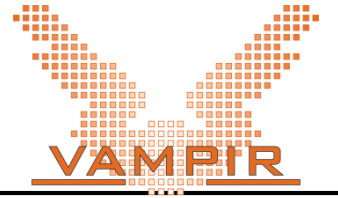
International HPC Summer School 2019: Performance analysis and optimization

Vampir, TAU, Extra-P, Darshan

VI-HPS Team
Ilya Zhukov – Jülich Supercomputing Centre

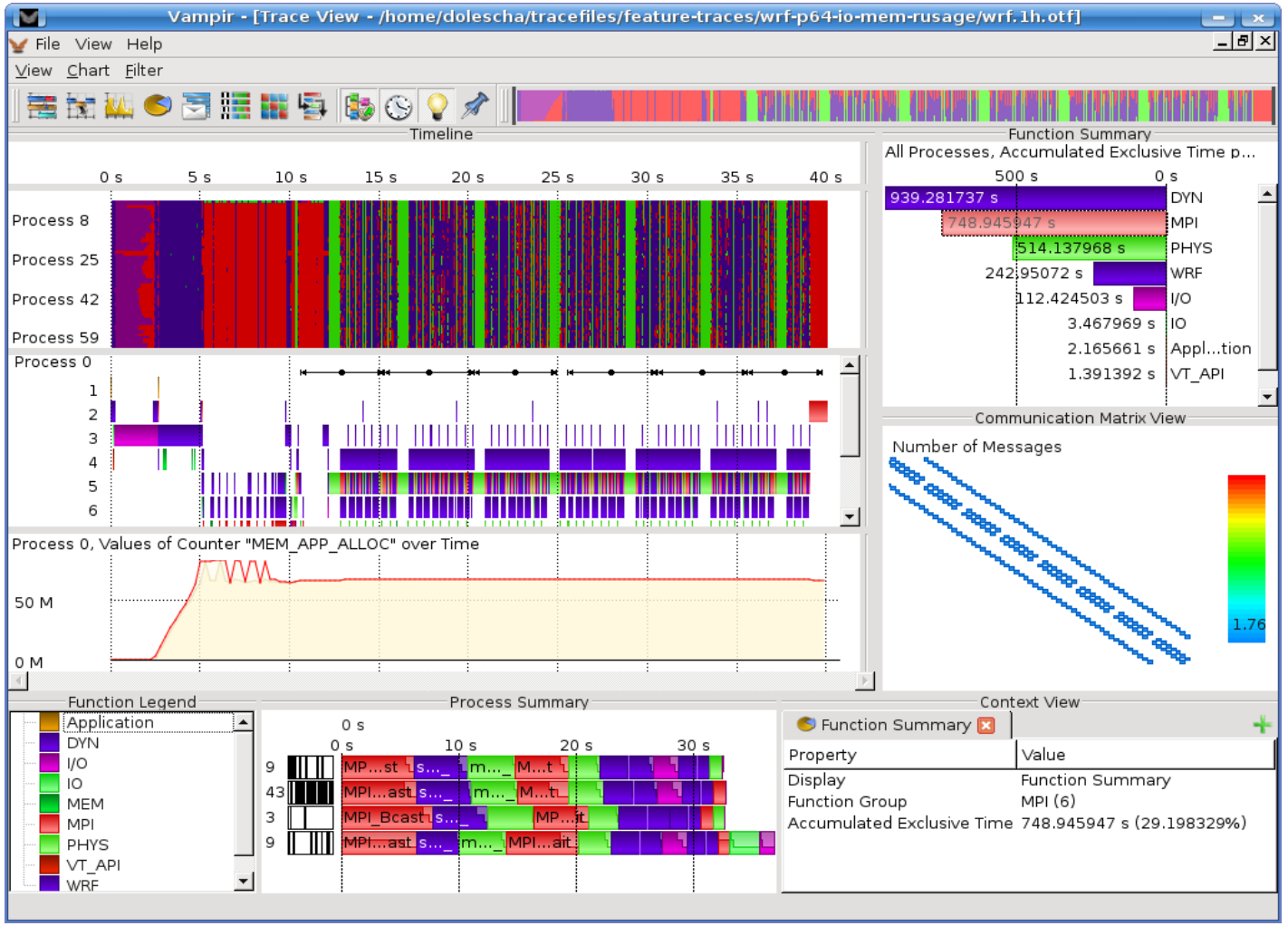
Vampir Event Trace Visualizer



- Offline trace visualization for Score-P's OTF2 trace files
- Visualization of MPI, OpenMP and application events:
  - All diagrams highly customizable (through context menus)
  - Large variety of displays for ANY part of the trace
- http://www.vampir.eu

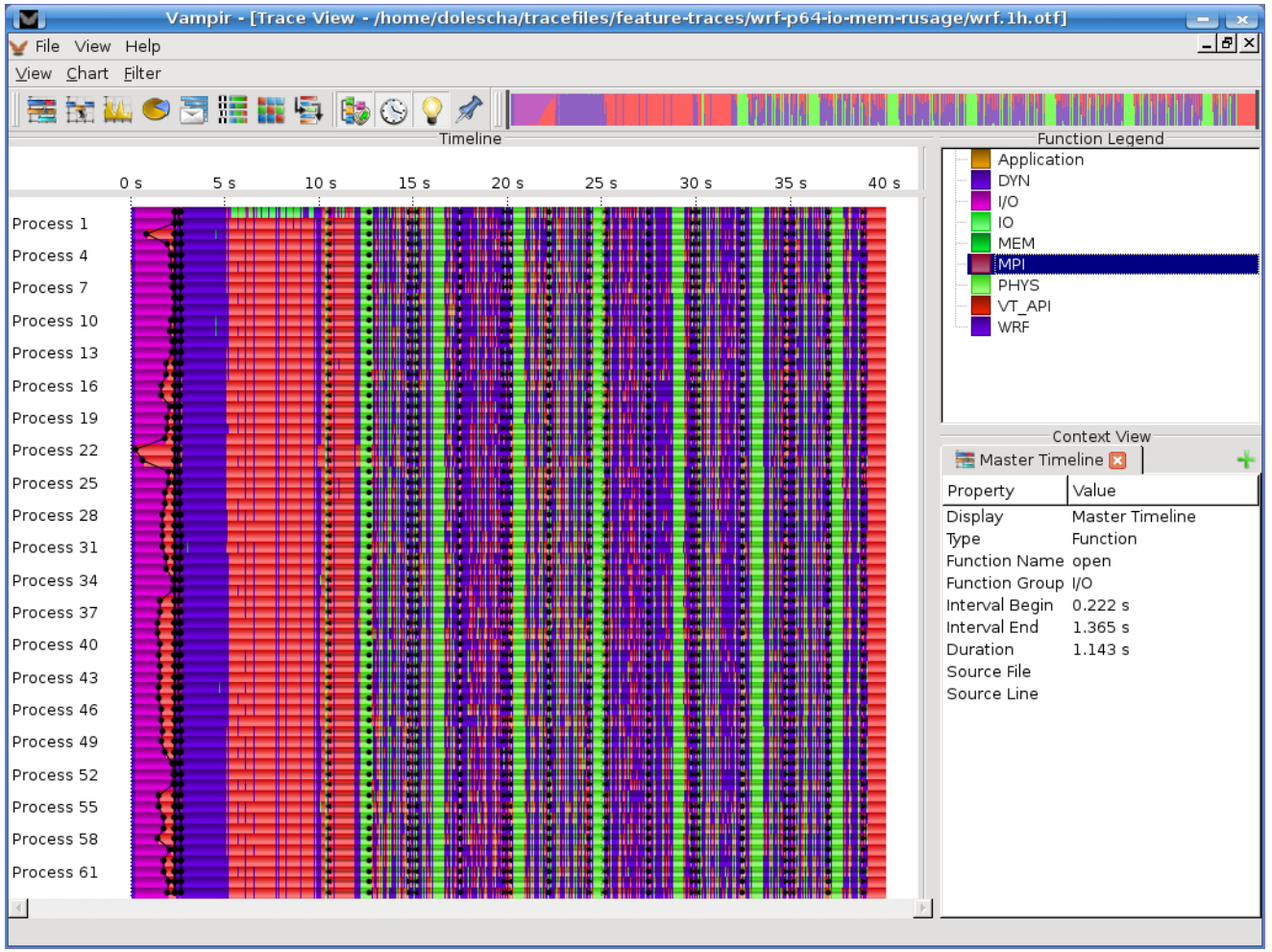
- Advantage:
  - Detailed view of dynamic application behavior
- Disadvantage:
  - Requires event traces (huge amount of data)
  - Completely manual analysis

Vampir Displays



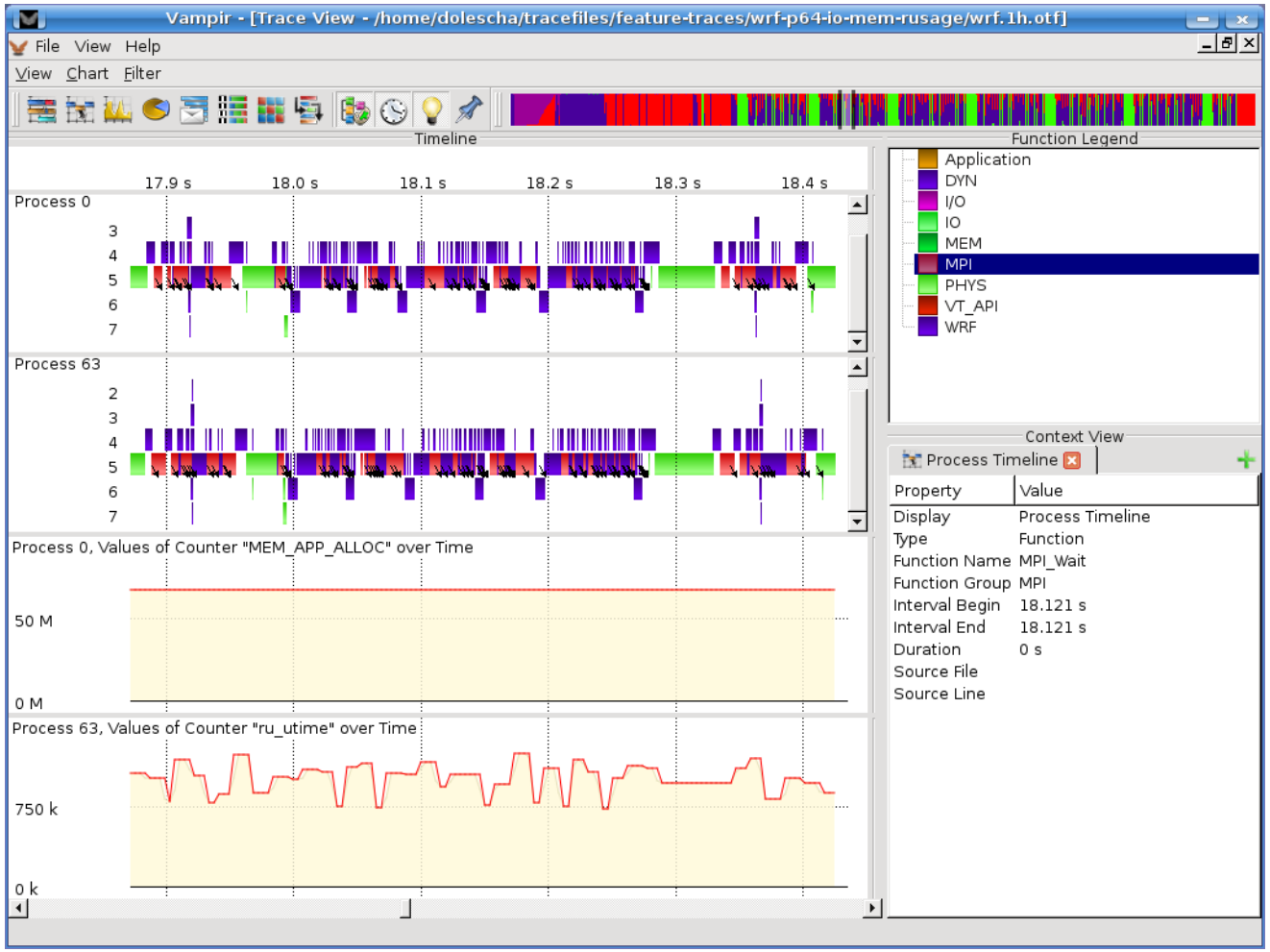
Vampir: Timeline Diagram

- Functions organized into groups
- Coloring by group
- Message lines can be colored by tag or size
- Information about states, messages, collective and I/O operations available through clicking on the representation



Vampir: Process and Counter Timelines

- Process timeline show call stack nesting
- Counter timelines for hardware and software counters

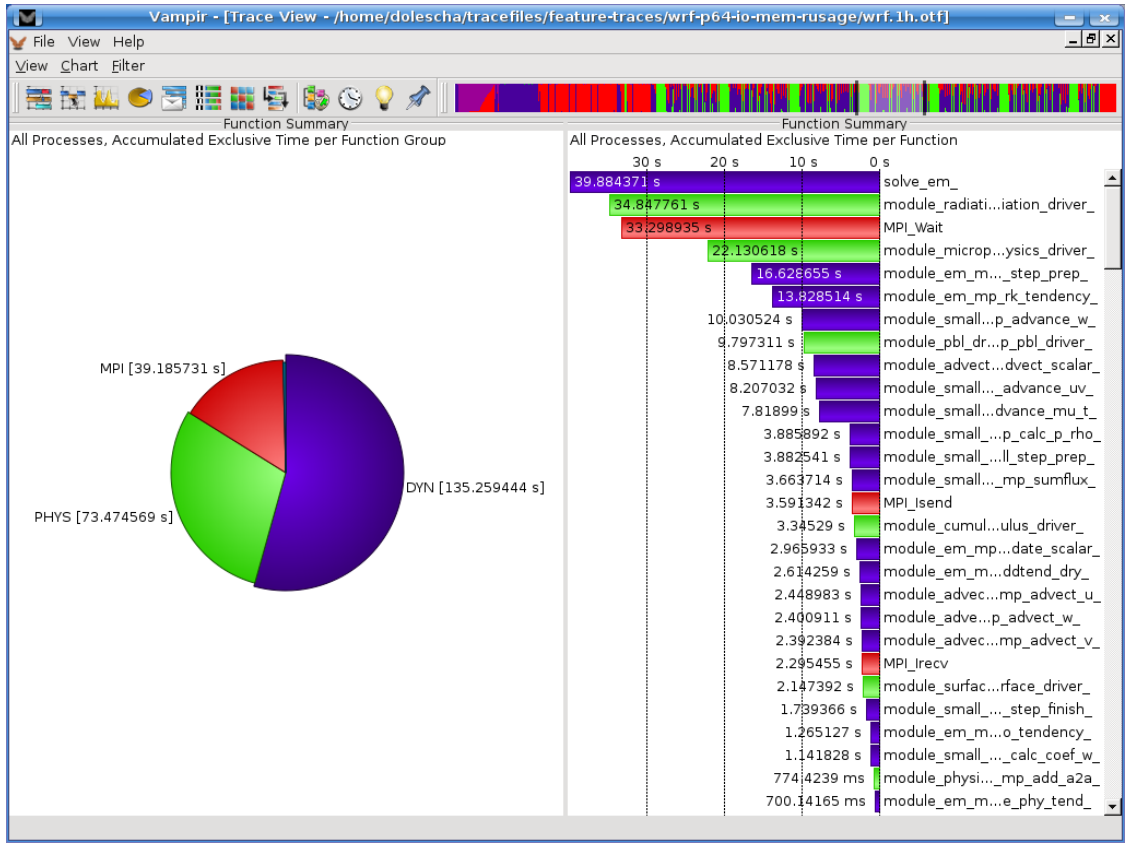


Vampir: Execution Statistics

- Aggregated profiling information:
  execution time,
  number of calls,
  inclusive/exclusive

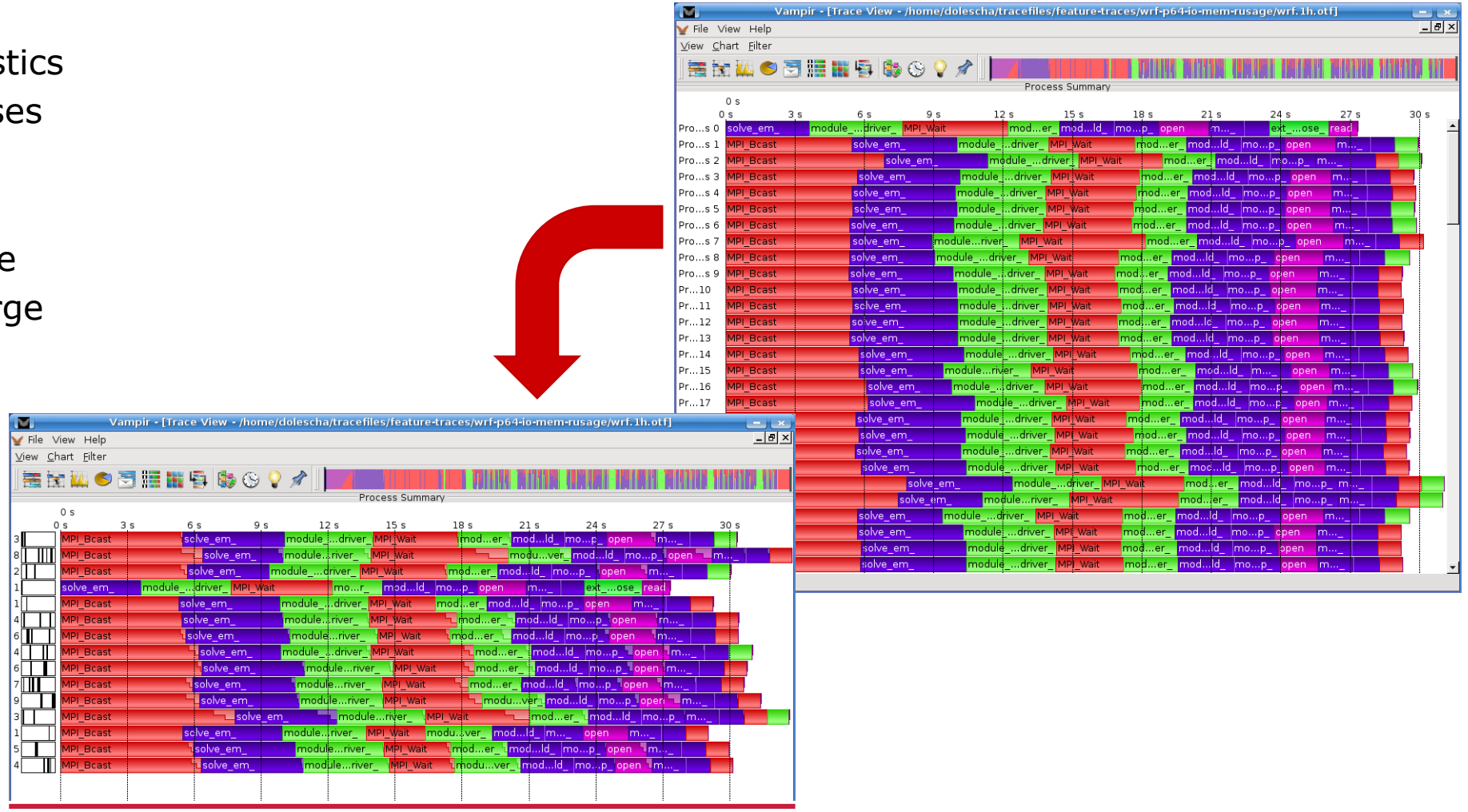
- Available for all / any group (activity) or all routines (symbols)

- Available for any part of the trace
  ⇨ selectable through time line diagram



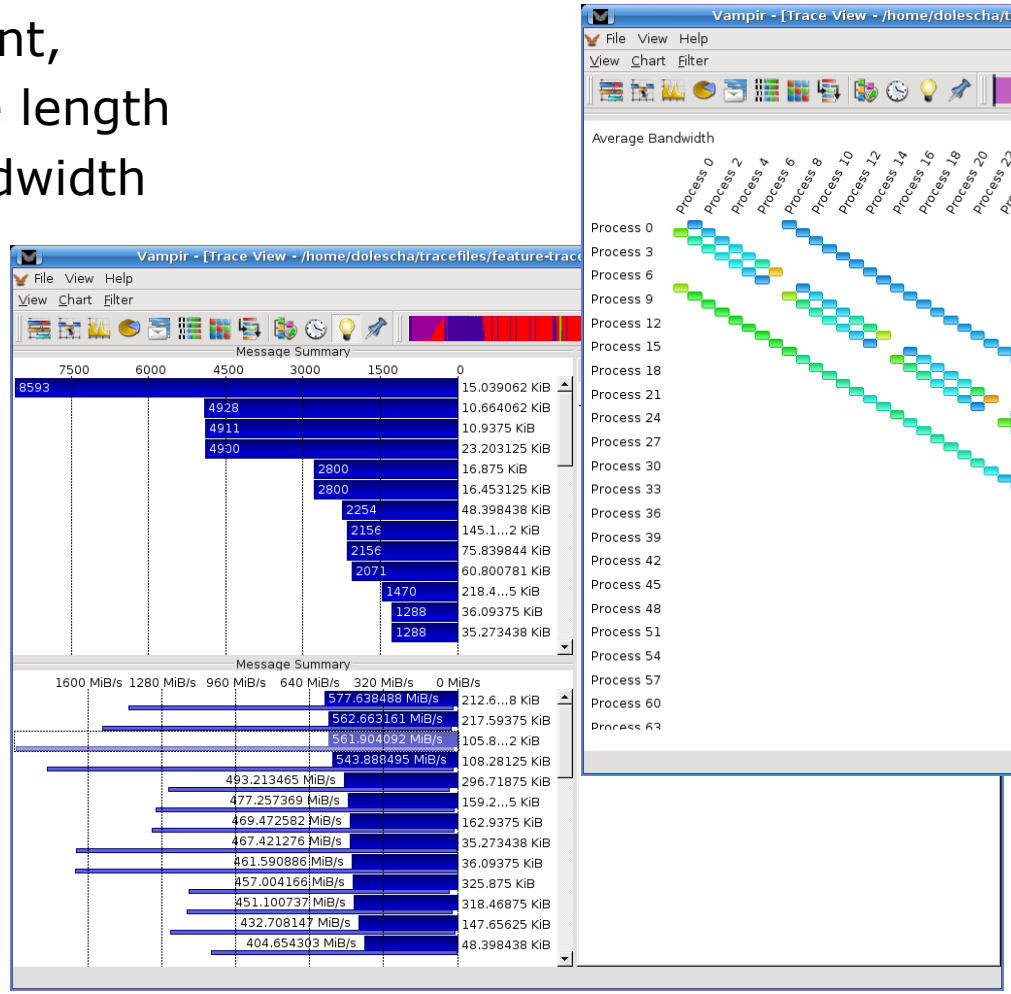
Vampir: Process Summary

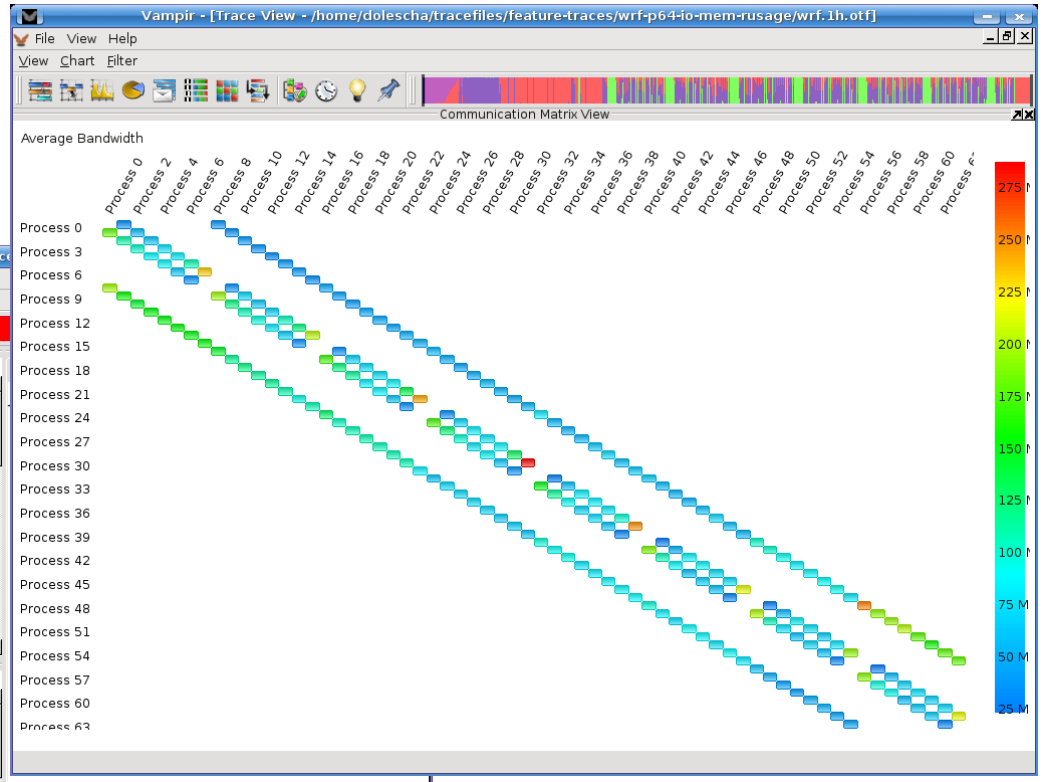
- Execution statistics over all processes for comparison
- Clustering mode available for large process counts



Vampir: Communication Statistics

- Byte and message count, min/max/avg message length and min/max/avg bandwidth for each process pair
- Message length statistics

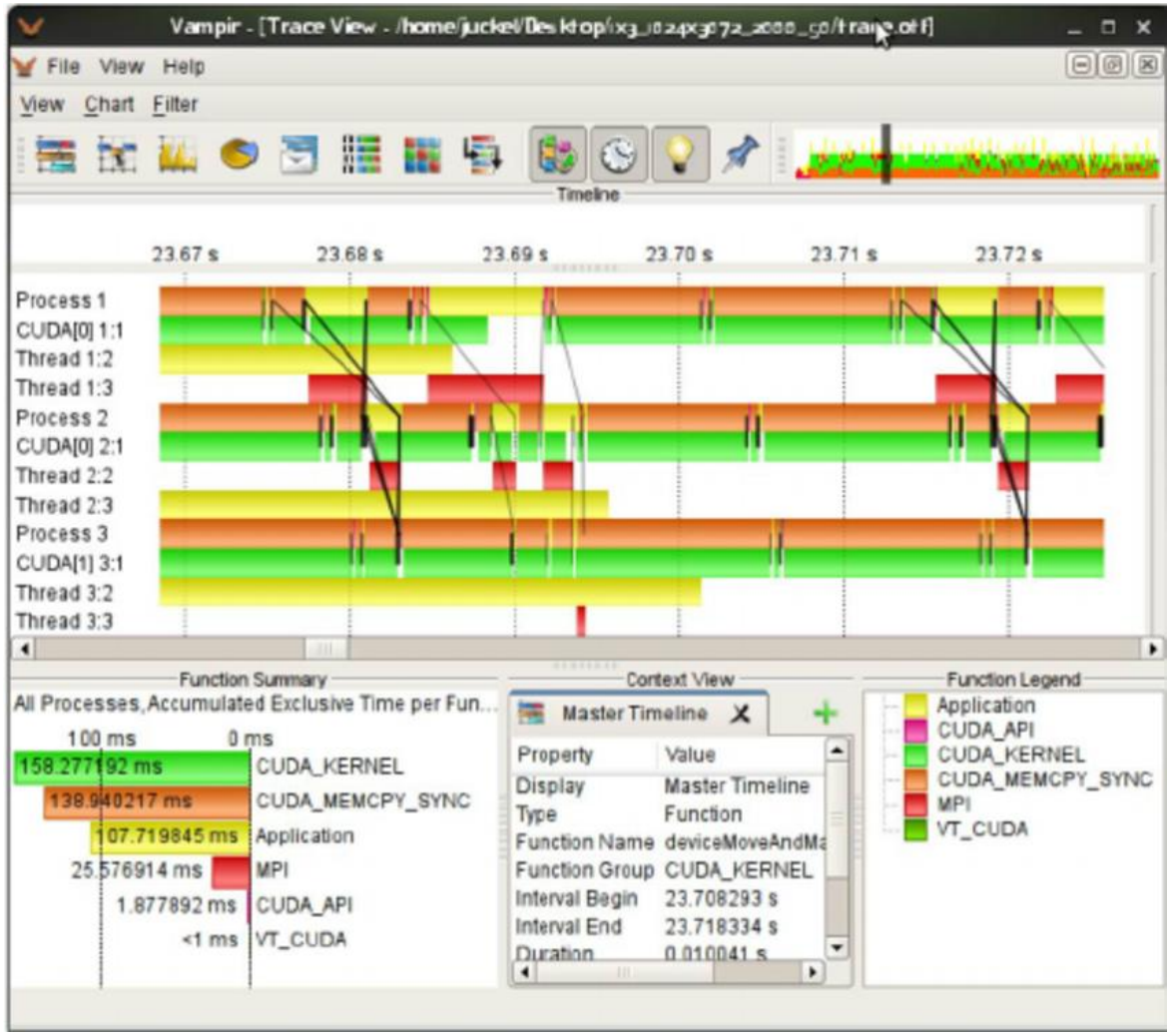




- Available for any part of the trace

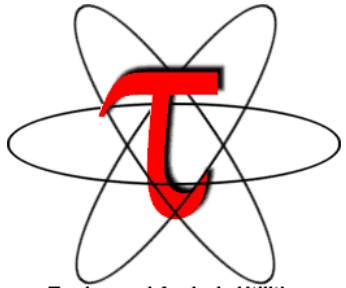
Vampir: CUDA Example

- Detailed information on kernel execution and memory transfers
- All statistics and displays also available for CUDA events



TAU

- Very portable tool set for instrumentation, measurement and analysis of parallel multi-threaded applications
- http://tau.uoregon.edu/

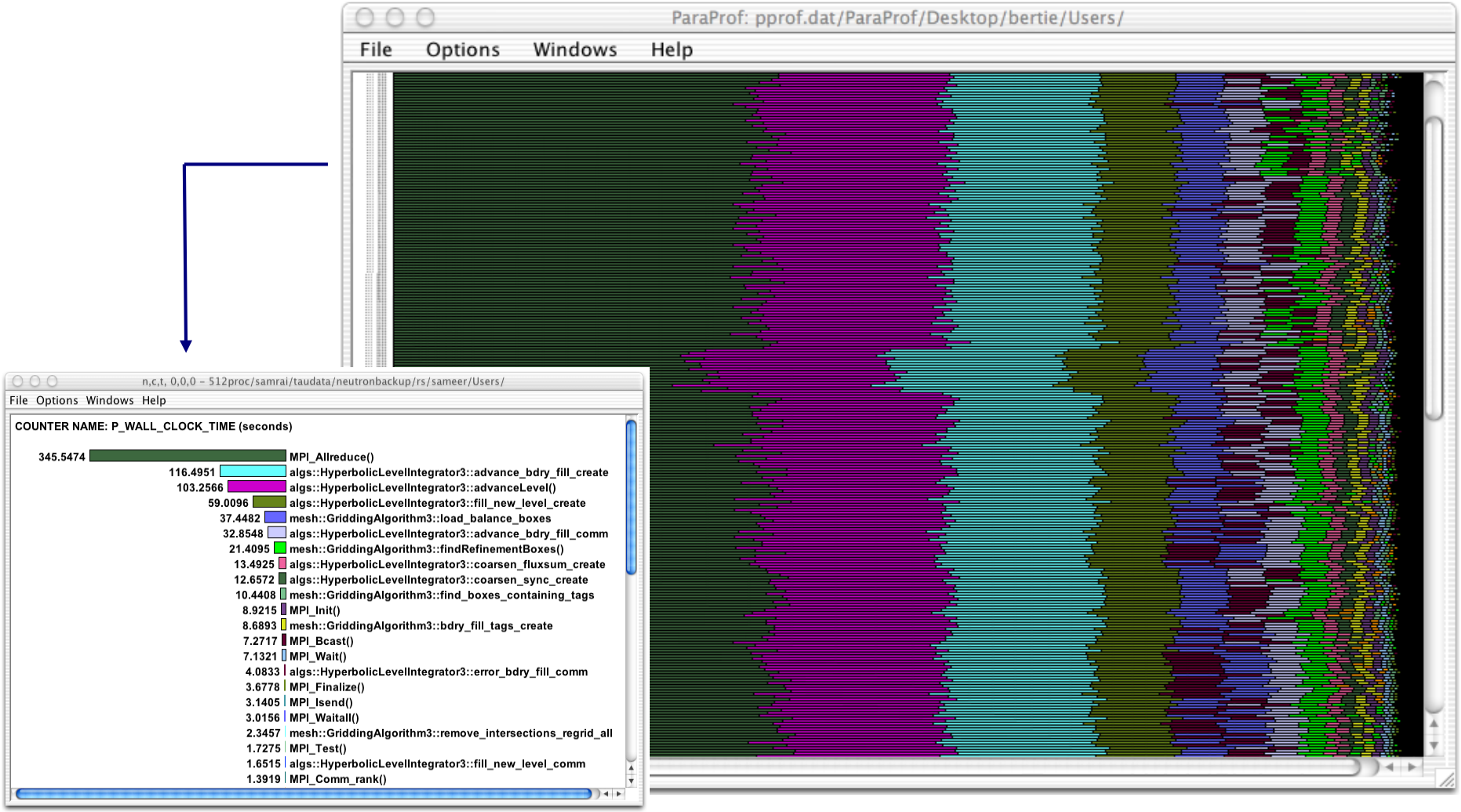


- Supports
  - Various profiling modes and tracing
  - Various forms of code instrumentation
  - C, C++, Fortran, Java, Python
  - MPI, multi-threading (OpenMP, Pthreads, …)
  - Accelerators

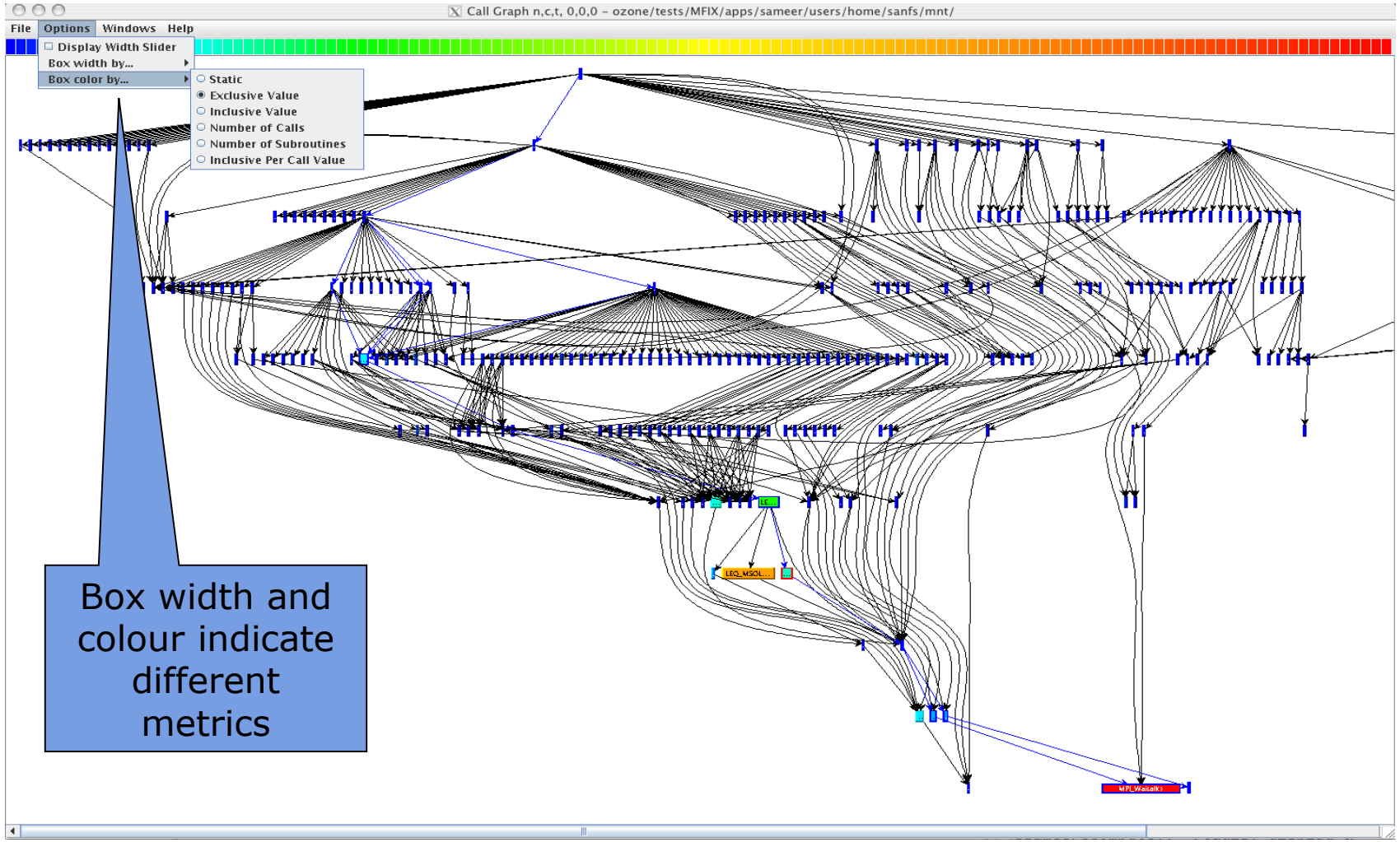
TAU: Instrumentation

- Flexible instrumentation mechanisms at multiple levels
  - Source code
    - manual
    - automatic
      - C, C++, F77/90/95 (Program Database Toolkit (PDT))
      - OpenMP (directive rewriting with Opari)
  - Object code
    - pre-instrumented libraries (e.g., MPI using PMPI)
    - statically-linked and dynamically-loaded (e.g., Python)
  - Executable code
    - dynamic instrumentation (pre-execution) (DynInst)
    - virtual machine instrumentation (e.g., Java using JVMPI)
- Support for performance mapping
- Support for object-oriented and generic programming

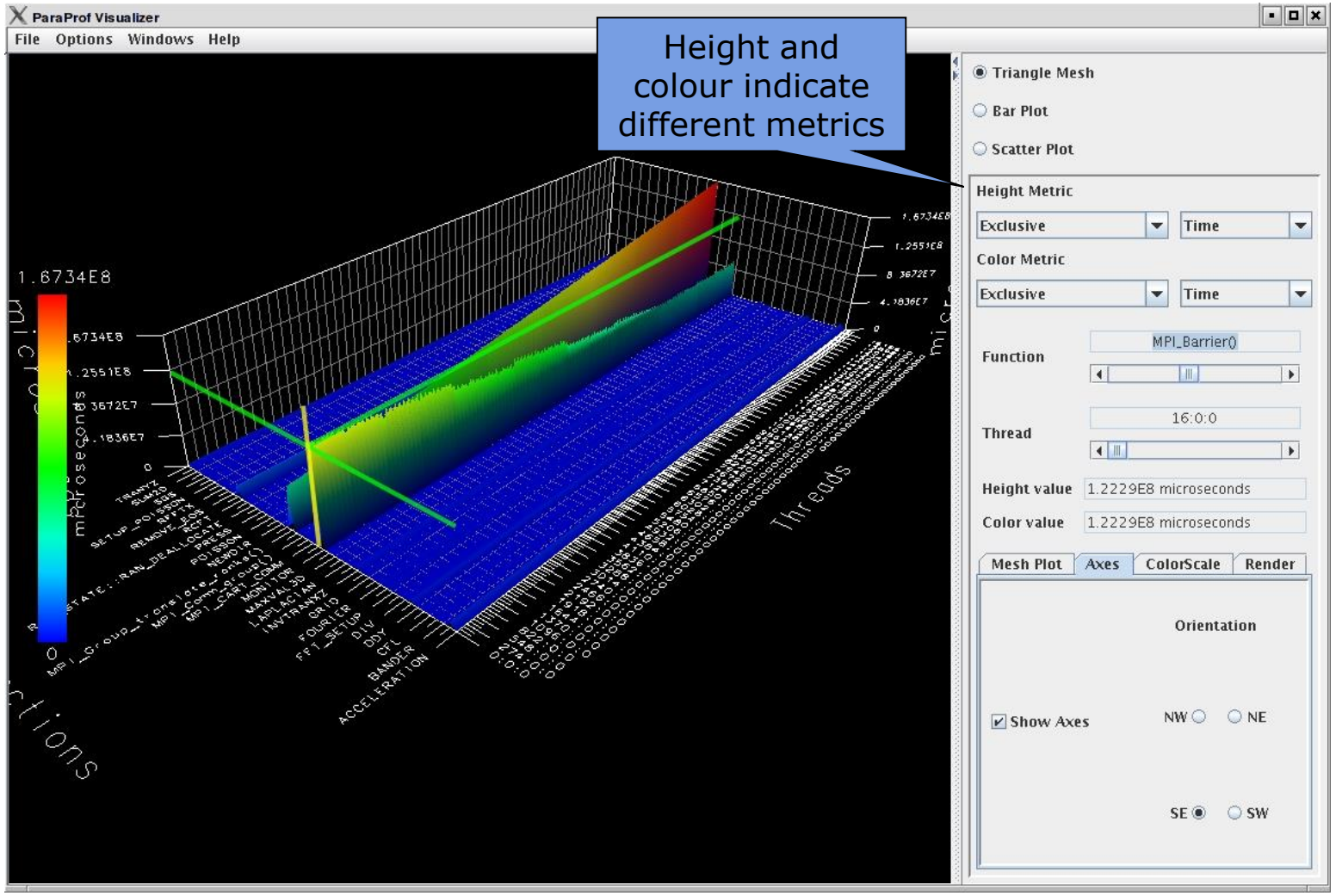
TAU: Basic Profile View



TAU: Callgraph Profile View



TAU: 3D Profile View



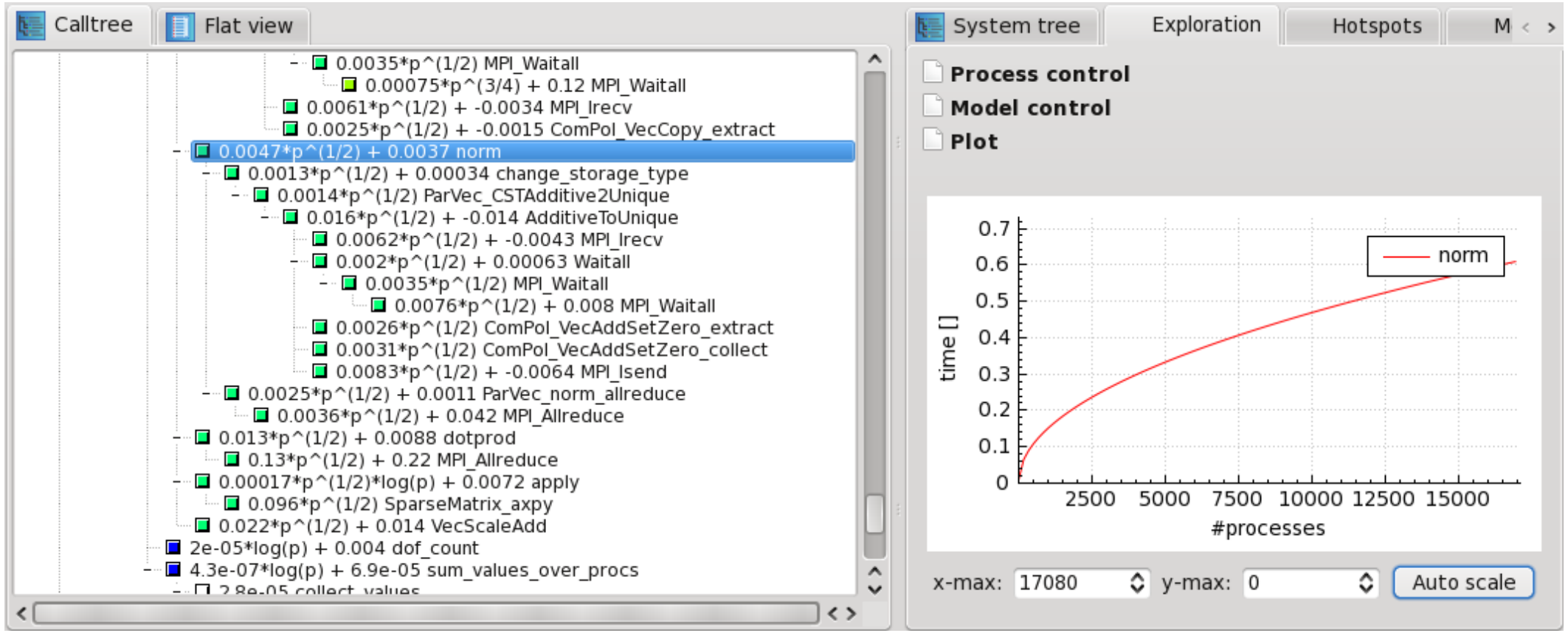
Extra-P

- **Goal**: identification of parts of the program which scaling behavior is unintentionally poor (much worse than expected) by means of automatic performance-modeling

- Supports Linux (x86x86_64/IA64/PPC/Power), Mac OS X (x86_64)
- Accepts input files in the Cube format and processes them into a condensed Cube format containing functions for each metric and call path

- http://www.scalasca.org
- Open Source: BSD 3-Clause License

Interactive exploration of performance models in Extra-P



Darshan

- I/O characterization tool logging parallel application file access
- Summary report provides quick overview of performance issues
- Works on unmodified, optimized executables
- Shows counts of file access operations, times for key operations, histograms of accesses, etc.

- Supports POSIX, MPI-IO, HDF5, PnetCDF, …
  - Doesn't support mpif90 on BlueGene systems (use mpif77)
- Binary log file written at exit post-processed into PDF report

- http://www.mcs.anl.gov/research/projects/darshan/
- Open Source: installed on many HPC systems

Example Darshan report extract

jobid:	uid:	nprocs: 4096	runtime: 175 seconds