Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Performance Analysis:

Methodology, Tools, and Metrics

Joan Vinyals Ylla-Català
joan.vinyals@bsc.es
Barcelona Supercomputing Center

The objective

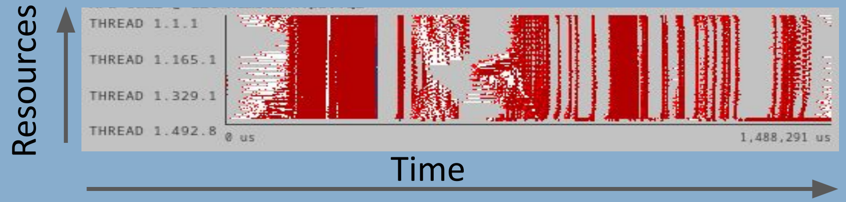
Learn a methodology that allows you to identify flaws in the parallelization of HPC applications.

Provide a set of tools used to follow this methodology, and a quick start on how to use them.

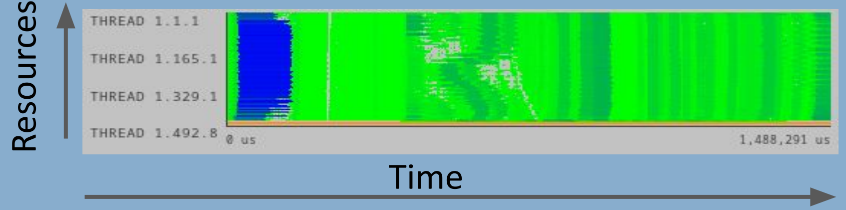
A quick view of Paraver figures

Timeline views

Color coded views, each color has some semantic. In this example each color maps to an MPI call.

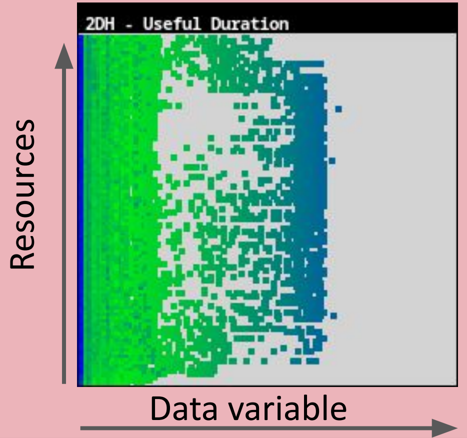


Color gradient views, range from green to blue (low and high values respectively). Here showing duration between MPI calls.

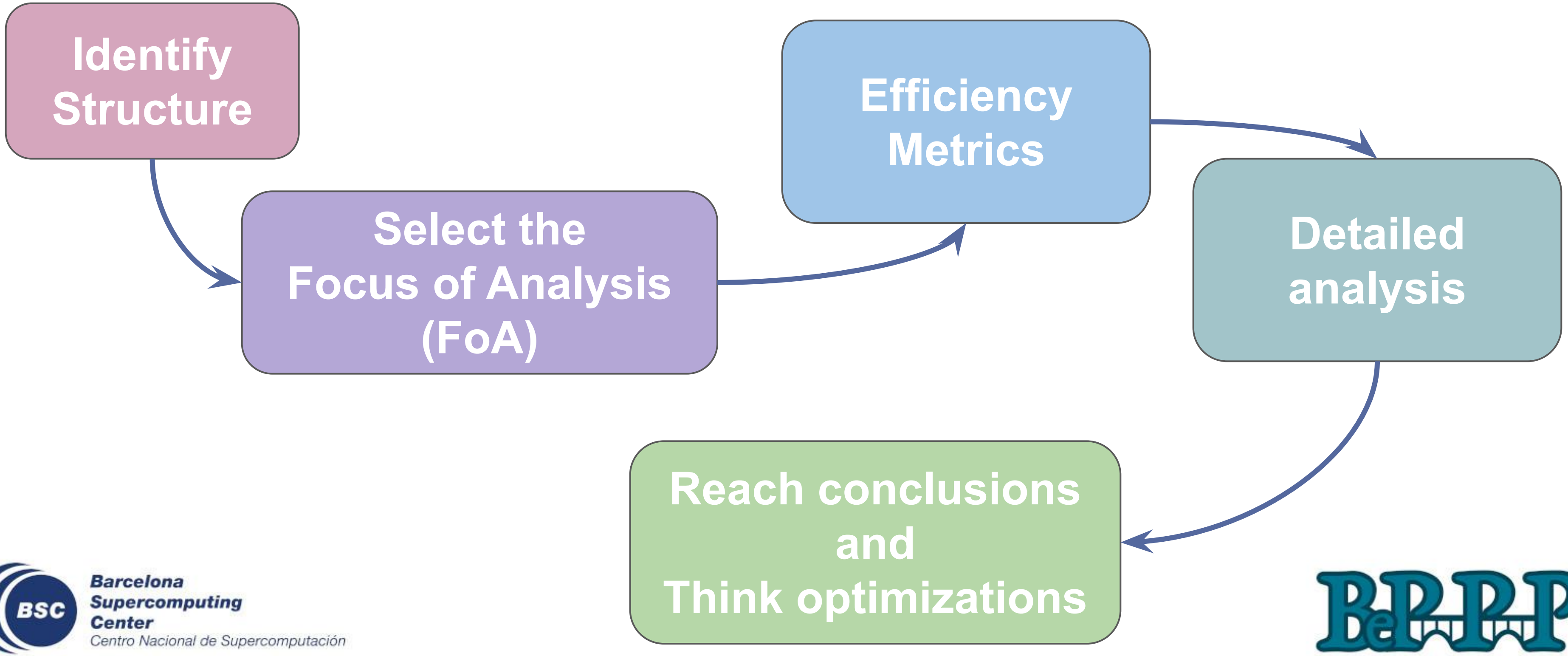


Histogram views

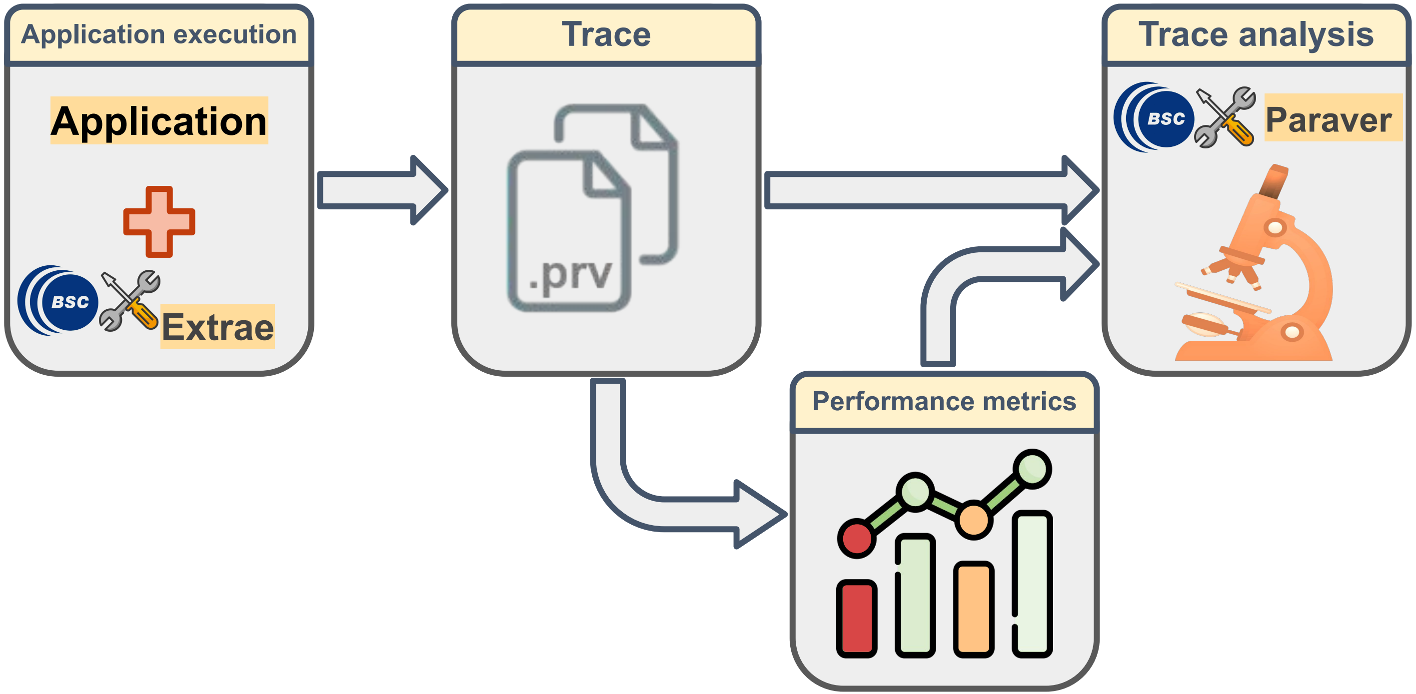
In 2D Histograms, each row is the histogram for each resource object (Proces, Threads, etc…), columns are the histogram bins, and color represent the height of each bin. In the example the data variable is time between MPI calls.



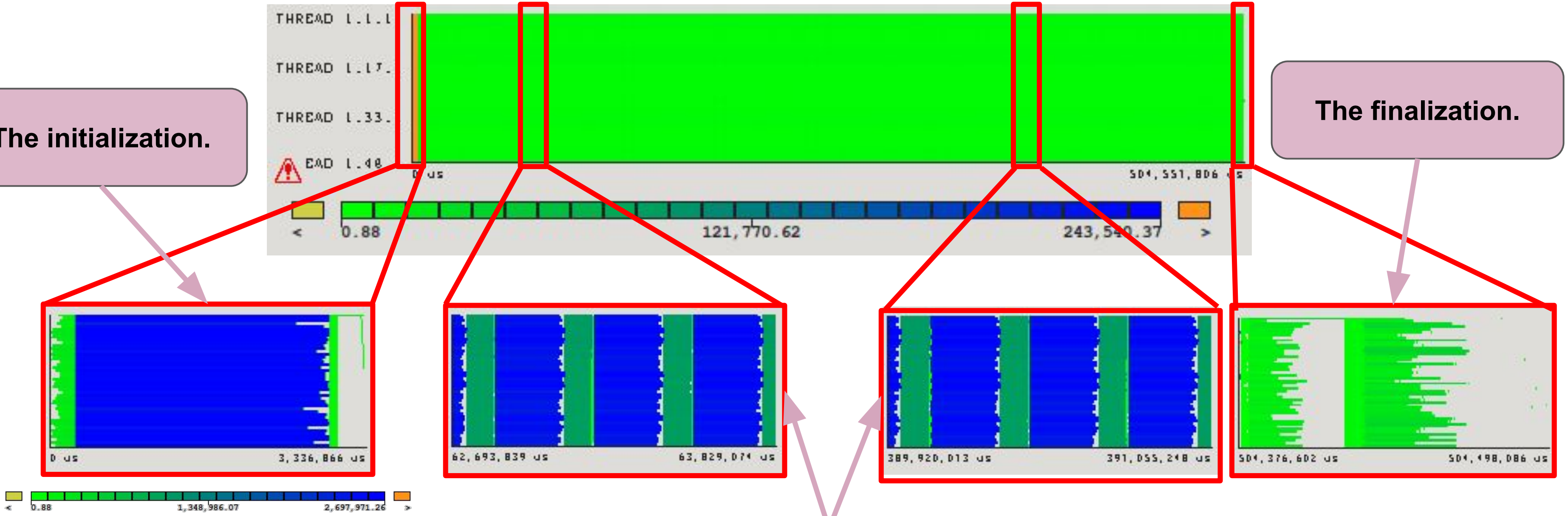
The methodology



Demo


Application execution
Application
Extrae
Trace
.prv
Trace analysis
Paraver
Performance metrics

Identify Structure



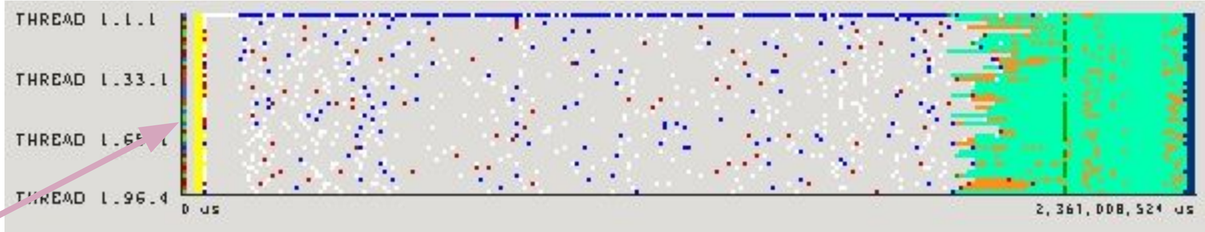
Regular iterative pattern: we observe that all the iterations across the execution have identical (visually) patterns.

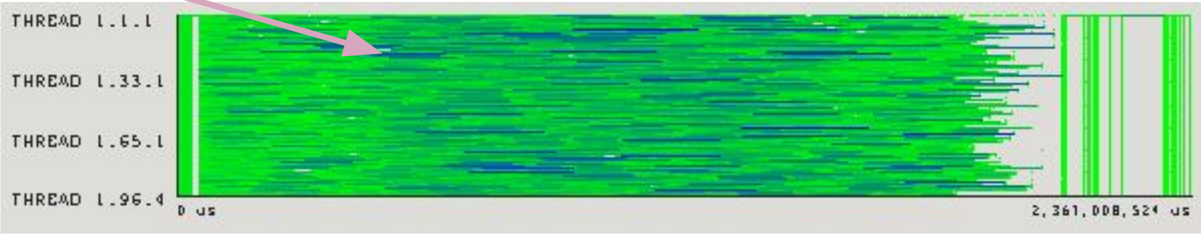
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Identify Structure

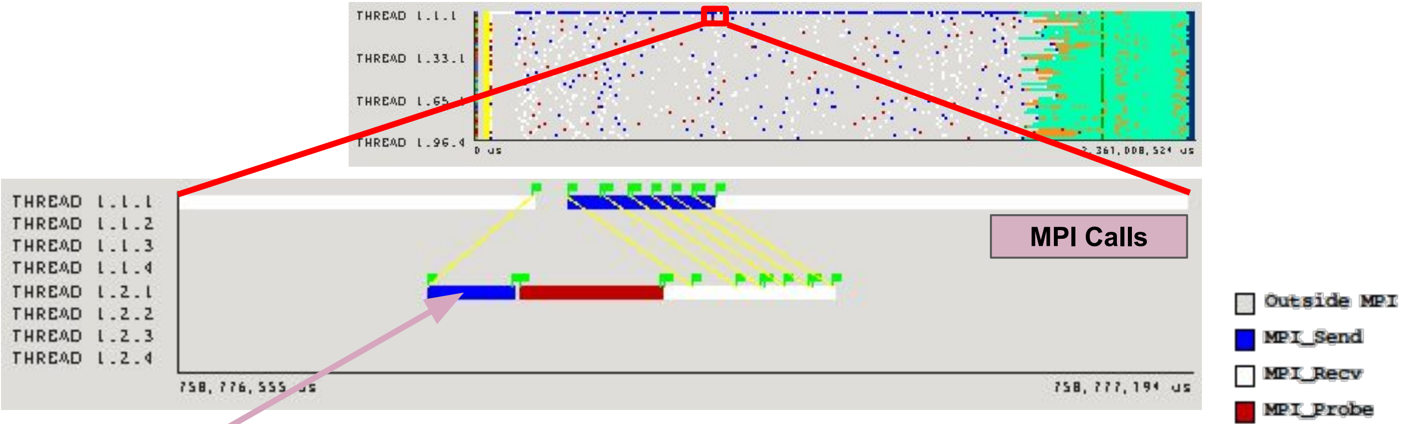
Not always its iterations!

In this example we observe: a random pattern of communication and granularities across the whole execution.





Identify Structure



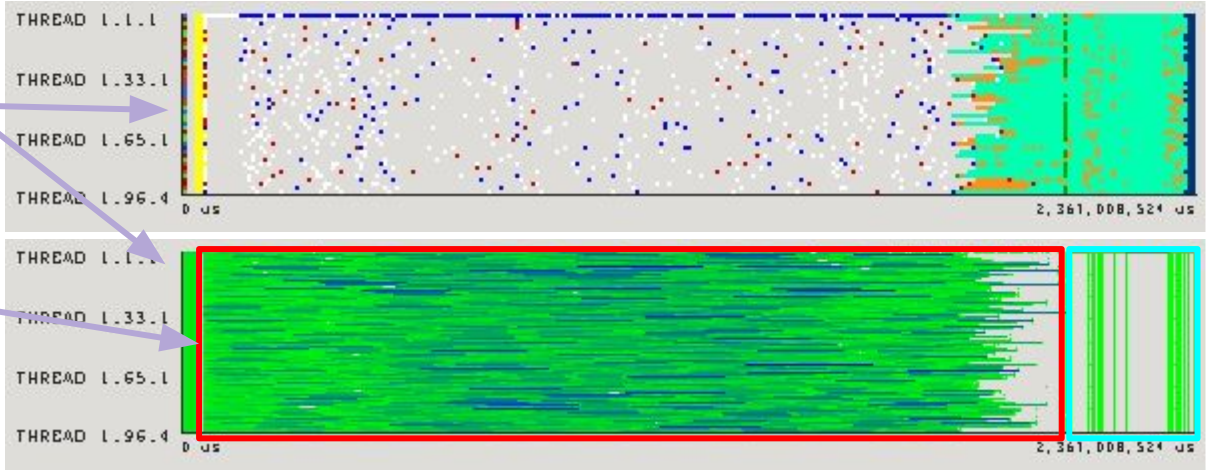
Then more detail is needed:
when the communications are studied we learn that this application uses the master/slave parallelization strategy.

Select the Focus of Analysis (FoA)

Objective: Select a region to analyze
- Not a correct answer, depends on the objective
- Usually the region that hoards more time of the execution.
- Different FoAs might be selected, then we would repeat the next steps for all regions.

Usually we use the MPI Calls and Useful duration views.

The RED and CYAN regions are two potential FoAs for this application.



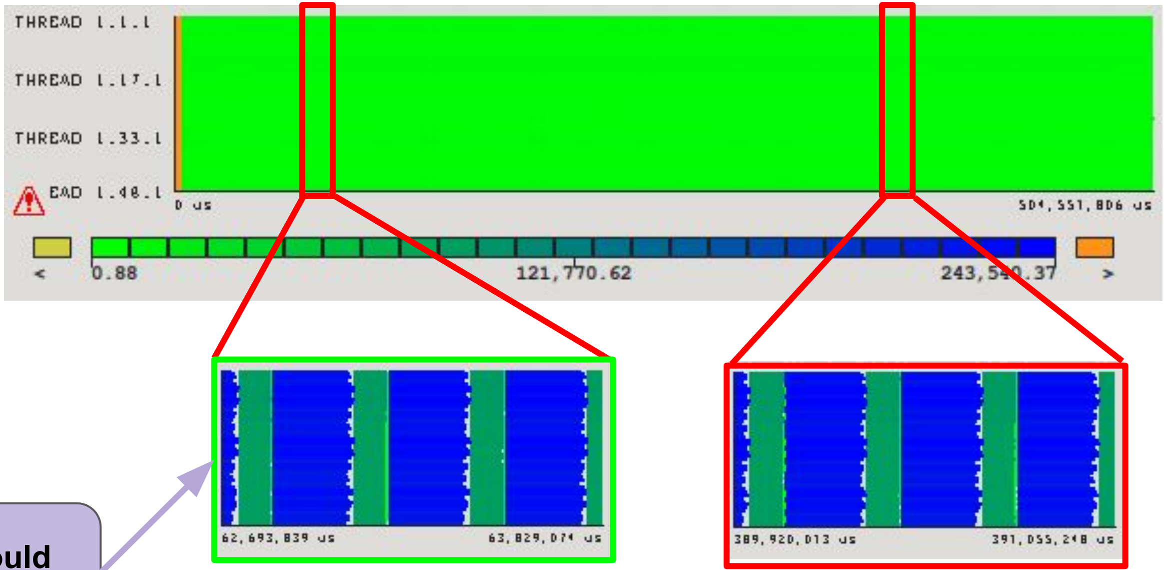
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Select the Focus of Analysis (FoA)

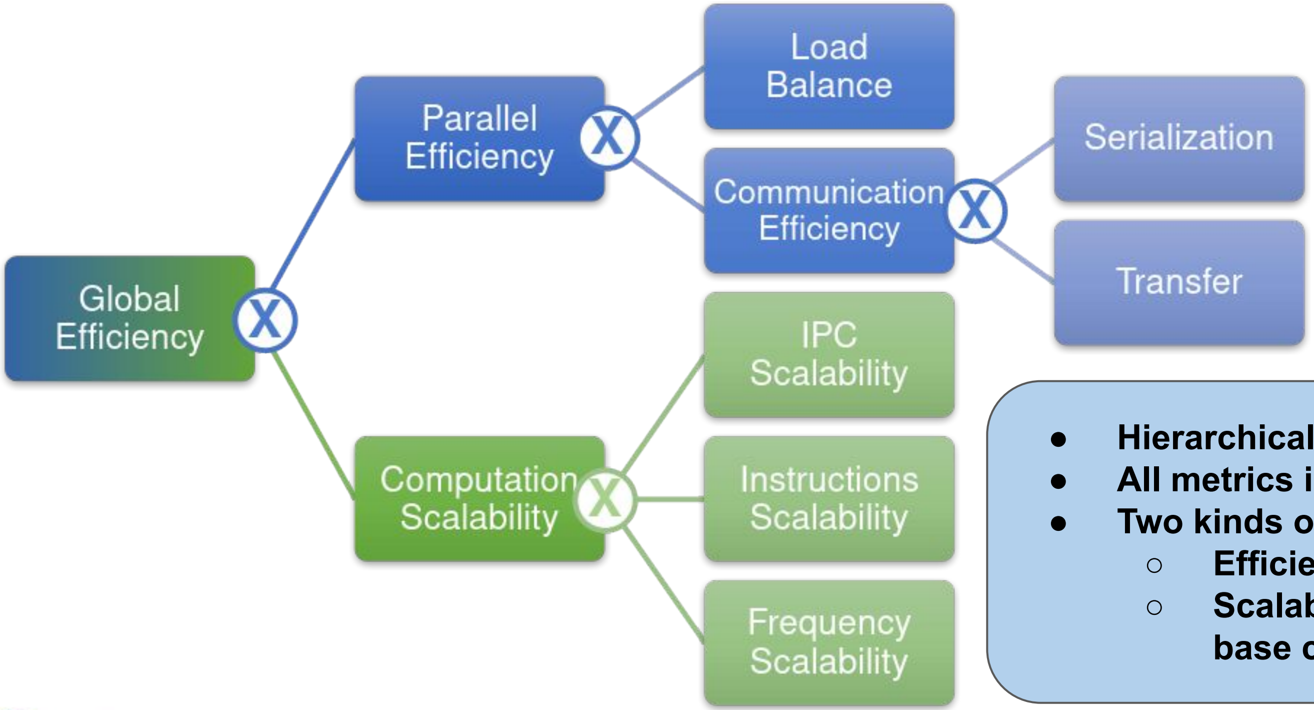
For the iterative application

We can isolate a few iterations to reduce the amount of data to process, when we observe that iterations across the execution have similar behaviour.

For example this could be the FoA



Efficiency Metrics



Model for MPI only applications!

- **Hierarchical (multiplicative) model.**
- **All metrics in %.**
- **Two kinds of metrics:**
  - **Efficiencies ( ). Are absolute.**
  - **Scalabilities ( ). Are relative to a base case.**

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Efficiency Metrics

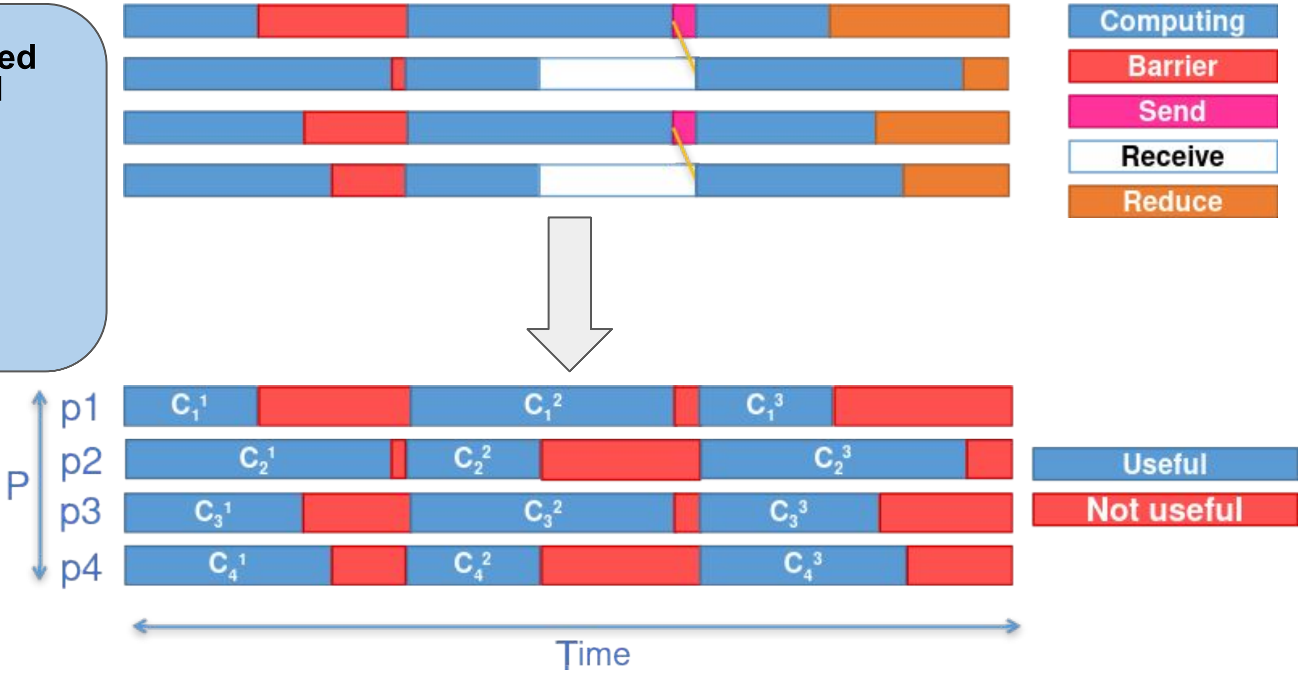
The **metrics** are computed based on **two states**: “Useful”, and “Not useful”.

All time in “**Computing**” is considered “**Useful**”.

Otherwise it's considered “**Not useful**”

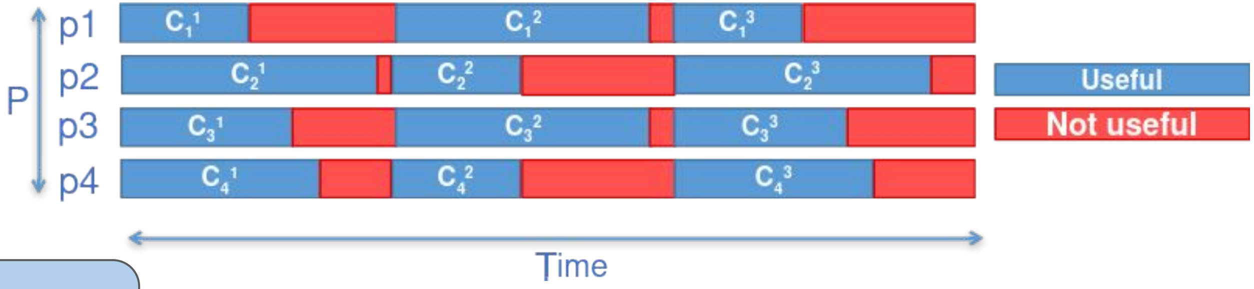
C_i^j = Compute time of the process i of the burst j.

A burst is uninterrupted computation between two programming model calls (MPI in this example)



Efficiency Metrics

Global Efficiency – Parallel Efficiency



Quantifies: The extent to which all resources in the system are kept actife doing useful work.

How it is computed: Ratio between time in the "Useful" state and time in ANY state (i.e. all consumed cpu time).

Parallel Efficiency = / (+)

Interpretation: When low indicates that a considerable amount of cpu time is spent doing not useful work.

Next step: Look at child metrics!

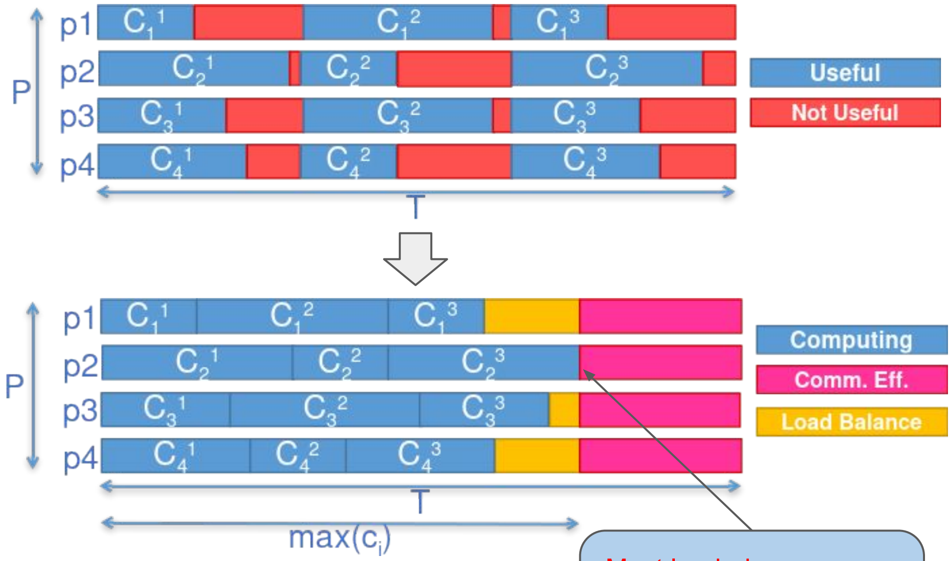
Efficiency Metrics

Global Efficiency – Parallel Efficiency – Load Balance

Quantifies: Efficiency loss caused by uneven distribution of useful computation time.

How it is computed: Ratio between the average time spent in "Useful" among processes, and the maximum time spent in "Useful" among processes.

Load Balance Efficiency = / (+)



Most loaded process draws the line between Load Balance and Comm. Eff.

Interpretation: When low indicates that computational load among different processors is uneven.

Next step: Look for computational cause to the unbalance.

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

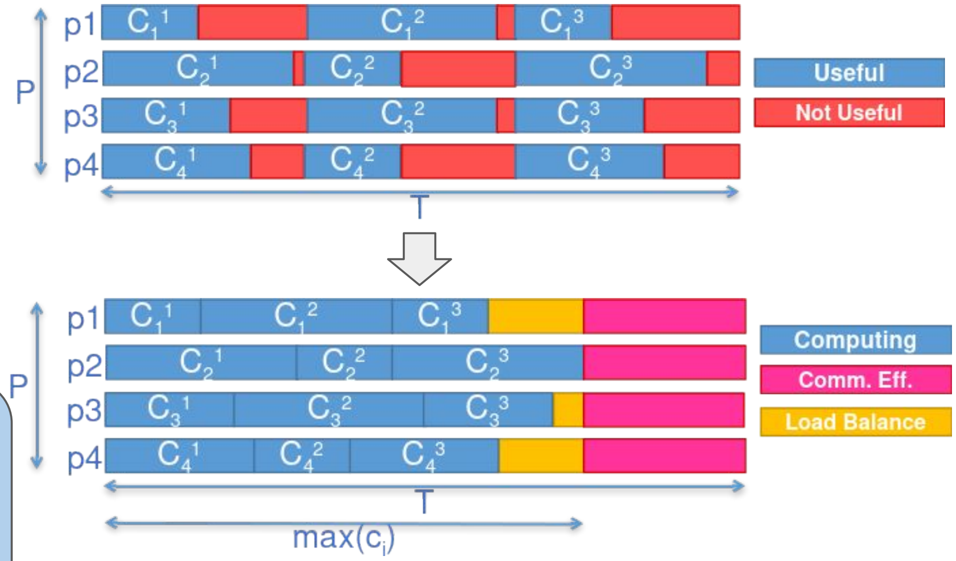
Efficiency Metrics

Global Efficiency – Parallel Efficiency – Communication Efficiency

Quantifies: Efficiency loss caused communications among processes. Including synchronizations, moving data or runtime overhead.

How it is computed: Ratio between the useful computation time of the most loaded process and the total cpu time.

Communication Efficiency = (+)/(+ +)



Interpretation: When low indicates that the interaction between processes is impacting the performance.

Next step: Look at child metrics!

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Efficiency Metrics

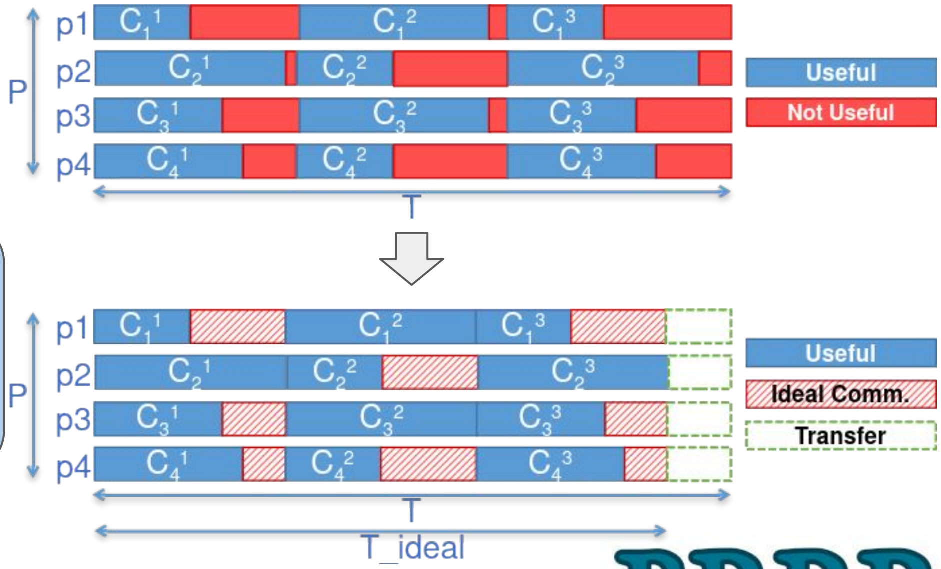
Global Efficiency – Parallel Efficiency – Communication Efficiency – Transfer

Interpretation: When low indicates overhead in the communications (including: runtime overhead, and moving data).

Quantifies: Efficiency loss caused by physical constraints in the communication system (i.e. time spent because the network and communication runtime are not ideal).

How it is computed: Ratio between the elapsed time in the ideal simulation and the elapsed time in the real execution.

Transfer Efficiency = (+)/(+ +)

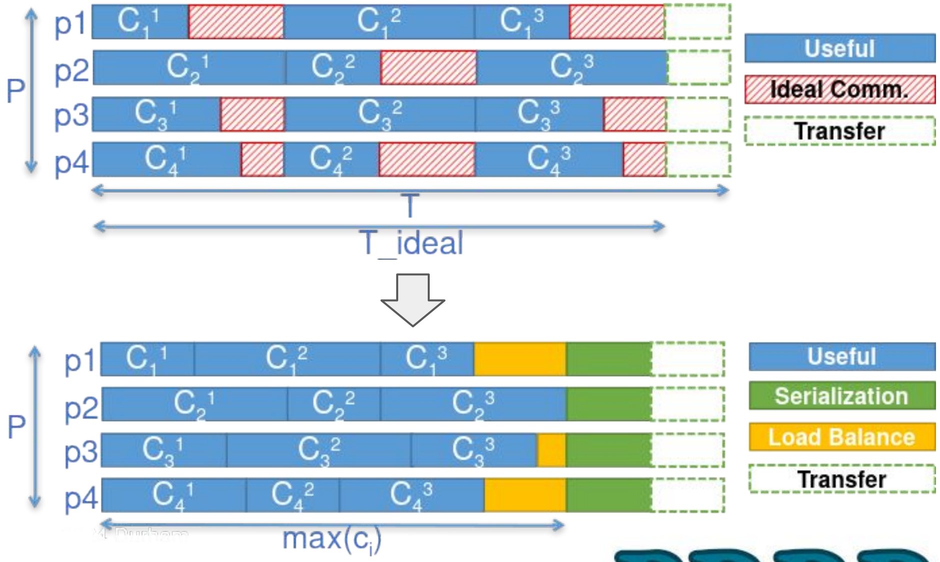


Efficiency Metrics

Global Efficiency – Parallel Efficiency – Communication Efficiency – Serialization

Interpretation: When low indicates overhead in the communications (including: runtime overhead, and moving data).

Quantifies: Efficiency loss caused by synchronization among processes, without the global load imbalance.



How it is computed: Ratio between time spent in "Useful" of the most loaded processes, and elapsed time of the simulation with ideal network.

Serialization Efficiency = (+)/(+)

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Efficiency Metrics

Global Efficiency | **Computation Scalability** | **IPC Scalability** | **Instructions Scalability** | **Frequency Scalability**

Global Efficiency / Computation Scalability

Quantifies: How the time spent doing "Useful" computation scales with respect to a reference case.

How it is computed: Ration between the useful computation time of the reference case and the useful computation time on the current run.

Interpretation: A low value indicates it takes more time to do the useful computation wrt. the reference case.

IPC Scalability

Quantifies: How the instructions per cycle change compared to the reference case.

Interpretation: A value lower than 100% indicates that the IPC is worse.

Instructions Scalability

Quantifies: How the number of instructions executed change compared to the reference case.

Interpretation: A value below 100% indicates that there might be code replication.

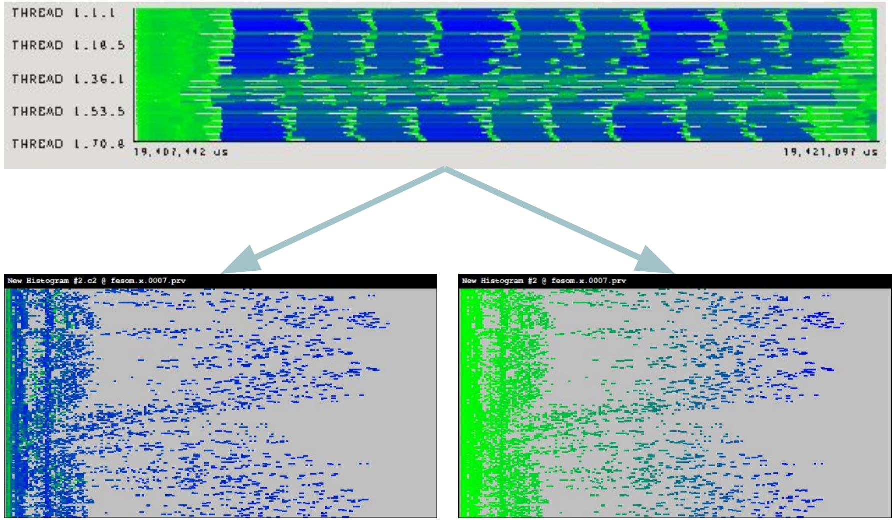
Frequency Scalability

Quantifies: How the frequency changes compared to the reference case.

Interpretation: A value lower than 100% indicates that les CPU cycles are dedicated to useful computation per unit of time. Might indicate "preemptions".

Detailed analysis

Global Efficiency – Parallel Efficiency – Load Balance



What to look for?

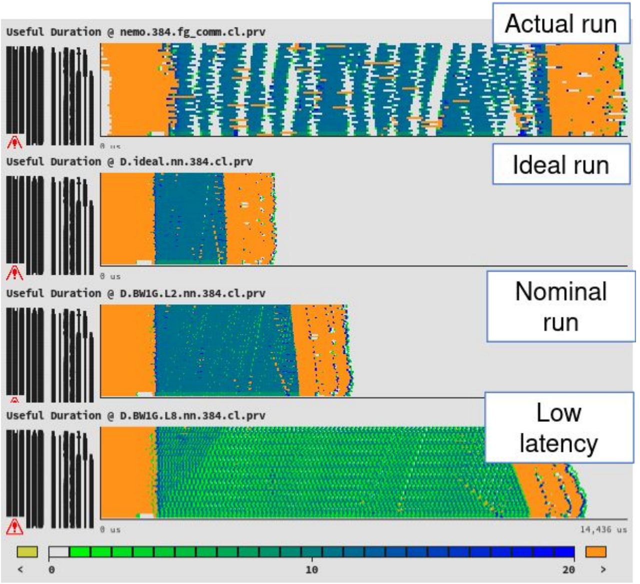
- Try to understand the cause of the imbalance.
- Generally we have 3 options, T = N * IPC * Freq, therefore the change in durations can come from:
    - Number of instructions
    - Instructions per Cycle
    - Frequency

Detailed analysis

Global Efficiency – Parallel Efficiency – Communication Efficiency – Transfer

What to look for? Find whether the transfer problem is bound to the bandwidth or the latency.

- We can use Dimemas simulations to study different conditions.
    - For example different bandwidth and latencies.



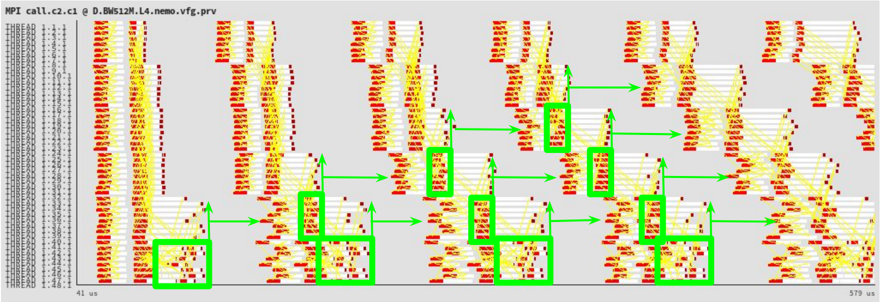
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Detailed analysis

Global Efficiency – Parallel Efficiency – Communication Efficiency – Serialization

Most common causes:

- Load Balance can hide as Serialization if the loaded processes change over the execution.
- Imbalance in number of communication calls among processes.
- Try to find the critical path in bursts of communication.

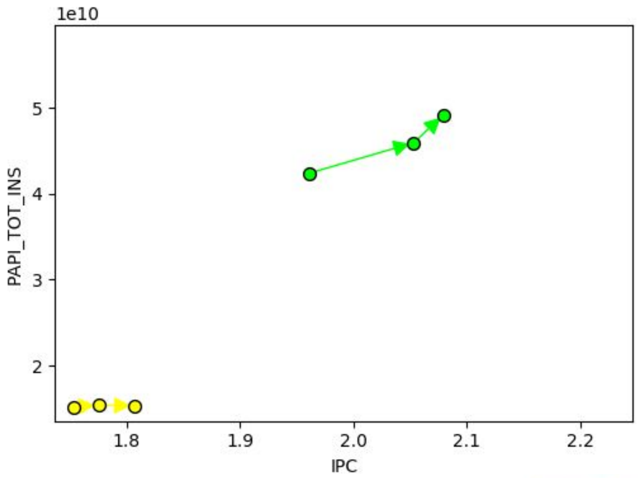


Detailed analysis

Global Efficiency | Computation Scalability | IPC Scalability | Instructions Scalability | Frequency Scalability

What is next?

- Try to correlate the changing affected metric with other counters.
    - For example: this figure show how different regions of the code evolve regarding IPC and number of instructions.
- Two other tools: Clustering and Tracking.



Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Detailed analysis

We observe that parallel efficiency has a tendency to decrease, which hints that this might be a bottleneck when running with more resources.

Instruction scalability is the main limiting factor of this application!

	4	8	16
Global efficiency	93.73	66.79	43.52
-- Parallel efficiency	93.73	90.86	89.07
-- Load balance	99.27	99.05	96.74
-- Communication efficiency	94.42	91.73	92.07
-- Serialization efficiency	98.89	94.94	94.28
-- Transfer efficiency	95.48	96.61	97.65
-- Computation scalability	100.00	73.51	48.86
-- IPC scalability	100.00	100.23	100.16
-- Instruction scalability	100.00	74.96	51.02
-- Frequency scalability	100.00	97.85	95.63

Percentage(%): 0, 20, 40, 60, 80, 100

Demo time!

Demo

We compile the application with debug symbols:

```
$ mpiicc -o demo demo.c -g
```

Then we run with Extrae loaded

```
# export EXTRAE_HOME to its installation path
$ export EXTRAE_CONFIG_FILE=”./extrae.xml”
$ mpirun -n 8 env LD_PRELOAD=”${EXTRAE_HOME}/lib/libmpitrace.so” ./demo
```

Demo

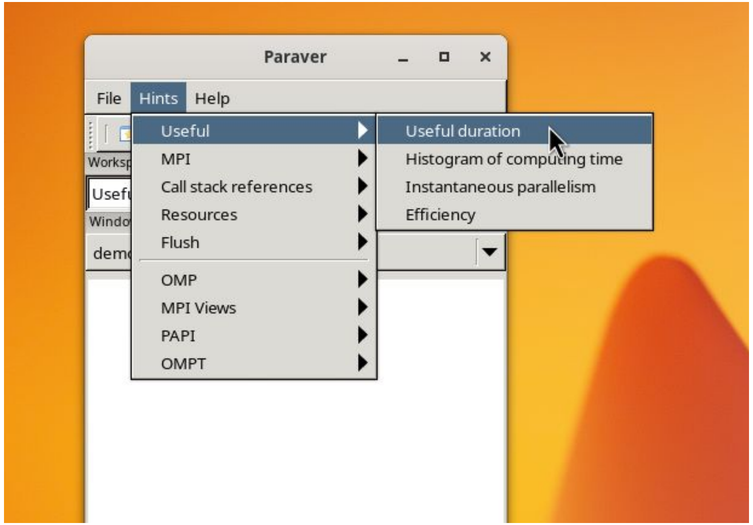
We got the trace:

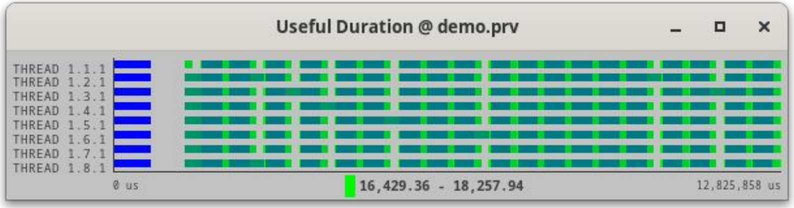
```
mpi2prv: Elapsed time removing temporal files: 0 hours 0 minutes 0 seconds
mpi2prv: Congratulations! demo.prv has been generated.
```

We can open the trace with wxparaver:

```
$ rename demo demo.8mpi demo.{prv,pcf,row}
$ wxparaver demo.8mpi.prv &
```

Demo





Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Demo

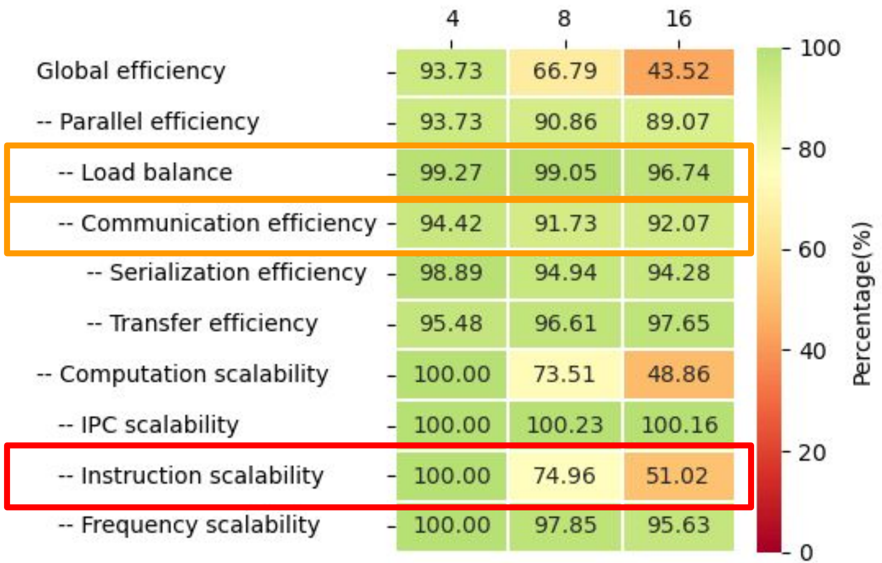
Then we generate traces for different amounts of processes:

```
# export EXTRAE_HOME to its installation path
$ export EXTRAE_CONFIG_FILE=”./extrae.xml”
$ mpirun -n 4 env LD_PRELOAD=”${EXTRAE_HOME}/lib/libmpitrace.so” ./demo
$ mpirun -n 16 env LD_PRELOAD=”${EXTRAE_HOME}/lib/libmpitrace.so” ./demo
```

And run the basicanalysis tool to compute the metrics:

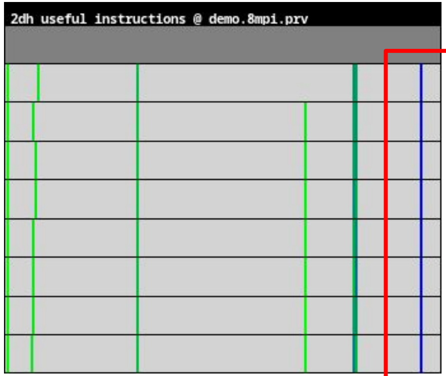
```
$ modelfactors.py demo.4mpi.prv demo.8mpi.prv demo.16mpi.prv
```

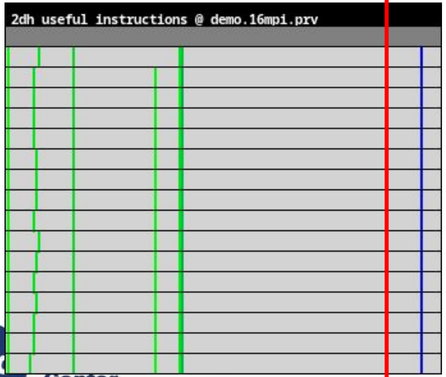
Demo

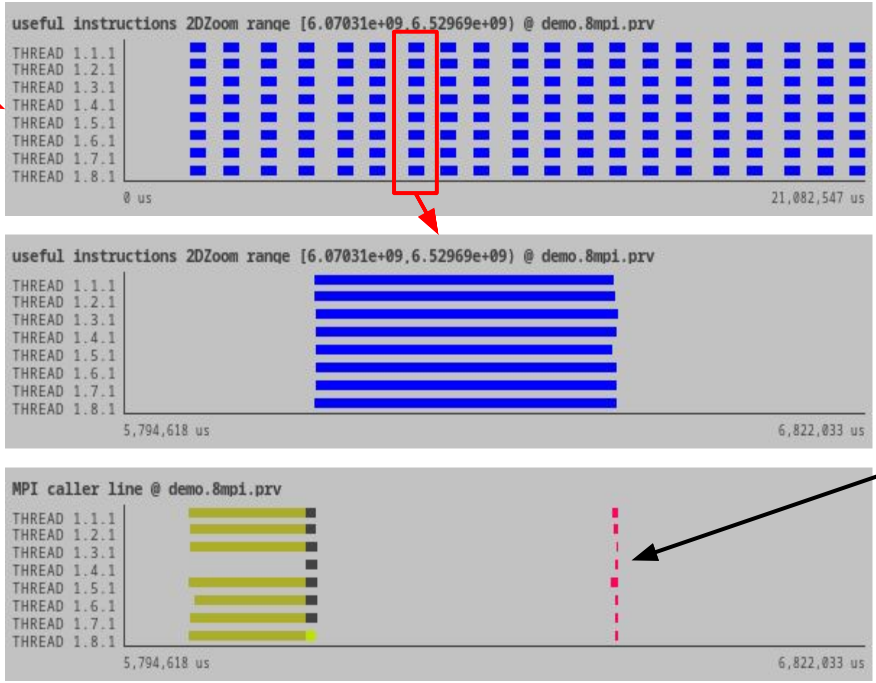

4
8
16
Global efficiency
93.73
66.79
43.52
-- Parallel efficiency
93.73
90.86
89.07
-- Load balance
99.27
99.05
96.74
-- Communication efficiency
94.42
91.73
92.07
-- Serialization efficiency
98.89
94.94
94.28
-- Transfer efficiency
95.48
96.61
97.65
-- Computation scalability
100.00
73.51
48.86
-- IPC scalability
100.00
100.23
100.16
-- Instruction scalability
100.00
74.96
51.02
-- Frequency scalability
100.00
97.85
95.63
100
80
60
40
20
0
Percentage(%)

Demo

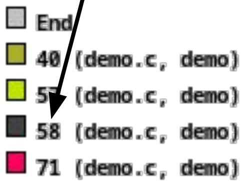
Instruction scalability







Low instruction scalability happens between the MPI calls at lines 58 and 71 of demo.c



Demo

Load balance

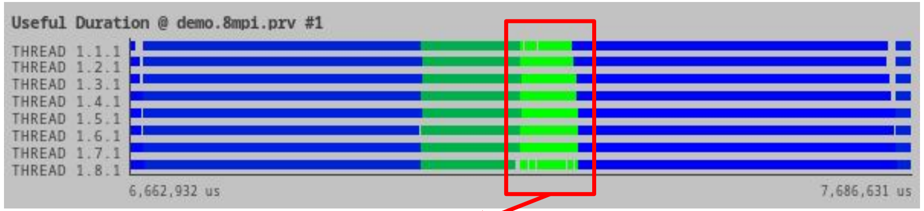
Load imbalance happens between lines 25 and 40 of the file demo.c

- End
- 25 (demo.c, demo)
- 40 (demo.c, demo)
- 57 (demo.c, demo)
- 58 (demo.c, demo)
- 71 (demo.c, demo)

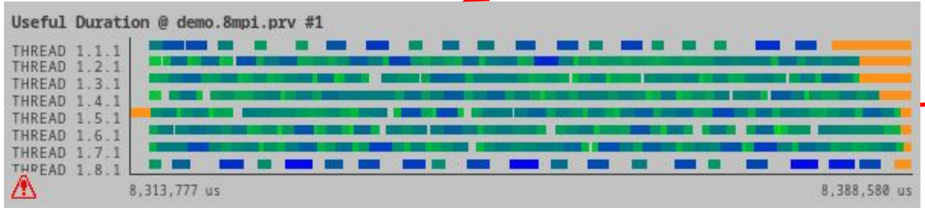
Barcelona Supercomputing Center
Centro Nacional de Supercomputación

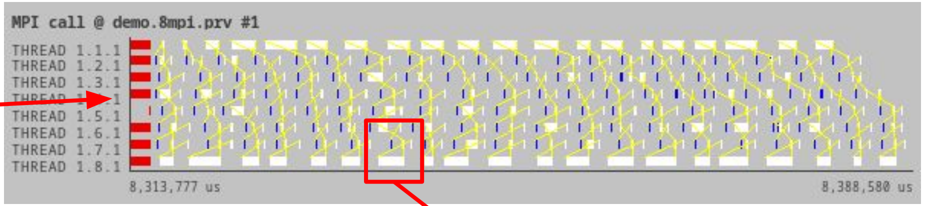
Demo

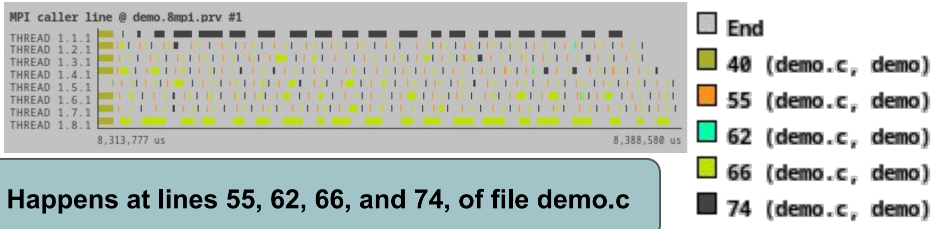
Communication Efficiency

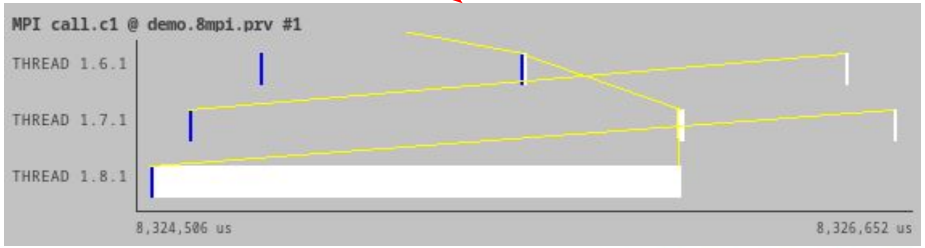


Check low granularity regions for communication inefficiencies.









Happens at lines 55, 62, 66, and 74, of file demo.c

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Demo

How to compute the metrics without a trace?

We run with TALP (DLB) loaded

```
# export DLB_HOME to its installation path
$ export DLB_ARGS="--talp --talp-papi --talp-summary=pop-metrics"
$ mpirun -n 8 env LD_PRELOAD="${DLB_HOME}/lib/libdlb_mpi.so" ./demo
```

Demo

TALP Output looks like that:

```
DLB[gs07r3b40:387149]: dlb 3.5a
DLB[gs07r3b40:387149]: ############### Monitoring Region POP Metrics ###############
DLB[gs07r3b40:387149]: ### Name:                                     Application
DLB[gs07r3b40:387149]: ### Elapsed Time:                             20.16 s
DLB[gs07r3b40:387149]: ### Average IPC:                              5.13
DLB[gs07r3b40:387149]: ### Parallel efficiency:                      0.94
DLB[gs07r3b40:387149]: ### MPI Parallel efficiency:                  0.94
DLB[gs07r3b40:387149]: ###   - MPI Communication efficiency:         0.95
DLB[gs07r3b40:387149]: ###   - MPI Load Balance:                     0.99
DLB[gs07r3b40:387149]: ###       - MPI Load Balance in:              0.99
DLB[gs07r3b40:387149]: ###       - MPI Load Balance out:             1.00
DLB[gs07r3b40:387149]: ### OpenMP Parallel efficiency:               1.00
DLB[gs07r3b40:387149]: ###   - OpenMP Load Balance:                  1.00
DLB[gs07r3b40:387149]: ###   - OpenMP Scheduling efficiency:         1.00
DLB[gs07r3b40:387149]: ###   - OpenMP Serialization efficiency:      1.00
```

Barcelona Supercomputing Center
Centro Nacional de Supercomputación

Demo time!

https://beppp-tutorials.readthedocs.io/en/latest/bsctools/tutorial.html

For more documentation also

https://beppp-tutorials.readthedocs.io/

Performance Analysis:

Methodology, Tools, and Metrics

Joan Vinyals Ylla-Català
joan.vinyals@bsc.es
Barcelona Supercomputing Center