The BSC performance tools and their role in POP3 services

Marta Garcia-Gasulla, BSC

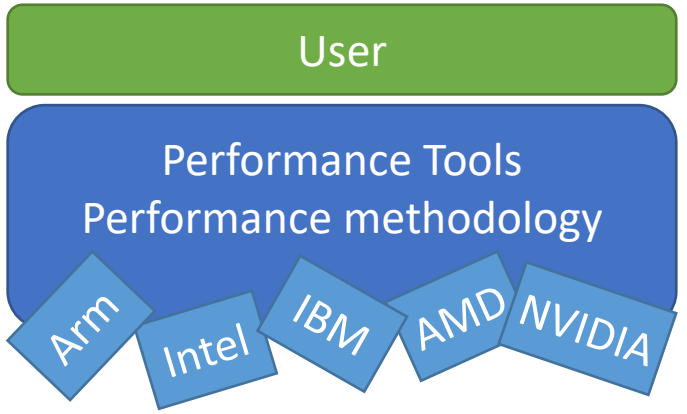
HORIZON-EUROHPC-JU-2023-COE

EuroHPC Joint Undertaking

1 January 2024– 31 December 2026

Grant Agreement No 101143931

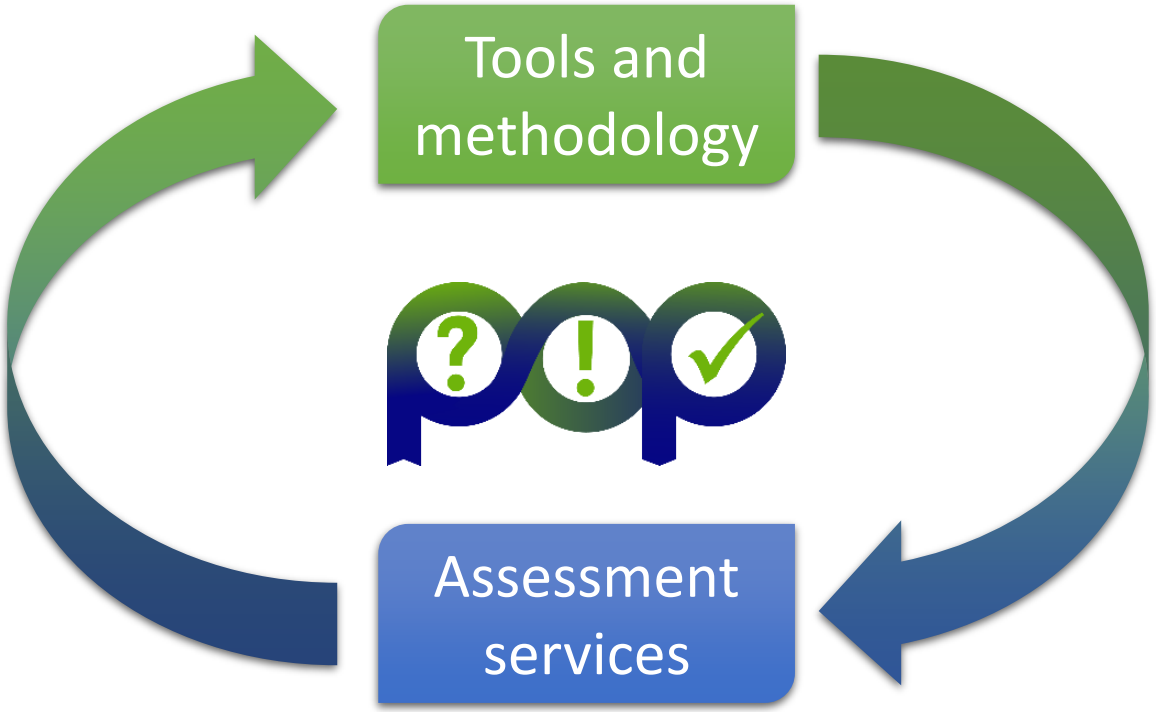
POP CoE



- A **Centre of Excellence**
  - On **Performance Optimisation and Productivity**
  - Promoting **best practices in parallel programming**
- Providing **FREE Services**
  - Precise understanding of application and system behaviour
  - Suggestion/support on how to refactor code in the most productive way
- **Horizontal**
  - Transversal across application areas, platforms, scales
- **For (EuroHPC) academic AND industrial codes and users !**

POP3 Vision

POP3 Vision

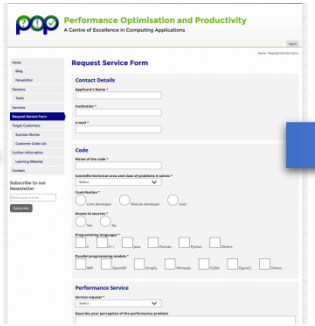


POP3 services

POP3 services

Online questionnaire

https://pop-coe.eu/request-service-form



User or developer

Who will obtain the traces?

expert

Client provides code, input and instructions on how to compile and run.

User or developer

POP expert provides instructions on how to install tracing tools and obtain traces.

Parallel Application Performance Assessment

traces

expert

Report

The report includes suggestions to improve performance

User or developer

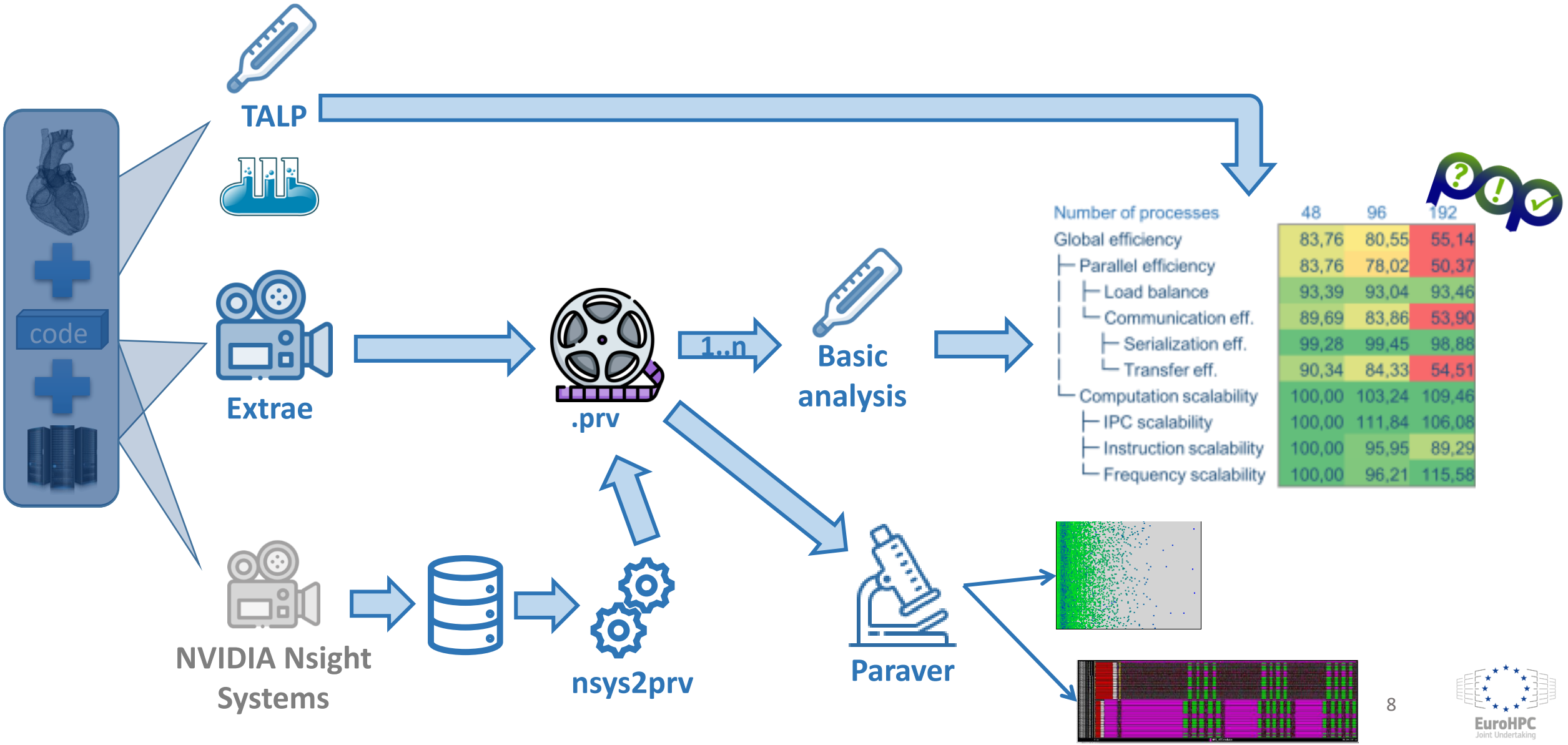
Code developer asks for a second level service

Second level services:
- **Proof of concept**
- **Advisory**
- **Correctness check**
- **Energy efficiency**

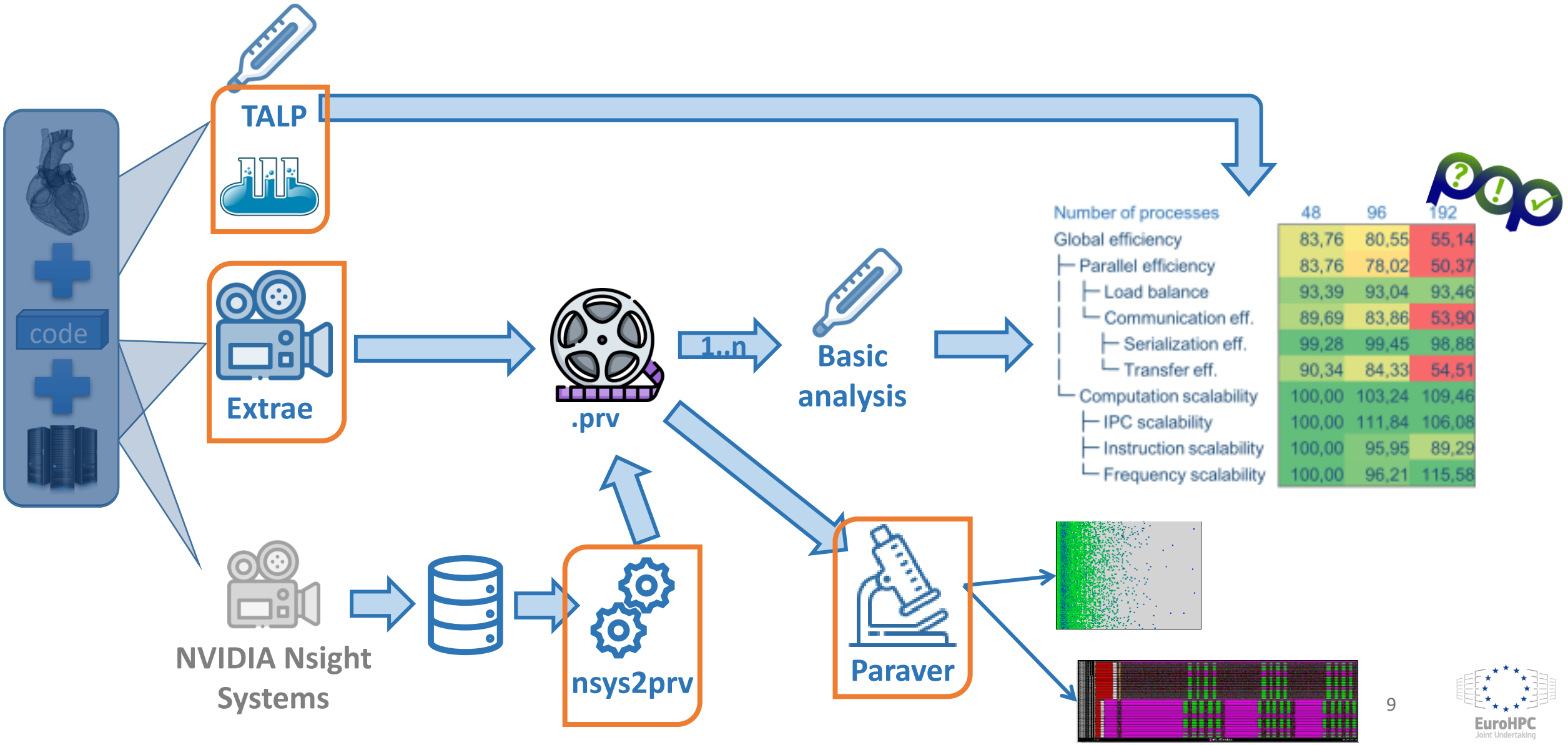
EuroHPC Joint Undertaking

The ecosystem of BSC performance tools

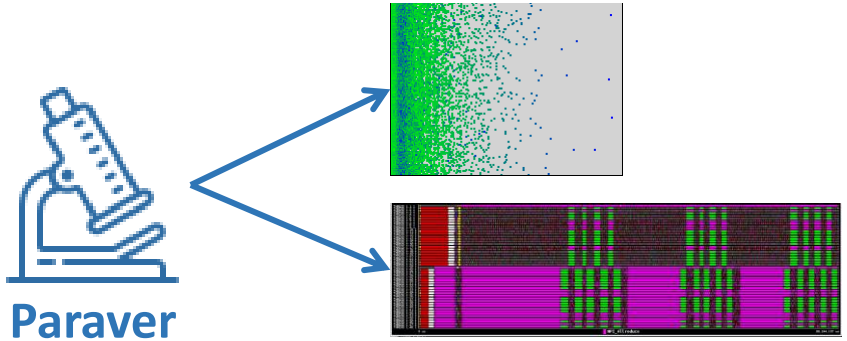
Tracing and profiling tools



Tracing and profiling tools



Advanced analysis tools

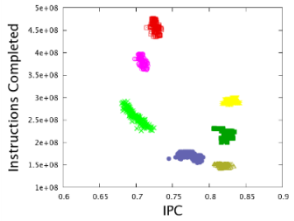


Paraver

.prv

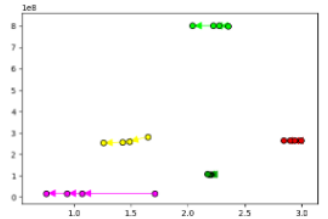
Clustering

.prv
+ clusterID



1..n

Tracking



Dimemas

.prv

Extrae

No need to recompile or relink

- Tracing library transparent to the application
- Platform agnostic
- Parallel programming models:
  - MPI, OpenMP, pthreads, OmpSs, CUDA, OpenCL, Java, Python
- Hardware counters
  - Through PAPI
- Link to source:
  - Callstack at MPI routines
  - OpenMP outlined routines
  - Selected user functions (Dyninst)
- Periodic sampling
- User friendly API to annotate your code with custom events

How to use Extrae

- Symbol substitution through LD_PRELOAD

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

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

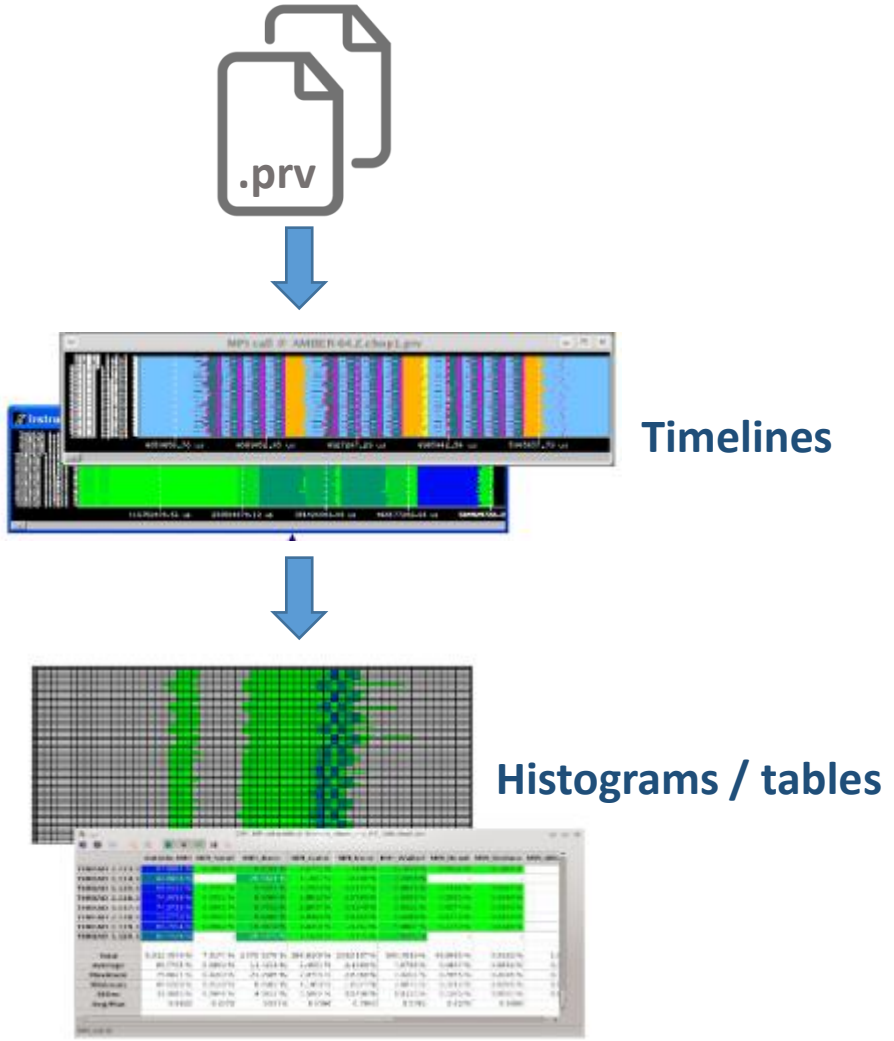
```
libmpitrace.so      libomptrace.so
libmpitracecf.so    libptmpitrace.so
libmpitracef.so     libptmpitracecf.so
libompitrace.so     libptmpitracef.so
libompitracecf.so   libpttrace.so
libompitracef.so    libseqtrace.so
```

- Detailed configuration in XML file

```
export EXTRAE_CONFIG_FILE=../extrae.xml
```

Paraver

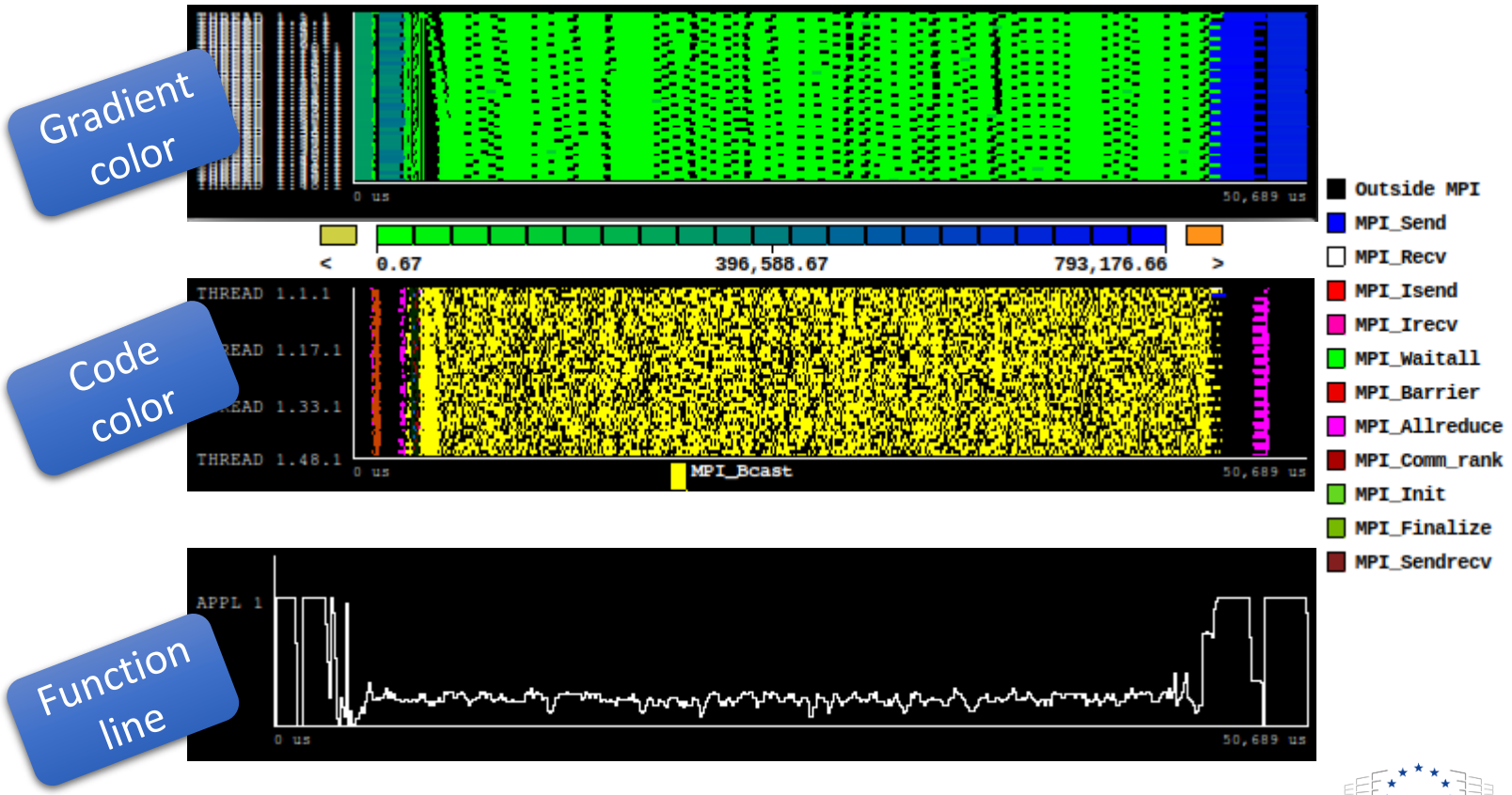
- (Performance) Data Browser
  - Any kind of timestamped data
  - Trace Visualization and analysis
  - Trace manipulation
  - Flexible
    - No pre-assumed semantics
    - Fully programmable
  - 2 Kind of views:
    - Timeline
    - Histograms, 2D and 3D tables
  - Multiple loaded traces
    - Allow comparative analysis



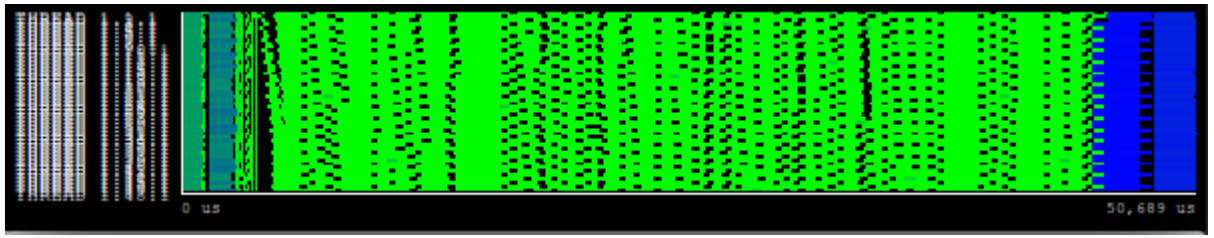
Paraver: Timelines

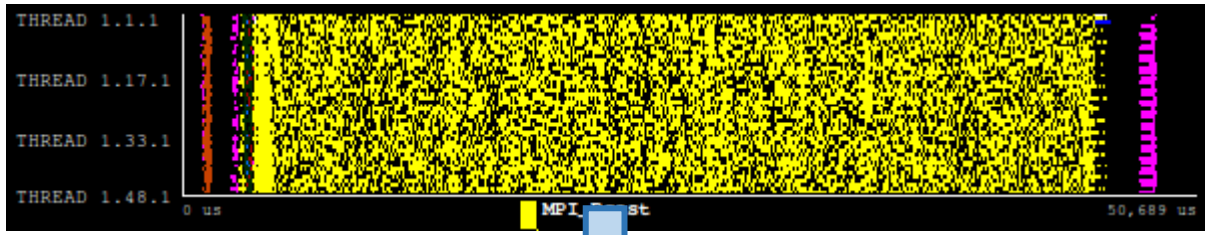
Separate the *What* from the *How*

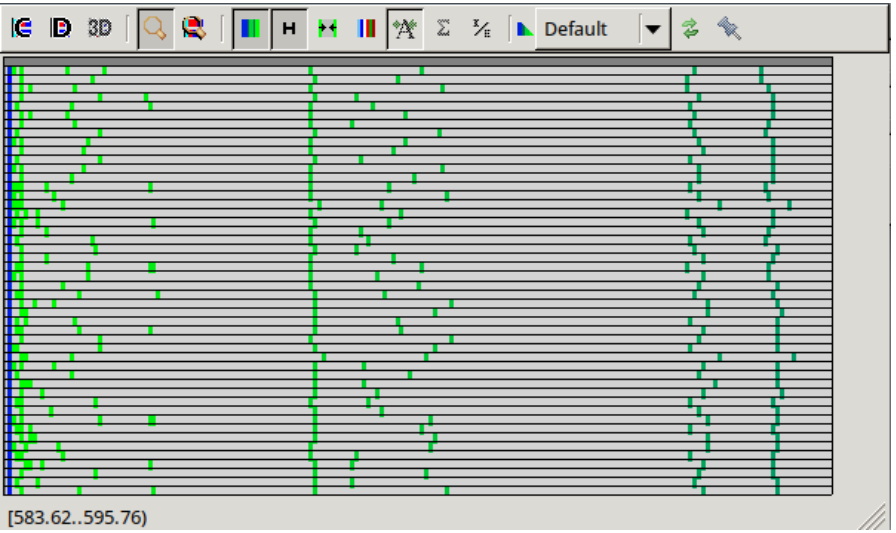
- *What* -> Semantics
  - Which event to visualize
  - Which value to use
  - How to combine them
- How -> Visual representation



Paraver: Histograms







	Outside MPI	MPI_Send	MPI_Recv	MPI_Bcast	MPI_Allreduce	MPI_Comm_rank	MPI_Comm_size
THREAD 1.1.1	23.74 %	0.26 %	1.40 %	72.80 %	0.48 %	0.03 %	0.03 %
THREAD 1.2.1	24.55 %	0.19 %	0.23 %	72.50 %	0.77 %	0.03 %	0.03 %
THREAD 1.3.1	22.63 %	1.47 %	0.34 %	73.42 %	0.38 %	0.03 %	0.03 %
THREAD 1.4.1	23.79 %	-	-	72.27 %	2.29 %	0.03 %	0.03 %
THREAD 1.5.1	23.47 %	-	-	73.83 %	1.06 %	0.03 %	0.02 %
THREAD 1.6.1	23.34 %	-	-	72.99 %	2.03 %	0.03 %	0.03 %
THREAD 1.7.1	22.74 %	-	-	73.19 %	2.36 %	0.04 %	0.03 %
THREAD 1.8.1	23.58 %	-	-	74.02 %	0.69 %	0.03 %	0.03 %
THREAD 1.9.1	22.66 %	-	-	73.59 %	2.05 %	0.04 %	0.03 %
THREAD 1.10.1	22.93 %	-	-	73.34 %	2.10 %	0.04 %	0.02 %
THREAD 1.11.1	23.02 %	-	-	74.22 %	1.11 %	0.03 %	0.03 %
THREAD 1.12.1	23.94 %	-	-	72.64 %	1.78 %	0.03 %	0.03 %
THREAD 1.13.1	23.04 %	-	-	73.28 %	1.99 %	0.04 %	0.03 %
THREAD 1.14.1	24.67 %	-	-	72.64 %	0.99 %	0.03 %	0.03 %
THREAD 1.15.1	22.88 %	-	-	73.61 %	1.80 %	0.03 %	0.03 %
THREAD 1.16.1	22.96 %	-	-	73.46 %	1.98 %	0.04 %	0.03 %
THREAD 1.17.1	22.96 %	-	-	74.58 %	0.88 %	0.03 %	0.03 %
THREAD 1.18.1	23.72 %	-	-	72.61 %	2.09 %	0.03 %	0.03 %
THREAD 1.19.1	23.33 %	-	-	72.71 %	2.28 %	0.04 %	0.03 %
THREAD 1.20.1	22.98 %	-	-	74.26 %	1.09 %	0.03 %	0.03 %
THREAD 1.21.1	23.03 %	-	-	73.04 %	2.27 %	0.03 %	0.03 %
THREAD 1.22.1	22.96 %	-	-	73.35 %	2.06 %	0.04 %	0.02 %

TALP: Tracking Application Live Performance

- TALP is a lightweight tool to gather efficiency metrics
  - At finalization
    - Allows continuous performance monitoring
    - Integration with CI/CD systems (TALP-pages)
  - At runtime
    - Allows dynamic adjustment of execution
      - Enable load balancing
      - Dynamic resource management
  - Provides API to annotate regions and maximize the information gathered
    - Allows to obtain metrics per region
  - Will indicate when detailed analysis using traces is needed

Version 3.6.0-beta1

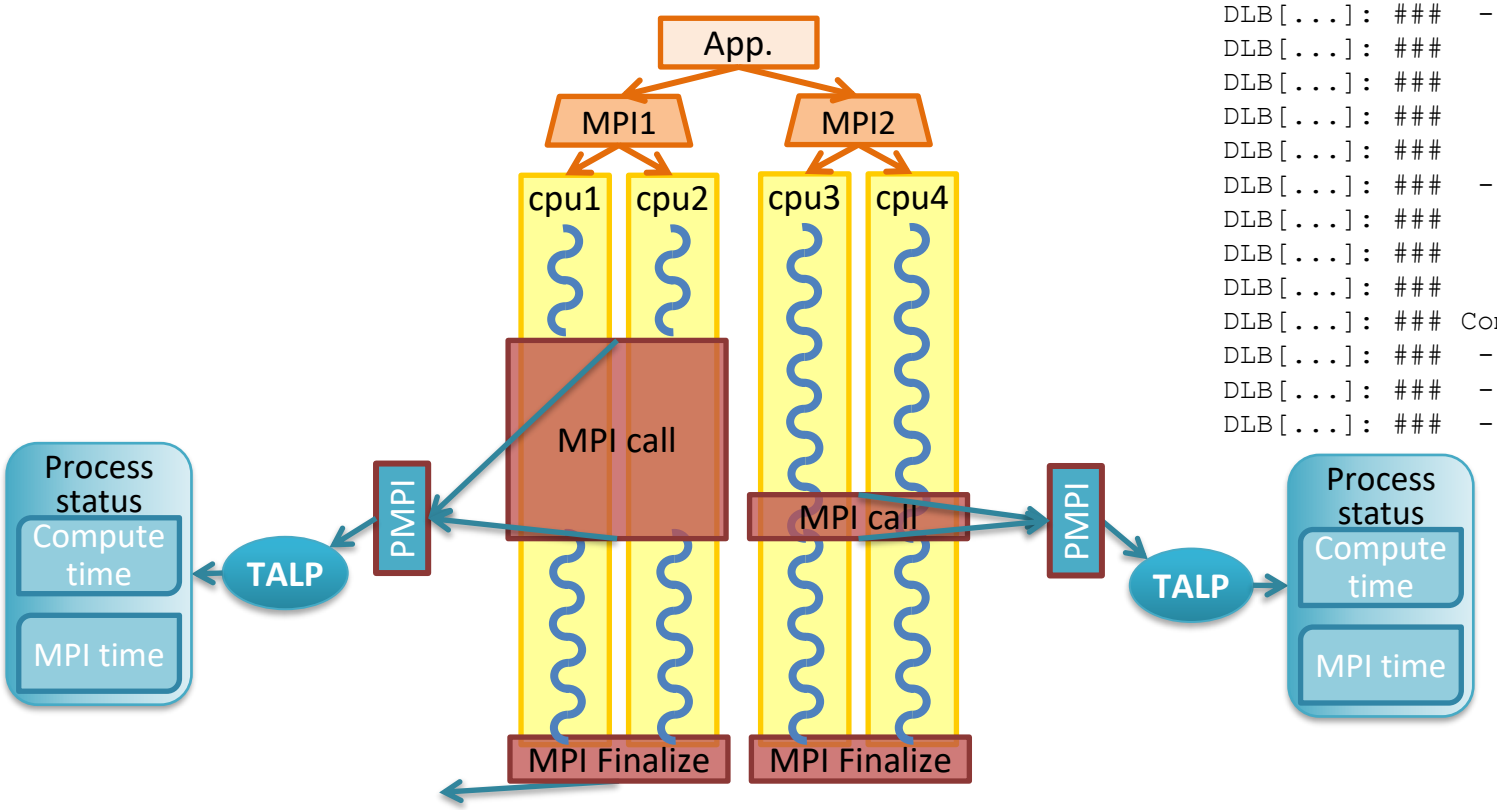
- MPI efficiency metrics
- Hardware counters (cycles, instructions, IPC, and frequency)
- OpenMP efficiency metrics
- TALP-pages (CI/CD integration)
- GPU efficiency metrics (support for CUDA, ROCm and OpenACC)

TALP a lightweight tool to Unveil Parallel Efficiency of Large Scale Executions. *In Proceedings of Performance Engineering, Modelling, Analysis, and Visualization Strategy (Permavost 2021).*

TALP: Transparent use for the user

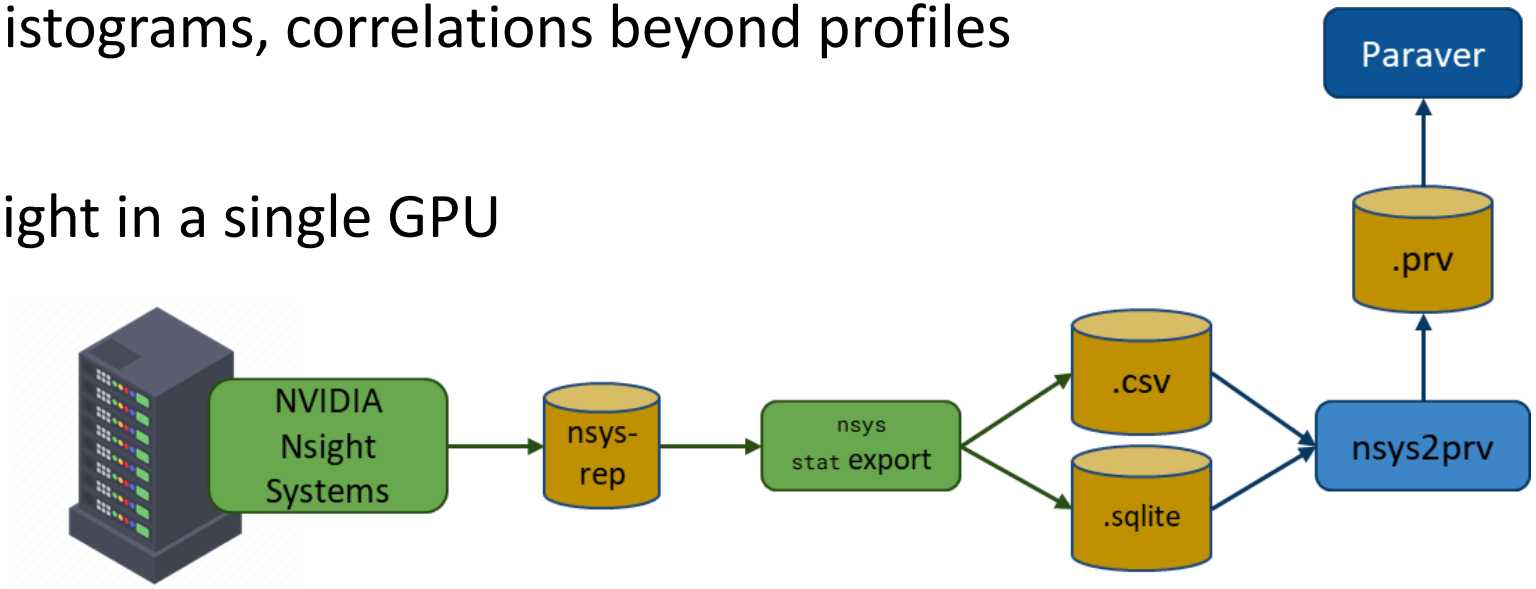
```
DLB_ARGS=" --talp"
env LD_PRELOAD="$DLB_LIBS/libdlb_mpi.so" ./app
```

```
DLB[...]: ############### Monitoring Region POP Metrics ###############
DLB[...]: ### Name:                                      Global
DLB[...]: ### Elapsed Time:                              31.76 s
DLB[...]: ### Parallel efficiency:                       0.70
DLB[...]: ###   - MPI Parallel efficiency:               0.70
DLB[...]: ###       - Communication efficiency:          1.00
DLB[...]: ###       - Load Balance:                      0.70
DLB[...]: ###           - In:                            0.70
DLB[...]: ###           - Out:                           1.00
DLB[...]: ###   - OpenMP Parallel efficiency:           0.84
DLB[...]: ###       - Load Balance:                      1.00
DLB[...]: ###       - Scheduling efficiency:             1.00
DLB[...]: ###       - Serialization efficiency:          0.84
DLB[...]: ### Computational metrics:
DLB[...]: ###   - Average useful IPC:                    0.59
DLB[...]: ###   - Average useful frequency:              2.95 GHz
DLB[...]: ###   - Number of instructions:                1.55E+11
```



nsys2prv

- Can we leverage NVIDIA´s acquisition capability …
- and use Paraver to better squeeze the information in NVIDIA traces ?
  - Scalability: dynamic range from very coarse to extremely fine grain
  - Analytics: timelines, histograms, correlations beyond profiles
- Allows us to analyze…
  - … from microscopic insight in a single GPU
  - … to large clusters



M. Clascá et al. **"NSYS2PRV: detailed and quantitative analysis of large-scale GPU execution traces with Paraver"**. *Cluster 2025*
https://pypi.org/project/nsys2prv/

TALP + Extrae + Paraver in action

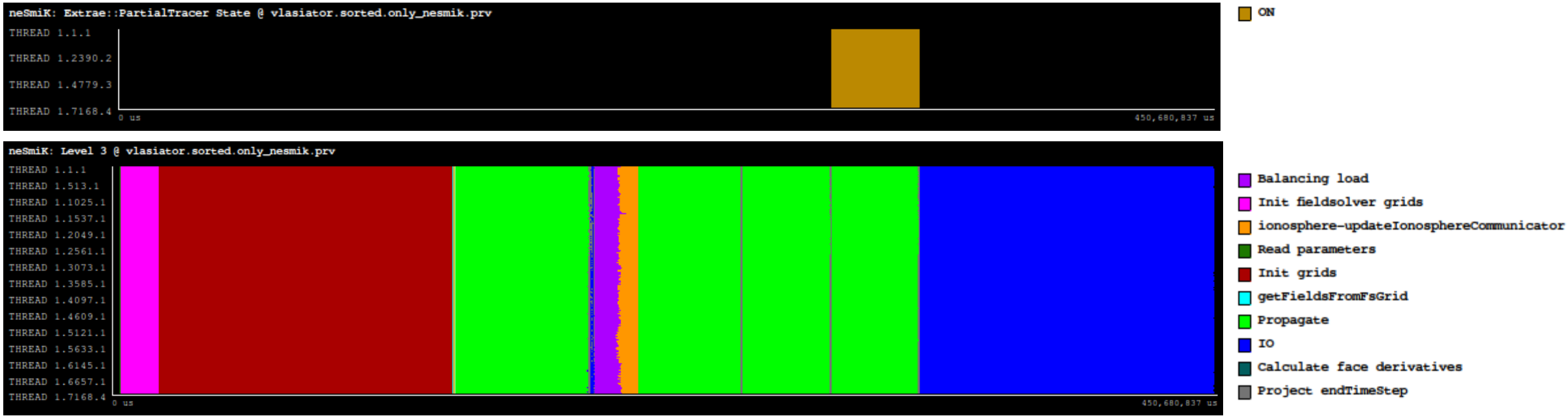
Metrics for **Propagate** slice

Metrics	7168xMPI 4xOpenMP
Global Effiency	0.15
- Parallel efficiency	0.15
-- MPI Parallel efficiency	0.25
--- MPI Communication efficiency	0.79
--- MPI Load balance	0.32
---- MPI In-node load balance	0.38
---- MPI Inter-node load balance	0.83
-- OpenMP Parallel efficiency	0.36
--- OpenMP Scheduling efficiency	1
--- OpenMP Load balance	1
--- OpenMP Serialization efficiency	0.36
- Computation Scalability	1
-- Instructions scaling	1
-- IPC scaling	1
-- Frequency scaling	1
Useful IPC	1.43
Frequency [GHz]	2.91
Elapsed time [s]	226.79

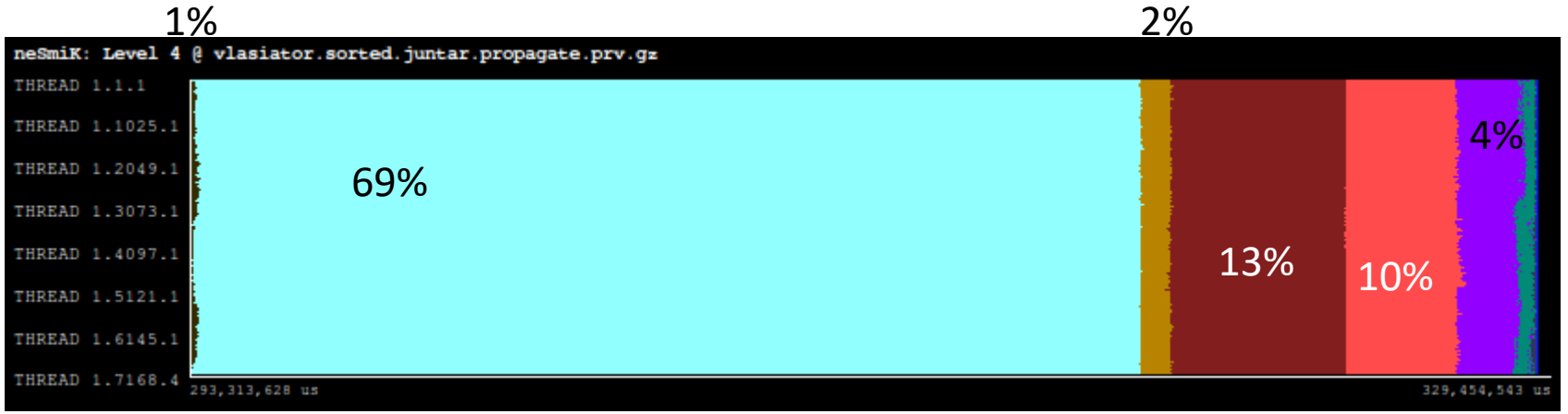
- We check the metrics obtained with TALP for the relevant region

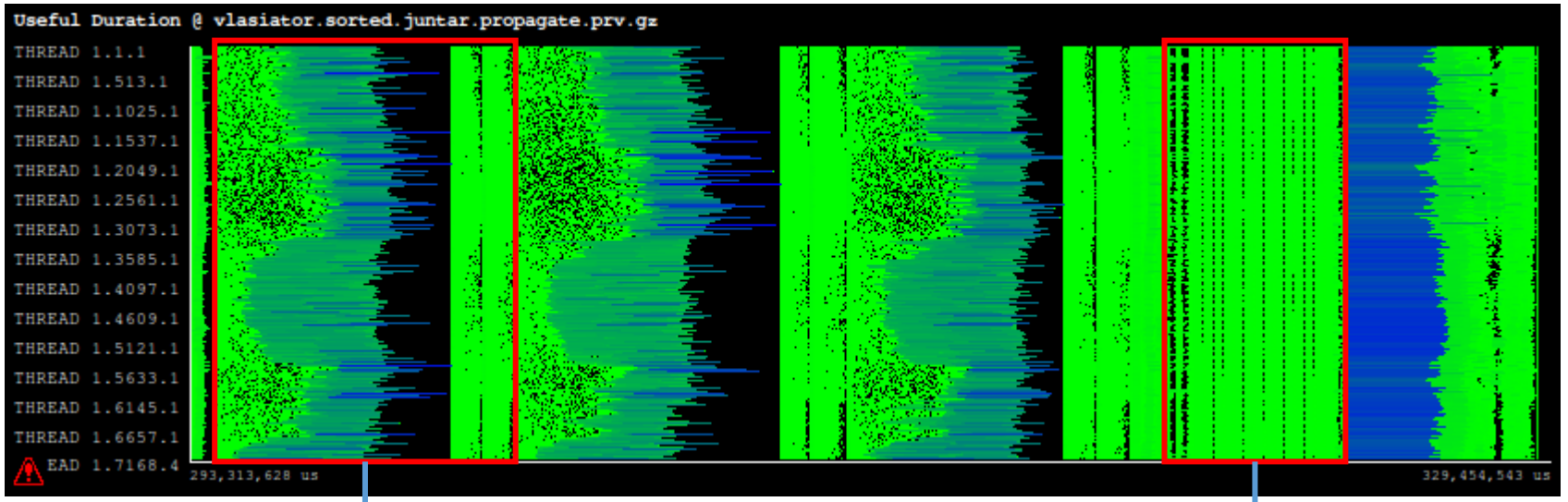
Macro-Structure

- Partial-tracing set to trace fourth iteration of time-step region: Propagate



Propagate





- Update system boundaries (Vlasov pre-translation)
- Velocity-space
- Propagate Fields
- Update system boundaries (Vlasov post-translation)
- Spatial-space
- Bailout-allreduce
- Update system boundaries (Vlasov post-acceleration)
- logfile-io
- write-restart

Spatial-space

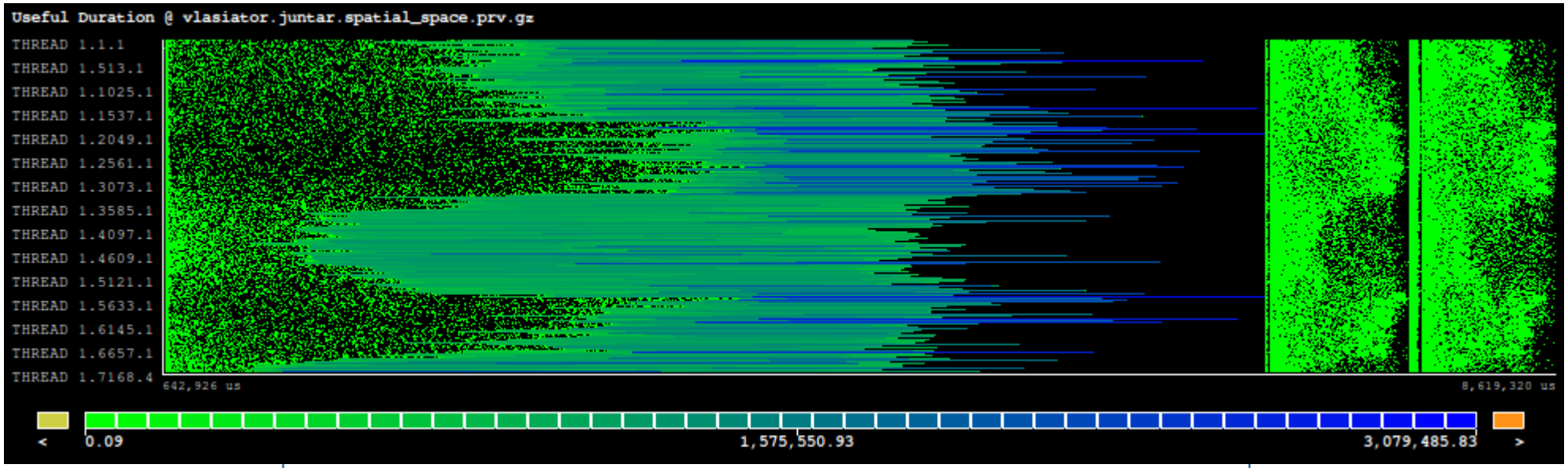
Propagate Fields

POP metrics Spatial-space

Metrics	Spatial-space
Global Effiency	0.11
- Parallel efficiency	0.11
-- MPI Parallel efficiency	0.15
--- MPI Communication efficiency	0.86
--- MPI Load balance	0.18
---- MPI In-node load balance	0.24
---- MPI Inter-node load balance	0.74
-- OpenMP Parallel efficiency	0.33
--- OpenMP Scheduling efficiency	1
--- OpenMP Load balance	1
--- OpenMP Serialization efficiency	0.33
- Computation Scalability	1
-- Instructions scaling	1
-- IPC scaling	1
-- Frequency scaling	1
Useful IPC	0.58
Frequency [GHz]	2.96
Elapsed time [s]	148

- Metrics for the Spatial-space region
- Main issues:
  - MPI Load imbalance
  - OpenMP serialization

Spatial-space

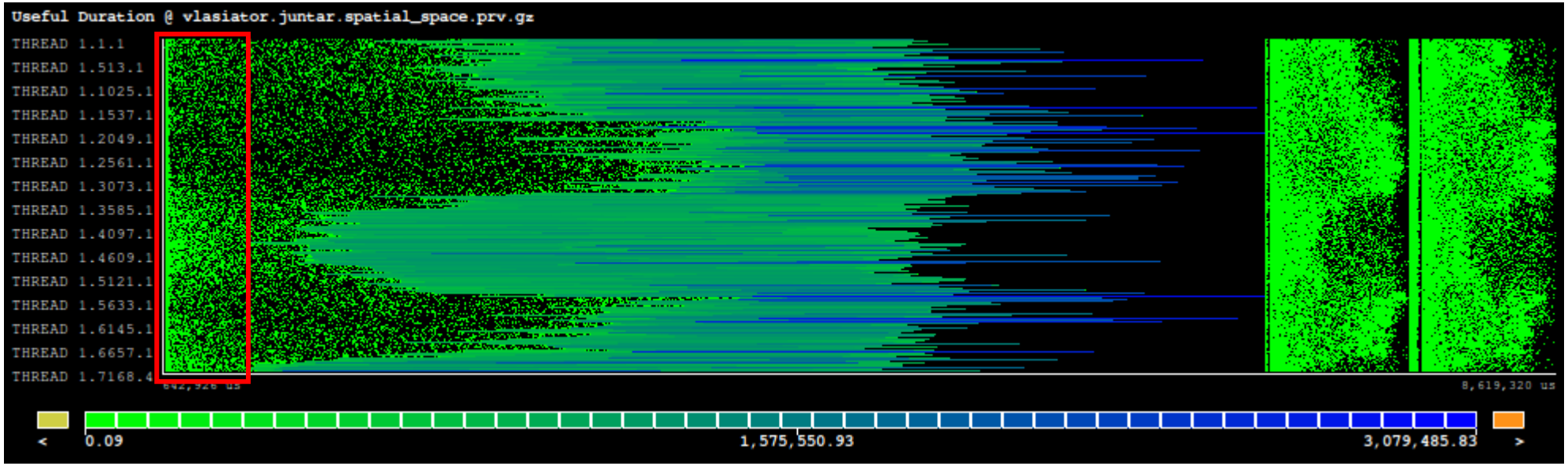


transfer-stencil-data-z
unneeded serialization of MPI_Waitalls

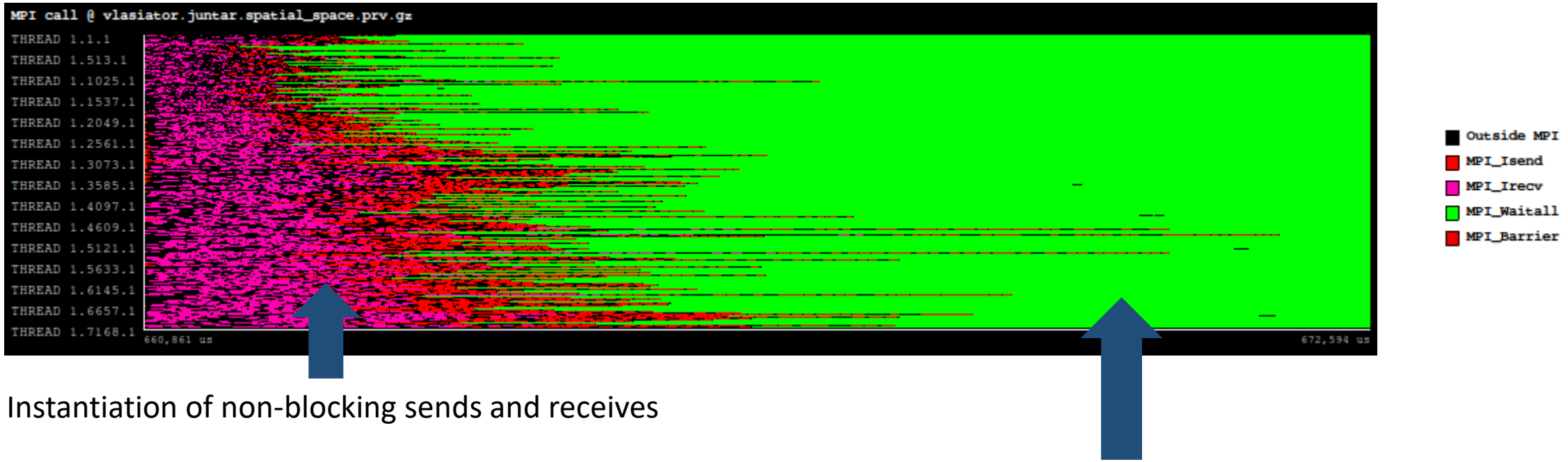
compute-mapping-z
Instruction load imbalance on MPI level

update_remote-z
same comm behavior, less data

Spatial-space::transfer-stencil-data-z



Spatial-space::transfer-stencil-data-z

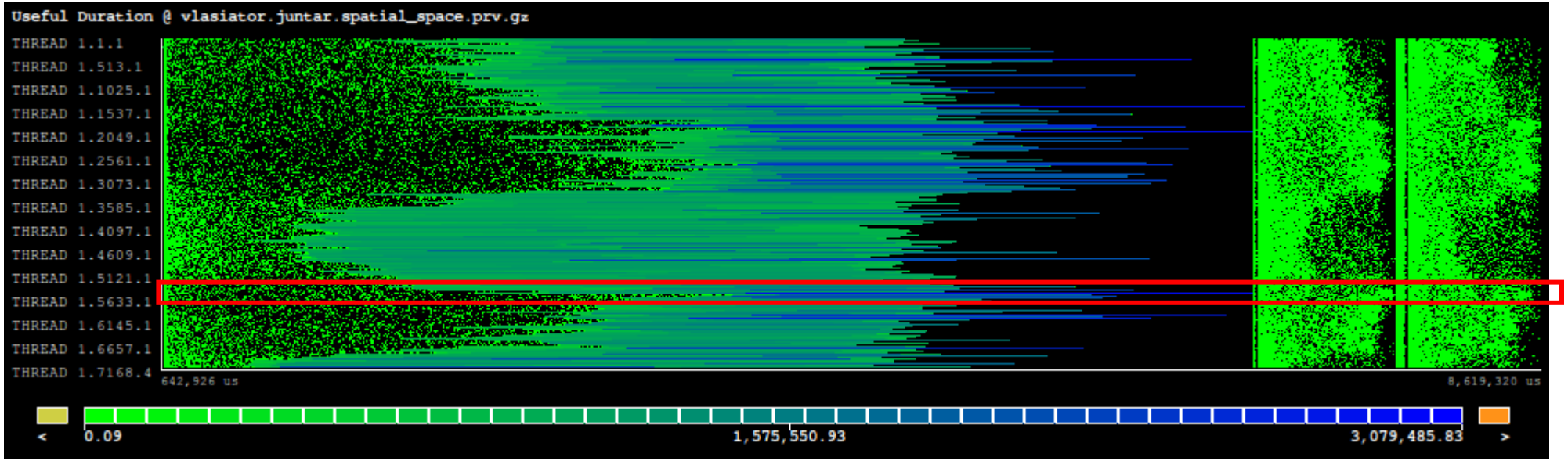


Instantiation of non-blocking sends and receives

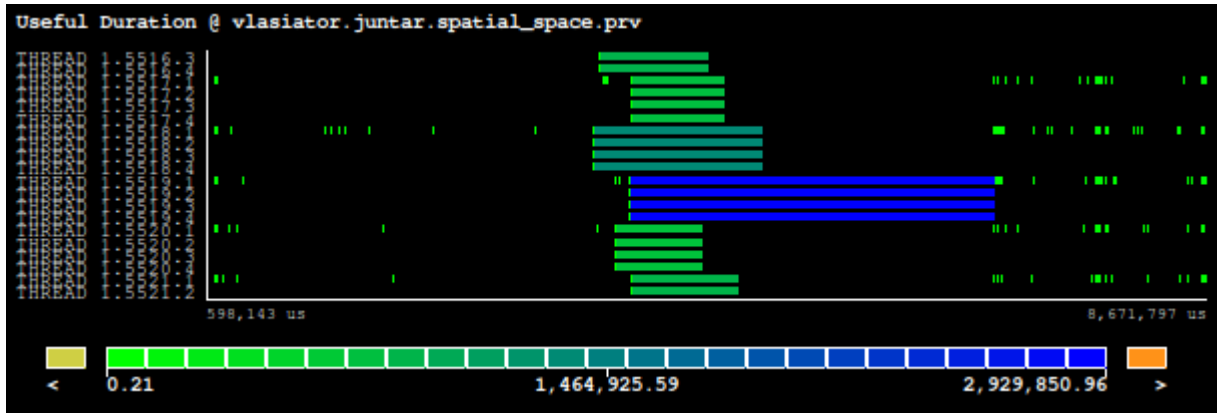
Forcing communication to complete in order of instantiation ->
Between 8/80 (min/max) MPI_Waitall calls per process

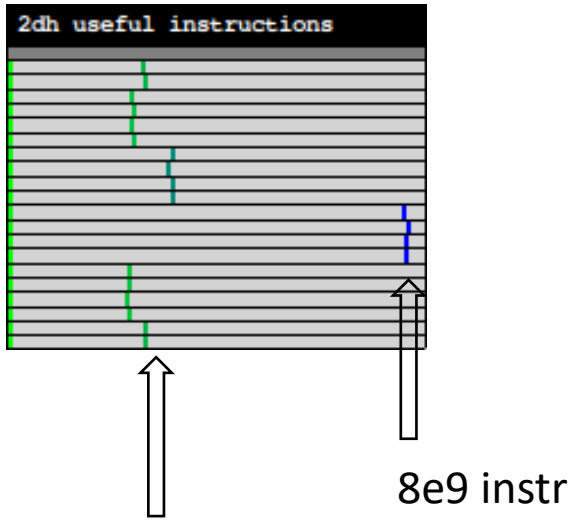
Average Bytes transferred: 800MB
Maximum Bytes transferred: 3.4GB
Minimum Bytes transferred: 173MB
Total Bytes transferred: 6.35TB

Spatial-space



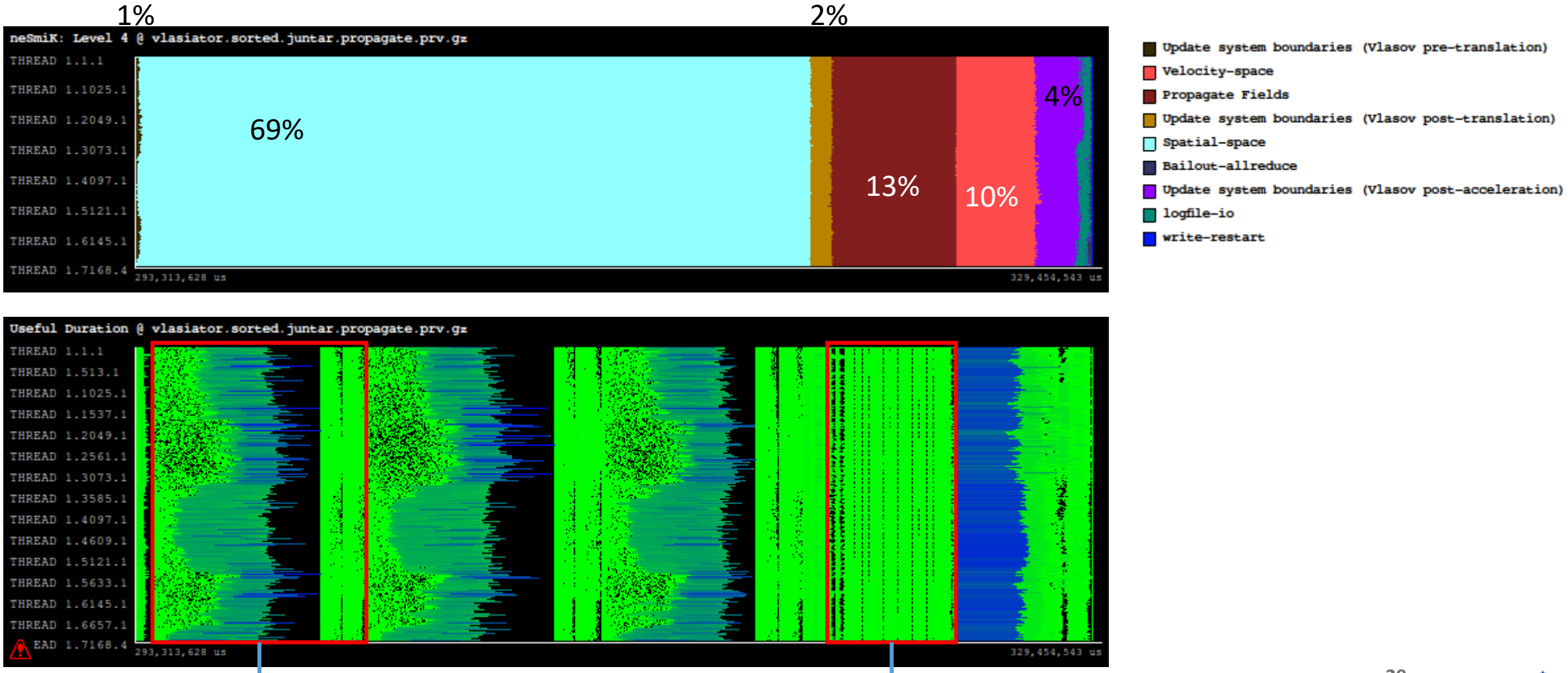
Spatial-space::compute-mapping-z





6 different processes arriving similarly after communication phase -> Then encountering load imbalance caused by instructions imbalance between the MPI processes

Propagate

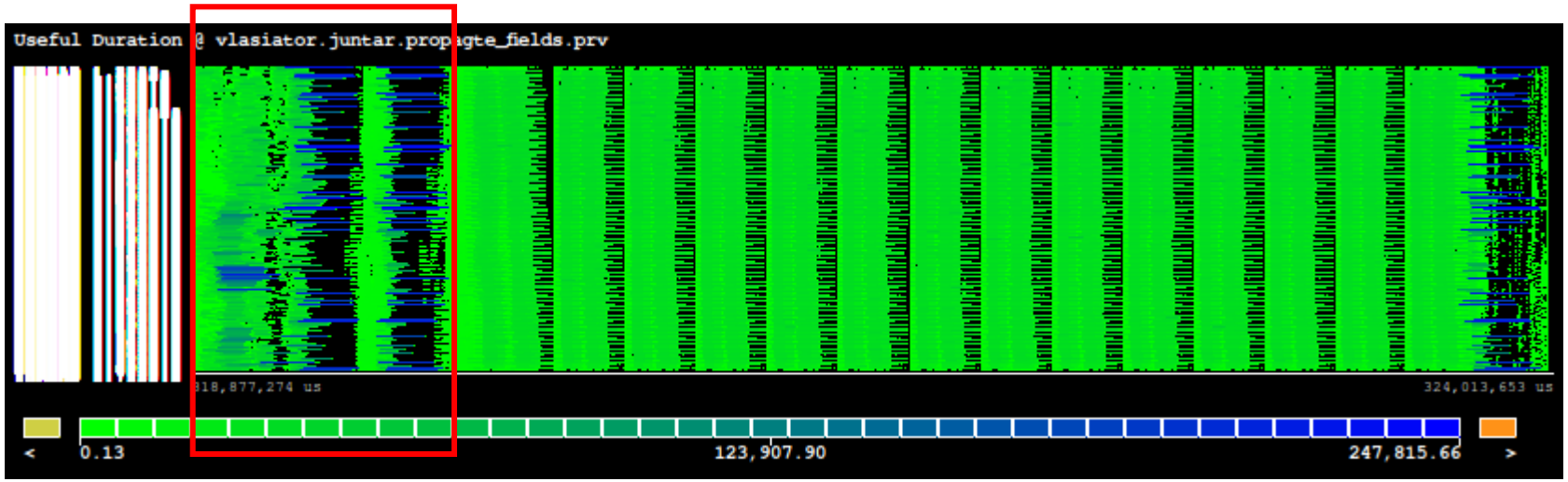

1%
2%
neSmiK: Level 4 @ vlasiator.sorted.juntar.propagate.prv.gz
THREAD 1.1.1
THREAD 1.1025.1
THREAD 1.2049.1
THREAD 1.3073.1
THREAD 1.4097.1
THREAD 1.5121.1
THREAD 1.6145.1
THREAD 1.7168.4
69%
13%
10%
4%
293,313,628 us
329,454,543 us
Update system boundaries (Vlasov pre-translation)
Velocity-space
Propagate Fields
Update system boundaries (Vlasov post-translation)
Spatial-space
Bailout-allreduce
Update system boundaries (Vlasov post-acceleration)
logfile-io
write-restart
Useful Duration @ vlasiator.sorted.juntar.propagate.prv.gz
THREAD 1.1.1
THREAD 1.513.1
THREAD 1.1025.1
THREAD 1.1537.1
THREAD 1.2049.1
THREAD 1.2561.1
THREAD 1.3073.1
THREAD 1.3585.1
THREAD 1.4097.1
THREAD 1.4609.1
THREAD 1.5121.1
THREAD 1.5633.1
THREAD 1.6145.1
THREAD 1.6657.1
EAD 1.7168.4
293,313,628 us
329,454,543 us
Spatial-space
Propagate Fields

POP metrics Propagate-fields

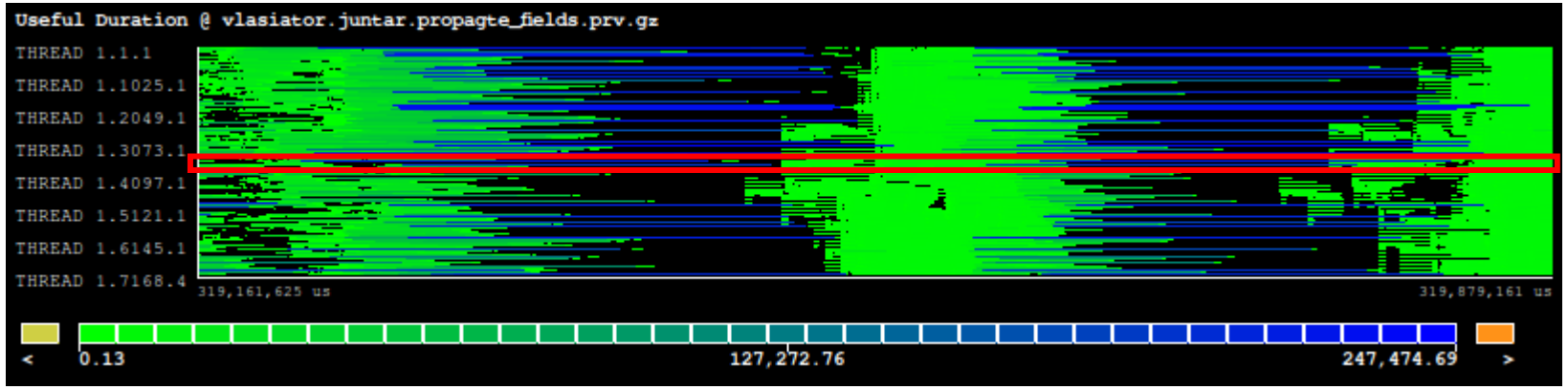
Metrics	Propagate Fields
Global Effiency	0.15
- Parallel efficiency	0.15
-- MPI Parallel efficiency	0.27
--- MPI Communication efficiency	0.85
--- MPI Load balance	0.32
---- MPI In-node load balance	0.41
---- MPI Inter-node load balance	0.77
-- OpenMP Parallel efficiency	0.36
--- OpenMP Scheduling efficiency	1
--- OpenMP Load balance	1
--- OpenMP Serialization efficiency	0.36
- Computation Scalability	1
-- Instructions scaling	1
-- IPC scaling	1
-- Frequency scaling	1
Useful IPC	2.84
Frequency [GHz]	2.93
Elapsed time [s]	56.27

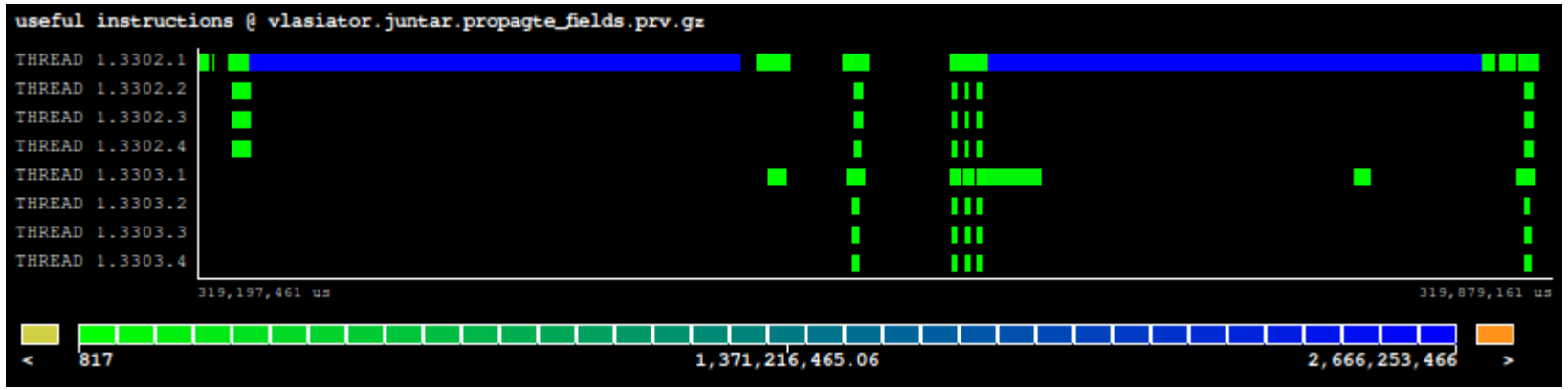
- Metrics for the Propagate-fields
- Main issues:
  - MPI Load imbalance
  - OpenMP serialization

Propagate-fields



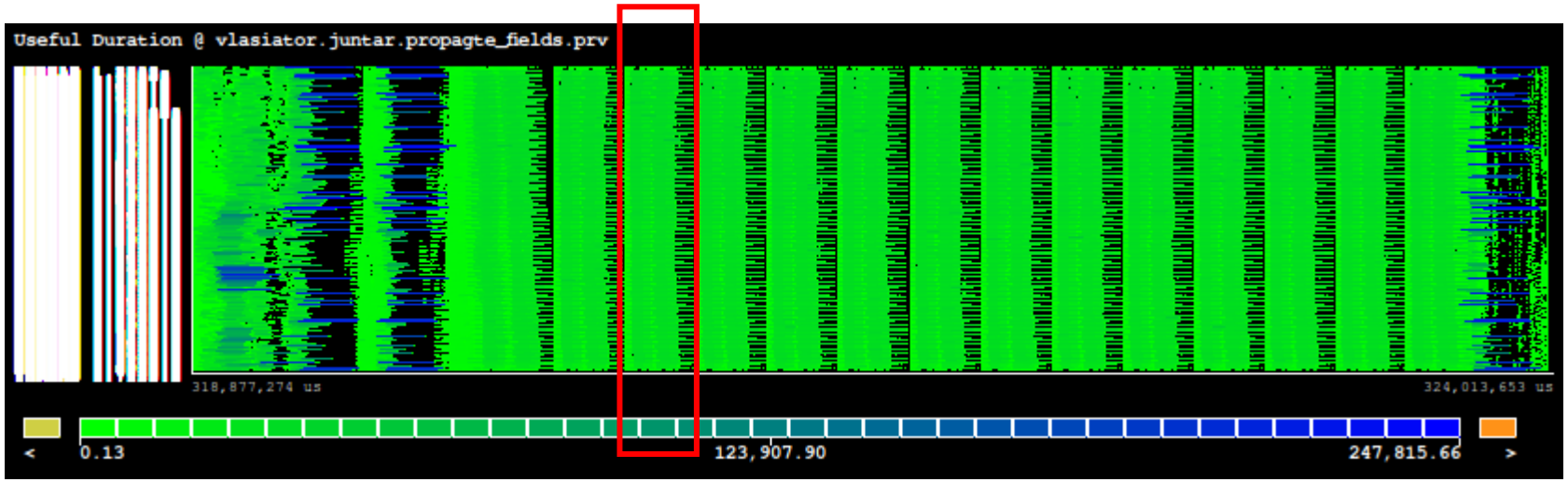
Propagate-fields



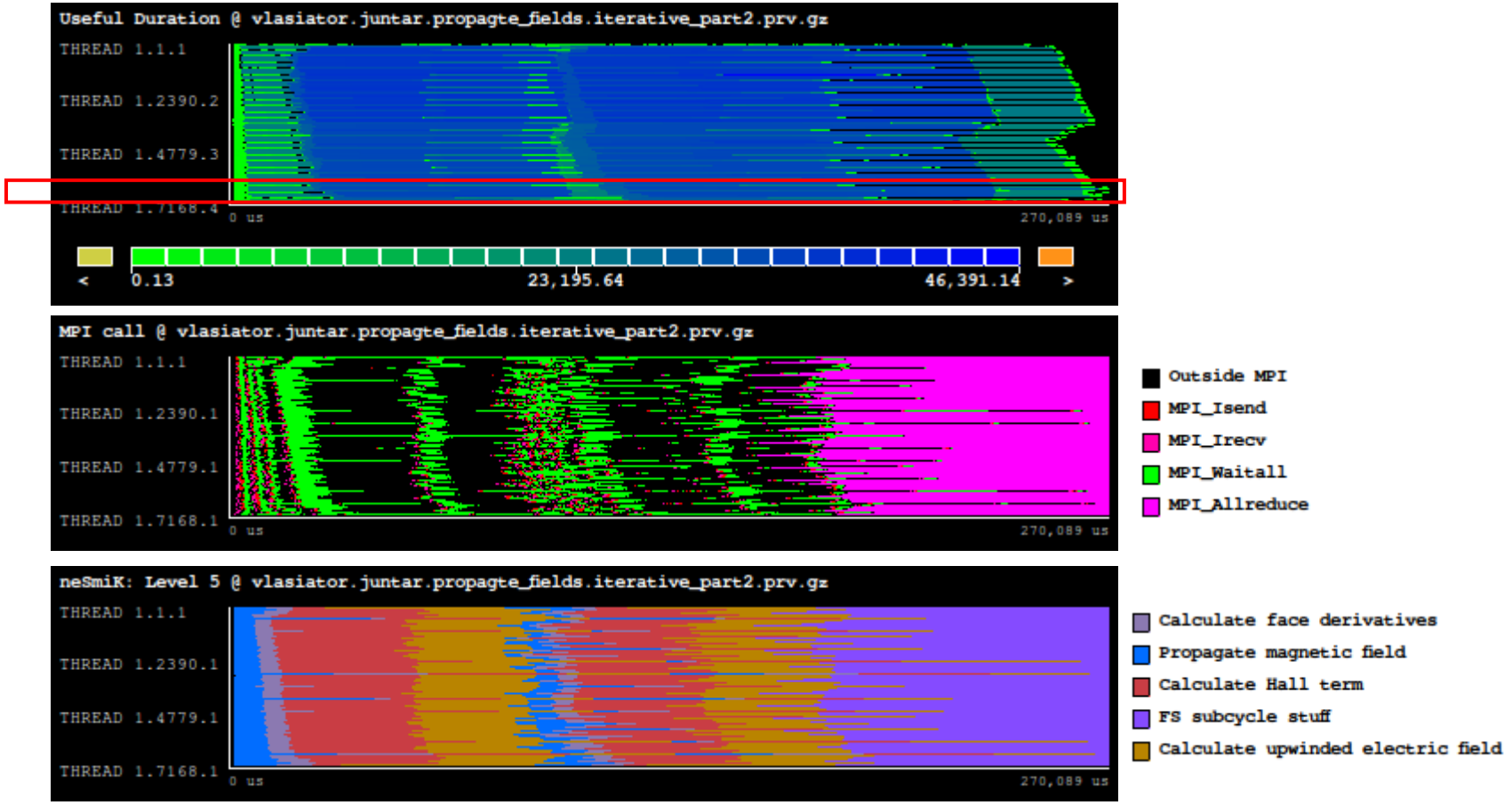


- Some processes more instructions than others
  - MPI LB
- Region not parallelized with OpenMP
  - OpenMP serialization

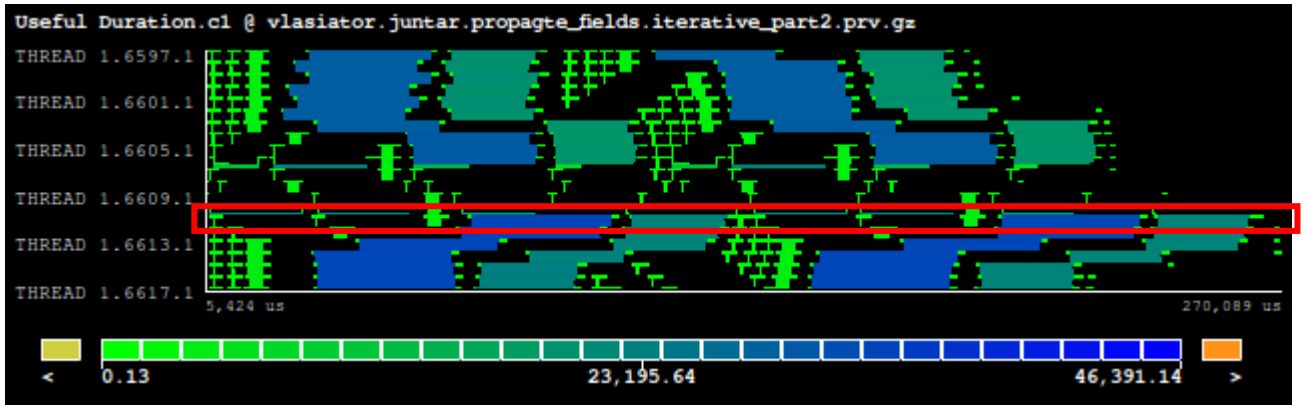
Propagate-fields



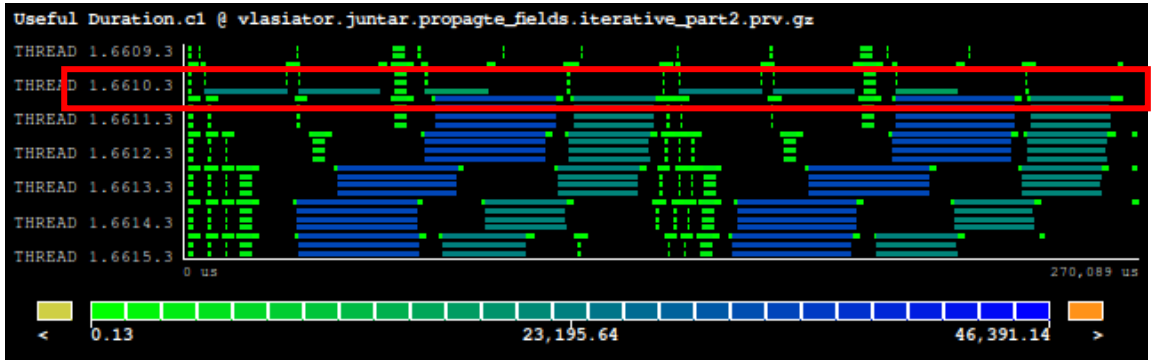
Propagate-fields

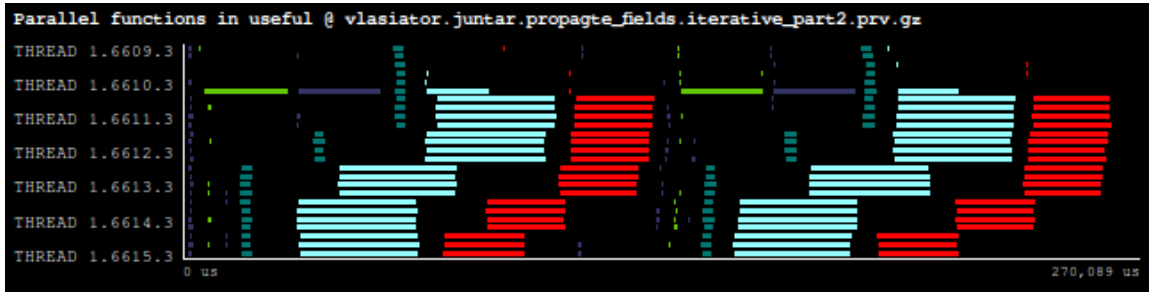


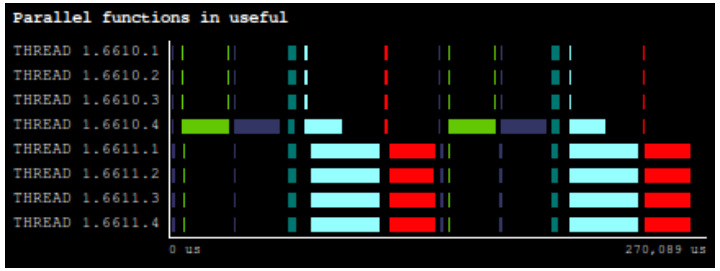
Propagate-fields

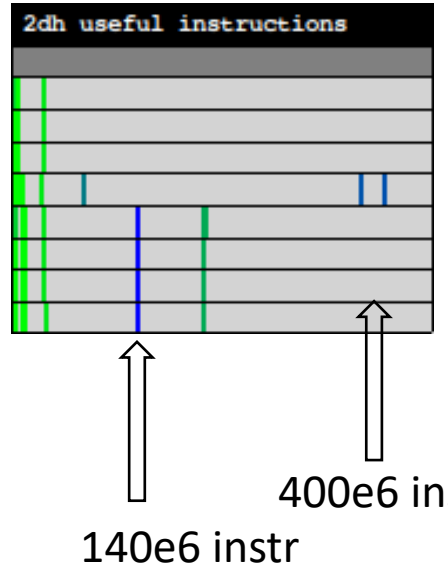


Propagate-fields









- Issues in 2 parallel regions:
  - Green/purple: MPI LB, not parallelized OpenMP
  - Blue/red: MPI and OpenMP LB

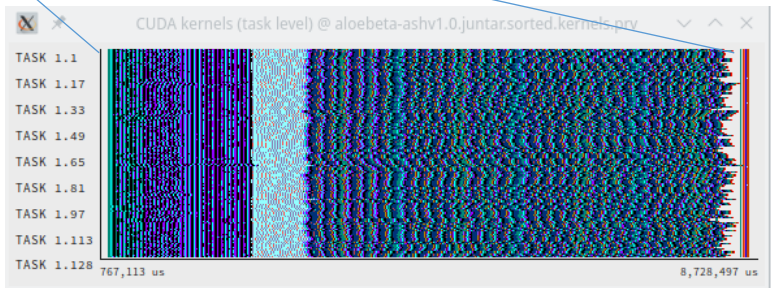
Nsys2prv and Paraver in action

Llama training @ 128 GPUs

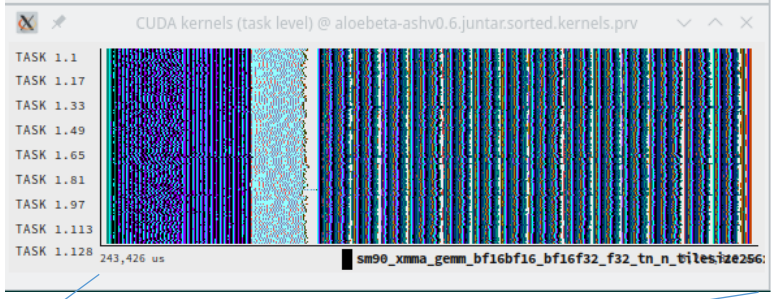
- Different microscopic behaviors … similar macroscopic duration !!!!

GPU || efficiency: 91.2 %
Load balance: 96 %
Comm. Eff.: 94.7 %
Comp. Eff.: 93.5 %

Overlapping



Non Overlapping

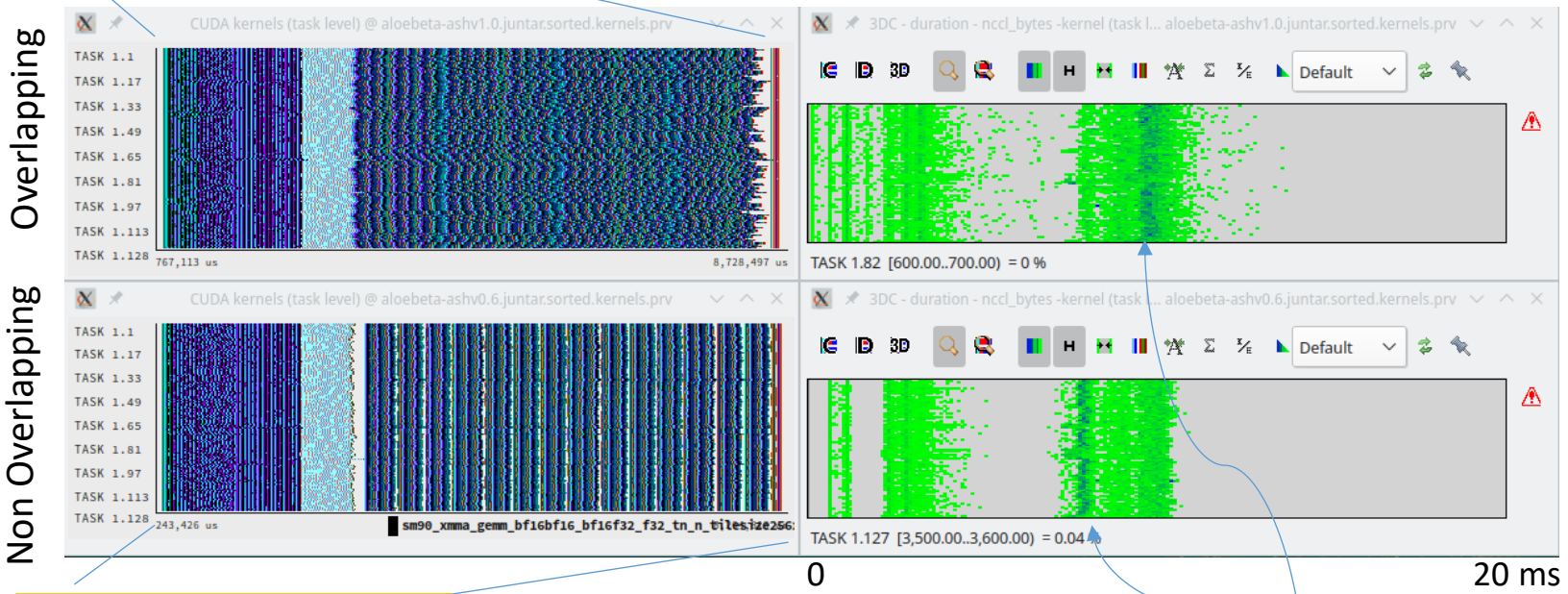


GPU || efficiency: 85.3 %
Load balance: 94 %
Comm. Eff. 90.3 %
Comp. Eff. 100 %

Llama training @ 128 GPUs

- Different microscopic behaviors … similar macroscopic duration !!!!

GPU || efficiency: 91.2 %
Load balance: 96 %
Comm. Eff.: 94.7 %
Comp. Eff.: 93.5 %



GPU || efficiency: 85.3 %
Load balance: 94 %
Comm. Eff. 90.3 %
Comp. Eff. 100 %

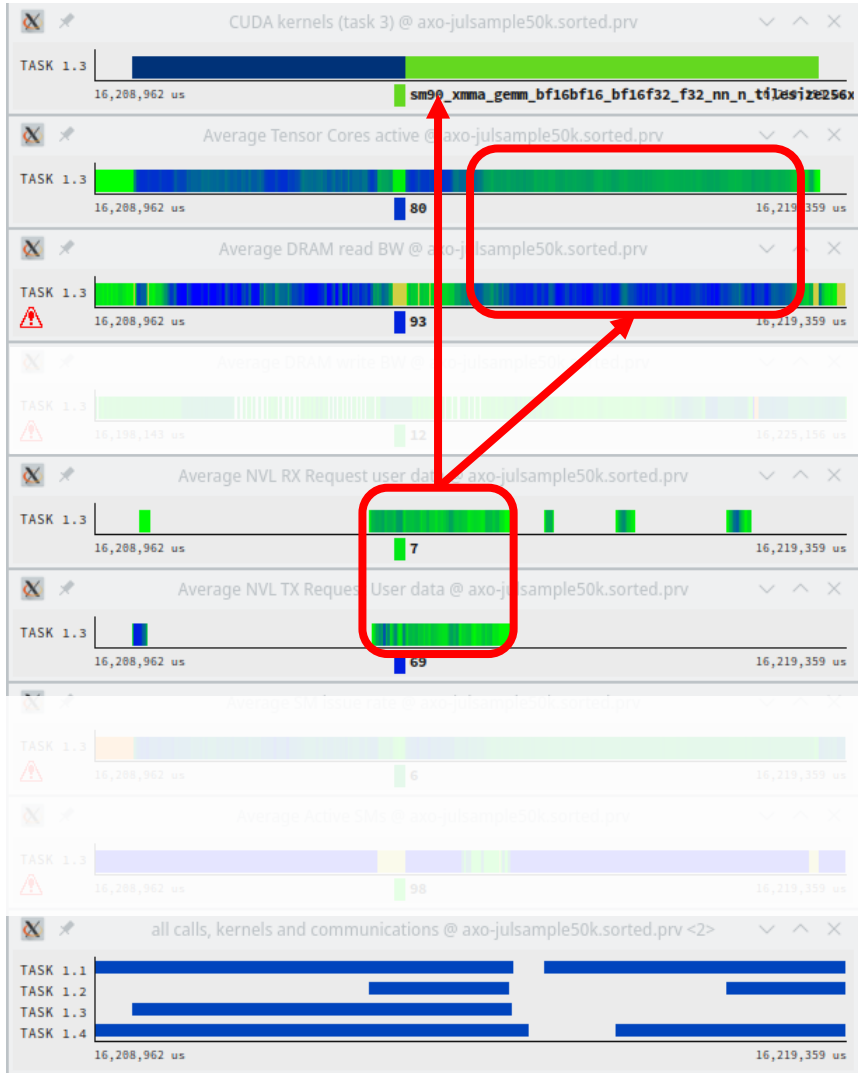
Same gemm kernel longer @ overlapping !

To overlap or not to overlap

- Gemms in two different iterations, same window duration, ….

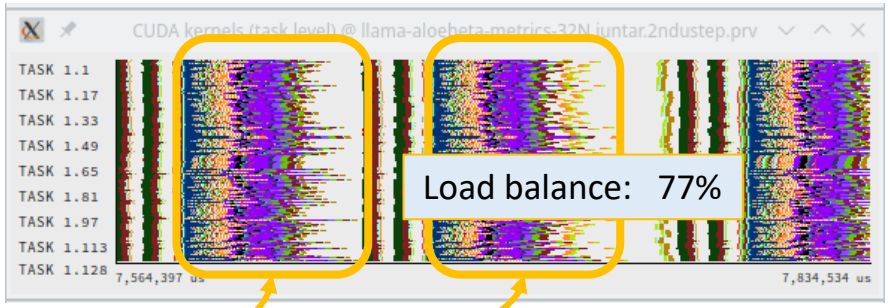
To overlap or not to overlap

- Gemms in two different iterations, same window duration, ….



Load Imbalance @ 128 GPUs?

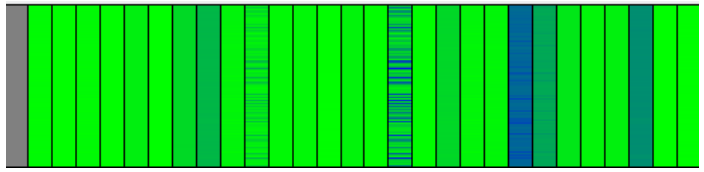
- Kernels and duration



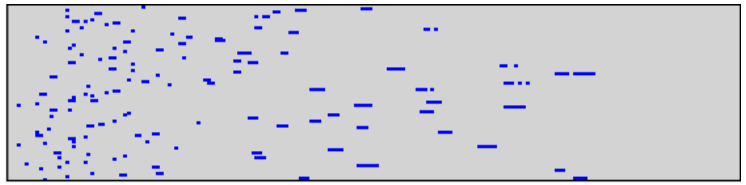
Load imbalance !!!

Repetitive pattern

Profile of kernels



Histogram of duration of flash_bwd_dq_dk_dv_… kernel

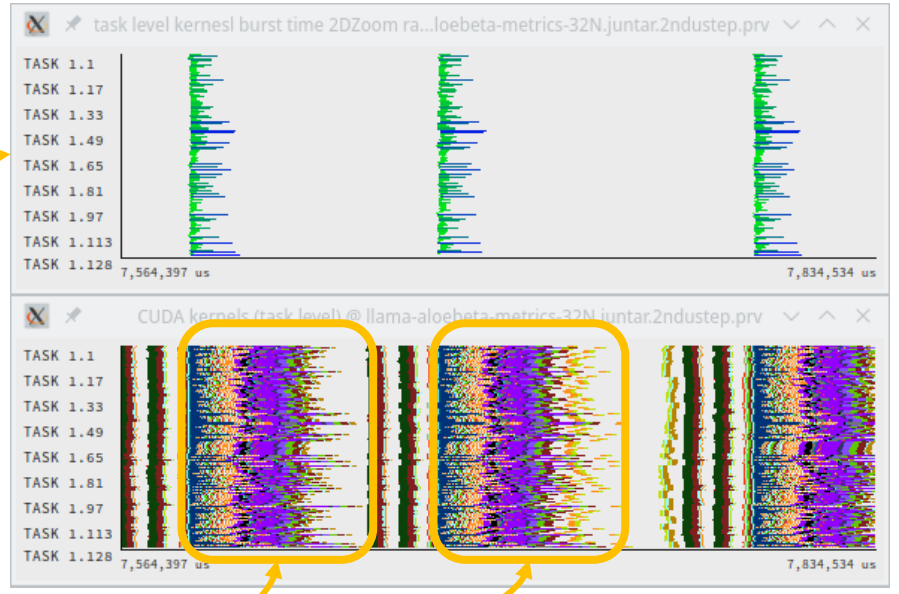


Load balance: 27 % !!

High variability across GPUs

Load imbalance@ 128 GPUs?

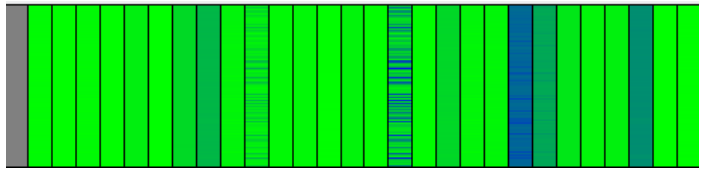
- Kernels and duration



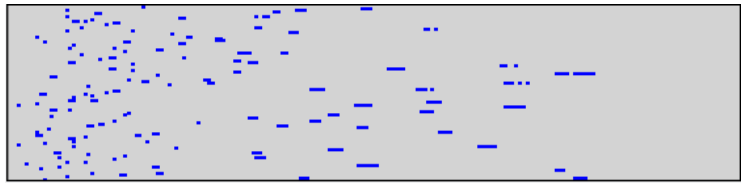
flash_bwd_dq... Kernel duration
Repetitive pattern

Load imbalance !!!
Repetitive pattern

Profile of kernels



Histogram of duration of flash_bwd_dq_dk_dv_... kernel

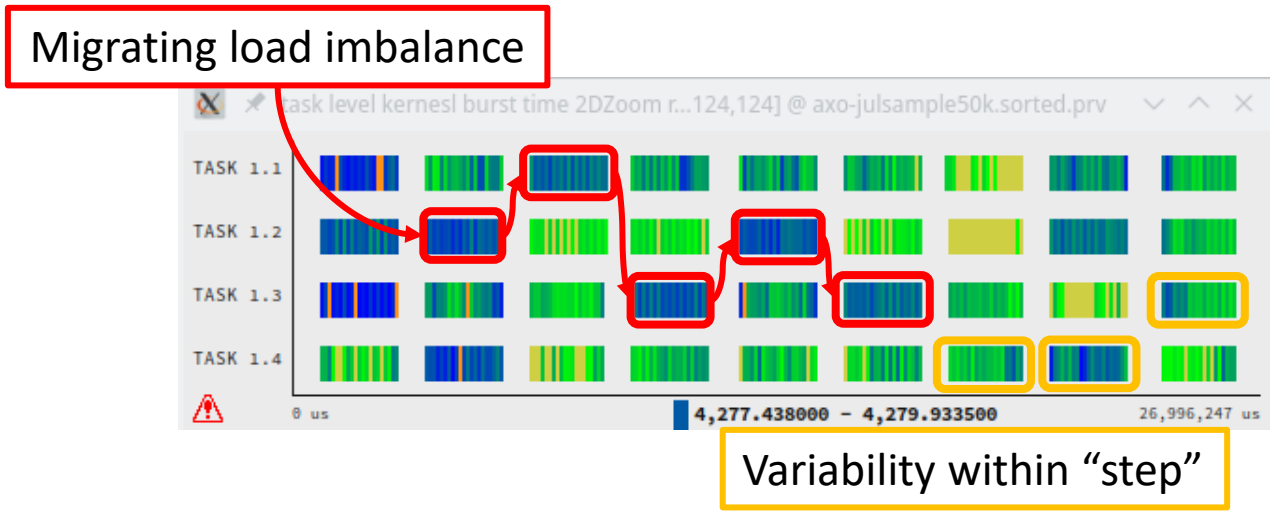


Load balance: 27 % !!

High variability across GPUs

Time varying load balance

- flash_bwd_dq_dk_dv_loop… kernel duration



Highlights

Highlights

- is providing **FREE Services** for European users
  - Precise understanding of application and system behaviour
  - Suggestion/support on how to refactor code in the most productive way
- We rely on a powerful set of tools
  - that cover traditional HPC codes and newly developed
  - that can analyse large scale executions..
  - … and provide insight at microscopic scale
  - that can help understand performance of AI frameworks
- Our tools are deployed in almost all EuroHPC systems

Next **training** on POP3 tools in Marenostrum5
in collaboration with CASTIEL:
Feb 9 – 13, 2026 Barcelona
Registration and more information here:
https://events.prace-ri.eu/event/1626/

Contact us: pop@bsc.es
Or request your assessment here:

SCAN ME

EuroHPC Joint Undertaking

Performance Optimisation and Productivity 3

A Centre of Excellence in HPC

Contact:

- https://www.pop-coe.eu
- pop@bsc.es
- @POP_HPC
- youtube.com/POPHPC

This project has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931. The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal and the Czech Republic.