a filtered trace of full duration run with a subset of the original information
- Load the **false-cc-32p.prv** trace



- Reduce trace size -> Yes
- Select Filter
- Discard Records: Event and Communication
- Keep states: Running
- Min. burst time: 10000 (=10us based on Stats)
- Apply

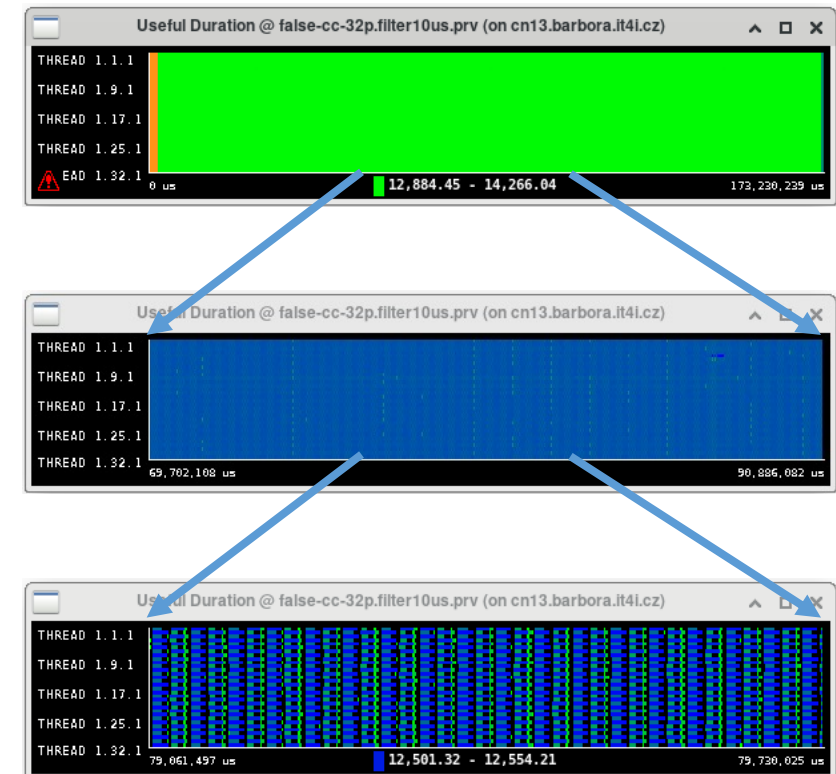


# FALSE-CC - PARAVR

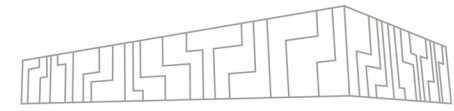


## Make a cut from filtered trace

- Open the **Useful duration** timeline from Main menu -> Hints -> Useful
- Zoom into the full green area
- Rescale the colors via right click -> Fit Semantic Scale -> Fit Both or click in the bottom left corner
- Repeat zooming and rescaling until a pattern can be recognized
- Repeat the cutting steps from the previous traces using Useful duration timelines
- **Important:** select the original non-filtered trace as the input in the Cutter dialog







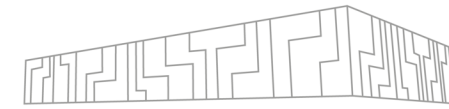
## Perform basic analysis

- Automatically evaluates the basic metrics of the whole trace – **thus cuts needed**
- Shows the directions for deeper analysis

```
| ml EXTRAE/3.8.3-OpenMPI-4.1.1  
| mkdir basic-analysis && cd basic-analysis  
| modelfactors.py ../false-cc-8p.chop10it.prv ../false-cc-  
16p.chop10it.prv ../false-cc-32p.chop10it.prv
```

- This may take quite a long time for larger data

# FALSE-CC – BASIC ANALYSIS

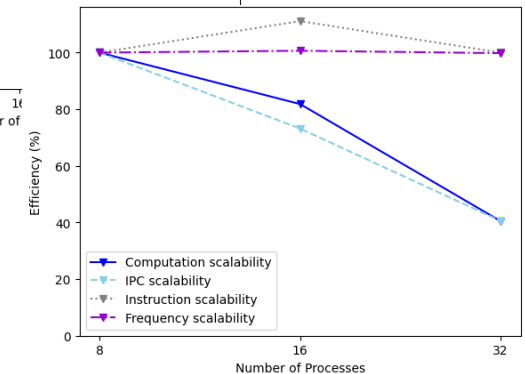
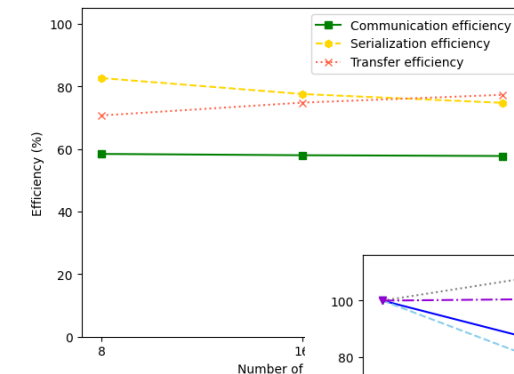
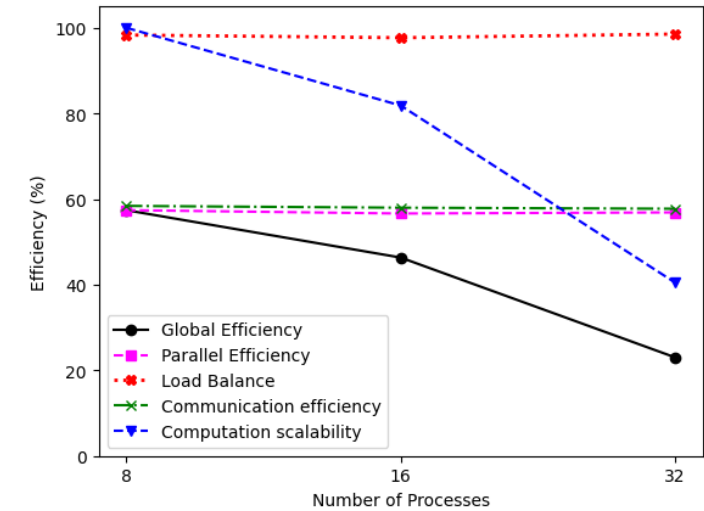


## Evaluate basic analysis results

- 100% ideal efficiency
- 90% means losing 10% of potential performance
- 80% all the lower efficiencies should be investigated

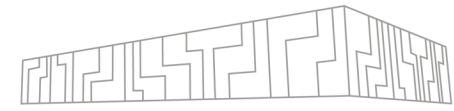


| MPI Processes           | 8    | 16   | 32   |
|-------------------------|------|------|------|
| Elapsed time (sec)      | 0.15 | 0.19 | 0.38 |
| Efficiency              | 1.00 | 0.81 | 0.40 |
| Speedup                 | 1.00 | 0.81 | 0.40 |
| Average IPC             | 1.43 | 1.05 | 0.58 |
| Average frequency (GHz) | 3.23 | 3.25 | 3.22 |



- The limiting factors are clearly IPC scaling, **Transfer efficiency**, and **Serialization**

# FALSE-CC – PARAVR

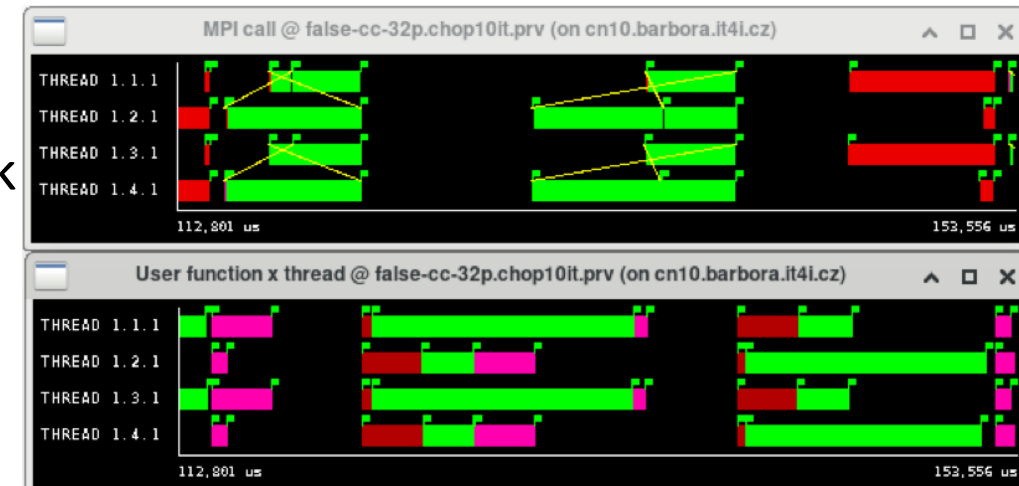


## Analyze the limiting factors

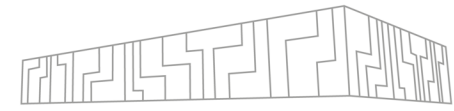
- We will focus on the **Transfer efficiency** now
- Open the **false-cc-32p.chop10it.prv**
- Main menu -> Hints -> MPI -> MPI calls
- Main menu -> Hints -> User functions -> User functions
- Synchronize windows via right click -> Synchronize -> 1 (both windows)
- Zoom to only one iteration and few processes
- Enable Communication Lines in MPI call window via right click -> View
- Send and Receive take very small time before the Waitalls
- Processes 1, 3, .. do two waitalls after the first pack
- Processes 2, 4, .. do two waitalls after the second pack
- They could start unpacking and computing instead

MPI profile @ false-cc-32p.chop10it.prv (on cn10.barbora.it4i.cz)

|              | MPI_Isend | MPI_Irecv | MPI_Waitall | MPI_Barrier |
|--------------|-----------|-----------|-------------|-------------|
| THREAD 1.1.1 | 0.12 %    | 0.09 %    | 22.07 %     | 17.83 %     |
| THREAD 1.2.1 | 0.06 %    | 0.05 %    | 40.36 %     | 4.93 %      |
| THREAD 1.3.1 | 0.14 %    | 0.11 %    | 22.36 %     | 17.99 %     |
| THREAD 1.4.1 | 0.06 %    | 0.04 %    | 40.36 %     | 5.53 %      |
| Total        | 0.38 %    | 0.29 %    | 125.15 %    | 46.28 %     |
| Average      | 0.09 %    | 0.07 %    | 31.29 %     | 11.57 %     |
| Maximum      | 0.14 %    | 0.11 %    | 40.36 %     | 17.99 %     |
| Minimum      | 0.06 %    | 0.04 %    | 22.07 %     | 4.93 %      |
| StDev        | 0.04 %    | 0.03 %    | 9.07 %      | 6.35 %      |
| Avg/Max      | 0.67      | 0.65      | 0.78        | 0.64        |



# FALSE-CC



## False-CC – mock-up fix

- The problem of the original version is equal treatment of receive and send operations
- Received data needed for computation, but send operations can wait
- The fix is to postpone the waiting on send buffers until its reuse

## Apply the fix and repeat the tracing and analysis

```
| cp -r /mnt/proj1/dd-22-26/
```

```
perf-handson-fixed/ ~
```

```
| cd ~/perf-handson/false-cc-fixed
```

- You can decrease the number of iterations and omit filtering

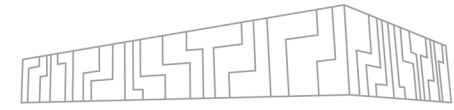
```
| TRACE_FILE=false-cc-fix-8p.prv mpirun -n 8 ./trace.sh ./false-cc.exe
```

```
| TRACE_FILE=false-cc-fix-16p.prv mpirun -n 16 ./trace.sh ./false-cc.exe
```

```
| TRACE_FILE=false-cc-fix-32p.prv mpirun -n 32 ./trace.sh ./false-cc.exe
```

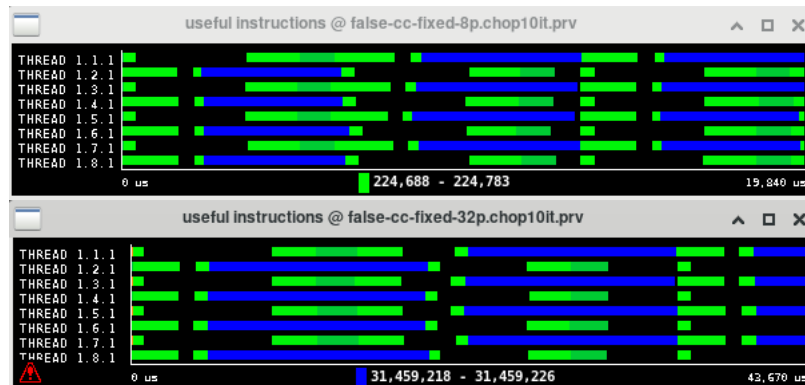


# FALSE-CC – PARAVR

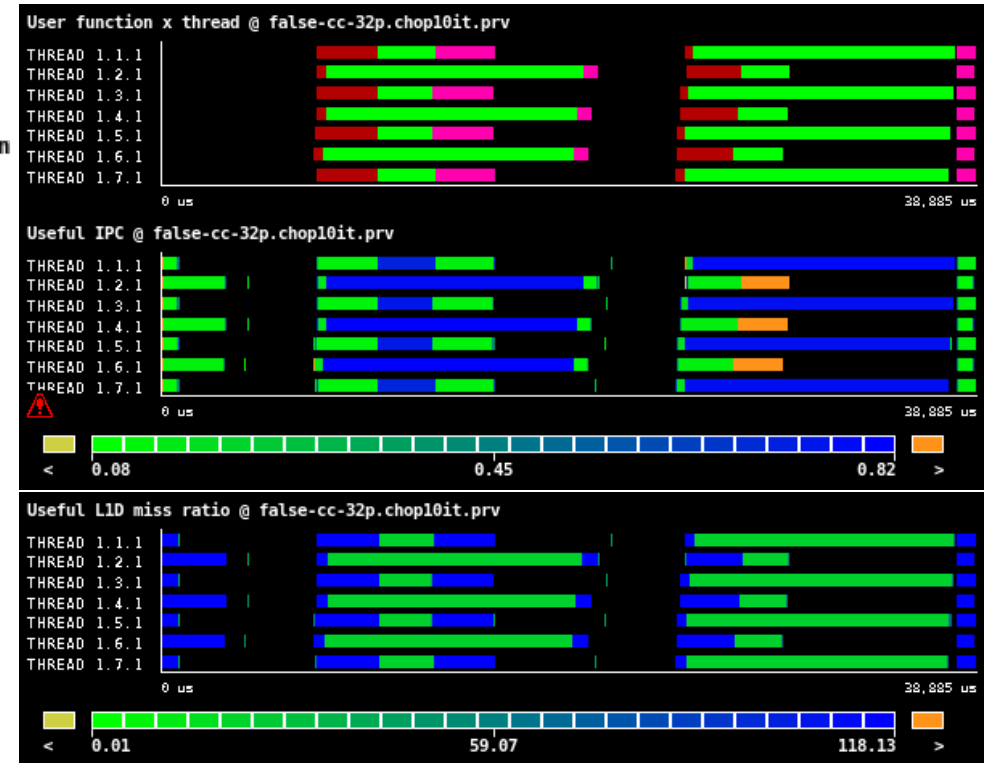


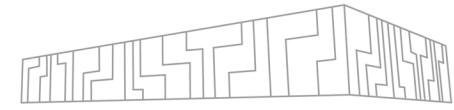
## Extra time? Analyze another limiting factor

- Very low IPC can be found in Pack and Unpack functions
- Check the cache misses!
- Compare traces of different scales



■ End  
■ Pack  
■ Unpack  
■ Computation





## Learning materials

- Paraver tutorials (Need to be installed for the first time)
- Extrae docs <https://tools.bsc.es/doc/html/extrae/index.html>
- Pop website <https://pop-coe.eu/further-information/learning-material>
- VI-HPS <https://www.vi-hps.org/training>