TECHNISCHE UNIVERSITÄT DRESDEN

Center for Information Services and High Performance Computing (ZIH)

Hauptseminar: Rechnerarchitektur und Programmierung

Evaluation von Visualisierungsmethoden zur skalierbaren Darstellung von Laufzeit- und Strukturunterschieden in parallelen Programmabläufen

Ronny Brendel (ronny.brendel@tu-dresden.de)

Tutor: Matthias Weber (matthias.weber@tu-dresden.de)

26th April, 2014

ZIH
Zentrum für Informationsdienste
und Hochleistungsrechnen

Contents

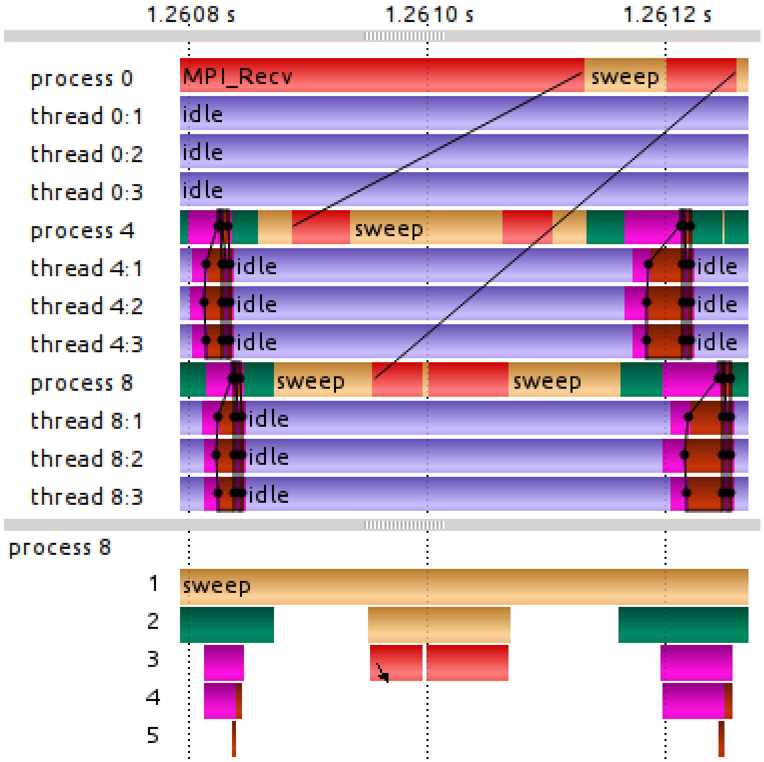
- Introduction
  - Tracing, Vampir, Challenges
  - Differences, Restrictions
- Ideas & Evaluation
  - Profile
  - Call Matrix
- Conclusion
- Future Work
- Sources

Introduction > Tracing

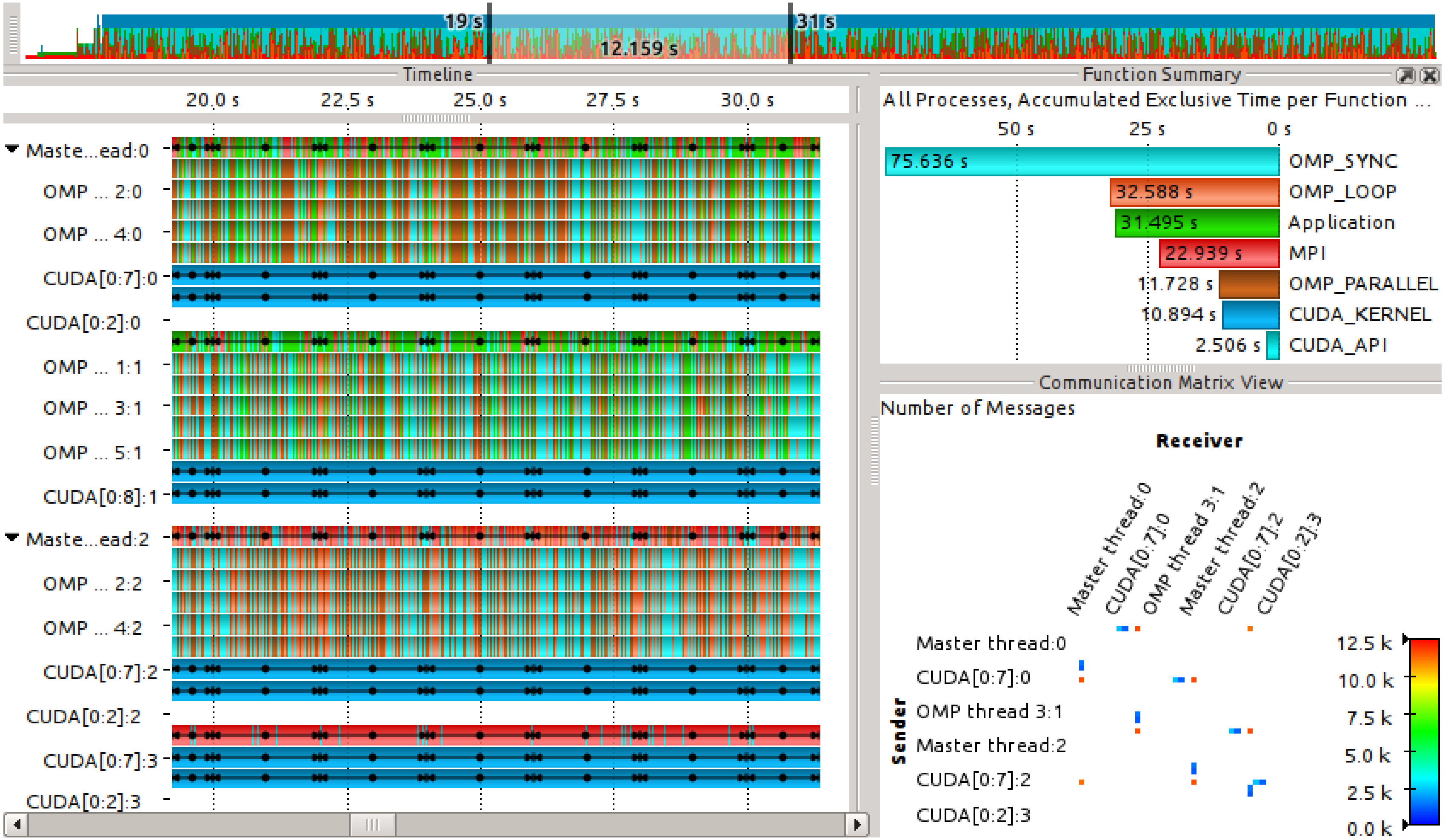
- Profiling
  - Information accumulated per function

```
cumulative self
seconds seconds calls name
   0.31    0.05 25195 QList::isEmpty()
   0.40    0.04 30239 QList::Node::t()
   0.44    0.04 12294 QList::end()
   0.55    0.03  3696 QList::end()
   0.70    0.03  4939 handleEnter
   0.73    0.03 36939 handleLeave
   0.88    0.02 99207 void std::swap()
```

- Tracing
  - Complete information about a program run



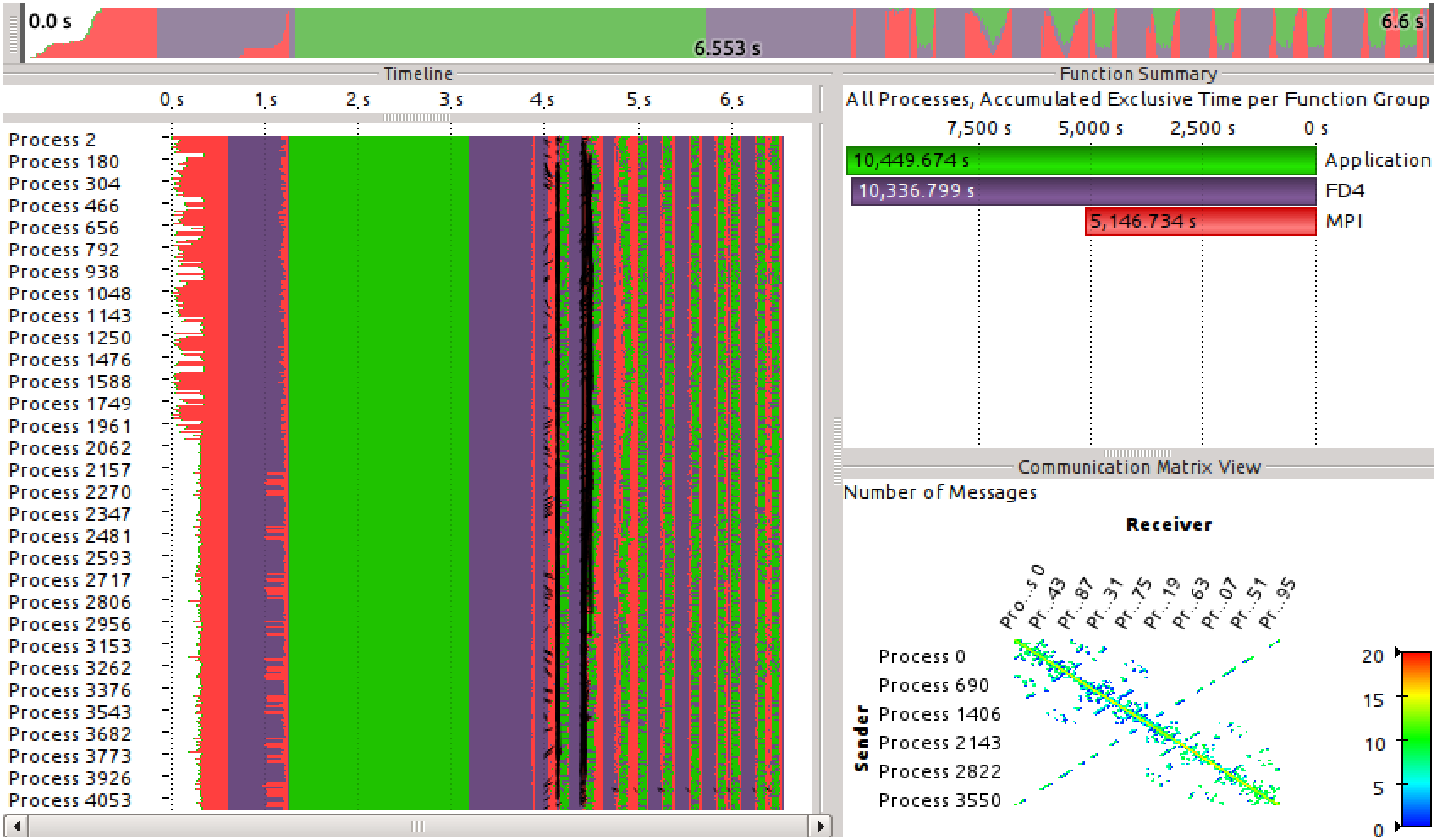
Introduction > Vampir



Introduction > Challenges

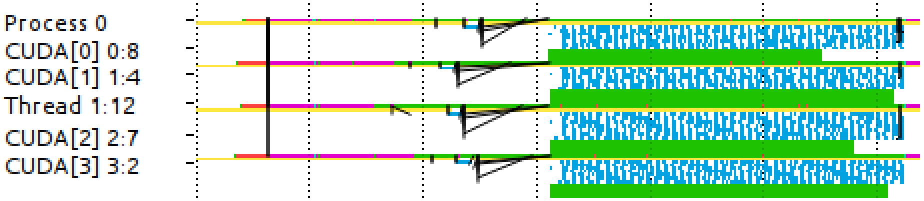
- Datastructures & algorithms:
  - Limited main memory size
  - Achieving scalibility wrt the:
    - Number of processes in the trace
    - Number of processes used for analysis
    - Trace length and detail
- Visualisation:
  - Limited number of Pixels on a screen
  - Achieving scalibilty wrt the number of processes in the trace
  - Aiding the user to gain insight into his program's behaviour and find performance problems

Introduction > Challenges



Introduction > Differences

- Why differences?
  - Present new information
    - Visualise timing differences between similar processes
    - Visualise the impact of optimising a program
    - Compare runs of the same program on different platforms
  - Improve scalibity of existing views
    - Preserve screen real estate by e.g. merging similar processes



  - Aid automatic analysis
    - Detect timing differences between structurally similar processes

Introduction > Restrictions

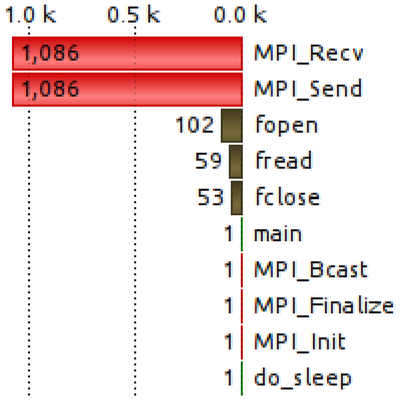
- Because visualising differences between traces in general is much ground to cover, we focus on ...
  - ... the function call stack + timing

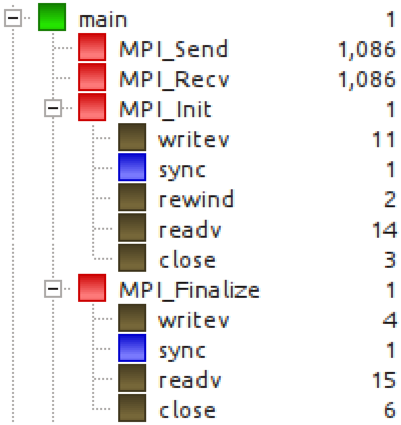
    (no communication, no performance counters, ...)
  - ... comparing processes inside the same program run
  - ... offline analysis

  - ... visualising/comparing profile-ish information

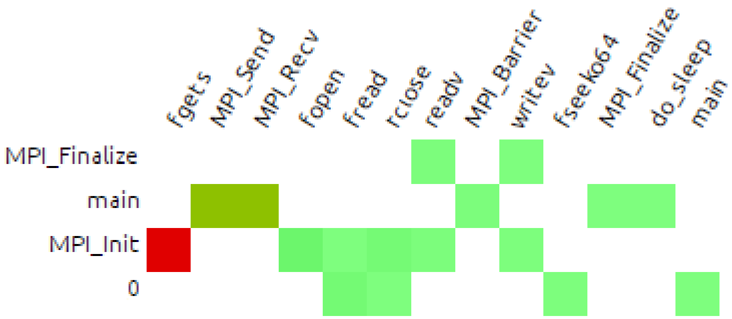
Ideas & Evaluation

- Profil-ish information:
  - Profile
    - Accumulated information for each function
  - Call Tree
    - Accumulated information per call stack configuration
  - Call Matrix
    - Accumulated information per caller/callee pair



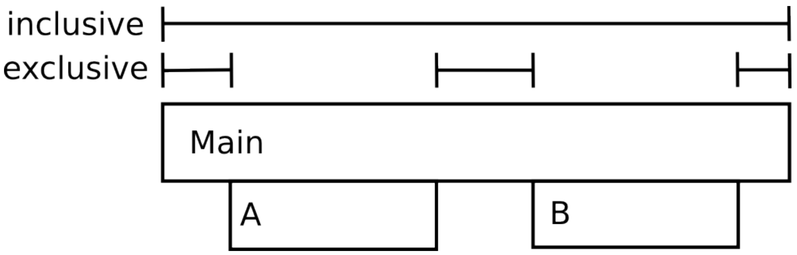


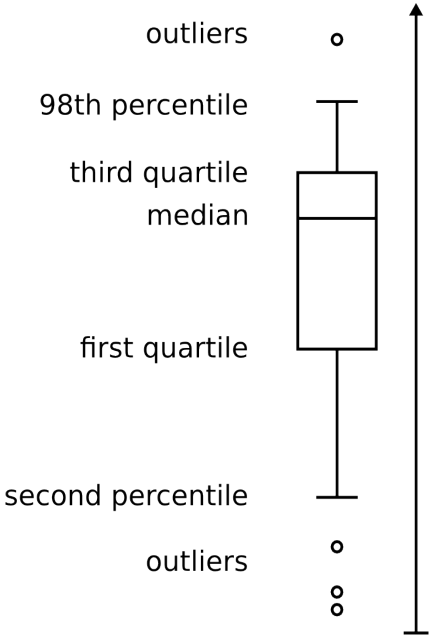
```
main                 1
  MPI_Send       1,086
  MPI_Recv       1,086
  MPI_Init           1
    writev          11
    sync             1
    rewind           2
    readv           14
    close            3
  MPI_Finalize       1
    writev           4
    sync             1
    readv           15
    close            6
```



Ideas & Evaluation

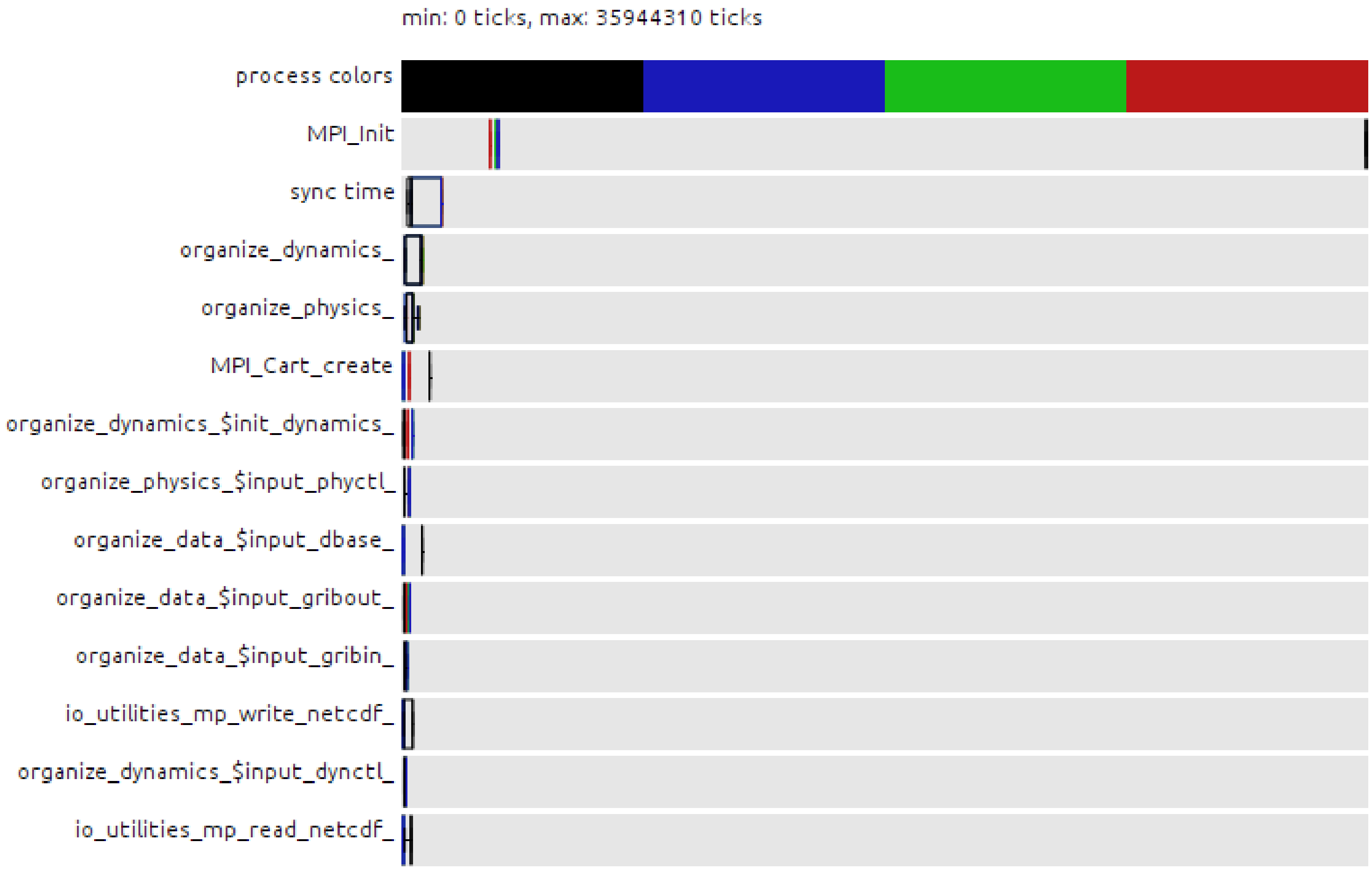
- What information are we comparing?
  - Execution times
    - Exclusive
    - Inclusive
    - Derived Values
      - Min, max, average, standard deviation
      - Median, Quartiles/Percentiles
        → Boxplot
  - Number of invocations





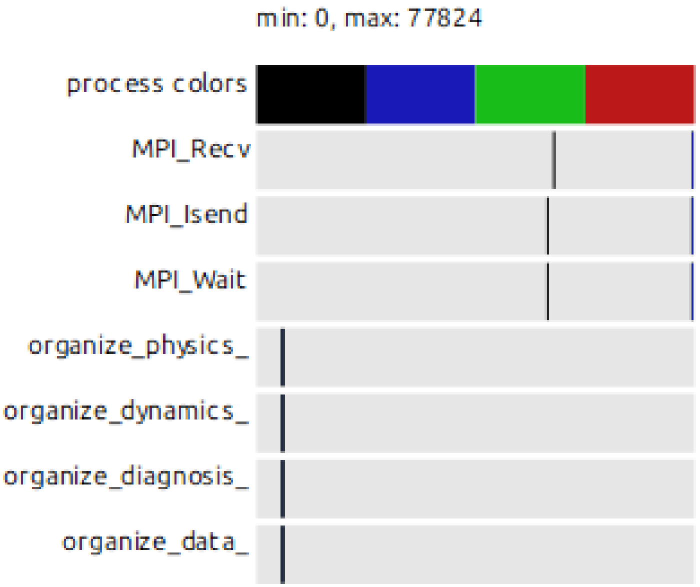
Ideas & Evaluation > Profile

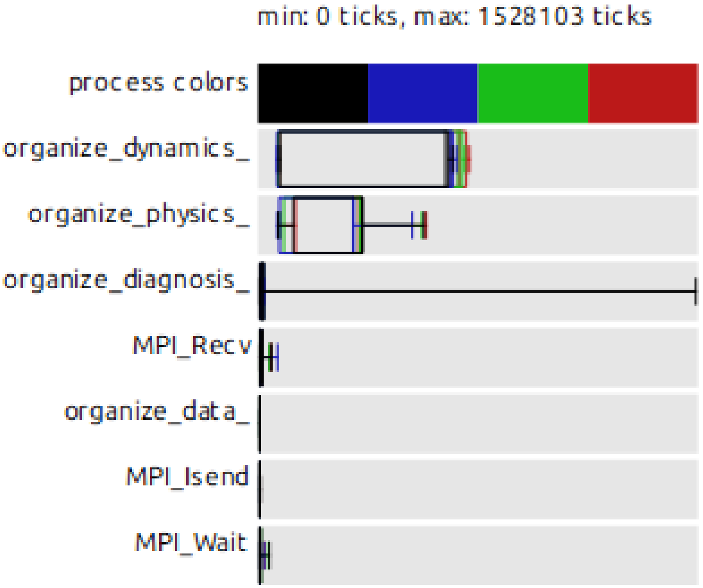
- Exclusive time, four processes


min: 0 ticks, max: 35944310 ticks
process colors
MPI_Init
sync time
organize_dynamics_
organize_physics_
MPI_Cart_create
organize_dynamics_$init_dynamics_
organize_physics_$input_phyctl_
organize_data_$input_dbase_
organize_data_$input_gribout_
organize_data_$input_gribin_
io_utilities_mp_write_netcdf_
organize_dynamics_$input_dynctl_
io_utilities_mp_read_netcdf_

Ideas & Evaluation > Profile

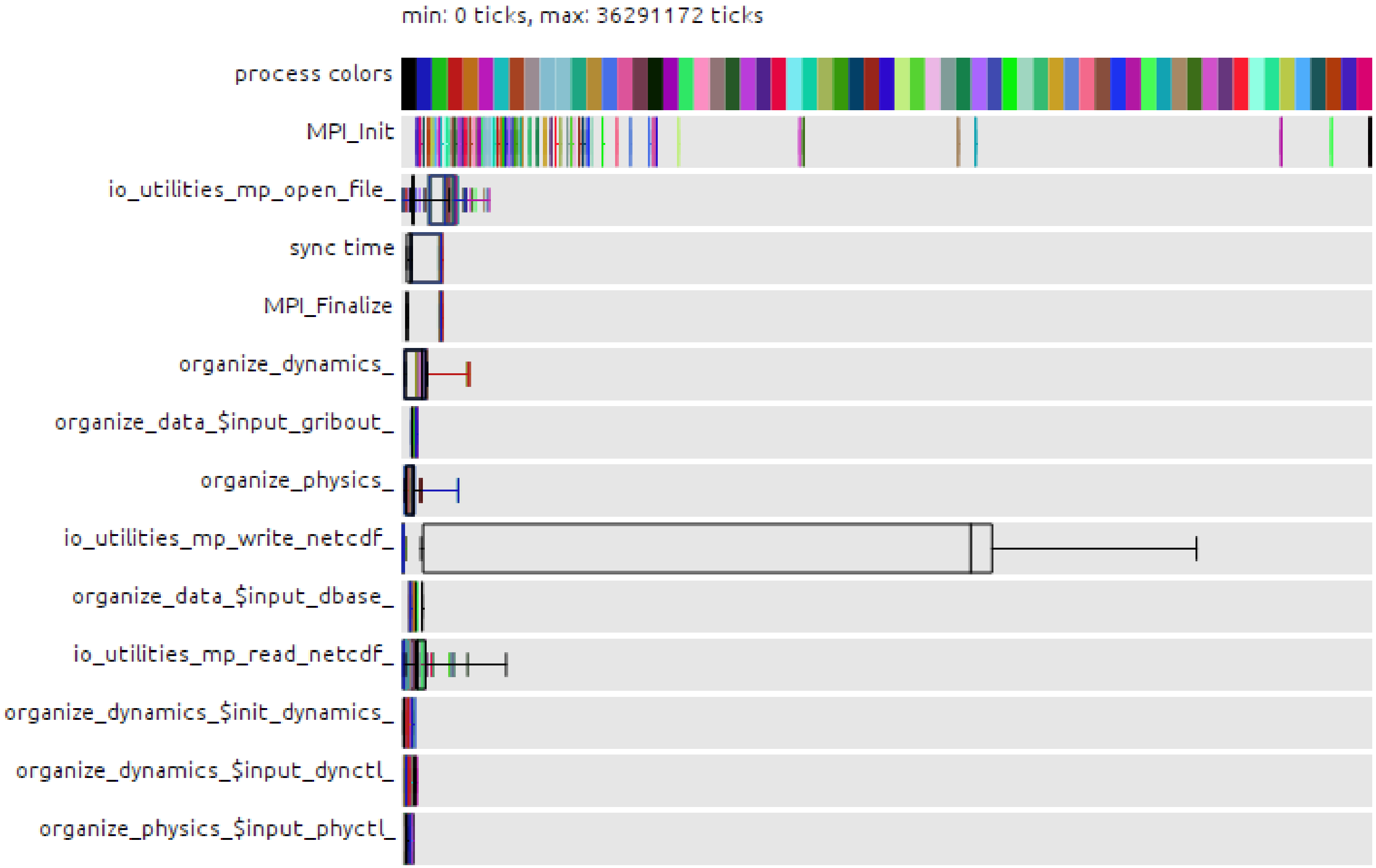
- Invocation count, four processes
- Exclusive time, four processes





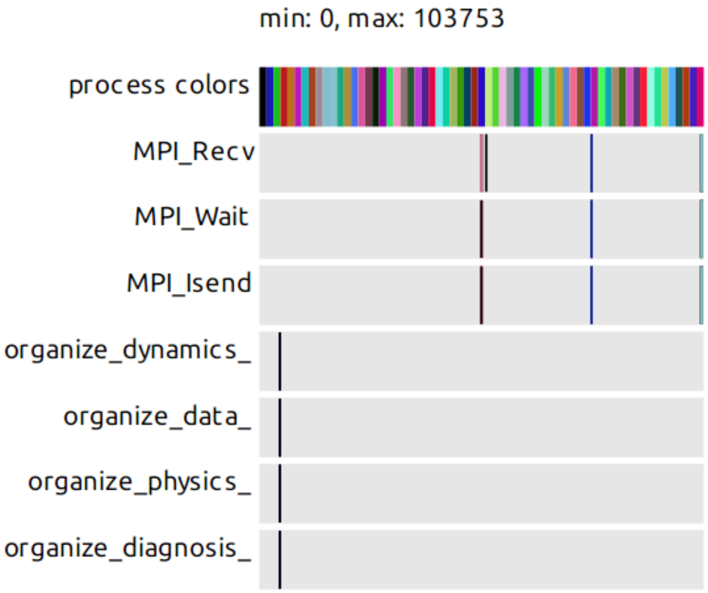
Ideas & Evaluation > Profile

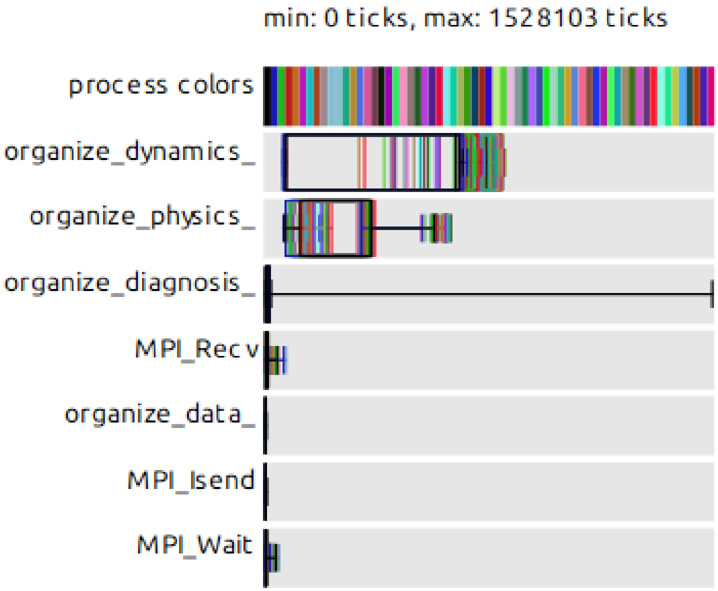
- Exclusive time, 64 processes



Ideas & Evaluation > Profile

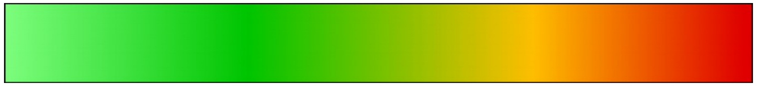
- Invocation count, 64 processes
- Exclusive time, 64 processes

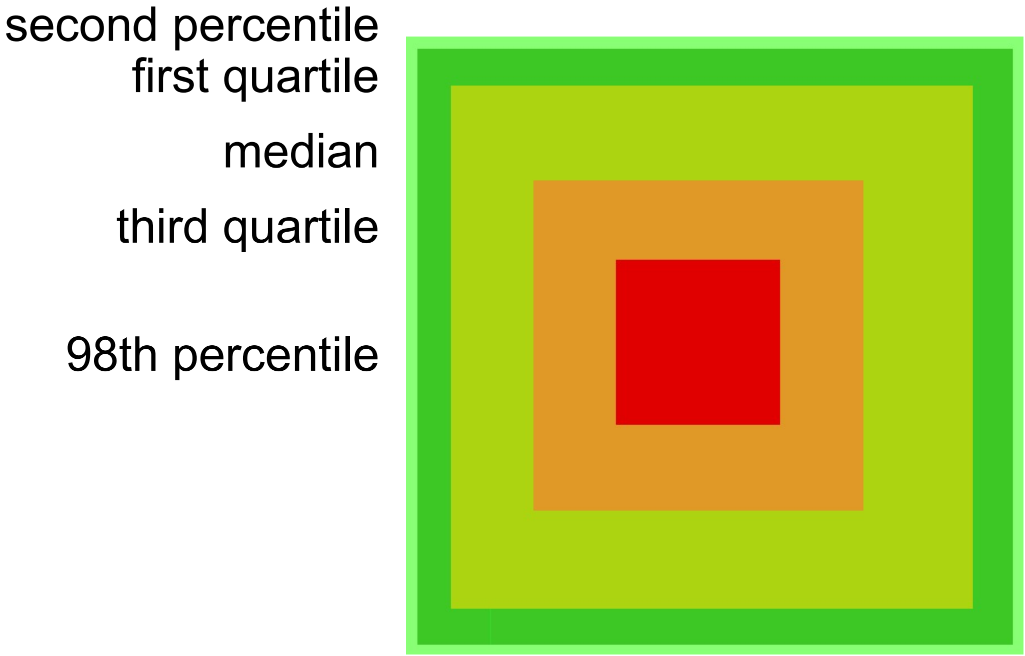


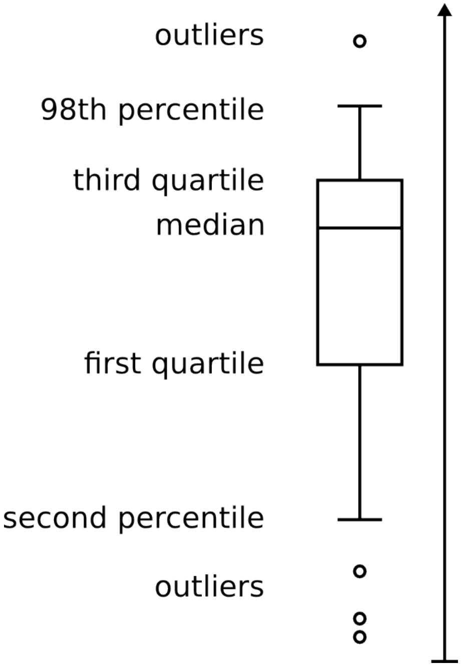


Ideas & Evaluation > Call Matrix

- Color-coded box plot
  - Area is divided in 1 + 3 + 6 + 3 + 1 parts
  - Linear gradient:

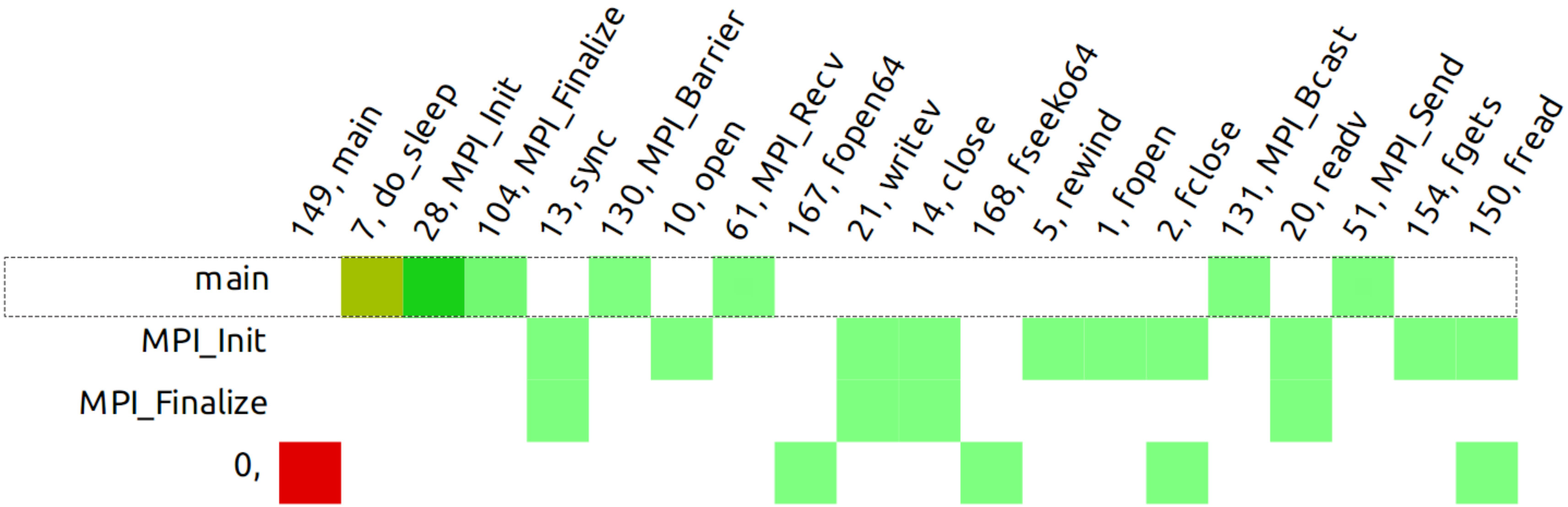






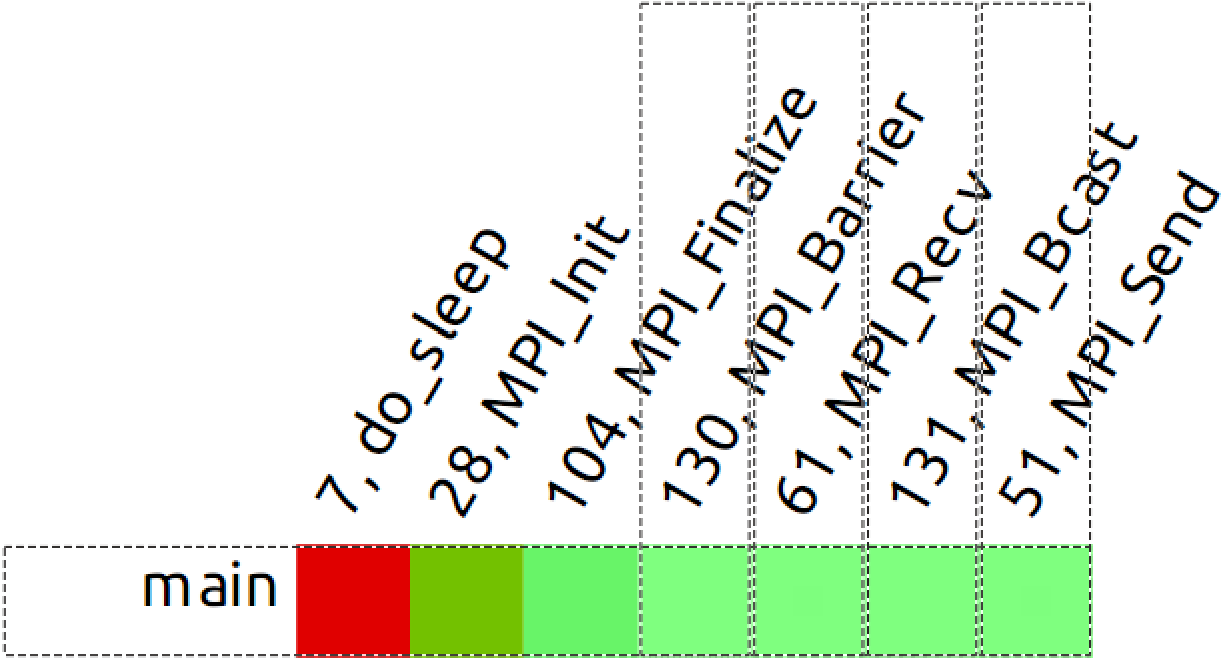
Ideas & Evaluation > Call Matrix

- Ping pong example, 1 process, inclusive time



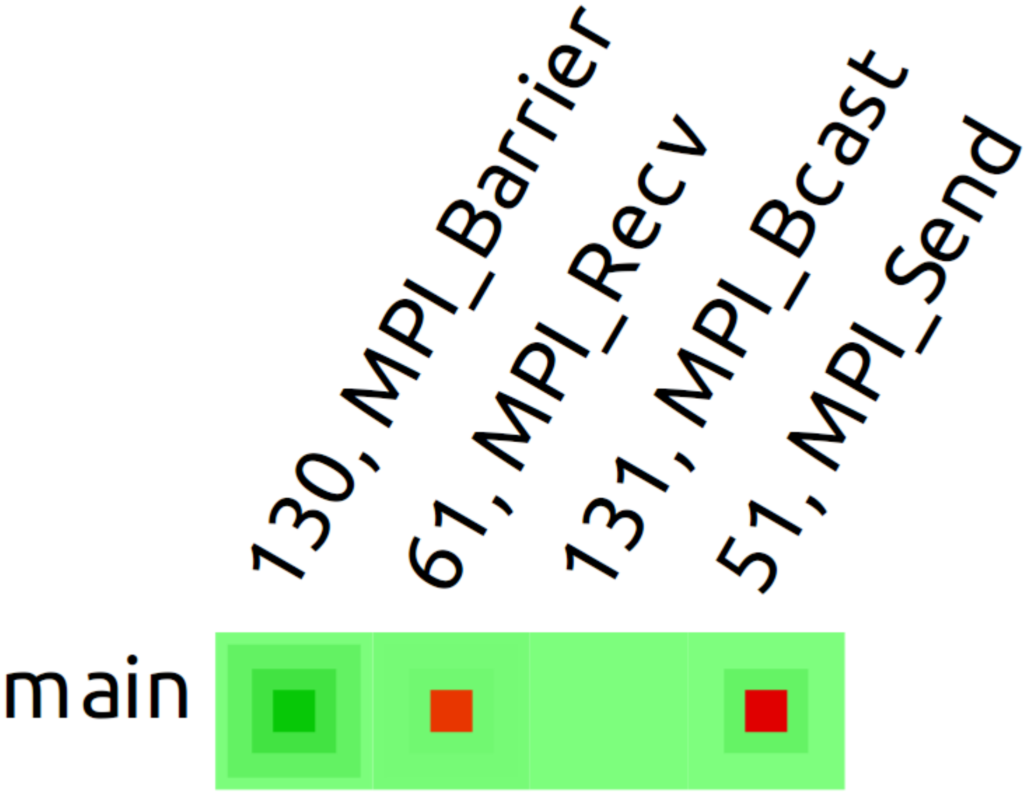
Ideas & Evaluation > Call Matrix

- Ping pong example, 1 process, inclusive time
- main selected



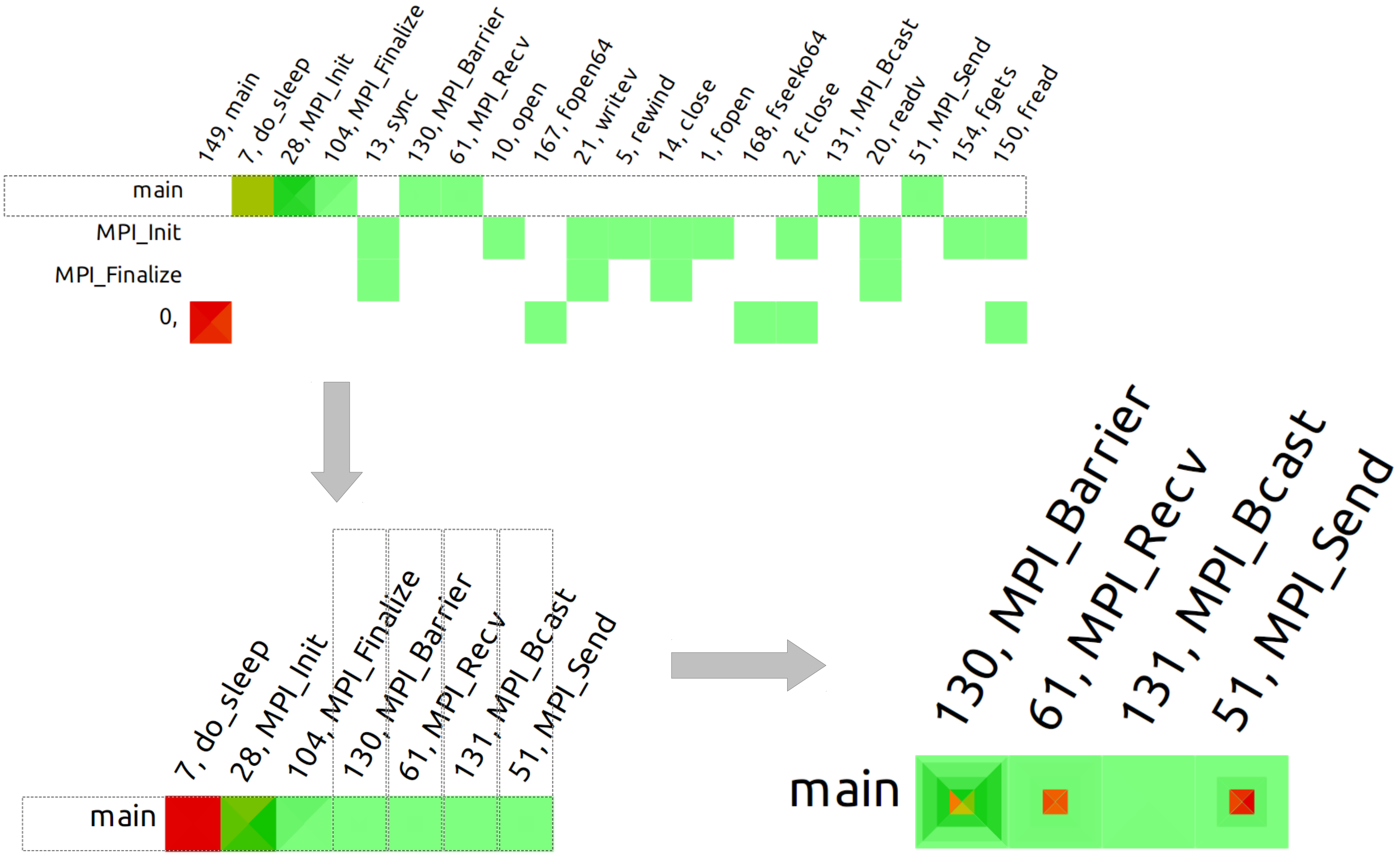
Ideas & Evaluation > Call Matrix

- Ping pong example, 1 process, inclusive time
- main + interesting sub-calls selected



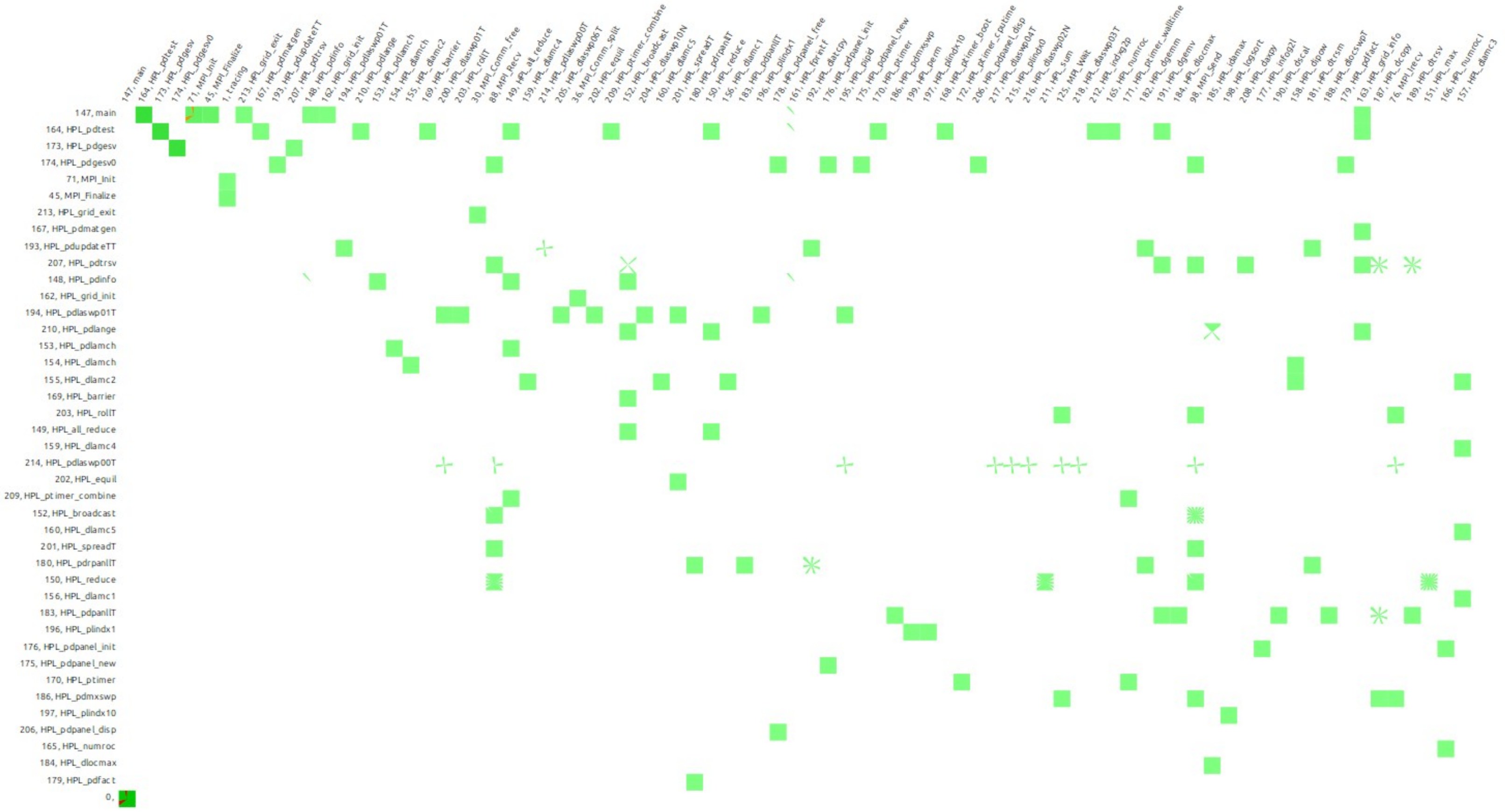
Ideas & Evaluation > Call Matrix

- Ping pong example, 4 processes, inclusive time



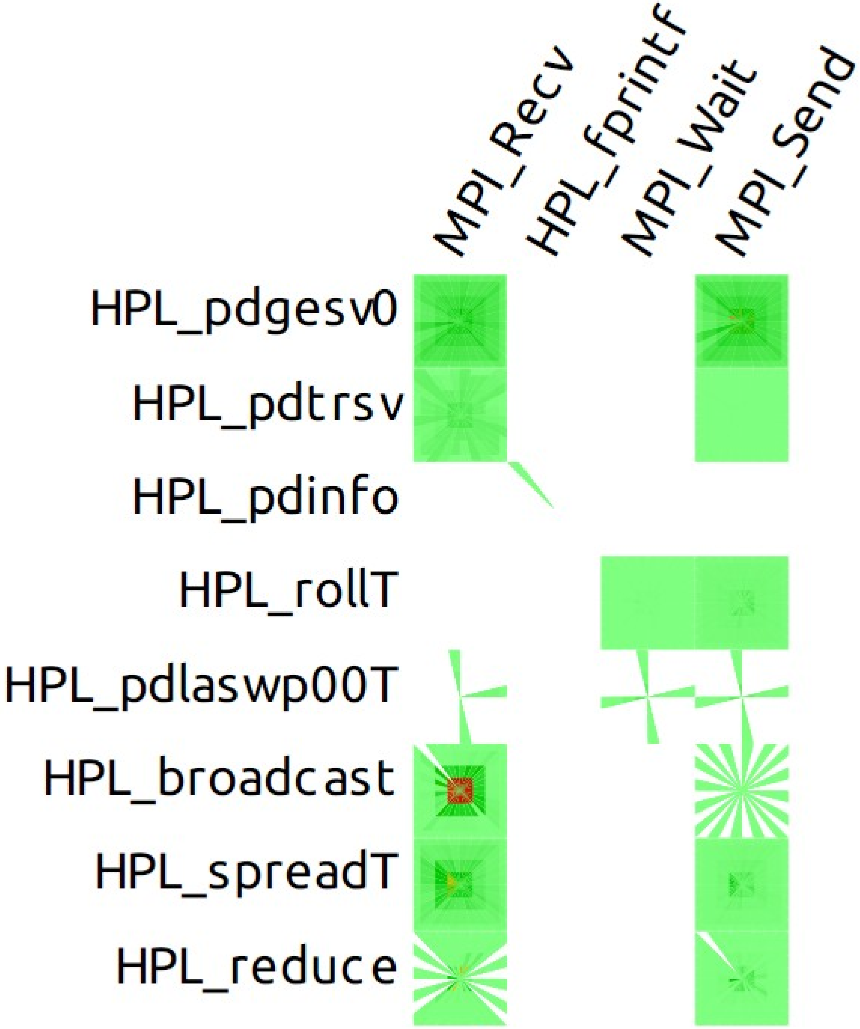
Ideas & Evaluation > Call Matrix

- Linpack, 32 processes, inclusive time



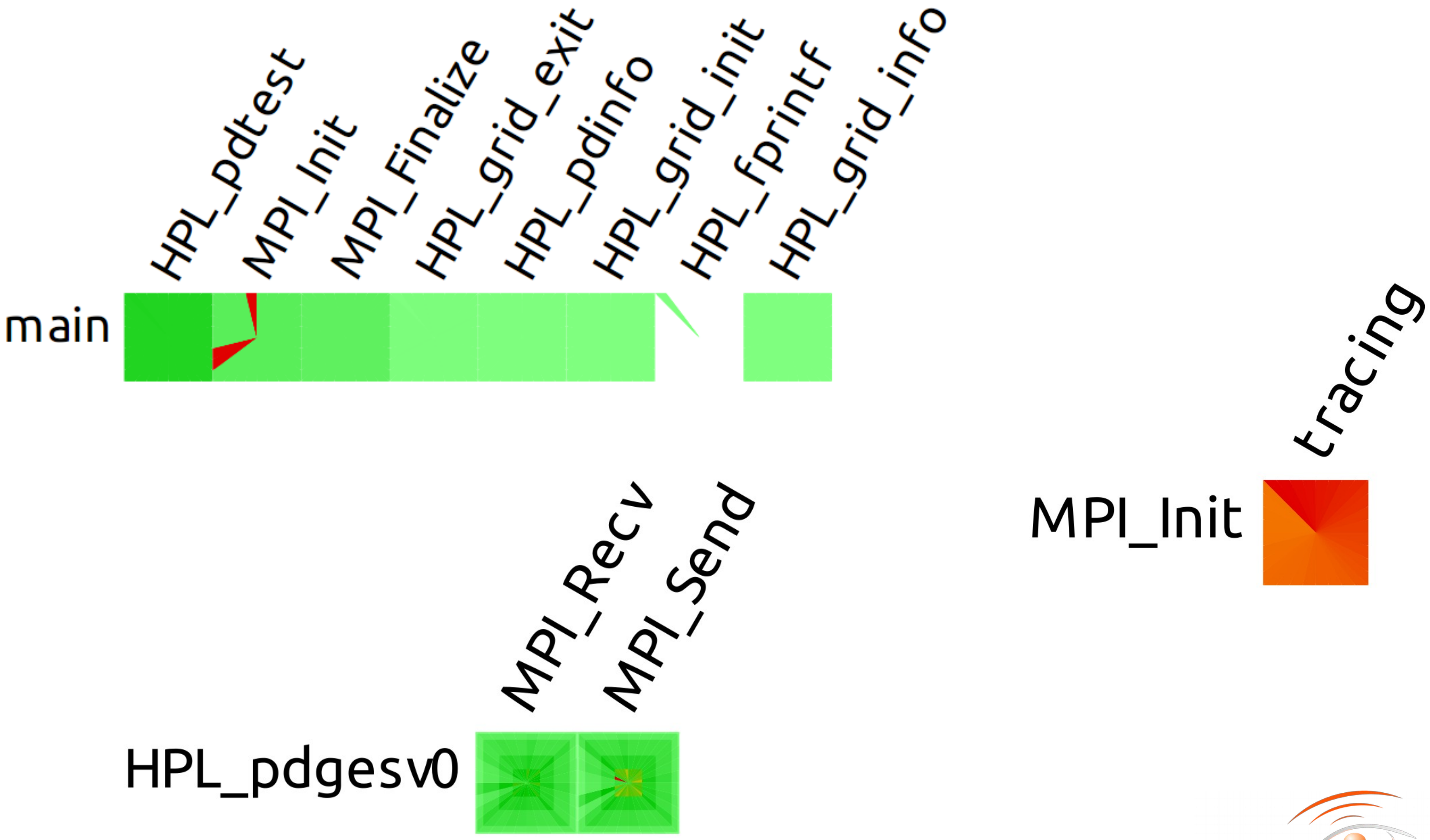
Ideas & Evaluation > Call Matrix

- Linpack, 32 processes, inclusive time



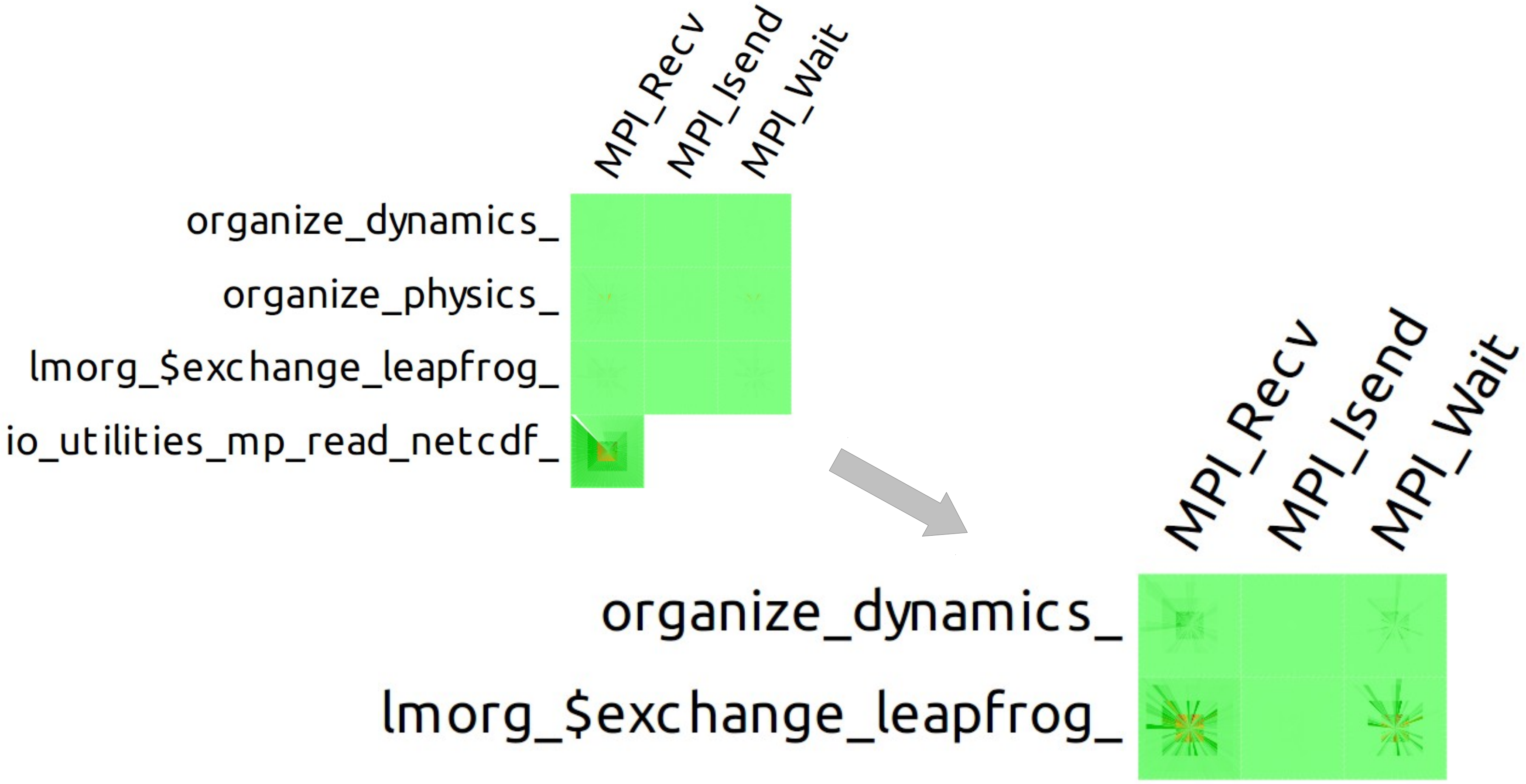
Ideas & Evaluation > Call Matrix

- Linpack, 32 processes, inclusive time



Ideas & Evaluation > Call Matrix

- COSMO-SPECS, 32 processes, exclusive time



Conclusion

- Comparing profiles:
  - Good for getting a qualitative overview
  - Not suitable for finding details
- Comparing call matrixes:
  - Not suitable for visualisation in its current state
  - Improvable through automatic filtering/grouping of functions and processes
  - Simple, scalable and helpful for automatic analysis
- Generally:
  - Boxplots are good
  - Median/Percentiles are good (more difficult to determine, though)
  - Filtering profiles by callers makes sense

Future Work

- Short-term:
  - Compare/Visualise call trees
  - Develop similarity metrics based on profiles and call matrixes

    → Automatically group processes

- Long-term:
  - Visualise differences and similiarities in structure and timing of traces (not profiles, but stack over time)

Sources

- [1] Andreas Knüpfer, Holger Brunst, Jens Doleschal, Matthias Jurenz, Matthias Lieber, Holger Mickler, Matthias S. Müller, and Wolfgang E. Nagel. The Vampir Performance Analysis Tool-Set. In *Tools for High Performance Computing*, pages 139–155. Springer, 2008.
- [2] Matthias Weber, Ronny Brendel, and Holger Brunst. Trace File Comparison with a Hierarchical Sequence Alignment Algorithm. In *IEEE 10th International Symposium on Parallel and Distribut ed Processing with Applications*, pages 247–254. IEEE, 2012.
- [3] Matthias Weber, Kathryn Mohror, Martin Schulz, Bronis R. de Supinski, Holger Brunst, and Wolfgang E. Nagel. Alignment-Based Metrics for Trace Comparison. In *Euro-Par 2013 Parallel Processing*, pages 29–40. Springer, 2013.

Thank You!