703308 VO High-Performance Computing WS2021/2022
Performance Analysis with Scalasca

Philipp Gschwandtner

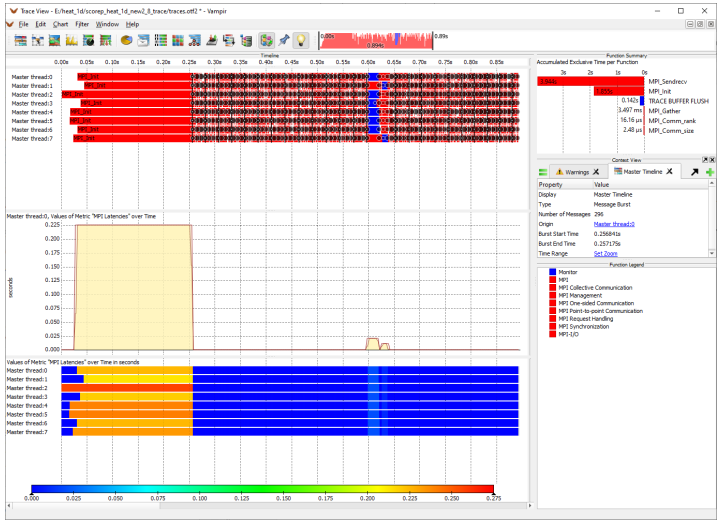
Overview

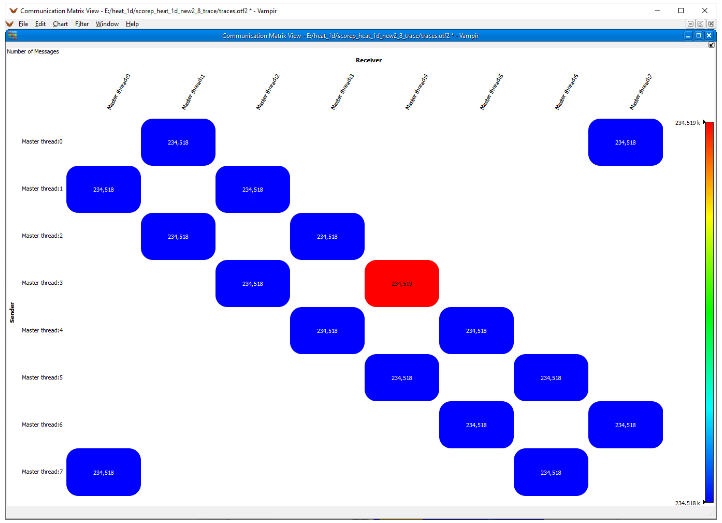
- Short introduction to the Scalasca performance analysis suite

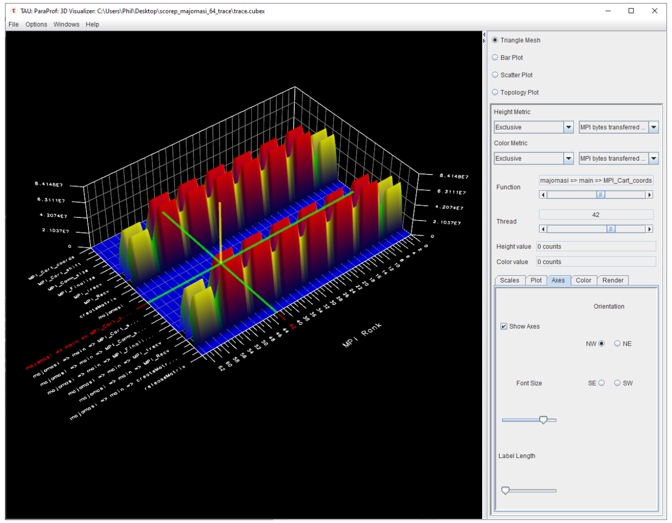
- Live demo showing
  - how it works,
  - how to **not** kill your hard disk when tracing, and
  - how to interpret the results

Motivation

- In the proseminar, we only saw text-based performance data so far
  - this is boring
- Data visualization can convey information much more effectively







Score-P Instrumentation and Measurement Tool

- jointly developed by the Jülich Supercomputing Centre and the Technical University of Dresden
- allows instrumenting multi-threaded, multi-processed, accelerated programs (OpenMP, Pthreads, MPI, CUDA, OpenCL, …)
- allows dynamic filtering of measurement data generation without re-compilation
- highly scalable (tested up to ~2 million threads)

Scalasca Analysis Tool

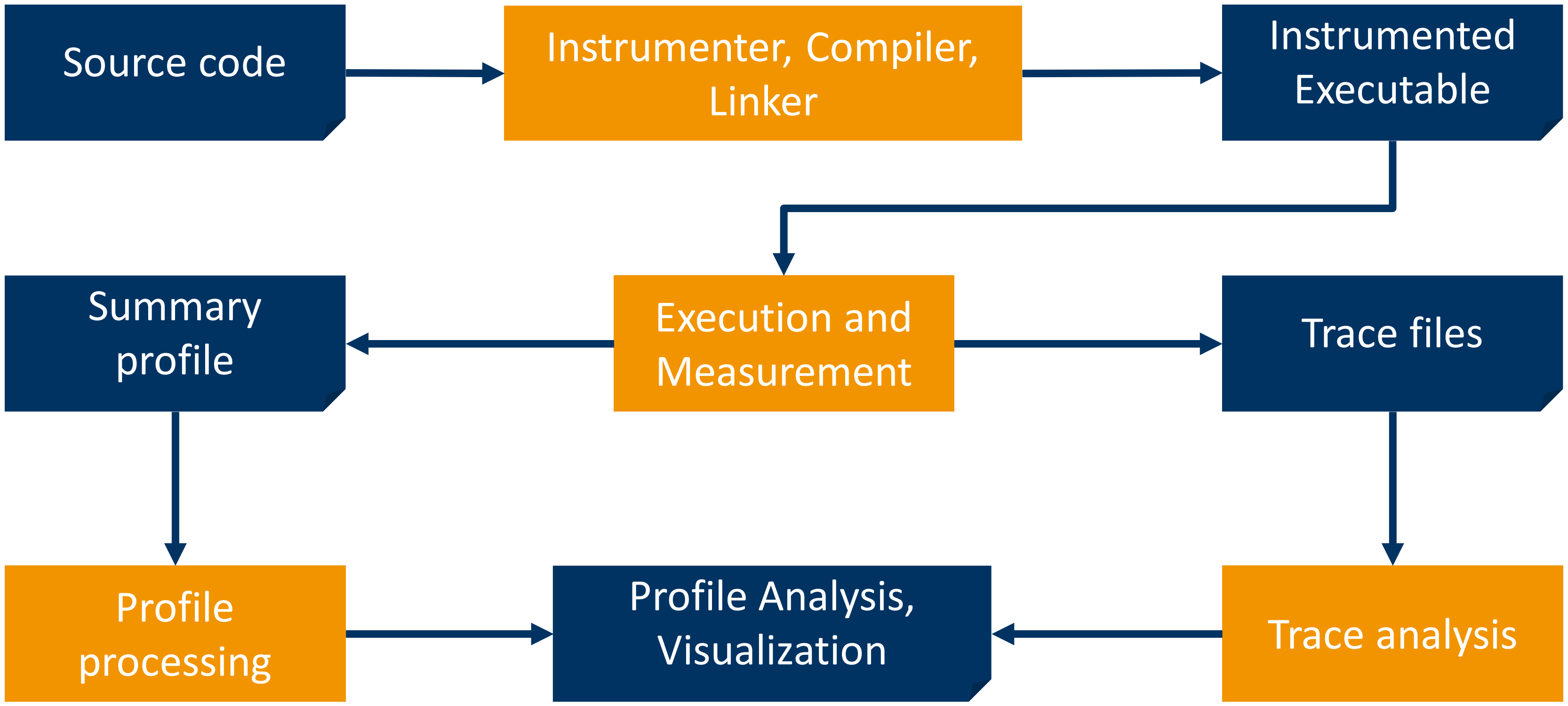
- developed by the Jülich Supercomputing Centre

- allows scalable analysis of
  - profile-based data (coarse-grained statistics of how much time spend where/how)
  - trace-based data (detailed statistics on MPI wait delays, etc.)

- comes with the CUBE reporting tool
  - can show individual metrics per region and per platform target
  - includes visualization of metrics in MPI topologies

Scalasca Suite: 3 Main Components

- Instrumentation
  - `scorep`
- Measurement
  - `scalasca`
- Analysis
  - `scalasca`
  - ultimately: `cube`



Selected Score-P Instrumentation Possibilities

- Compiler-based (default)
  - MPI calls
  - OpenMP primitives
  - all functions
- Manual
  - relies on the user marking regions of interest
- Source-to-source
  - using the Program Database Toolkit (PDT, experimental)
  - mainly for compatibility with TAU analysis suite

Workflow

1. Compile and run your application normally
    - to make sure it actually works and get a feel for the expected walltime
    - any subsequent errors are likely setup issues with the performance tool

2. Compile with `scorep` wrapper and run with `scalasca` to collect profile data

3. optionally analyze and re-run with (filtered) tracing

4. retrieve data and study in `cube` (or other tools)

Score-P and Scalasca Cheat Sheet

- `scorep mpicc ...`
  - instrument and compile application
- `scalasca -analyze mpiexec ...`
  - run and generate summary report in `score_p<app_name>_<num_ranks>_sum/`
- `scalasca -examine -s <report_dir>`
  - analyze and give feedback (buffer size, functions, …) in `<report_dir>/scorep.score`
- `cat <report_dir>/scorep.score`
  - check generated report, use for defining a filter, setting buffer size, any sanity checks, …
- define filter, test with `scorep-score -f myfilter <report_dir>/profile.cubex`
  - will show you updated estimates for buffer sizes and covered functions (with `-r`)
- `SCOREP_FILTERING_FILE=my.filter scalasca -analyze -q -t mpiexec …`
  - re-run with tracing, disabling profile

Further Useful Information

- `export SCOREP_TOTAL_MEMORY=256MB`
  - buffer size for tracing – if not set or too small, you'll receive warnings and skewed performance data due to buffer flushes
- `export SCAN_ANALYZE_OPTS="--time-correct"`
  - perform a time-correction pass in case of warnings about non-synchronized timers
- cube also supports plugins we did not look at
  - e.g. a basic timeline view

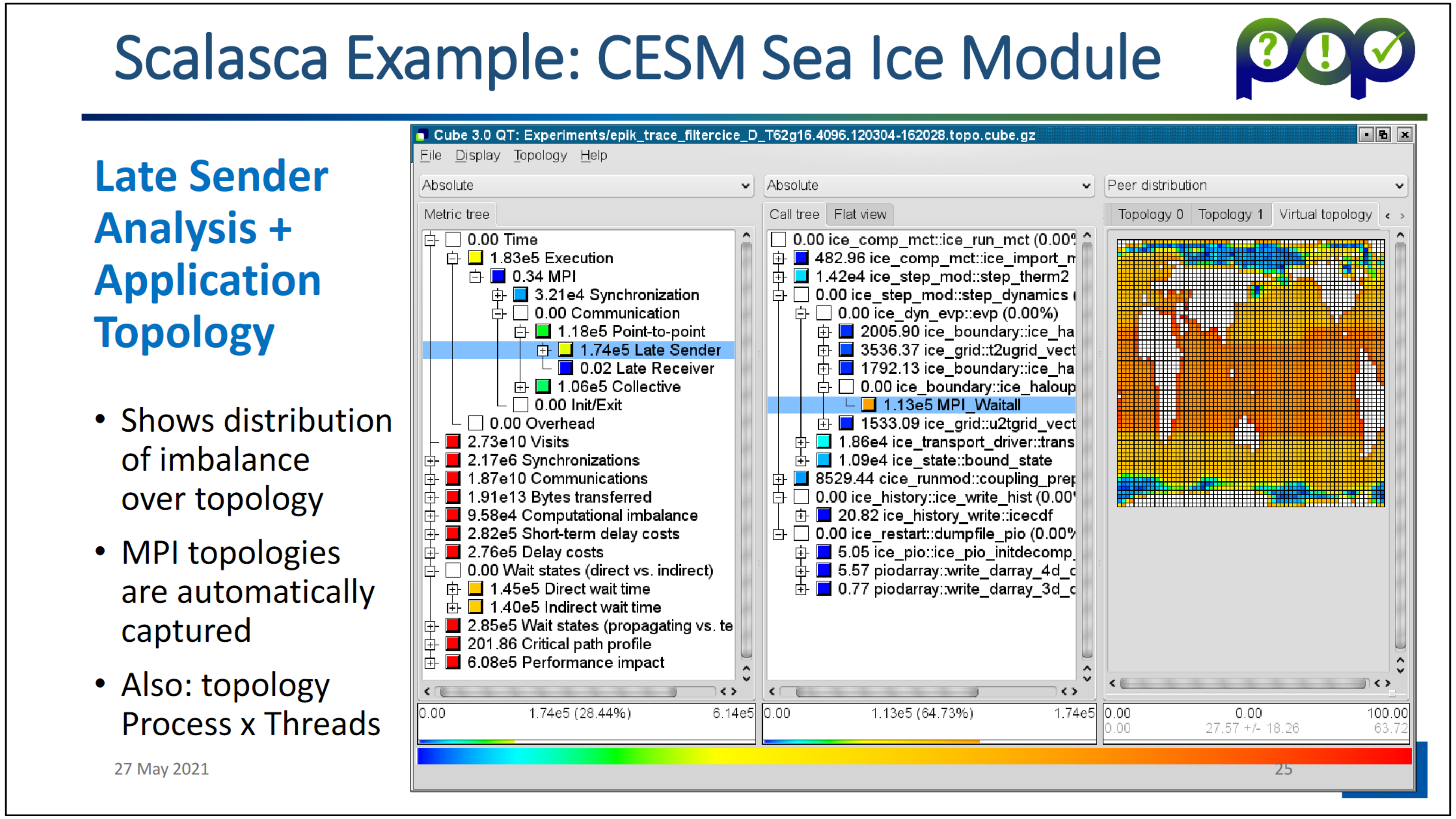
Example Output in `scorep.score`

```
Estimated aggregate size of event trace:                   1921GB
Estimated requirements for largest trace buffer (max_buf): 31GB
Estimated memory requirements (SCOREP_TOTAL_MEMORY):       31GB
(warning: The memory requirements cannot be satisfied by Score-P to avoid
 intermediate flushes when tracing. Set SCOREP_TOTAL_MEMORY=4G to get the
 maximum supported memory or reduce requirements using USR regions filters.)

flt     type     max_buf[B]          visits  time[s] time[%] time/visit[us]  region
         ALL 32,230,441,448 85,917,697,218 19186.46   100.0           0.22  ALL
         USR 32,212,254,864 85,899,346,178  9970.26    52.0           0.12  USR
         MPI     18,186,567     18,350,912  2222.72    11.6         121.12  MPI
      SCOREP             41             64     0.00     0.0          25.96  SCOREP
         COM             24             64  6993.48    36.5   109273165.13  COM

         USR 32,212,254,720 85,899,345,920  9970.25    52.0           0.12  computeTemperature
         MPI      7,127,127      4,587,584    17.63     0.1           3.84  MPI_Isend
         MPI      7,127,040      4,587,520     8.07     0.0           1.76  MPI_Irecv
         MPI      3,932,184      9,175,104  2167.78    11.3         236.27  MPI_Wait
         MPI          3,776             64     0.13     0.0        2088.58  MPI_Recv
                                  ...snip...
```

MPI Cartesian Topology Visualization



Source: Bernd Mohr, https://pop-coe.eu/sites/default/files/pop_files/pop-webinar-scalasca.pdf

Summary

- Score-P/Scalasca/Cube are very capable and efficient tools for performance analysis
  - do not provide the level of detail of e.g. Vampir timelines
  - but that can be a feature instead of limitation (consider very large applications and systems)

- Weblinks
  - Score-P: https://www.vi-hps.org/projects/score-p/
  - Scalasca & Cube: https://www.scalasca.org/
  - Scalasca Quick Reference http://apps.fz-juelich.de/scalasca/releases/scalasca/2.6/docs/QuickReference.pdf