30. March 2026

Analysis and Optimization at Production Scale: Insights from Vlasiator

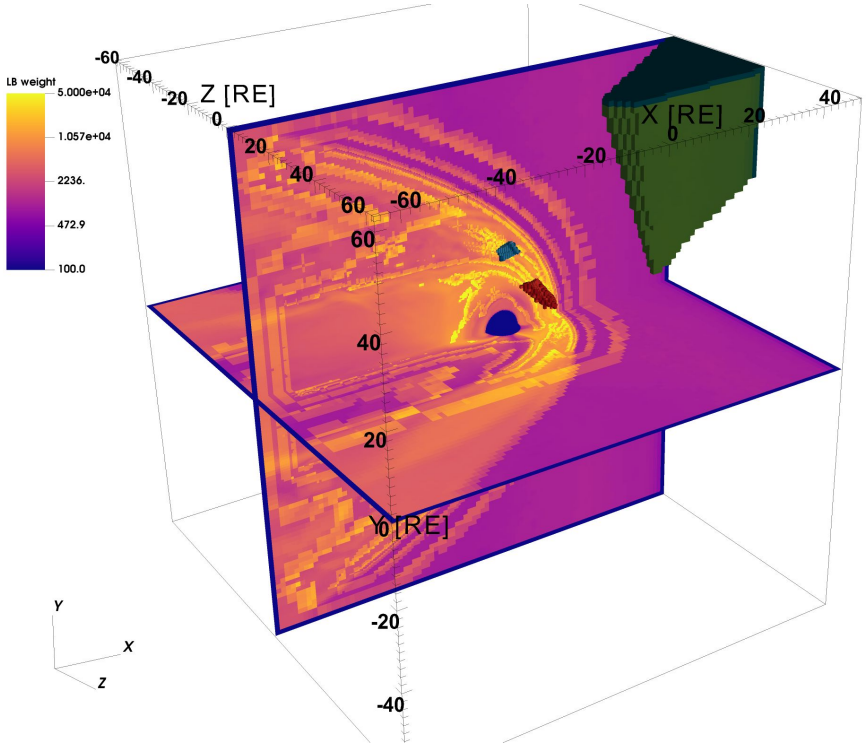
Valentin Seitz *(valentin.seitz@bsc.es)*

Outline

- Introduction
  - Vlasiator
  - Tools & Methodology
- Analysis
- Optimization

Motivation

Understand the performance of a production level simulation of Vlasiator, using a restart file produced from a 500 node production run on Mahti.



Vlasiator simulation state showcasing the heterogeneity in computational load of the domain for timestep 82848. Axis labels are in units of Earth radii (RE)
Plot provided by Vlasiator-Team

- Near-earth plasma simulations
- C++
- MPI+OpenMP
- Code annotations through phiprof
- Specific branch available on GitHub
- Restart file
  - ~5TB on disk
  - ~55TB main memory

Scaling run from 250 to 500 nodes in Marenostrum 5 GPP[1]

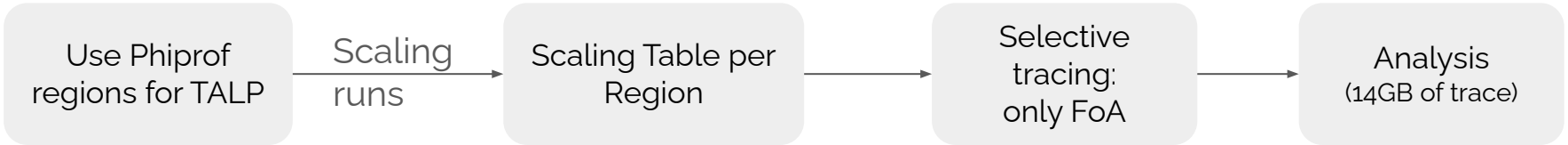
- 14 MPI processes with 8 OpenMP threads each (SMT disabled)
- Block distribution of processes (7 per socket), threads bound to close cores.

1: Introducing MareNostrum 5, Banchelli et al. 2026

Methodology

- Typical approach: trace everything & select focus of analysis (FoA) based on structure
  - In this case: trace sizes between 500-900GB
  - not easily applicable at this scale

- Phiprof annotations in the trace
- Profiling + selective tracing of only 1 timestep
- Based on profiling results, choose regions to analyse



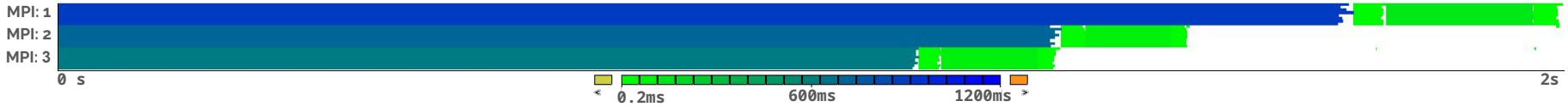
Tools

- Analysis done using BSC tools: Extrae, Paraver & TALP
- Very similarly doable with Score-P, Scalasca, Cube and Vampir
- Hardware counters gathered using PAPI

Overhead around 5-10%

Paraver traces

Useful duration



OpenMP parallel function

MPI Call

Analysis

- Structure of a typical timestep
- Scaling results
  - Analysis of the spatial space translations
  - Analysis of the field solvers

Structure of a timestep (Propagate)

- Spatial-space (z,y,x)
- Propagate Fields
- Velocity-space
- Update system boundaries

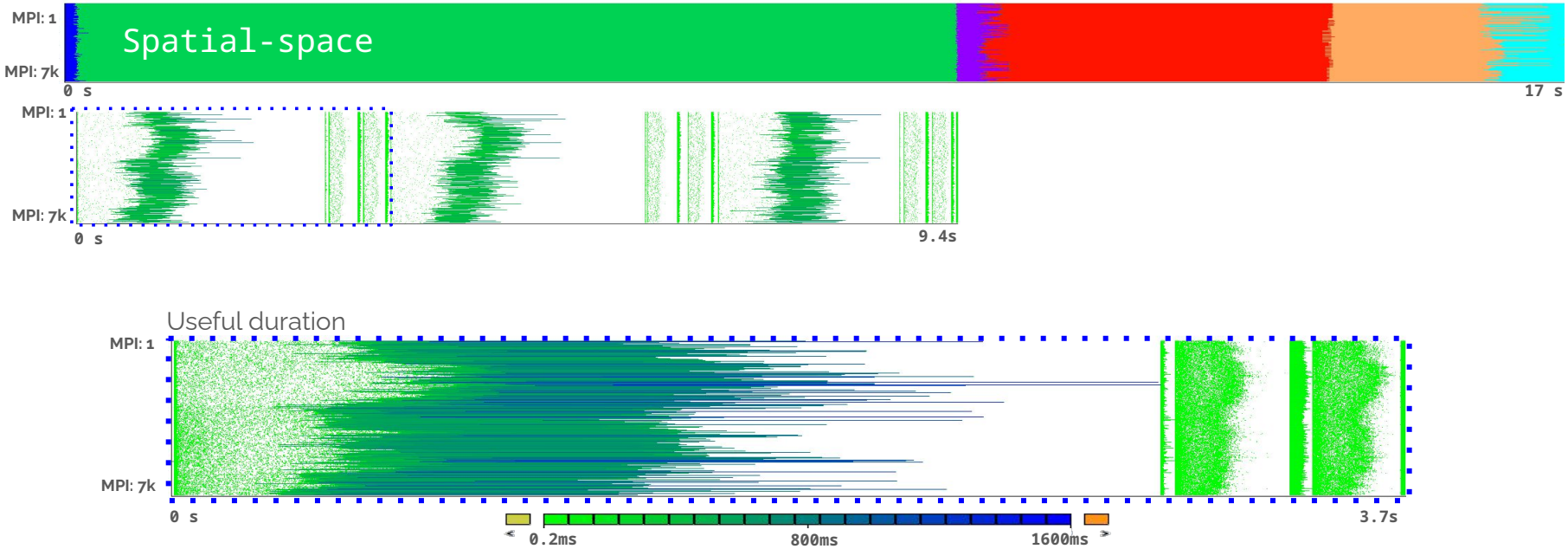
Phiprof regions with their respective relative runtime per timestep of the 500 node execution

Scaling

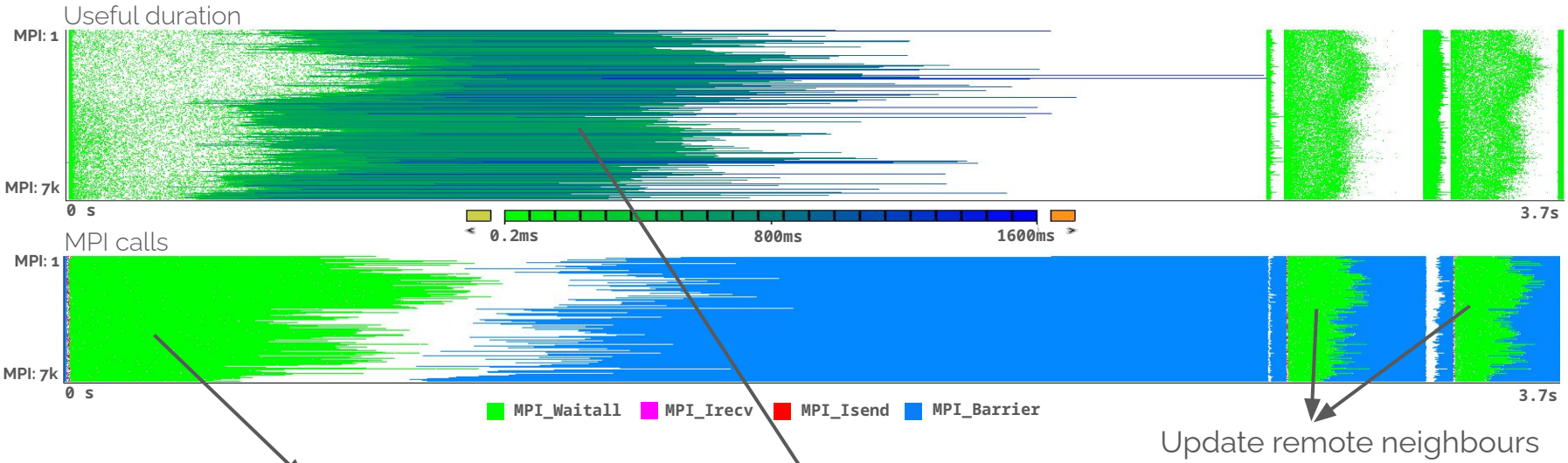
	Propagate		Propagate					
	Propagate		Spatial-space		Propagate Fields		Velocity-space	
Number of CPUs	28000	56000	28000	56000	28000	56000	28000	56000
MPI Parallel Efficiency	0.23	0.21	0.19	0.17	0.18	0.16	0.52	0.43
MPI Load Balance	0.51	0.53	0.35	0.39	0.31	0.27	0.54	0.51
MPI Communication Efficiency	0.46	0.40	0.55	0.43	0.58	0.63	0.97	0.84
Computation scalability	1.00	1.07	1.00	1.08	1.00	1.05	1.00	1.01
Average IPC	1.52	1.59	0.99	1.08	2.38	2.40	1.68	1.68
Average Frequency [GHz]	2.89	2.91	2.95	2.95	2.84	2.87	2.91	2.92
Average Runtime [s]	36.75	17.27	21.89	9.43	8.39	4.70	4.83	2.64
Speedup		2.13		2.32		1.79		1.83

Efficiency table for the timestep region and 3 child regions scaling Vlasiator from 250 to 500 nodes in Marenostrum 5 GPP.

Spatial Space



Spatial Space: z-direction

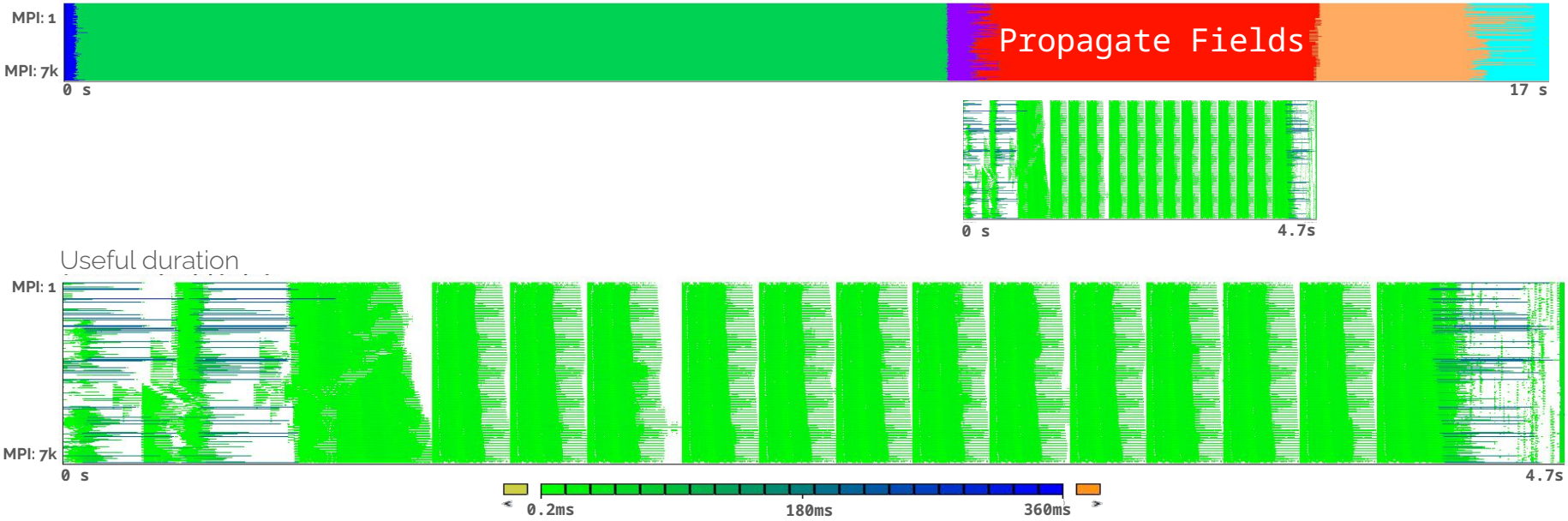


Update remote neighbours

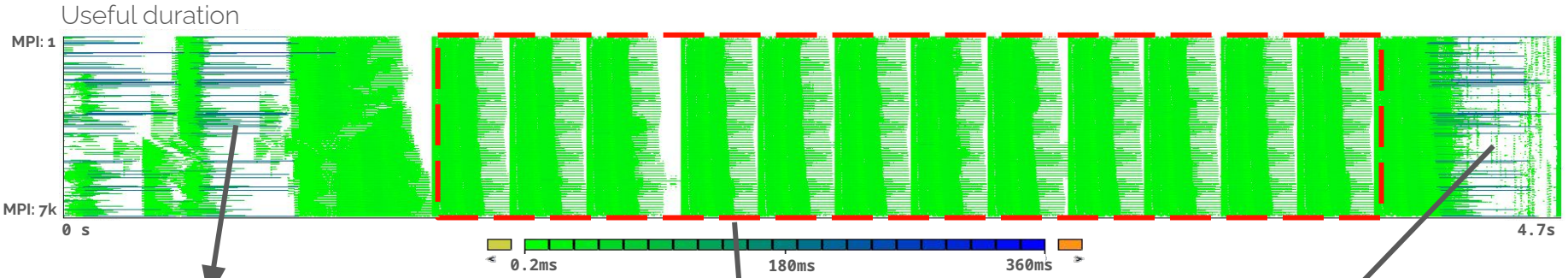
Communication of 6.35TB of data.

Computation load imbalance by uneven work distribution during the “stencil computation”

Propagate Fields



Propagate Fields

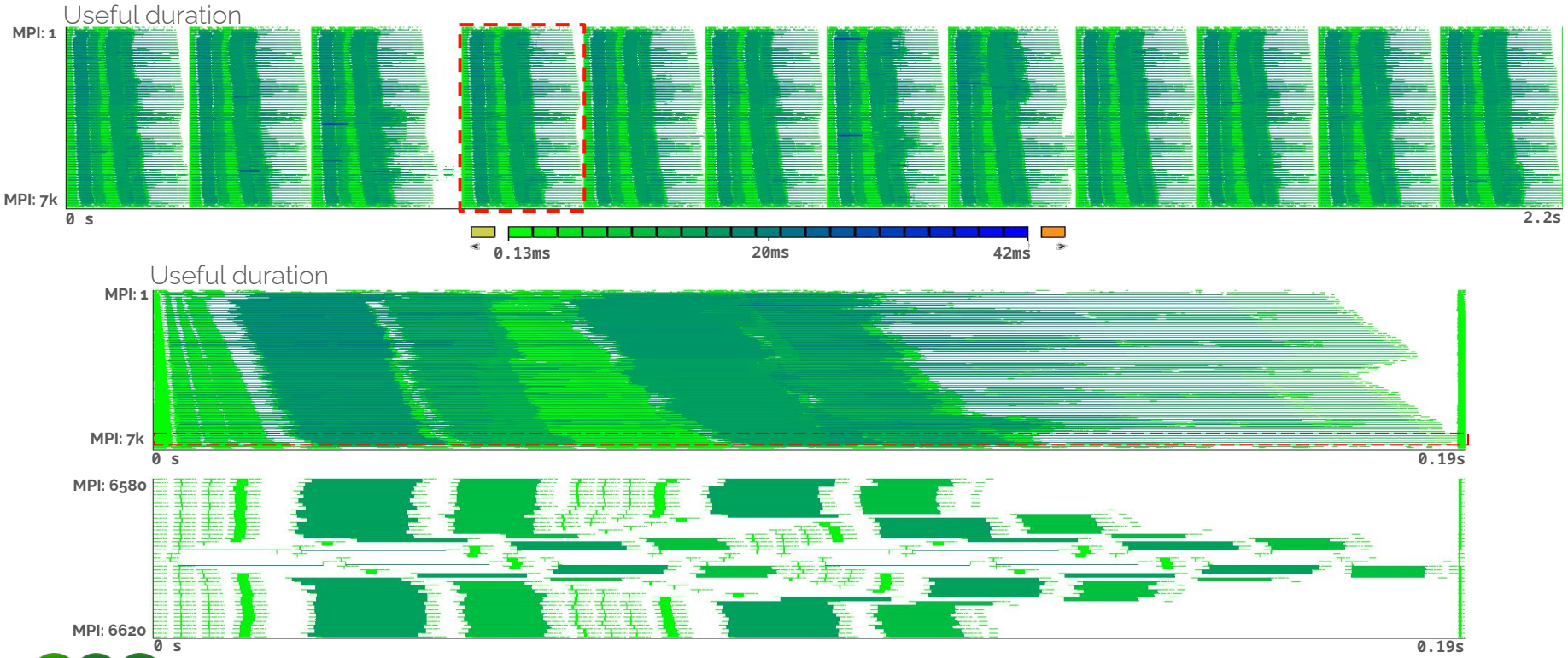


Feed moments from kinetic solver into the field solver:
serial code & load imbalance!

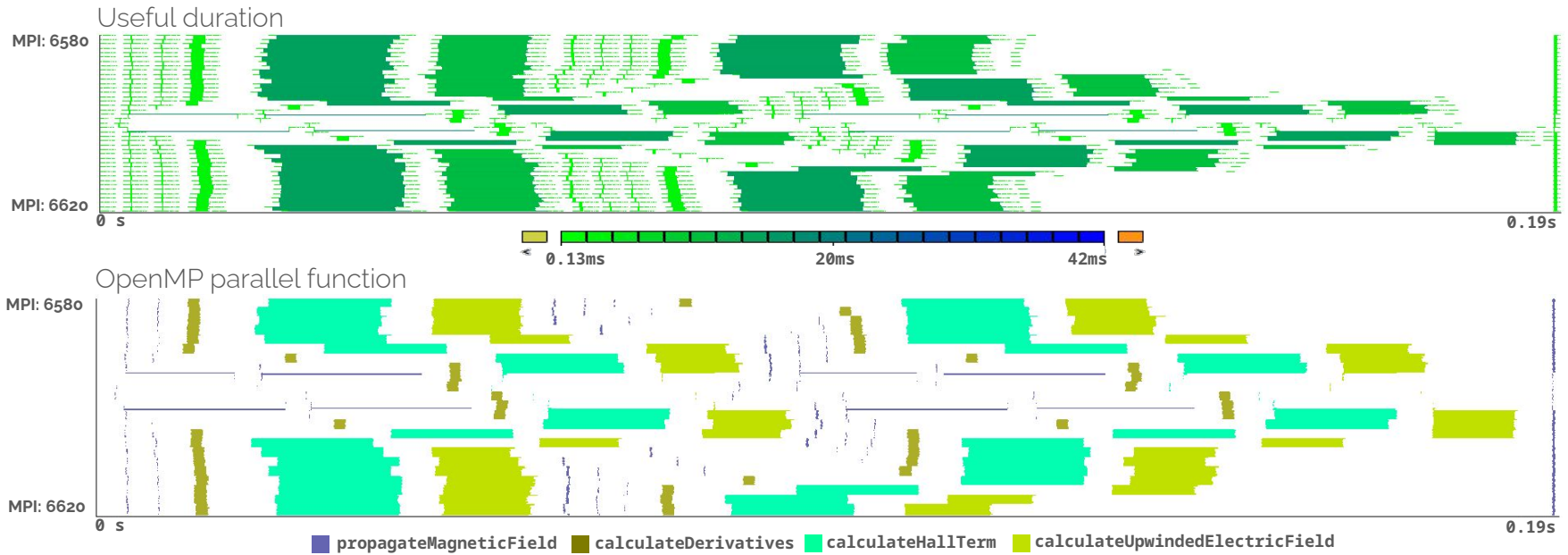
Iterative part of the actual solver

Copy solved fields back:
Similar problems as initial setup

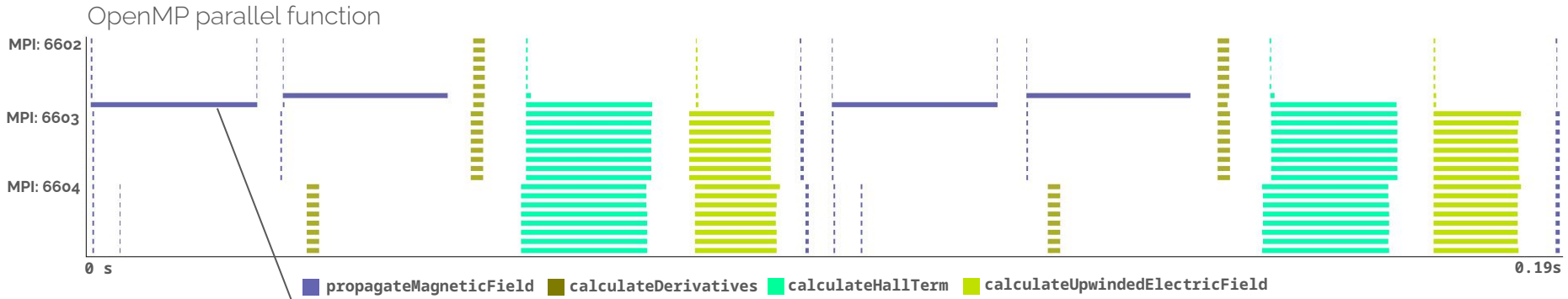
Propagate Fields: 12 iterations



Propagate Fields: 1 iteration; few processes



Propagate Fields: 1 iteration; 3 processes



MPI Load imbalance & only one OpenMP thread got all the work

Recap

Spatial Space

- non-overlapping communication and computation
- Load balance in the computation part

Propagate Fields

- Serial code at start and end of field solvers
- `propagateMagneticField`:
  MPI Load imbalance & unfavorable OpenMP worksharing

Advisory Study: magnetic field boundary handling

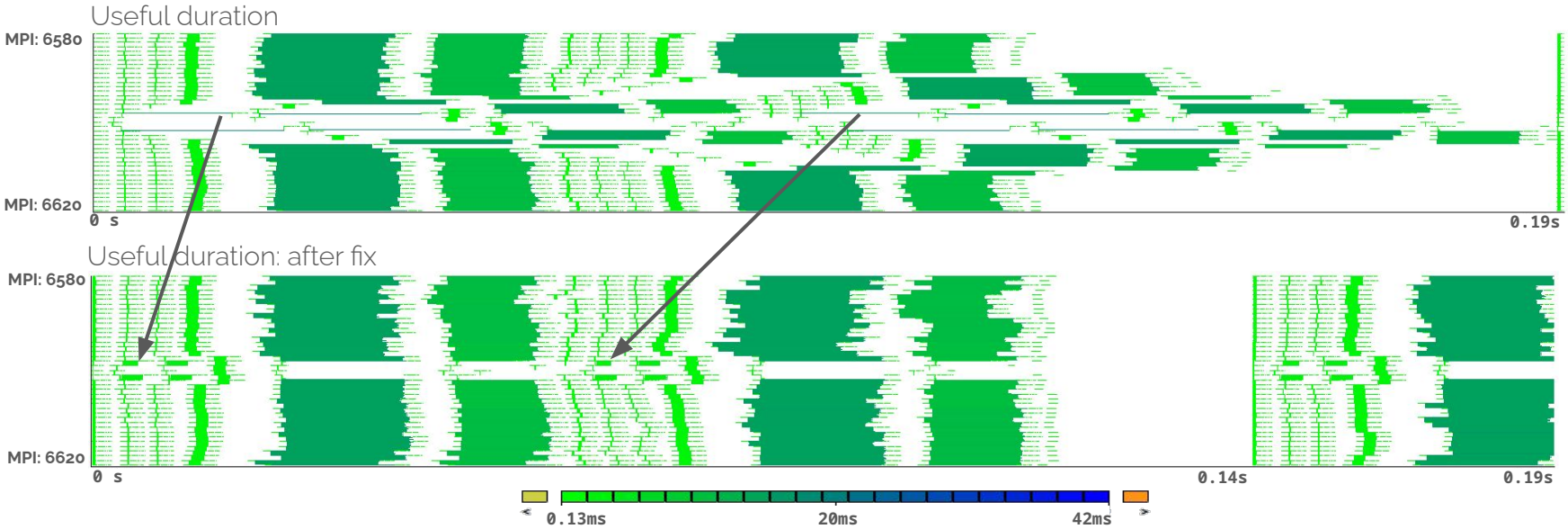
```
#pragma omp parallel {
  #pragma omp for collapse(2)
  for (int k = 0; k < gridDims[2]; k++) {
    for (int j = 0; j < gridDims[1]; j++) {
      for (int i = 0; i < gridDims[0]; i++) {
        if (checkIfBoundary(i, j, k)) {
          if (computeX(i, j, k)) {
            compute(i, j, k, 0);
          }
          if (computeY(i, j, k)) {
            compute(i, j, k, 1);
          }
          if (computeZ(i, j, k)) {
            compute(i, j, k, 2);
          }

        }
      }
    }
  }
}
```

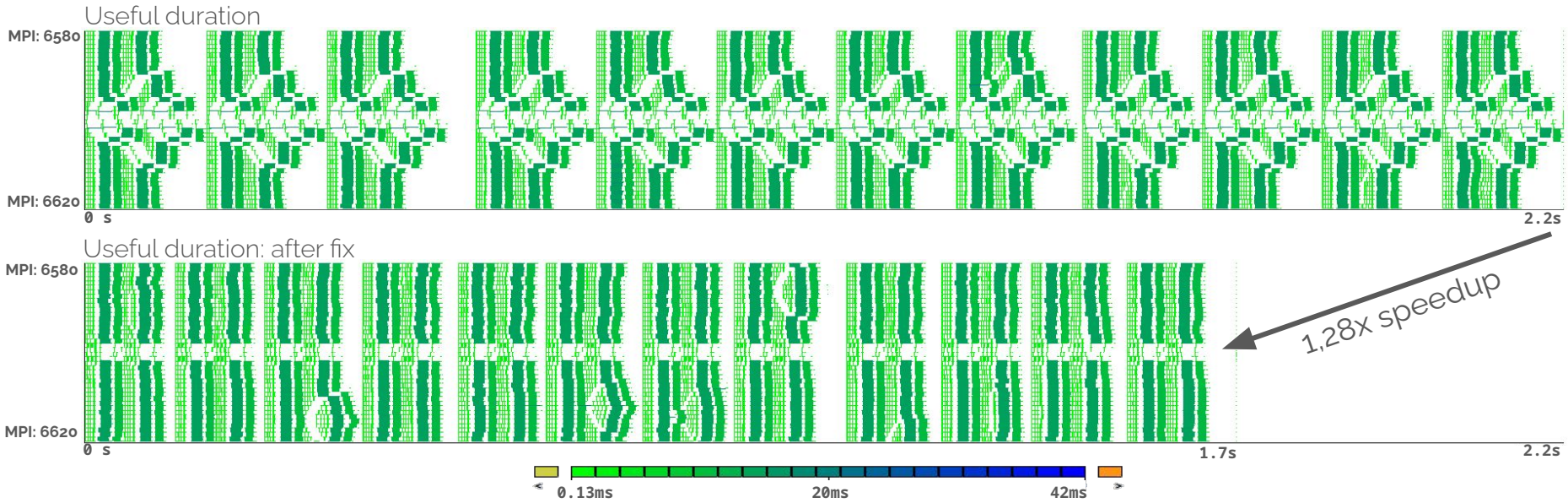
```
std::vector<std::array<int,4>> toSolve;
#pragma omp parallel {
  #pragma omp for collapse(2)
  for (int k = 0; k < gridDims[2]; k++) {
    for (int j = 0; j < gridDims[1]; j++) {
      for (int i = 0; i < gridDims[0]; i++) {
        if (checkIfBoundary(i, j, k)) {
          if (computeX(i, j, k)) {
            toSolve.push_back({i, j, k, 0});
          }
          if (computeY(i, j, k)) {
            toSolve.push_back({i, j, k, 1});
          }
          if (computeZ(i, j, k)) {
            toSolve.push_back({i, j, k, 2});
          }
        }
      }
    }
  }
}
#pragma omp for
for (const auto[i,j,k,dim] : toSolve ) {
    compute(i,k,k,dim);
}
```

Code available at: https://github.com/fmihpc/vlasiator/pull/1110

Advisory Study: one iteration; few processes



Advisory Study: 12 iterations; few processes



Advisory Study: Efficiency Metrics

Code Version	before	after
MPI Parallel Efficiency	0.34	0.43
MPI Load Balance	0.49	0.85
MPI Communication Efficiency	0.70	0.50
Computation scalability	1.00	0.99
Average IPC	2.41	2.40
Average Frequency [GHz]	3.0	3.0
Average Runtime [s]	2.25	1.75
Speedup		1.28

Summary and Outlook

- Detailed analysis of the two most time-consuming regions even at 56000 CPUs.
- Spatial Space
  - could benefit from overlapping computations and communications
  - spatial space stencils experience heavy load imbalance
  - Working on a prototype **(proof of concept)** to load balance the spatial space stencils using DLB
- Propagate Fields
  - Restructuring the code in an **advisory study** improved the field solver performance by 10%

Acknowledgements

This work has received funding from the European High-Performance Computing Joint Undertaking (JU) under grant agreement No 101143931 (POP3 project) and No 101093261 (Plasma-PEPSC project).
The JU receives support from the European Union's Horizon Europe research and innovation programme and Spain, Germany, France, Portugal, the Czech Republic, Sweden, Finland, Greece, and Slovenia.
The Plasma-PEPSC project (PCI2022-135050-2) and the POP3 project (PCI2024-153419) are also co-funded by MICIU/AEI /10.13039/501100011033 and by the UE NextGenerationEU/PRTR.
Access to the computing resources was provided by the EuroHPC Development Access Call EHPC-DEV-2025D05-038