Scalasca Trace Tools

Case study: TeaLeaf

Markus Geimer
Jülich Supercomputing Centre

trace tools
scalasca

TeaLeaf

- HPC mini-app developed by the UK Mini-App Consortium
  - Solves the linear 2D heat conduction equation on a spatially decomposed regular grid using a 5 point stencil with implicit solvers
  - Part of the Mantevo 3.0 suite
  - Available on GitHub: http://uk-mac.github.io/TeaLeaf/

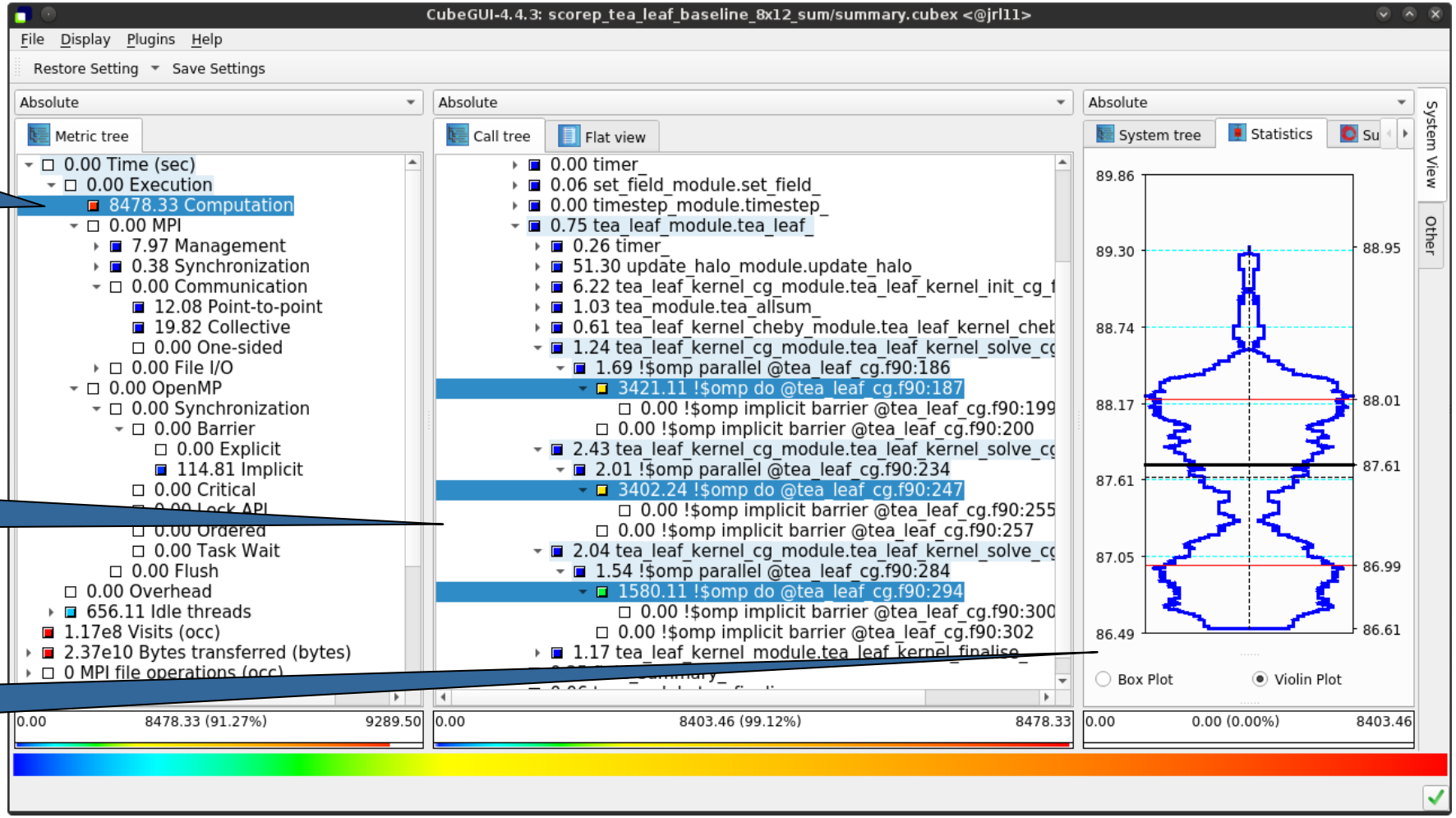
- Measurements of TeaLeaf reference v1.0 taken on Jureca cluster @ JSC
  - Using Intel 19.0.3 compilers, Intel MPI 2019.3, Score-P 5.0, and Scalasca 2.5
  - Run configuration
    - 8 MPI ranks with 12 OpenMP threads each
    - Distributed across 4 compute nodes (2 ranks per node)
    - Test problem “5”: 4000 × 4000 cells, CG solver

TeaLeaf summary report analysis (I)

91% of the execution time is computation…

…almost entirely spent in 3 OpenMP `do` loops…

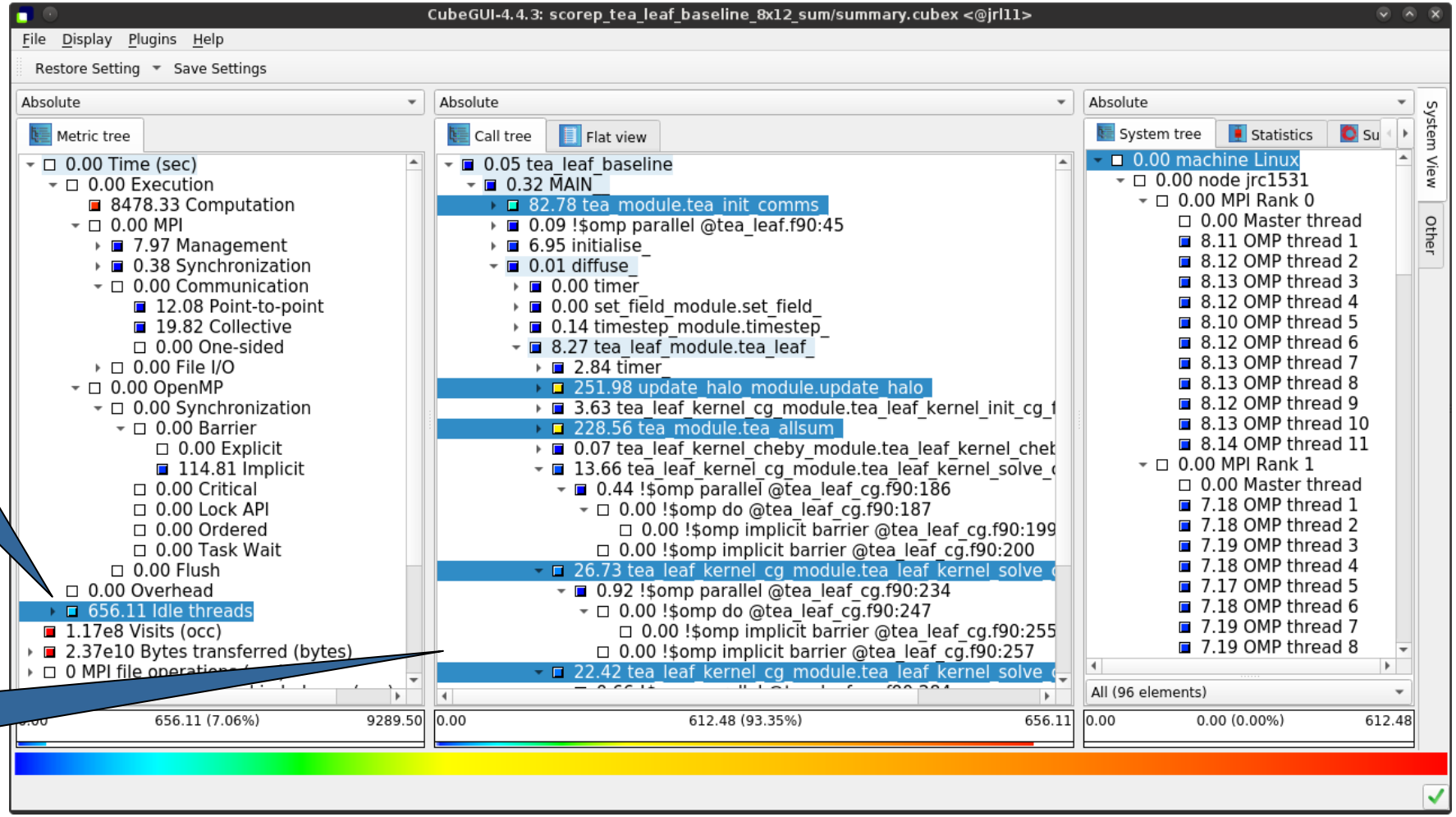
…with a slight imbalance across ranks & threads



TeaLeaf summary report analysis (II)

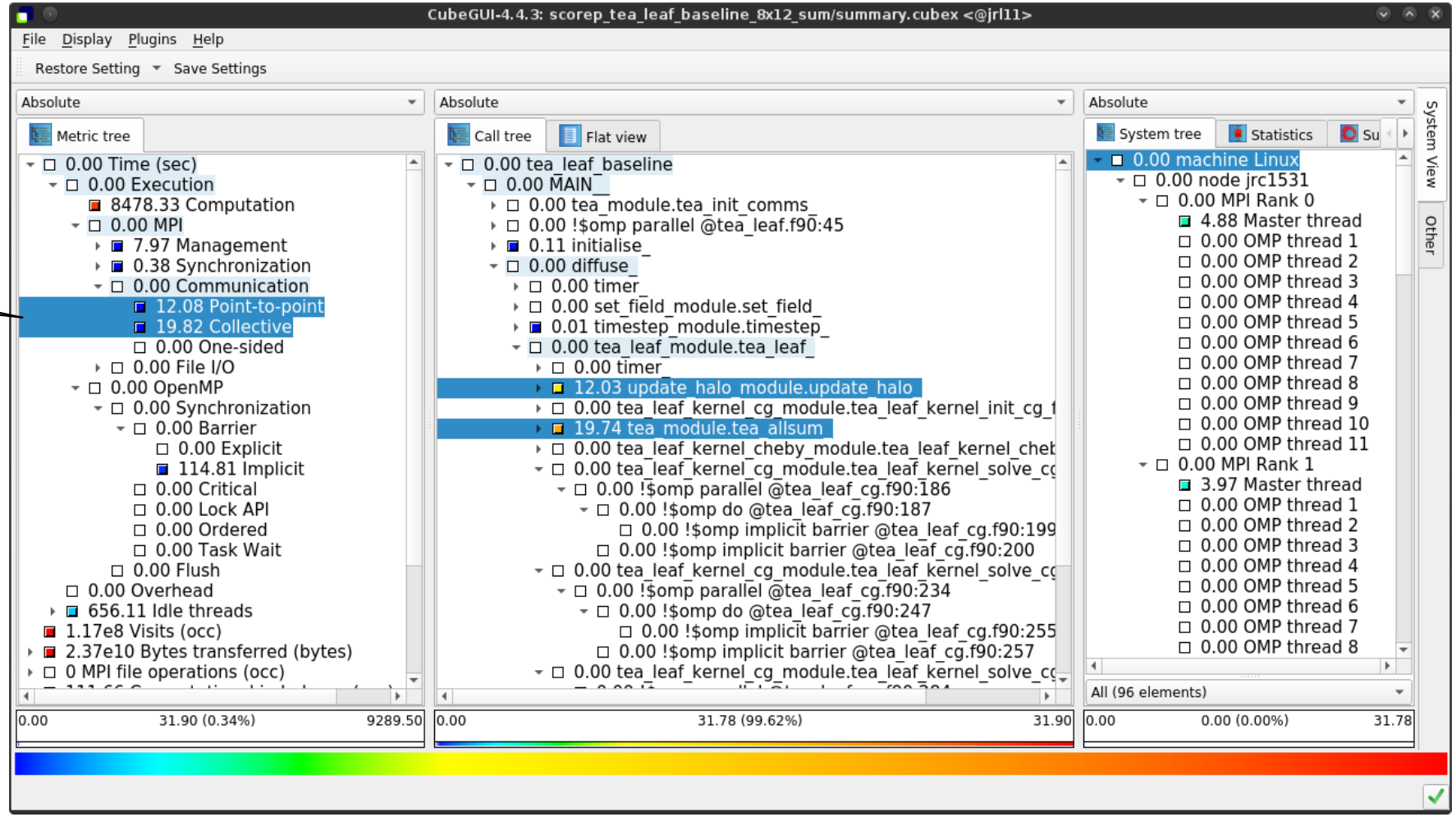
7% of the execution time are lost due to idle threads…

…in non-OpenMP parallelized code regions



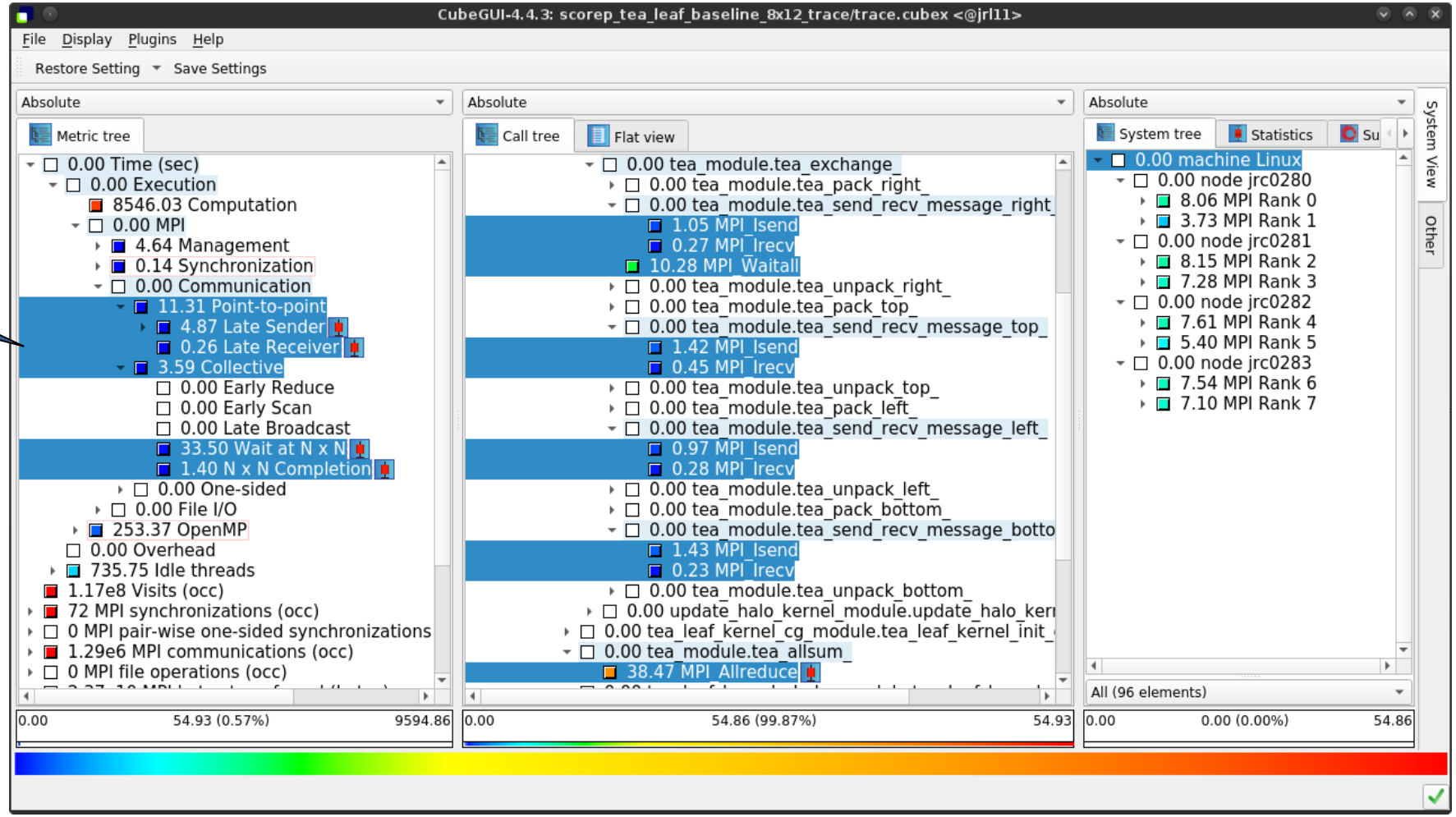
TeaLeaf summary report analysis (III)

MPI communication time is negligible (0.34%); communication is only on the master threads (MPI_THREAD_FUNNELED)



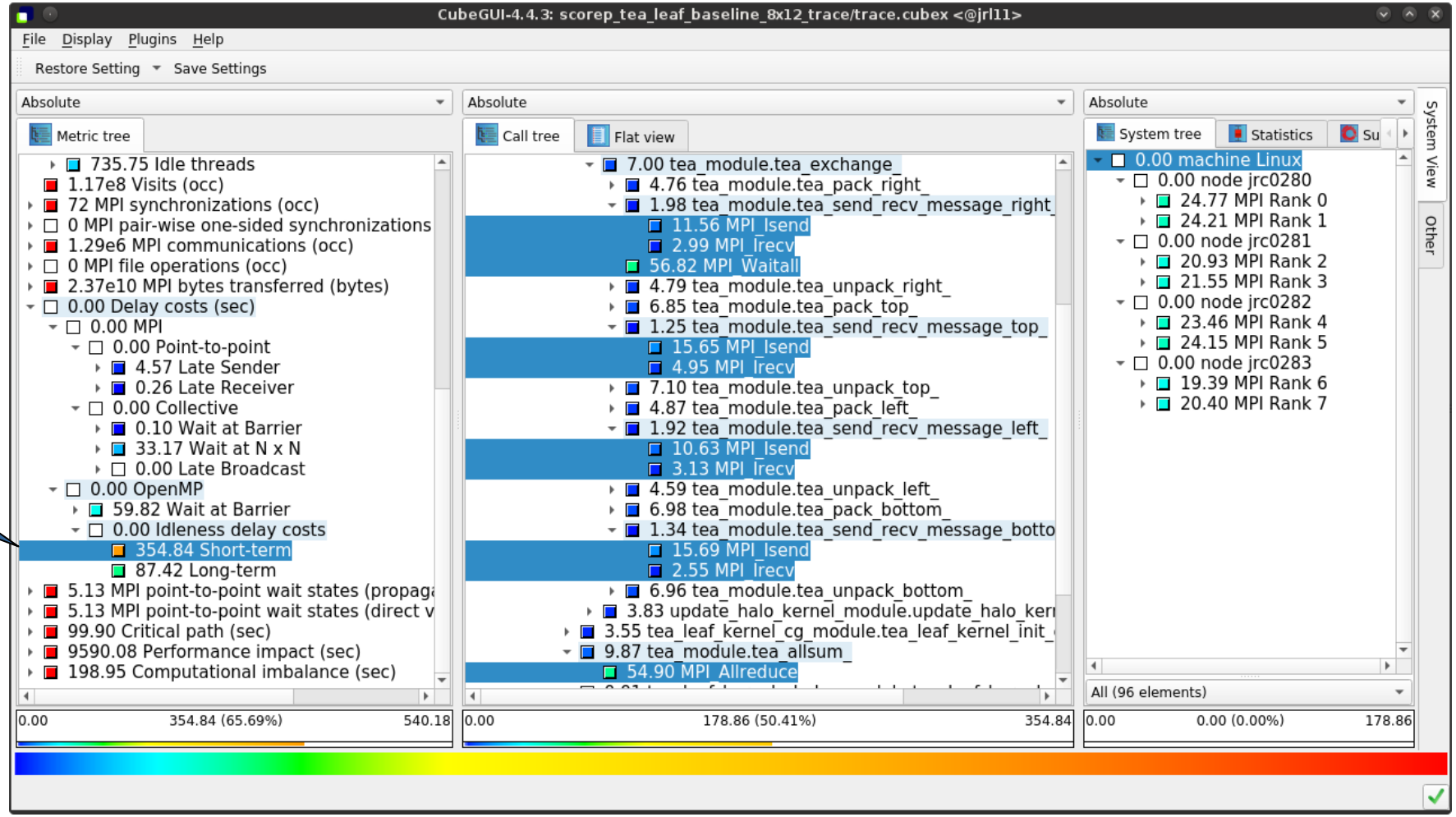
TeaLeaf Scalasca report analysis (I)

While MPI communication time and wait states are small (~0.6% of the total execution time)…



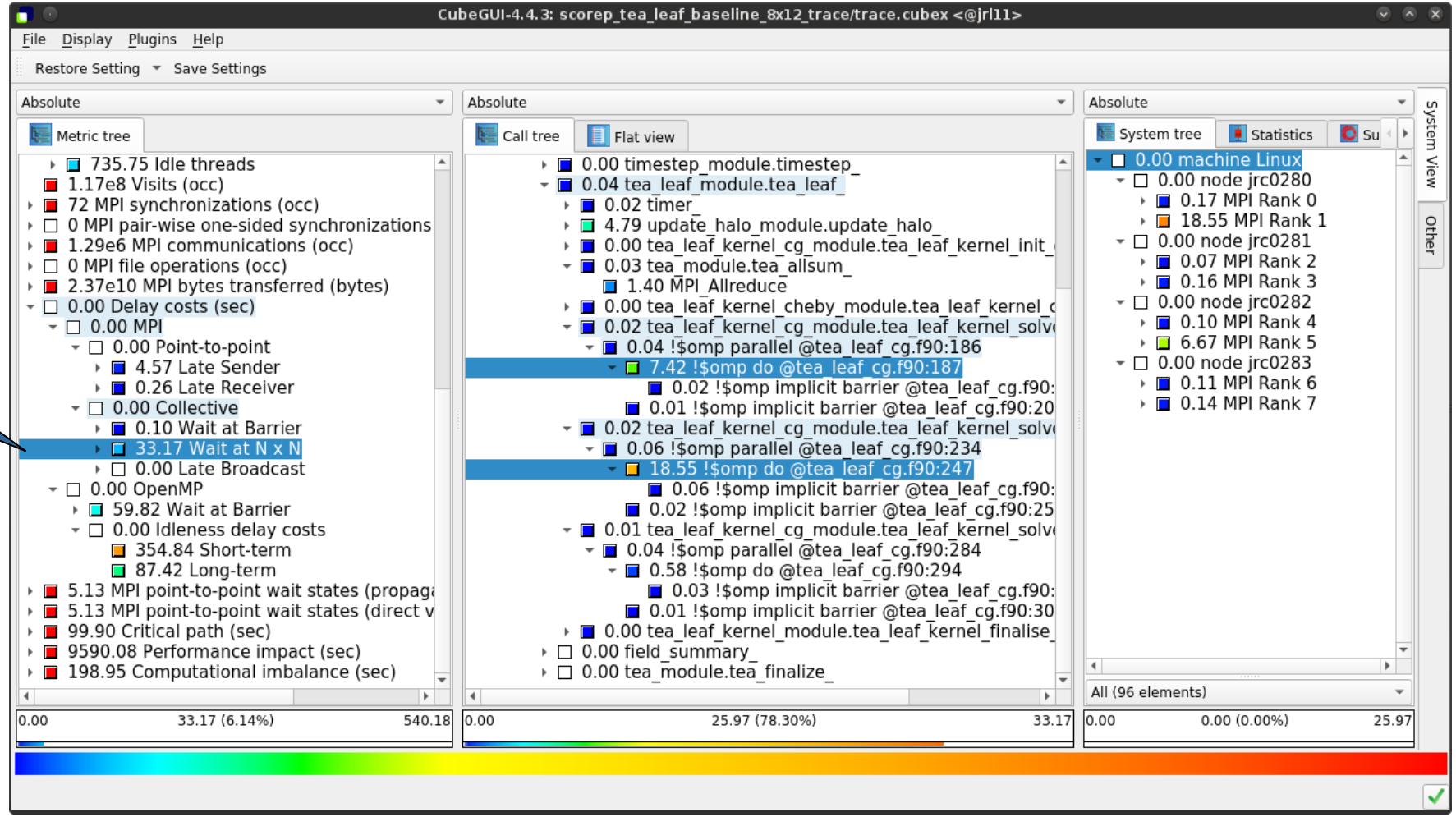
TeaLeaf Scalasca report analysis (II)

…they directly cause a significant amount of the OpenMP thread idleness



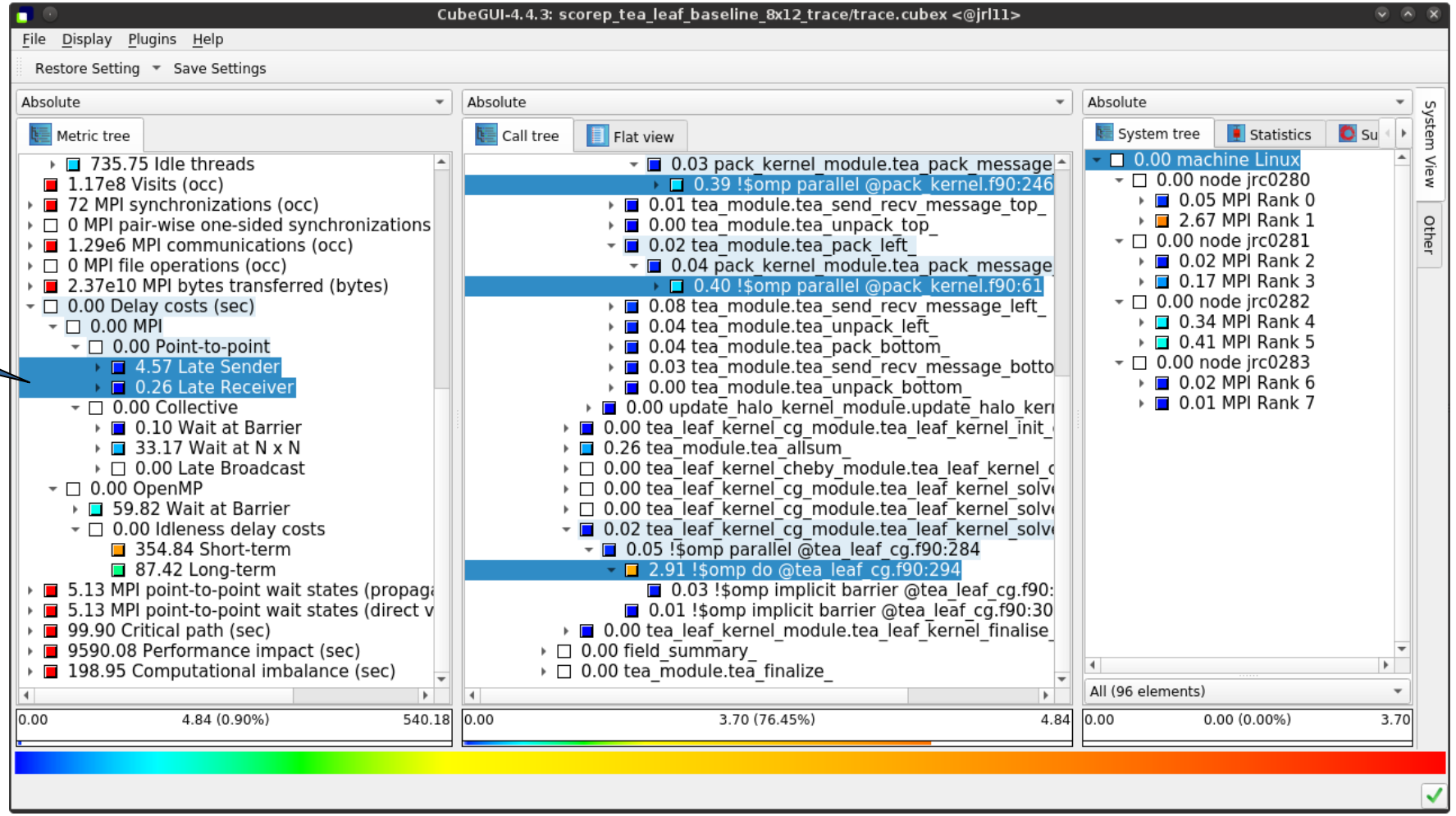
TeaLeaf Scalasca report analysis (III)

The “Wait at NxN” collective wait states are mostly caused by the first 2 OpenMP `do` loops of the solver (on ranks 5 & 1, resp.)…



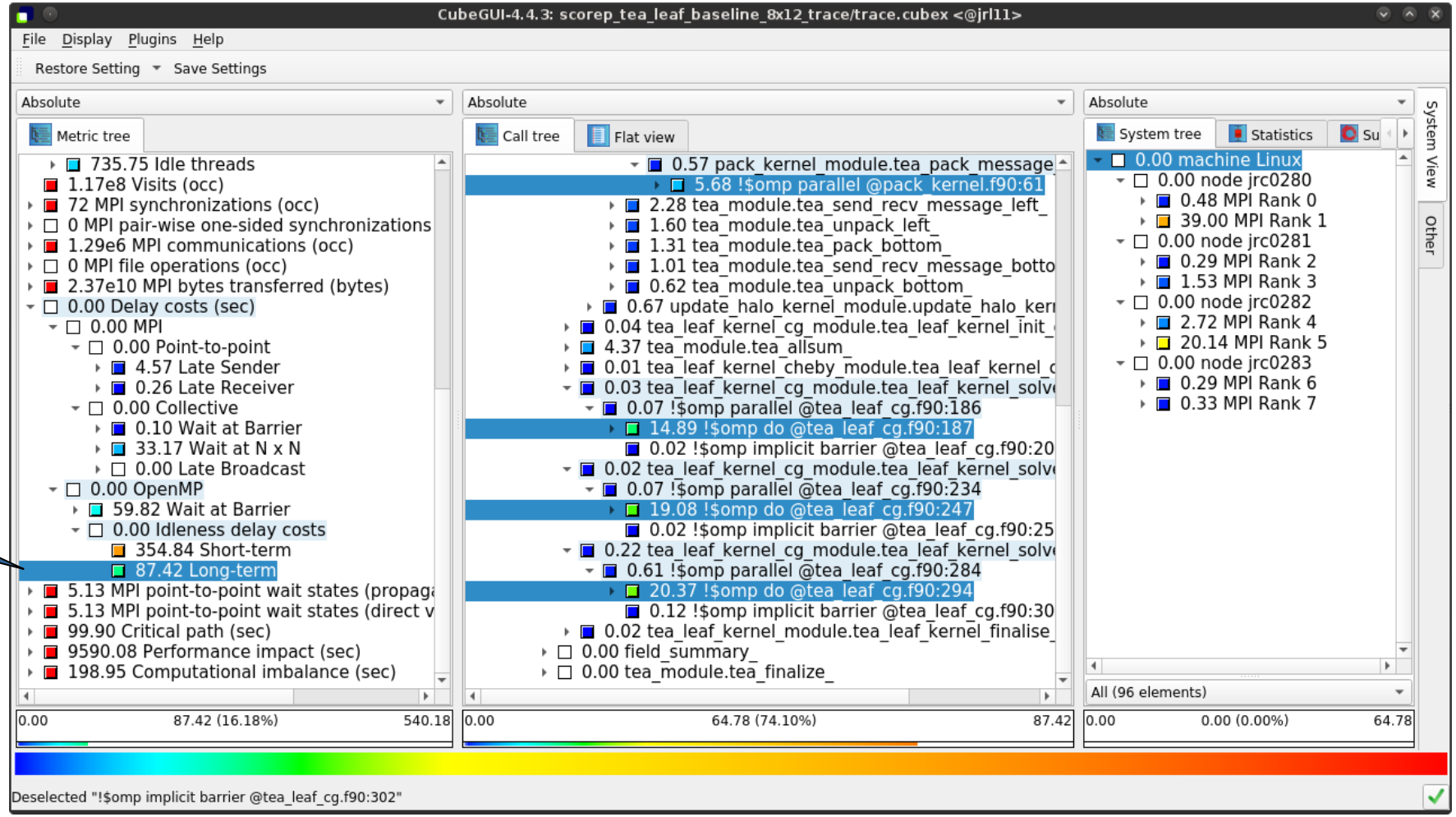
TeaLeaf Scalasca report analysis (IV)

…while the MPI point-to-point wait states are caused by the 3rd solver `do` loop (on rank 1) and two loops in the halo exchange



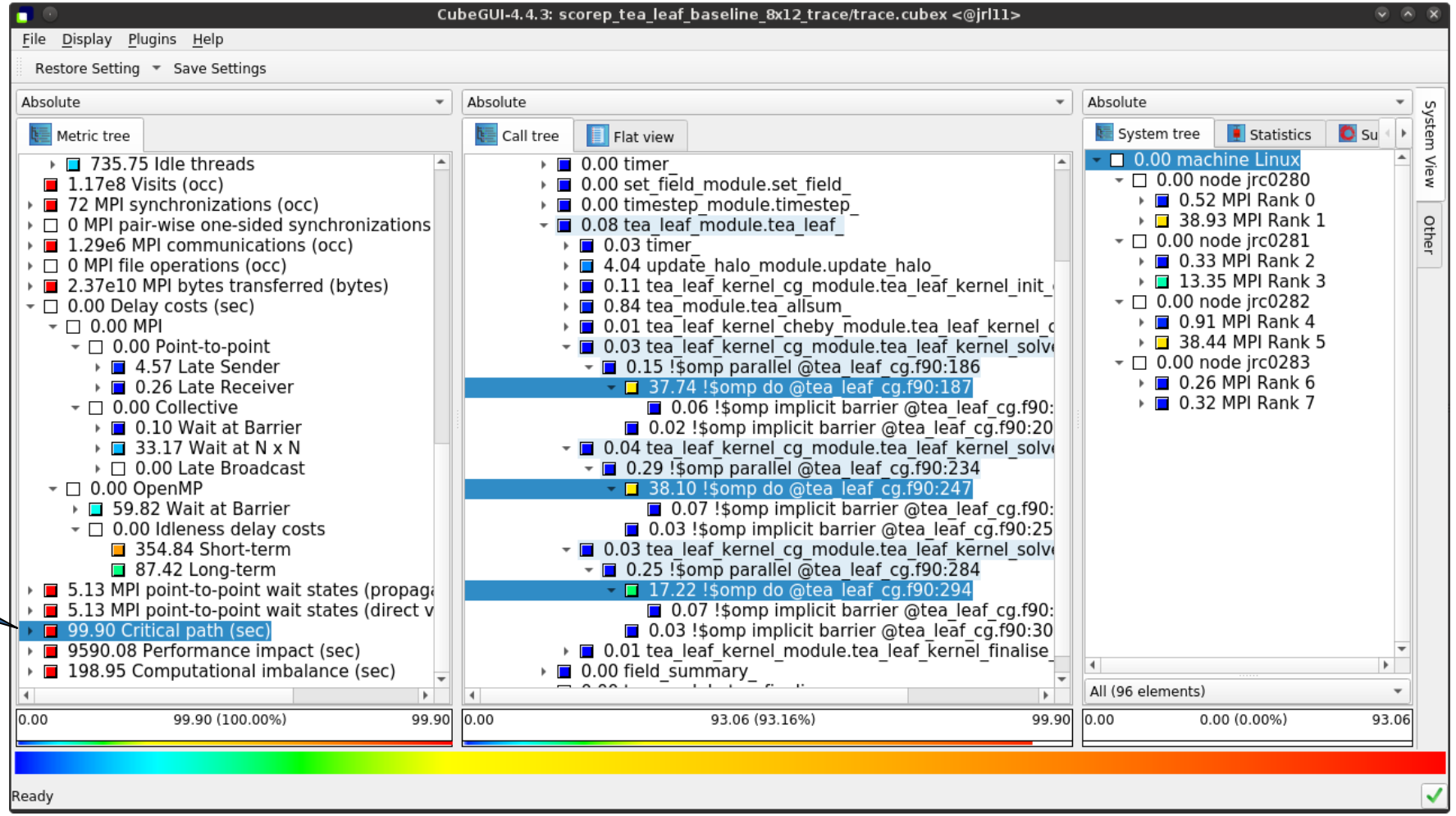
TeaLeaf Scalasca report analysis (V)

Various OpenMP `do` loops (incl. the solver loops) also cause OpenMP thread idleness on other ranks via propagation



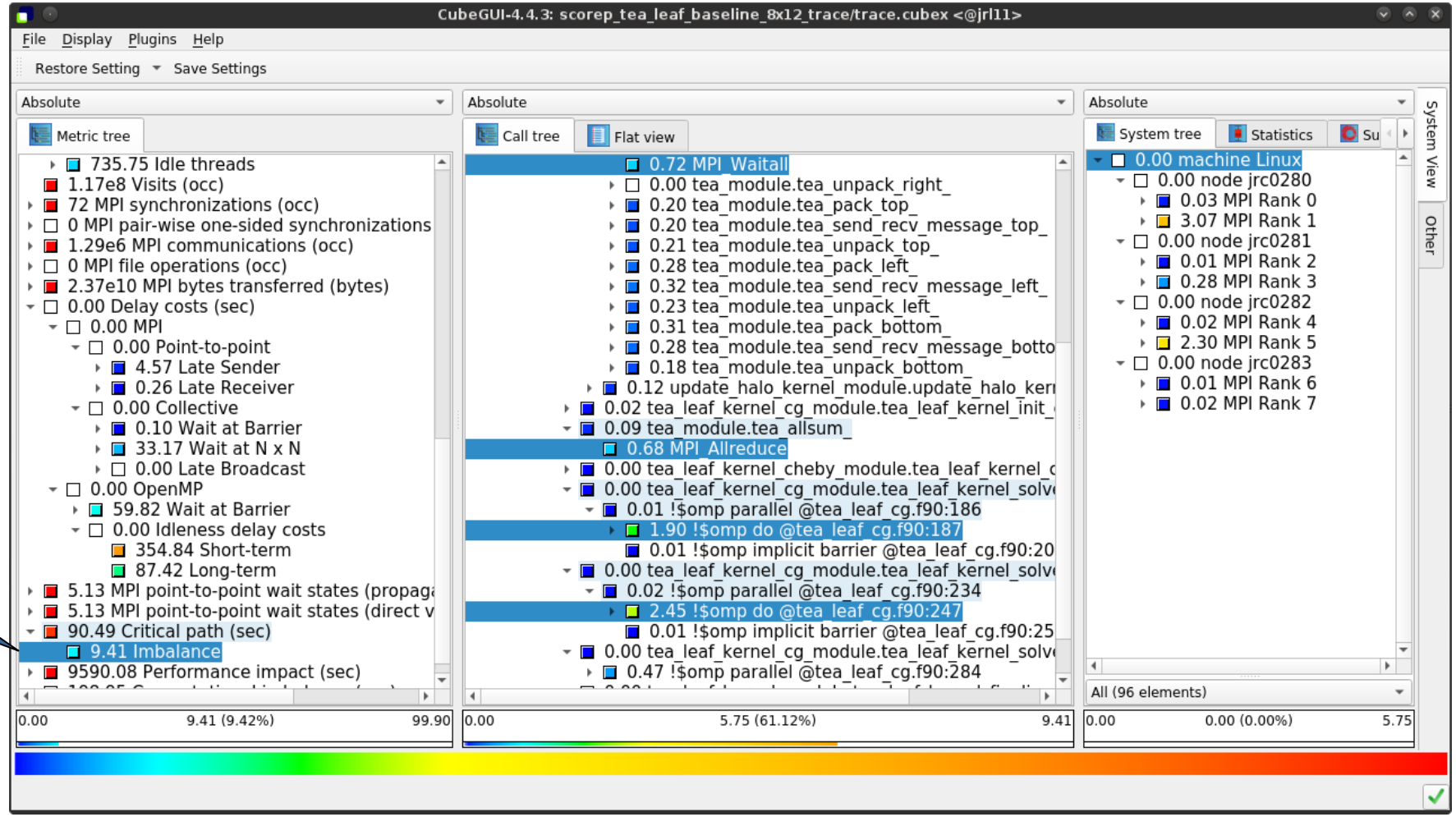
TeaLeaf Scalasca report analysis (VI)

The Critical Path also highlights the three solver loops…



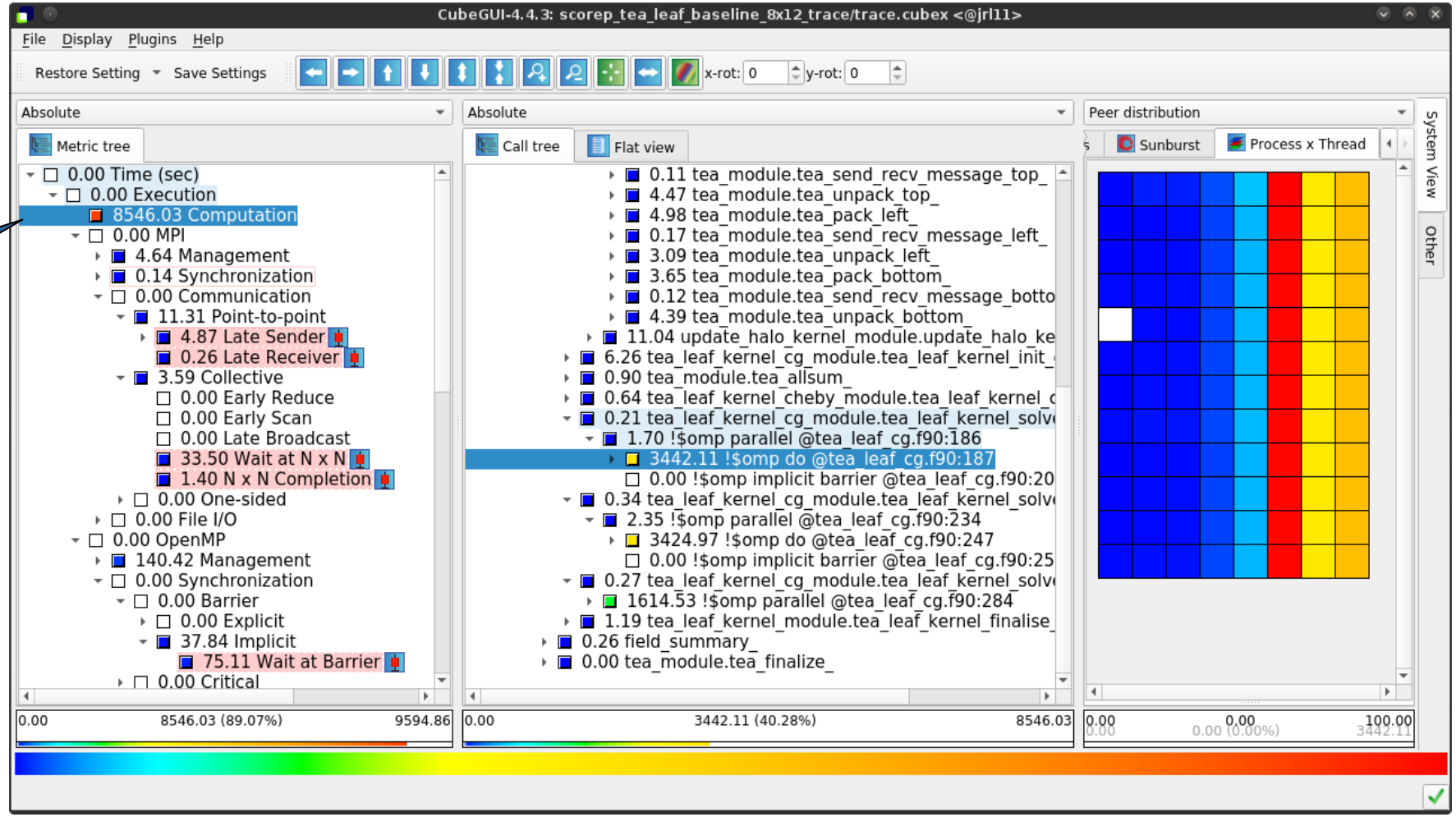
TeaLeaf Scalasca report analysis (VII)

…with imbalance (time on critical path above average) mostly in the first two loops and MPI communication



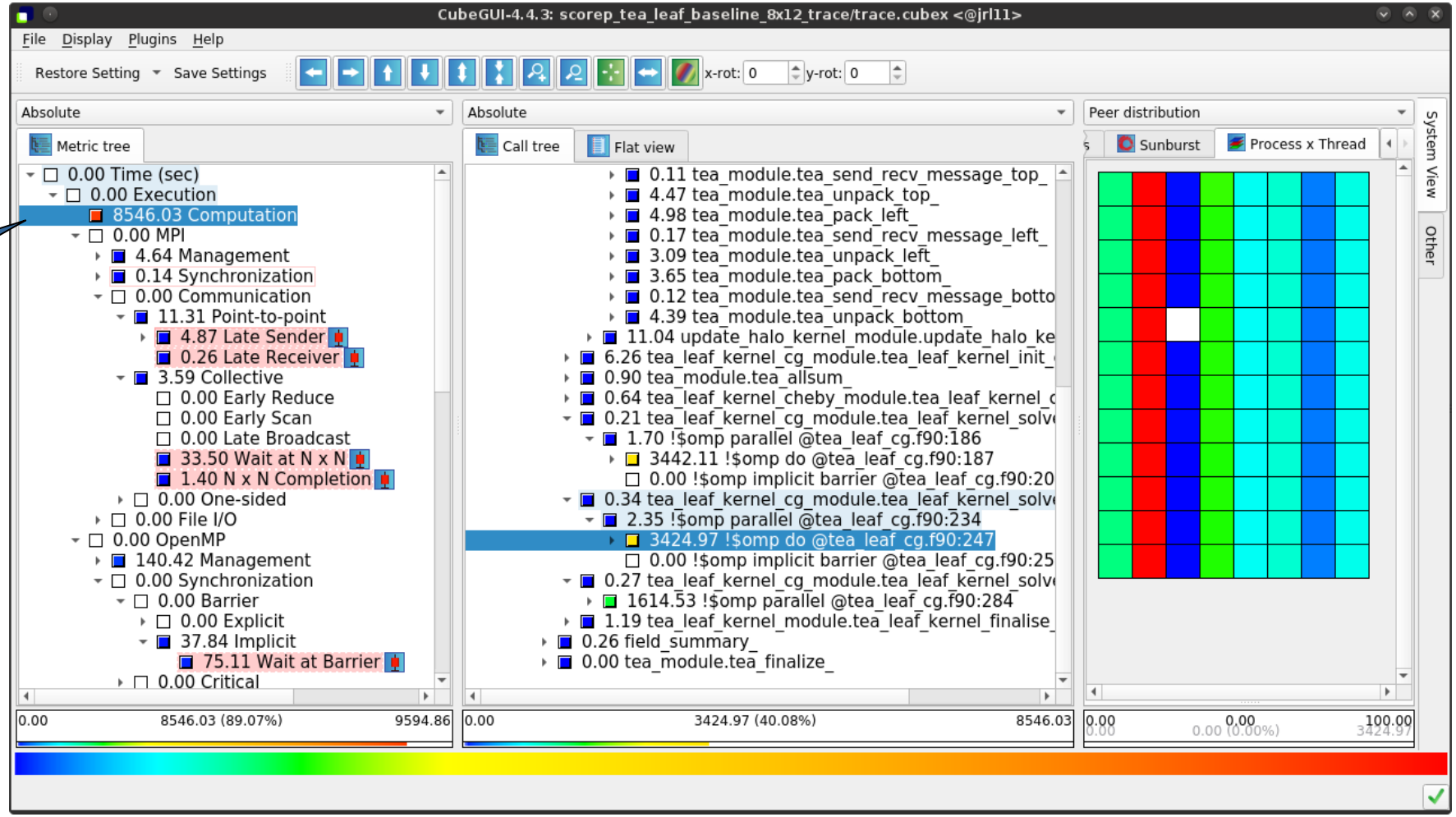
TeaLeaf Scalasca report analysis (VIII)

Computation time of 1st…



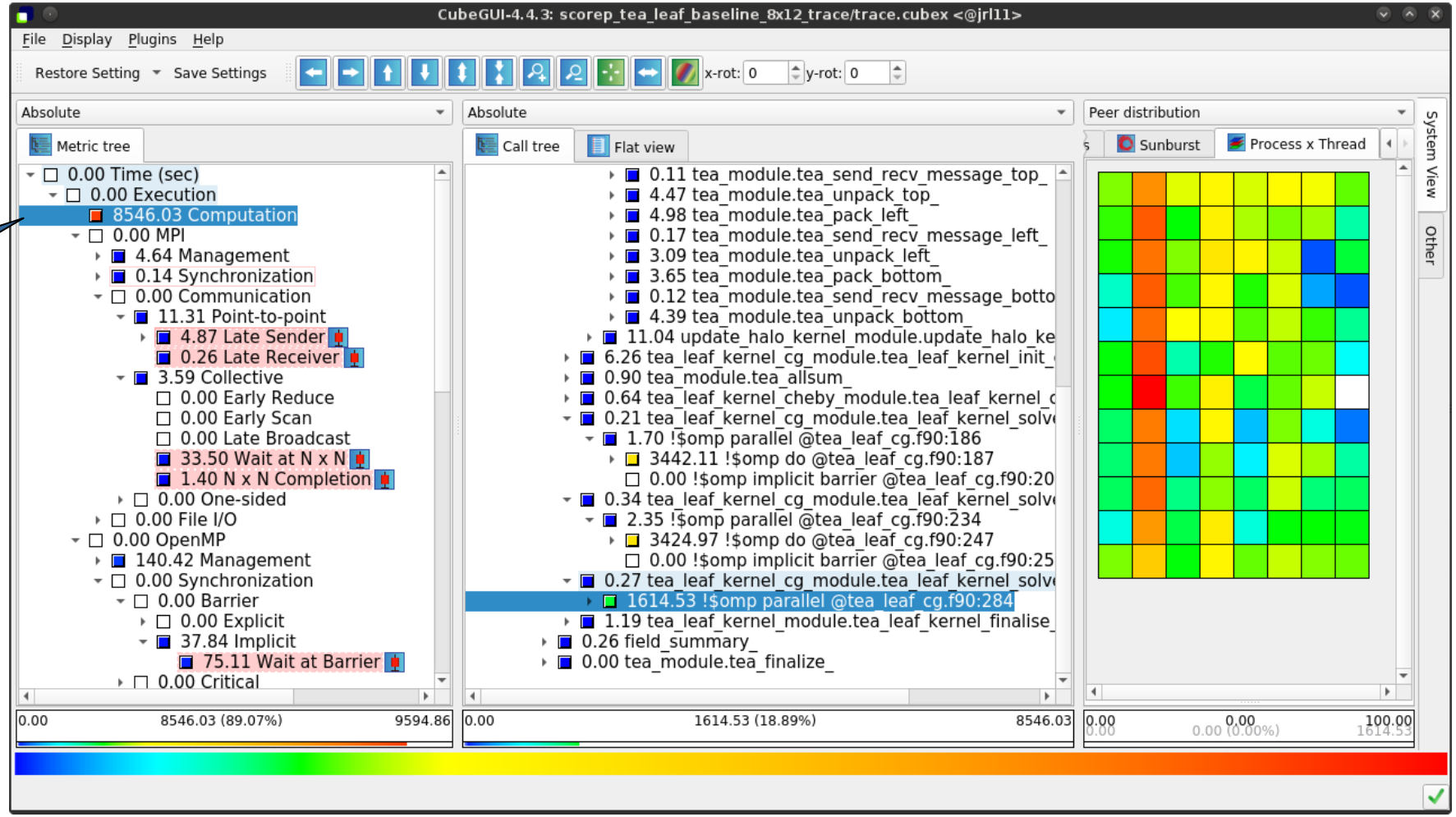
TeaLeaf Scalasca report analysis (IX)

…and 2nd `do` loop mostly balanced within each rank, but vary considerably across ranks…



TeaLeaf Scalasca report analysis (X)

…while the 3rd `do` loop also shows imbalance within each rank



TeaLeaf analysis summary

- The first two OpenMP do loops of the solver are well balanced within a rank, but are imbalanced across ranks
  - ➔ Requires a global load balancing strategy
- The third OpenMP do loop, however, is imbalanced within ranks,
  - causing direct "Wait at OpenMP Barrier" wait states,
  - which cause indirect MPI point-to-point wait states,
  - which in turn cause OpenMP thread idleness
  - ➔ Low-hanging fruit

- Adding a `SCHEDULE(guided)` clause reduced
  - the MPI point-to-point wait states by ~66%
  - the MPI collective wait states by ~50%
  - the OpenMP "Wait at Barrier" wait states by ~55%
  - the OpenMP thread idleness by ~11%
  - ➔ **Overall runtime (wall-clock) reduction by ~5%**