HP-SEE

SCALASCA

www.hp-see.eu

Dusan Stankovic
Scientific Computing Laboratory
Institute of Physics Belgrade
sdule@ipb.ac.rs

HP-SEE

High-Performance Computing Infrastructure
for South East Europe's Research Communities

The HP-SEE initiative is co-funded by the European Commission under the FP7 Research Infrastructures contract

Introduction

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- SCALASCA - a tool that supports measuring and analyzing runtime behavior of parallel applications
  - interprocess communication
  - and synchronization
- Stands for Scalable performance Analysis of Large-Scale parallel Applications
- Developed with large scale systems in mind (typically over 1000 processes running in parallel)
- Traces processed in parallel, efficient implementation
- Open Source software, available under BSD licence
- Developed in Jülich Supercomputing Centre
- Available for a wide range of current HPC platforms

Optimization workflow

HP-SEE

High-Performance Computing Infrastructure for South East Europe's Research Communities

- Optimization work consists of multiple tasks, profiling and performance analysis is an important one

Overview

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Supports parallel programming paradigms & languages
  - MPI, OpenMP & hybrid OpenMP/MPI
  - Fortran, C, C++
- Integrated instrumentation, measurement & analysis toolset
  - Automatic and/or manual customizable instrumentation
  - Runtime summarization (aka profiling)
  - Automatic event trace analysis
  - Analysis report exploration & manipulation

Performance analysis techniques

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Profile analysis
  - Summary of aggregated metrics per function/callpath and/or per process/thread
- Time-line analysis
  - Visual representation of the space/time sequence of events
  - Requires an execution trace
- Pattern analysis
  - Search for event sequences characteristic of inefficiencies
  - Can be done manually, e.g., via visual time-line analysis

Installation

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Download source code with:

```
wget http://www2.fz-juelich.de/zam/datapool/scalasca/scalasca-1.4.1.tar.gz
```

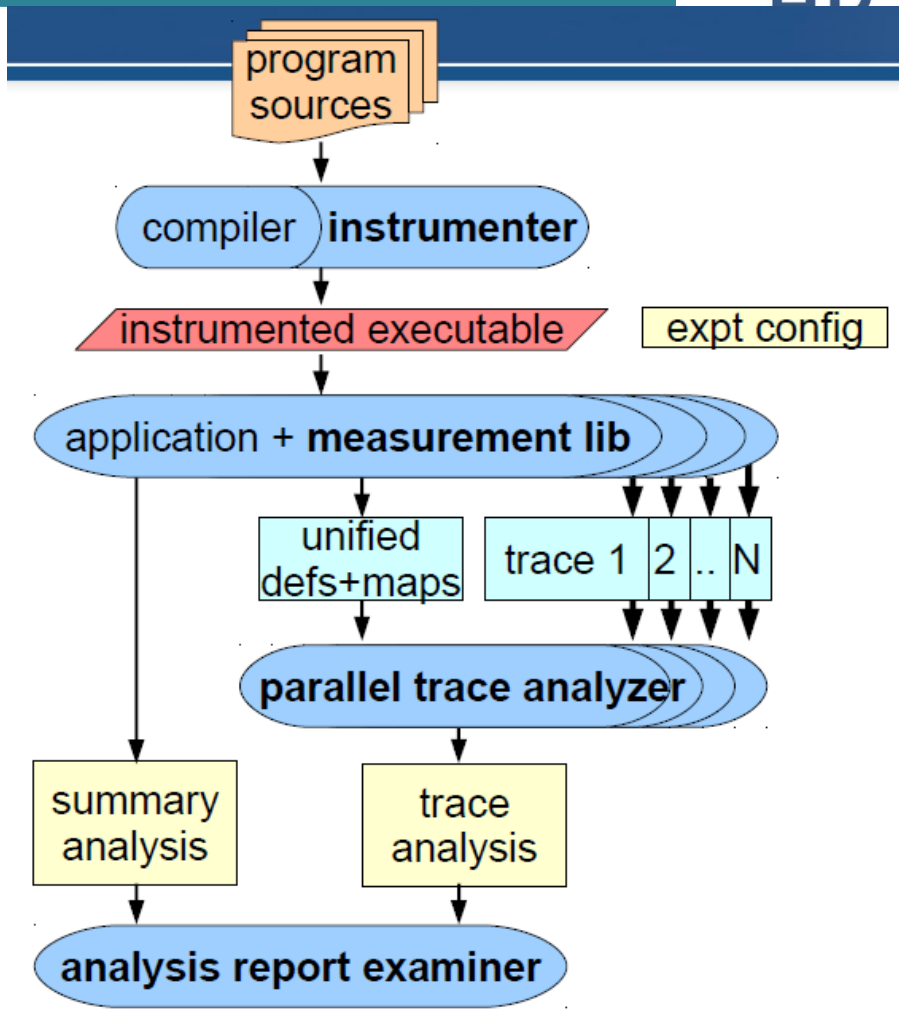
- Extract archive contents with:

```
tar xf scalasca-1.4.1.tar.gz ; cd scalasca-1.4.1
```

- Run configure, set location with `--prefix=`, compiler suite to use with `--compiler=` and graphical library to use when building GUI with `--with-qmake=`
- Build as usual with make ; make install
- Needs Qt graphical library in order to create GUI viewer

Generic parallel tools architecture

- Automatic/manual code instrumenter
- Measurement library for runtime summary & event tracing
- Parallel (and/or serial) event trace analysis when desired
- Analysis report examiner for interactive exploration of measured execution performance properties



SCALASCA toolset components

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Scalasca instrumenter – SKIN
  - prepare application objects and executable for measurement:

    ```
    scalasca -instrument <compile-or-link-command>
    ```

- Scalasca analyzer – SCAN
  - run application under control of measurement system:

    ```
    scalasca -analyze <application-launch-command>
    ```

- Scalasca examiner – SQUARE
  - post-process & explore measurement analysis report:

    ```
    scalasca -examine <experiment-archive|report>
    ```

Instrumentation

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Prepend commands which invoke the compiler with `scalasca -instrument`, for example:

  `scalasca -instrument` `mpicc -o test test.c`

- For larger applications, when using makefiles to build, useful technique is to have prefix (e.g. PREP) as a variable
  - Set `CC` (or `MPICC`) to `$PREP gcc` (or `$PREP mpicc`)
  - Initially set to an empty string, leave like that for production builds
  - Invoke `make` with `PREP="scalasca -instrument"` for instrumented builds

Measurement and analysis

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Prepend commands which run the application executable with `scalasca -analyze`, for example:

  `scalasca -analyze` `mpirun -np 2 test`

- To enable trace collection and analysis, add the `-t` flag
- Each application run with Scalasca analyzer enabled will produce new archive named as:
  - `epik_`***`executablename`***`_A(xB)_{sum|trace}`, where
  - `A` is the number of MPI processes
  - `B` is the (optional) number of OMP threads/process
  - `sum` is for summary, `trace` for trace files

Analysis report examination

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Analysis of created summary or trace files can be done interactively, or by a textual score output
- CUBE3 is graphical viewer bundled with Scalasca
- If CUBE3 is built, invoke with **`scalasca -examine epik_archive_name`**
- Otherwise, specify `-s` flag to just print textual score output, for example:

```
scalasca -examine -s epik_ctest-mpi_4_sum
```

- Textual output is saved in file `epik.score` in the archive directory

Textual analysis report example

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Example is from Scalasca example directory
- **scalasca -examine -s epik_ctest-mpi_4_sum**
- **cat epik_ctest-mpi_4_sum/epik.score**

```
flt  type        max_tbc          time        % region
      ANY           7974         11.76   100.00 (summary) ALL
      MPI           1926          5.12    43.54 (summary) MPI
      COM            168          3.00    25.53 (summary) COM
      USR           5832          3.41    28.99 (summary) USR

      USR           3024          3.41    28.99 do_work
      USR           2592          0.00     0.00 step
      MPI            750          0.00     0.00 MPI_Isend
      MPI            690          0.00     0.00 MPI_Irecv
      MPI            480          0.01     0.05 MPI_Barrier
      MPI            360          0.00     0.00 MPI_Wait
      USR            216          0.00     0.00 sequential
      MPI            120          0.00     0.00 MPI_Bcast
      ...
```

Filtering of specific routines [1/2]

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- For real-world applications, profiling overhead can be significant
  - Lots of routines, most of them not relevant to parallel portions of the code
  - Execution time of some routines is very short, but they are called millions of times
- These factors make instrumented application runs last much longer than ordinary runs (up to a few orders of magnitude)
- Solution – use filter to ignore specific routines while collecting profiling or tracing data
- In filter file - list routines by their names, wildcard characters are allowed

Filtering of specific routines [2/2]

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Create a filter file with names of some of the routines
- Invoke execution of the application with `scalasca -analyze -f filter_file mpirun -np 2 test`
- Examine generated data as usual
- Re-run previous example with `filter.scan` that contains routine names `do_work` and `sequential`
- Runtime analyzer will report filtering 2 of 96 functions
- Now you can see that the textual report is missing measurements for `do_work` and `sequential` routines

Filtered textual output

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- The same example from Scalasca example directory
- **scalasca -analyze -f filter.scan mpirun -np 4 ctest-mpi**
- **scalasca -examine -s epik_ctest-mpi_4_sum**
- **cat epik_ctest-mpi_4_sum/epik.score**

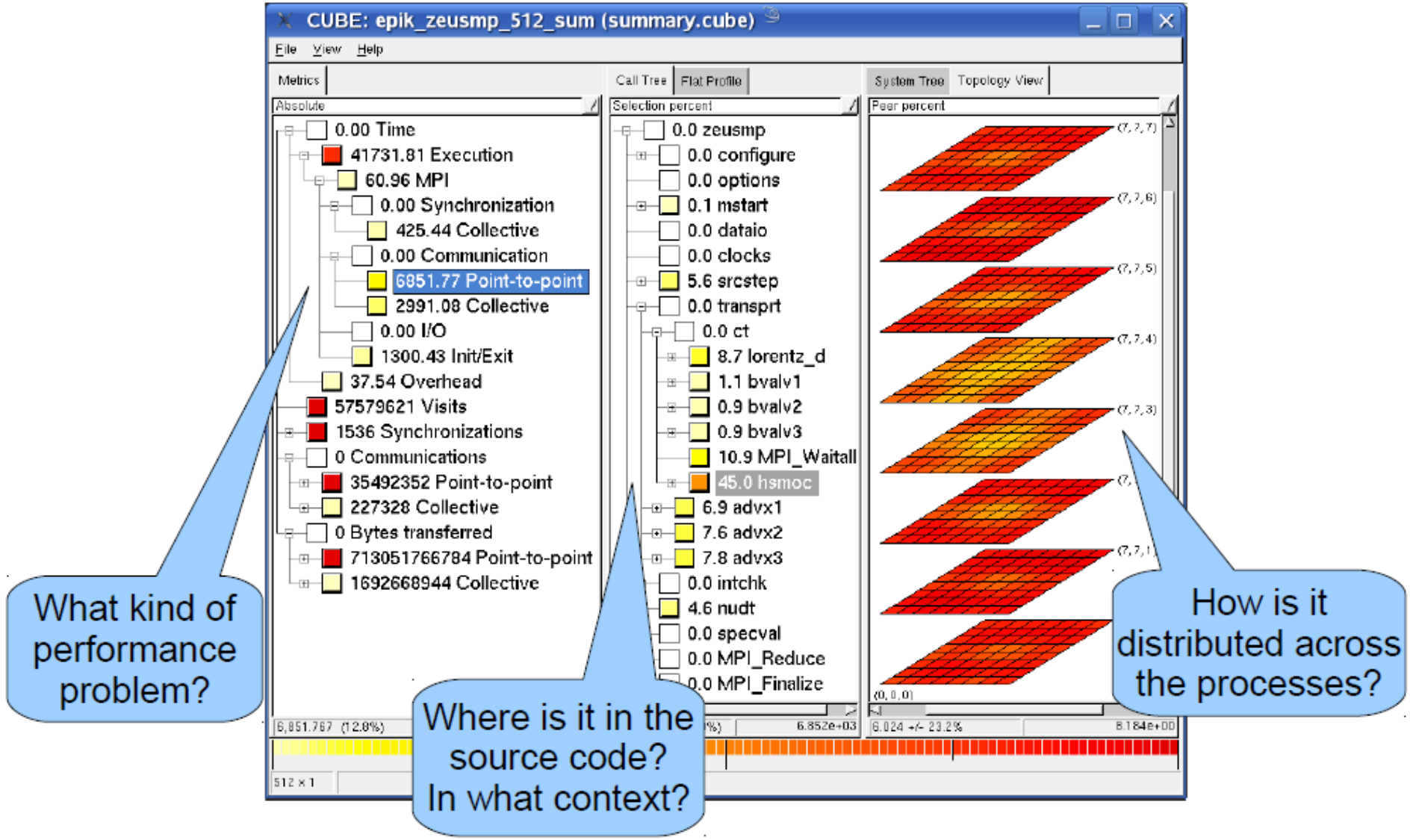
```
flt  type       max_tbc         time        %  region
      ANY          4734        11.62   100.00  (summary) ALL
      MPI          1926         5.11    44.00  (summary) MPI
      COM           168         3.49    30.05  (summary) COM
      USR          2592         2.92    25.13  (summary) USR

      USR          2592         2.92    25.13  step
      MPI           750         0.00     0.00  MPI_Isend
      MPI           690         0.00     0.00  MPI_Irecv
      MPI           480         0.00     0.01  MPI_Barrier
      MPI           360         0.00     0.00  MPI_Wait
      MPI           120         0.00     0.00  MPI_Bcast
      MPI           120         0.00     0.00  MPI_Allreduce
      COM            72         0.49     4.19  parallel
      ...
```

Analysis presentation and exploration with CUBE

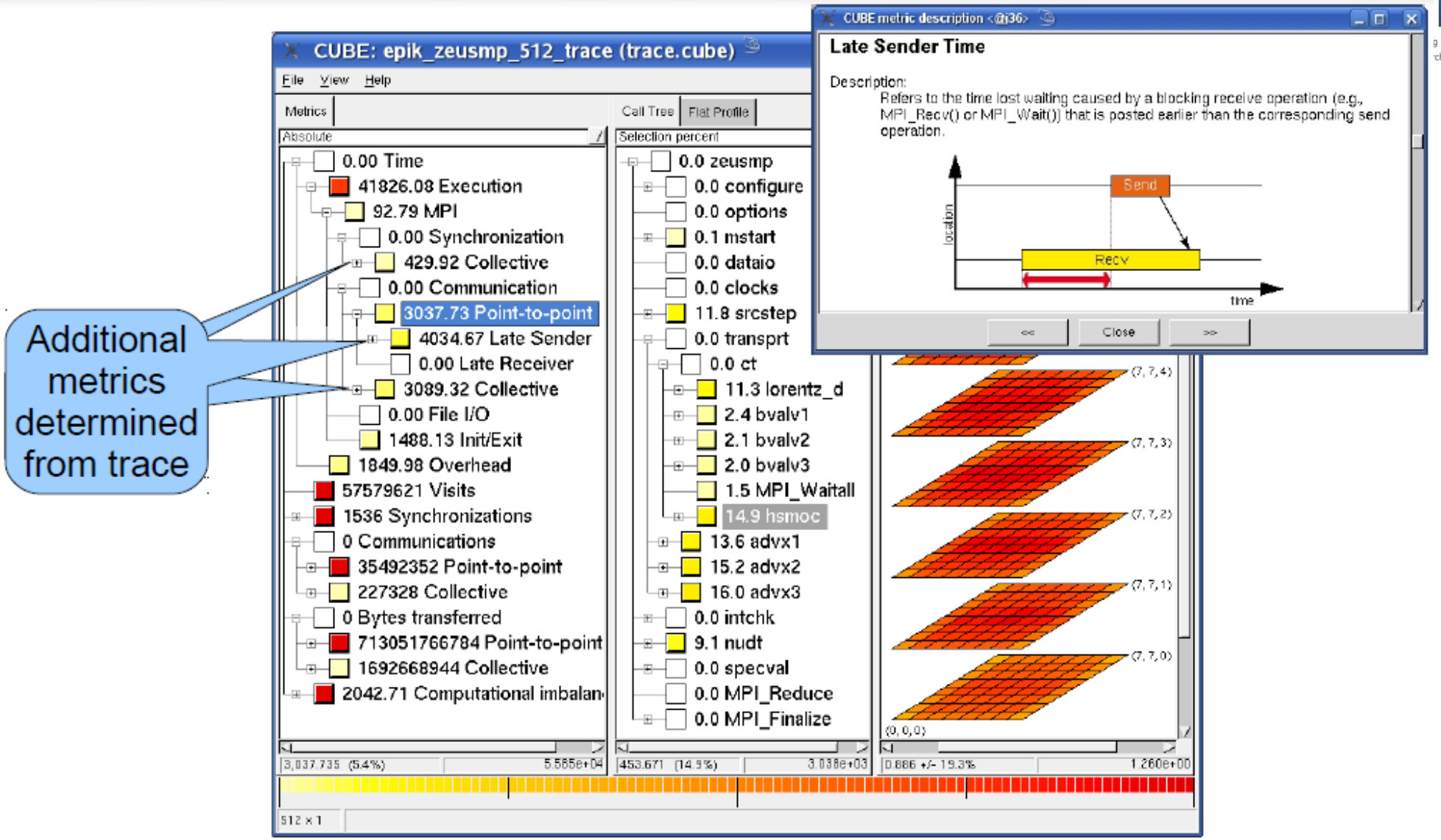
HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Representation of values on three hierarchical axes
  - Performance property (metric)
  - Call-tree path (program location)
  - System location (process/thread)
- Three coupled tree browsers represent different dimensions of the performance space (metric, program, system)
- CUBE displays severities
  - As value: for precise comparison
  - As colour: for easy identification of hotspots
  - Inclusive value when closed & exclusive value when expanded
  - Customizable via display mode

CUBE GUI overview [1/2]



CUBE GUI overview [2/2]



Known issues

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Automatic topology recording supported only for IBM BG and Cray XT systems
- Each toolset installation only supports one MPI implementation / compiler
- The same team of threads are expected to be used throughout execution
- OpenMP - If some parallel regions employ more threads (for example, by using `num_threads` or `set_num_threads`), those extra threads are ignored in the measurement
- Also, no support for MPI_THREAD_MULTIPLE

Conclusion

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- Powerful tool for analyzing and measuring parallel applications supporting most widely used technologes – MPI and OpenMP
- Great scalability achieved by using parallel trace analysis on the same CPUs the application was executed on
- Interoperability with other performance analysis software, examples:
  - TAU instrumentation with Scalasca measurement libraries
  - Trace conversion utilities for VT/OTF, Paraver, JumpShot
- Vampir visualization of Scalasca traces (without conversion)
- Alternative presentation with TAU Paraprof/PerfExplorer

References

HP-SEE
High-Performance Computing Infrastructure for South East Europe's Research Communities

- http://www.scalasca.org
- User guide: http://www2.fz-juelich.de/jsc/datapool/scalasca/UserGuide.pdf
- CUBE User guide (Qt version):

  http://www2.fz-juelich.de/jsc/datapool/scalasca/CubeGuide.pdf
- Images on previous slides taken from VI-HPS “Scalasca Overview” presentation by B. Wylie and M. Geimer

  www.vi-hps.org/datapool/vihpstw9/Scalasca_Overview.pdf