Scalable Performance Analysis of Large-Scale Applications

Felix Wolf
Forschungszentrum Jülich

July 16th 2007

Outline

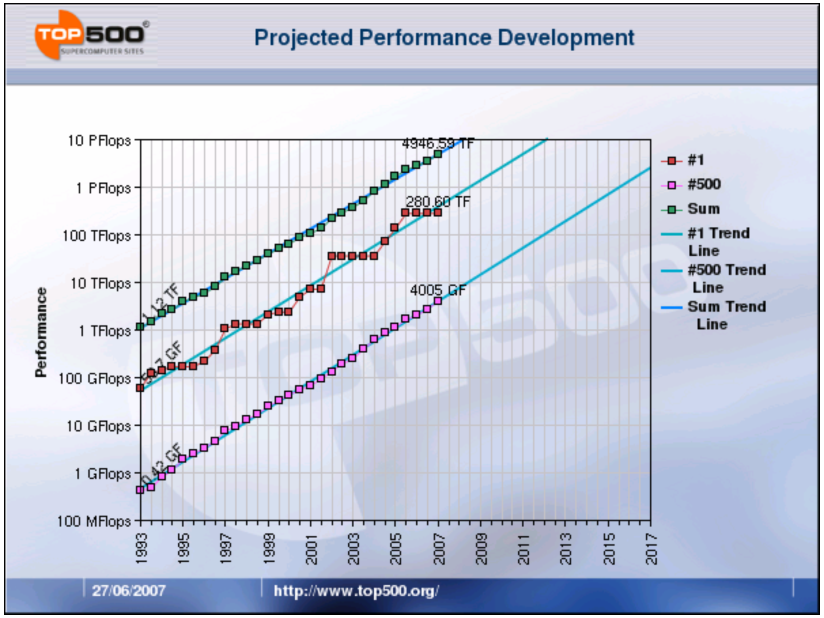
- Motivation
- Pattern search in event traces
- Parallel approach
- Experimental results
- Data formats, interfaces, and components
- Ongoing tool integration efforts

People involved

- Research Group
  Performance Analysis of Parallel Programs
  - Focus: scalability, grids, Cell, SCALASCA
  - Teaching at RWTH Aachen University
  - Felix Wolf, Ralph Altenfeld, Daniel Becker, Markus Geimer, Björn Kuhlmann, Matthias Pfeiffer, Brian Wylie, Liang Yang
- Research Group
  Performance Optimization and Programming Environments
  - Focus: multithreading, OpenMP, KOJAK
  - Bernd Mohr, Sebastian Flott, Christoph Geile, Marc-André Hermanns

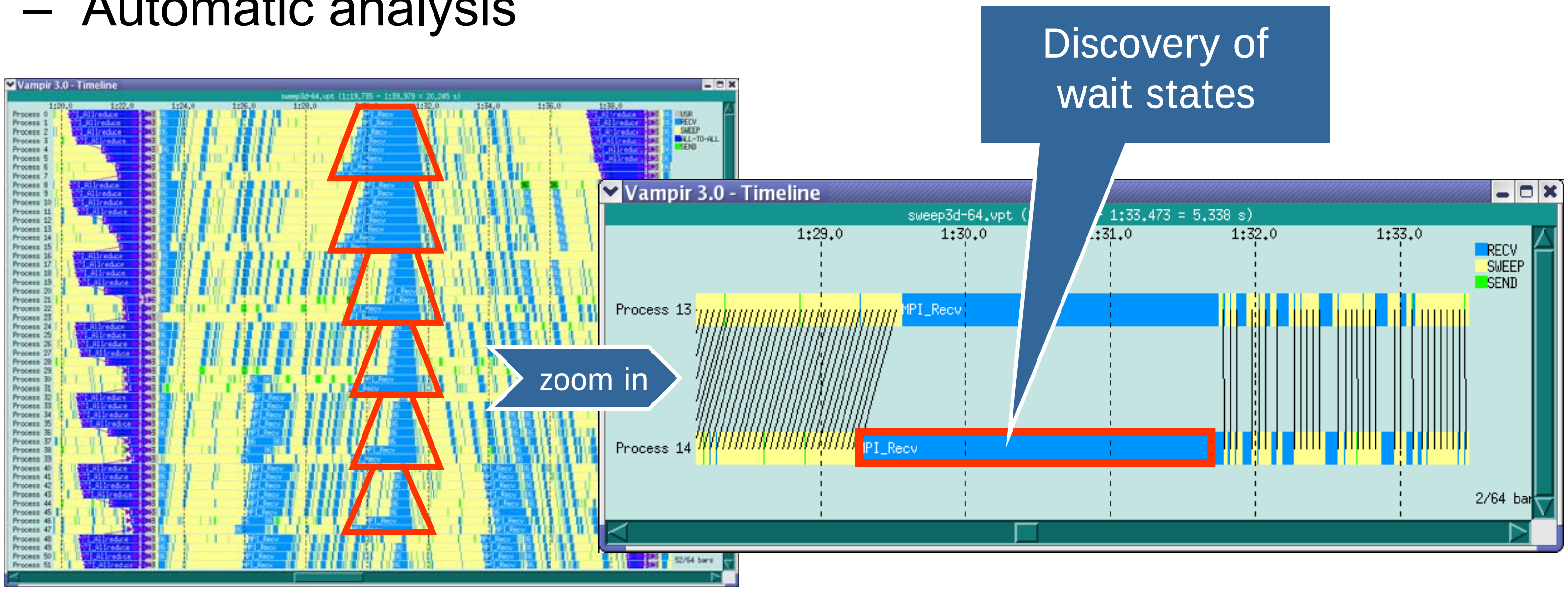
Increasing parallelism

- Advanced numerical simulations harness higher degrees of parallelism
  - Multiple cores instead of higher clock speeds
    - Tens of thousands of processors
  - Higher application complexity
    - Multi-physics & multi-scale
- Scalability hard to achieve
  - Overhead for managing concurrency at large scale
  - Hierarchy of latencies and bandwidths
  - Load balance
  - Access to shared resources



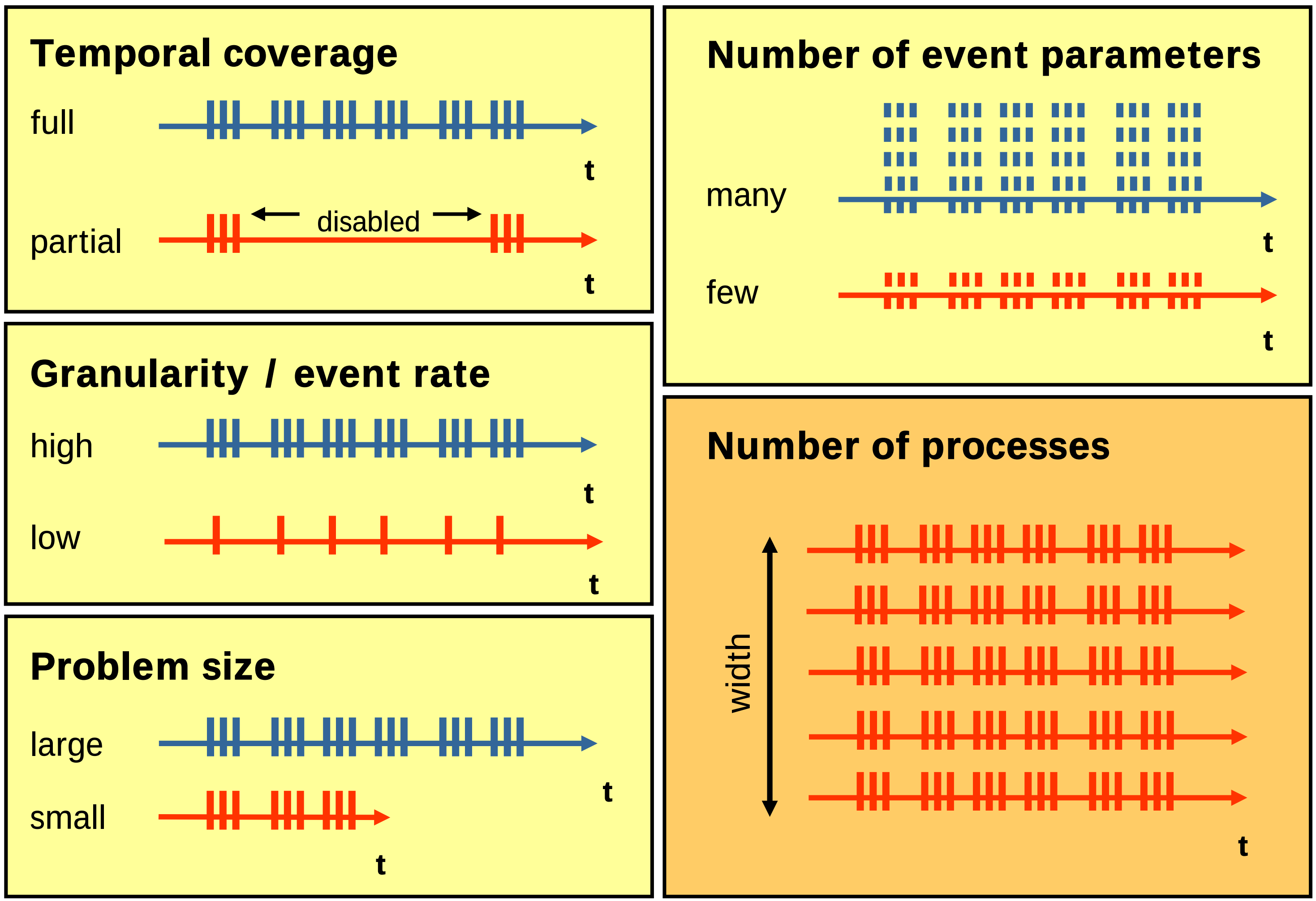
Why tracing?

- Traces preserve temporal and spatial relationships among individual runtime events
  - Suitable to study interactions among different processes
  - Allows identification of wait states
- Two approaches
  - Visual analysis
  - Automatic analysis



Why not tracing?

- Trace size is impediment to scalability

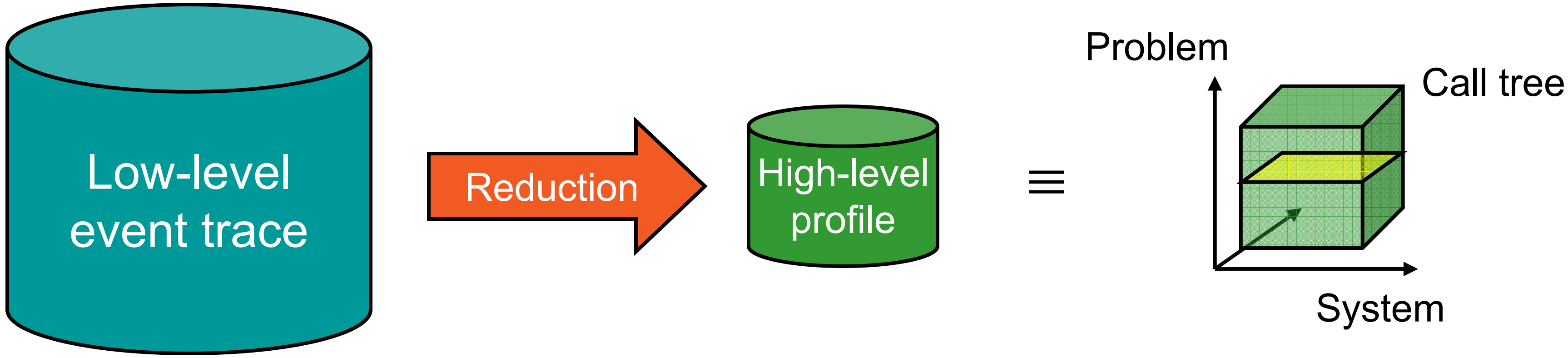


Scalability of tracing

- Excessive storage requirements
  - Especially during trace generation and in-memory analysis
- Intrusion when flushing event data to disk at runtime
- Costly file I/O when merging many potentially large process-local trace files
- Excessive processing times during analysis
- Failure, extended response times, and size of graphical displays
- Unexpected problems
  - Building robust and scalable tools is hard…

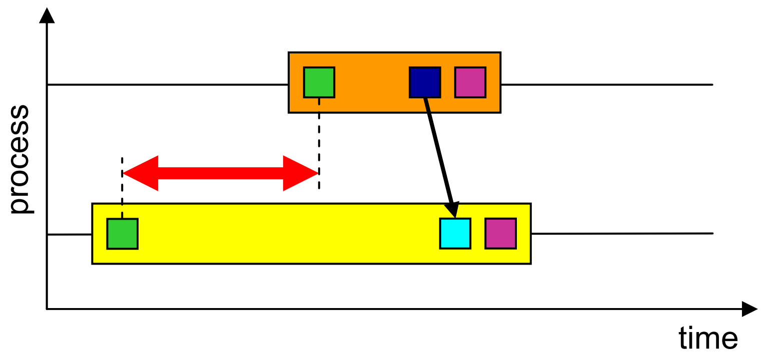
Automatic off-line trace analysis

- Idea:
  - Automatic search for *patterns* of inefficient behavior
  - Classification of behavior
  - Quantification of significance

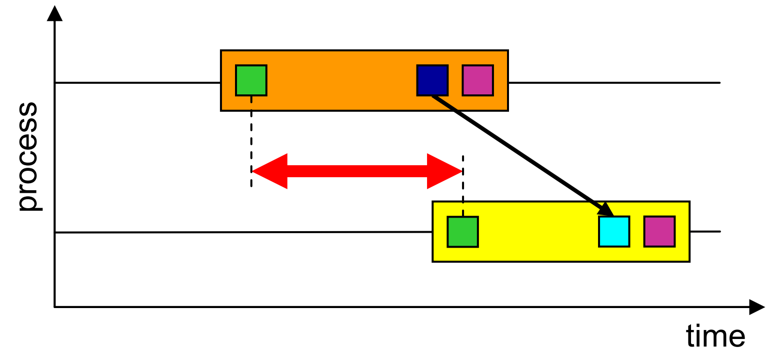


- Quicker than manual (visual) analysis
- Guaranteed to cover the entire event trace

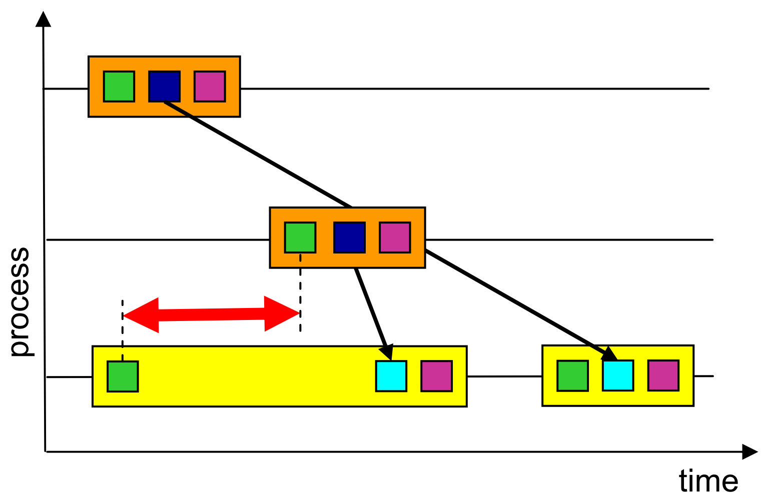
Example patterns



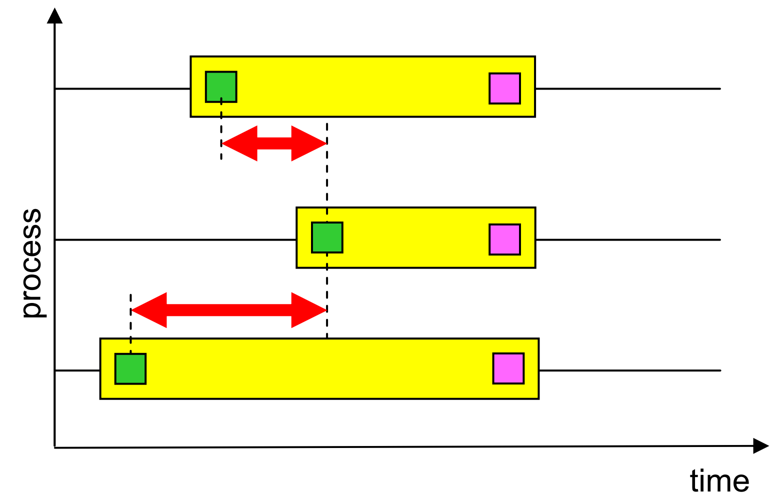
(a) Late sender



(b) Late receiver



(c) Late sender / wrong order



(d) Wait at n-to-n

- Follow-up project of KOJAK
- Started in January 2006
- Funded by Helmholtz Initiative and Networking Fund
  - Developed in partnership with University of Tennessee
- Objective: develop a scalable version of KOJAK
  - Basic idea: parallelization of analysis
  - Current focus: single-threaded MPI-1 applications
- http://www.scalasca.org/

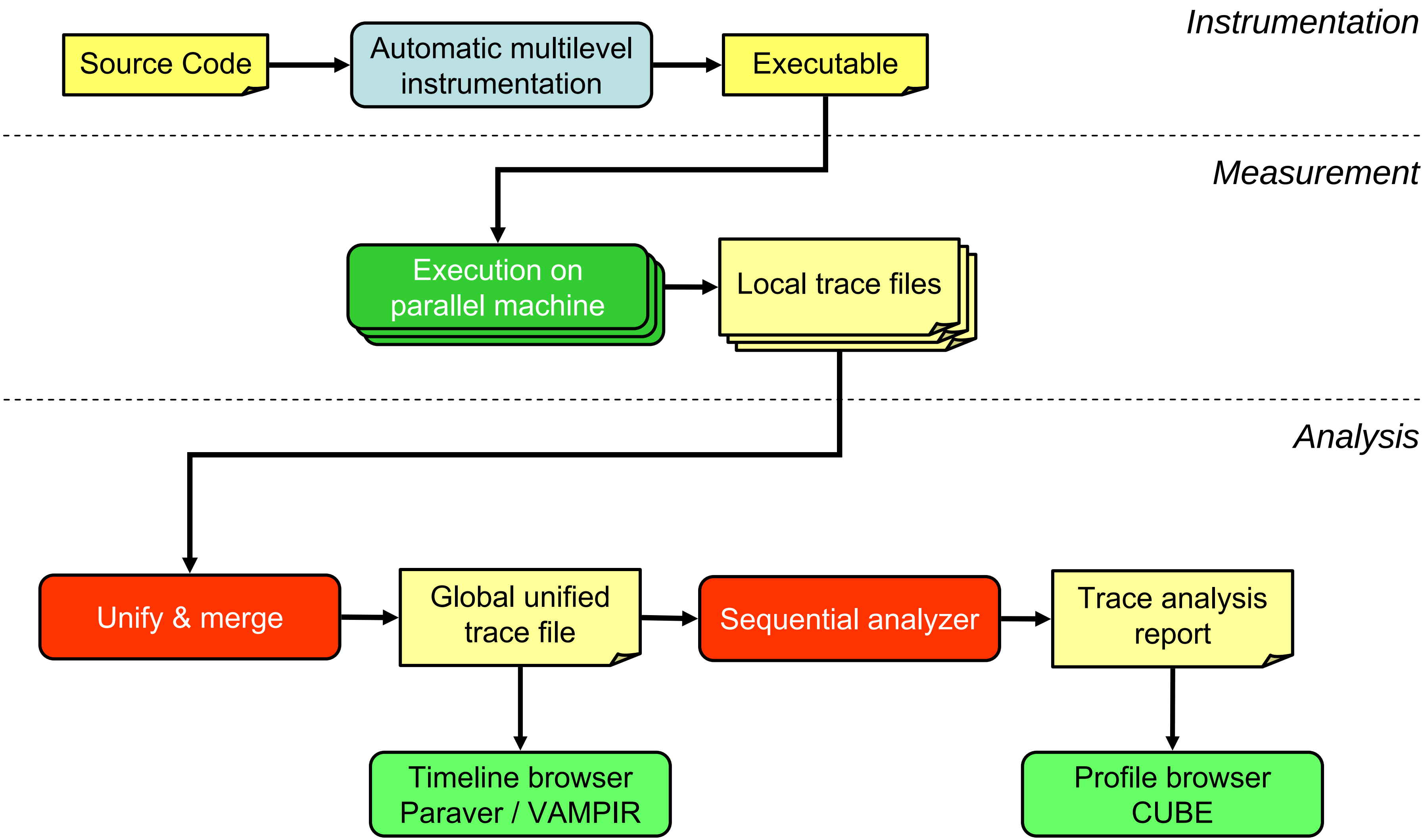
scalasca

Forschungszentrum Jülich
in der Helmholtz-Gemeinschaft

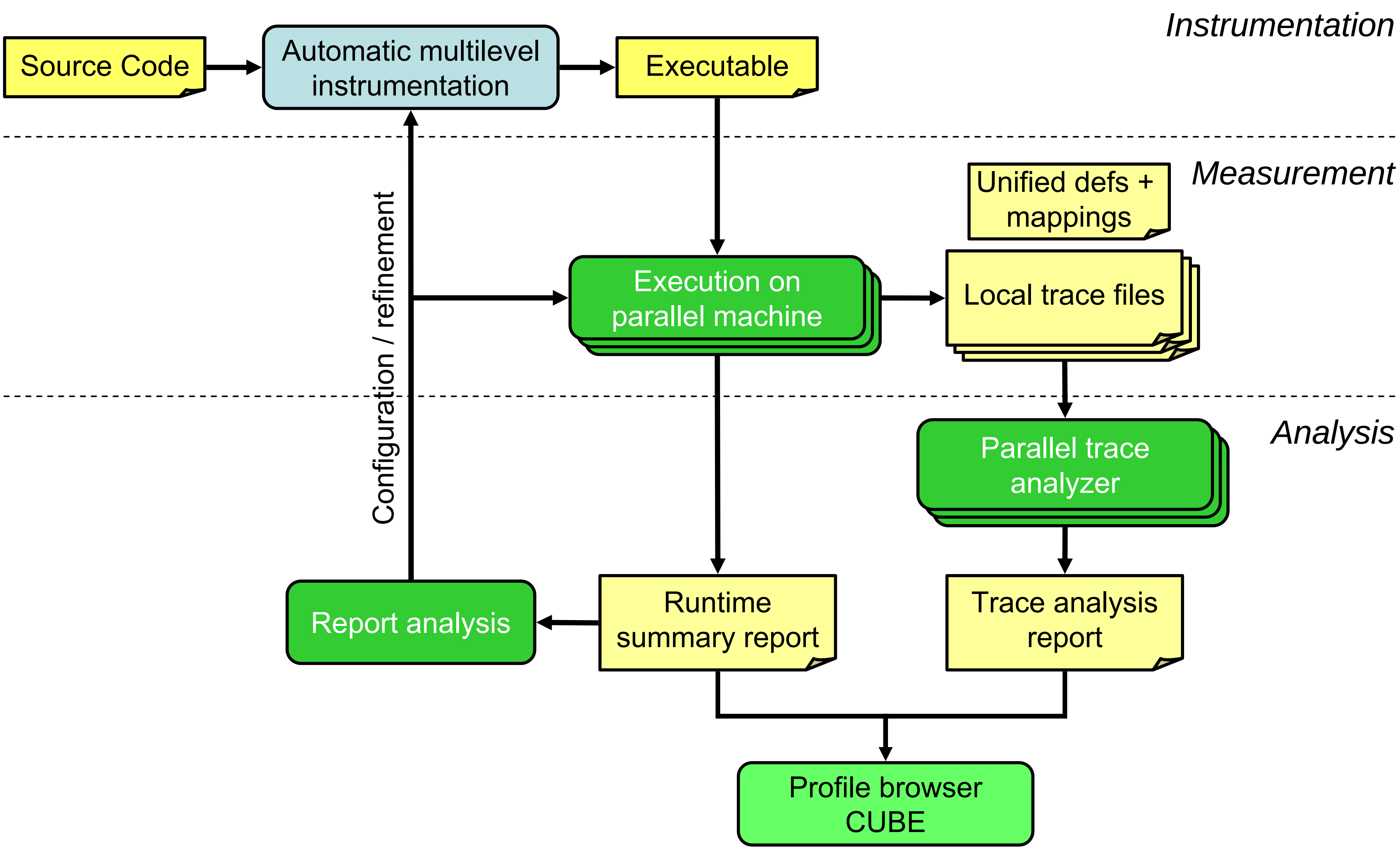
THE UNIVERSITY of TENNESSEE

HELMHOLTZ | ASSOCIATION

Sequential trace analysis (KOJAK)



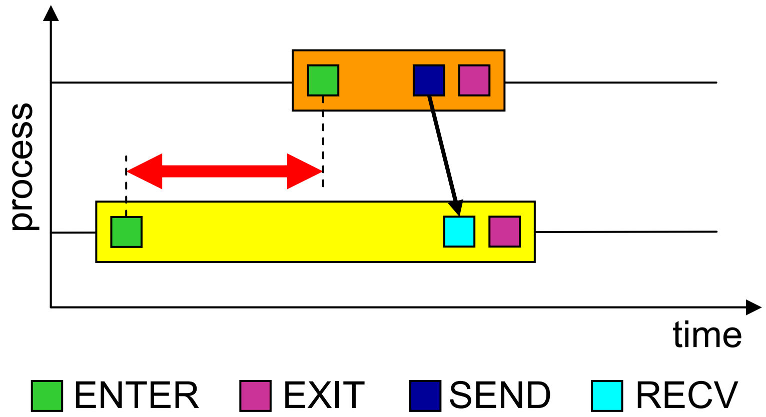
Parallel trace analysis (SCALASCA)

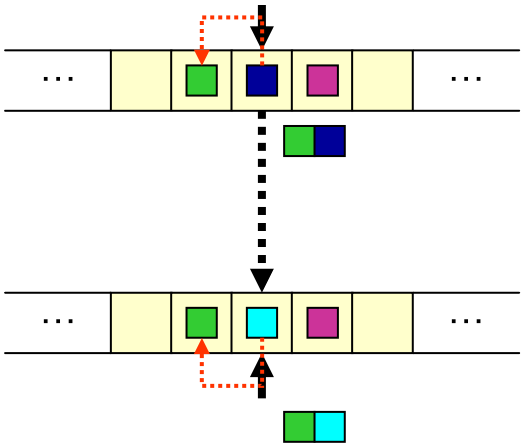


Parallel pattern analysis

- Analyze separate local trace files in parallel
  - Exploit distributed memory and processing capabilities
  - Often allows keeping whole trace in main memory
- Parallel replay of target application‘s communication behavior
  - Analyze communication with an operation of the same type
  - Traverse local traces in parallel
  - Exchange data at synchronization points of target application

Example: Late Sender





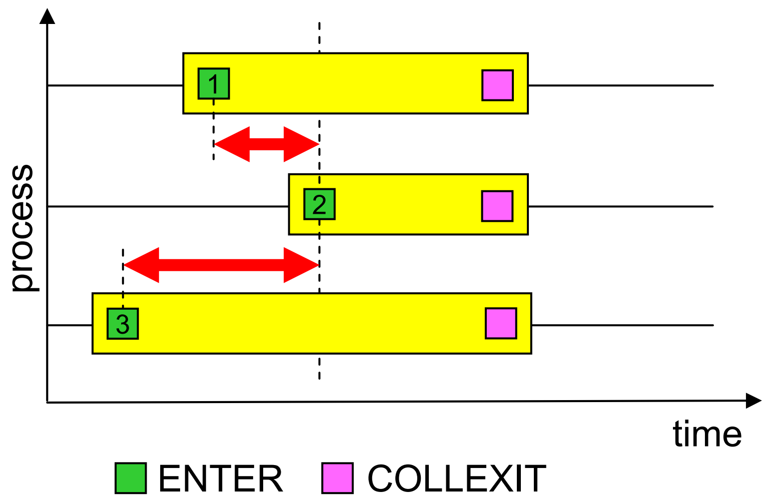
Sender

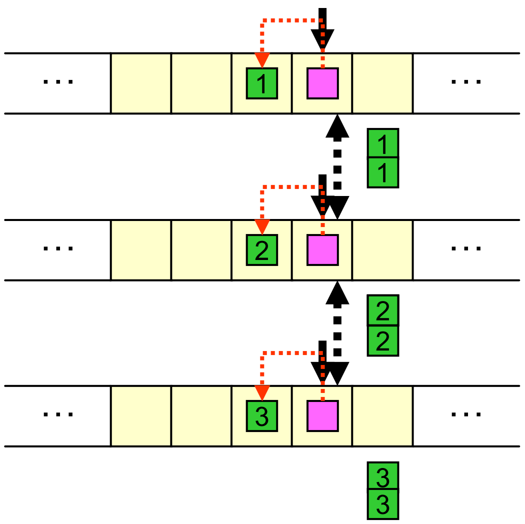
- Triggered by send event
- Determine enter event
- Send both events to receiver

Receiver

- Triggered by receive event
- Determine enter event
- Receive remote events
- Detect *Late Sender* situation
- Calculate & store waiting time

Example: Wait at N x N





- Triggered by collective exit event
- Determine enter events
- Determine & distribute latest enter event (max-reduction)
- Calculate & store waiting time

Test applications

- Scalability
  - ASCI SMG2000 benchmark
    - Semi-coarsening multi-grid solver
    - Fixed problem size per process - weak scaling behavior
  - ASCI SWEEP3D benchmark
    - 3D Cartesian (XYZ) geometry neutron transport model
    - Fixed problem size per process - weak scaling behavior
- Analysis results
  - XNS fluid dynamics code
    - FE simulation on unstructured meshes
    - Constant overall problem size – strong scaling behavior

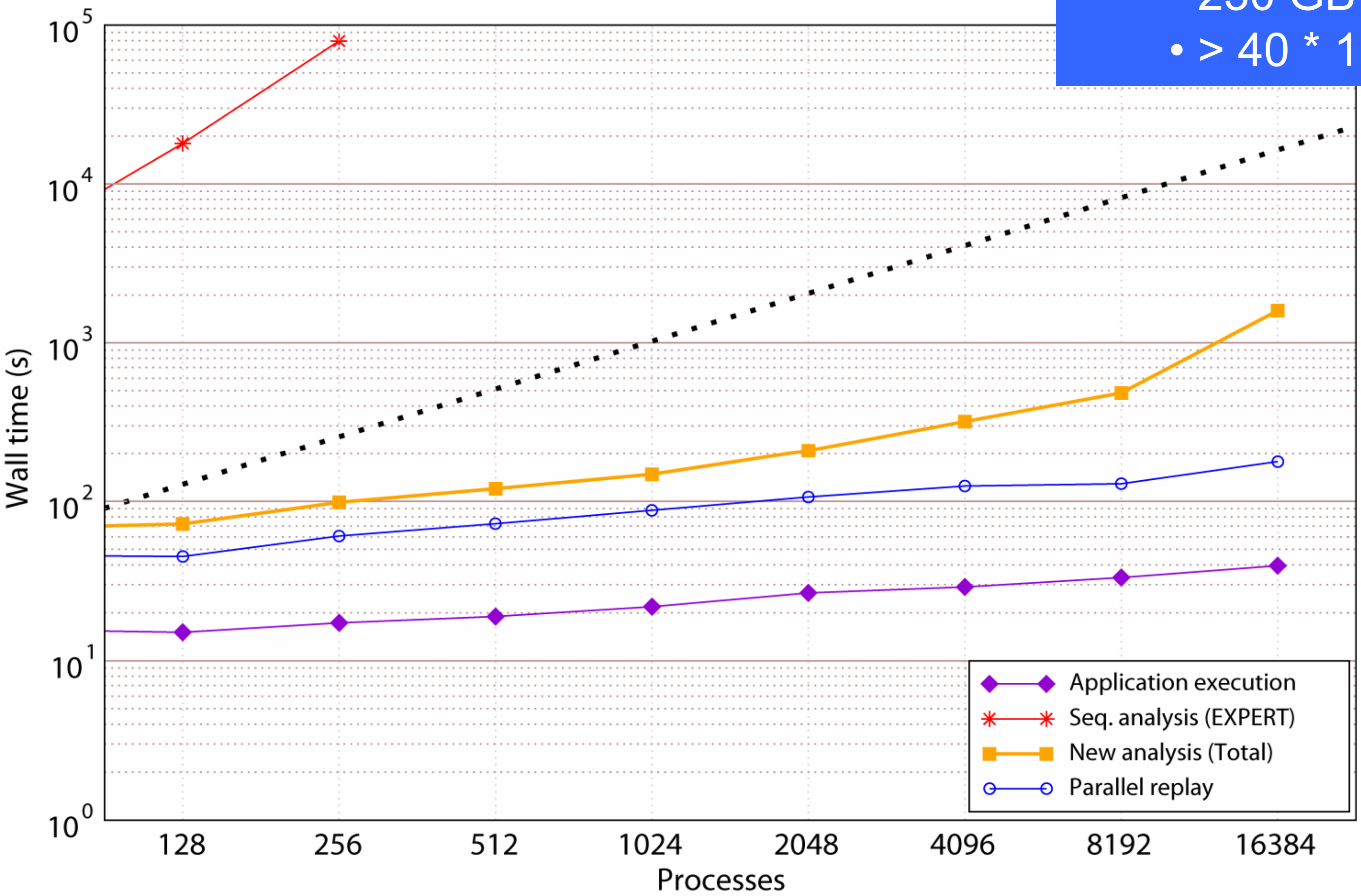
Test platforms

- IBM Blue Gene/L (JUBL) in Jülich
  - 8 Racks with 8192 dual-core nodes
  - 288 I/O nodes
  - GPFS parallel file system

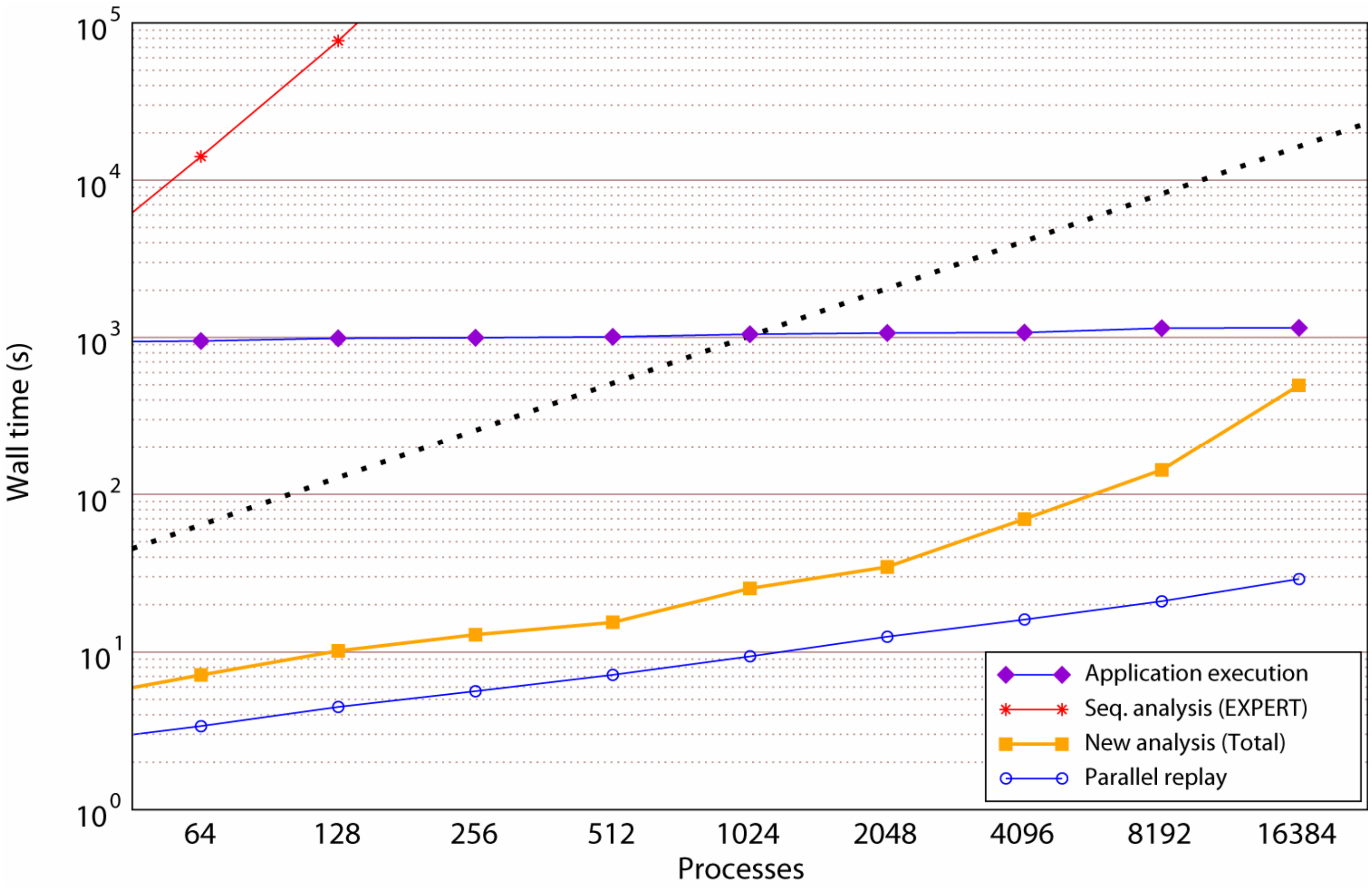
- Cray XT3/4 (Jaguar) in Oak Ridge
  - 11706 dual core nodes
  - Lustre parallel file system
  - Time provided by PEAC end station, (Performance Evaluation and Analysis Consortium), Pat Worley

SMG2000 trace analysis on BG/L

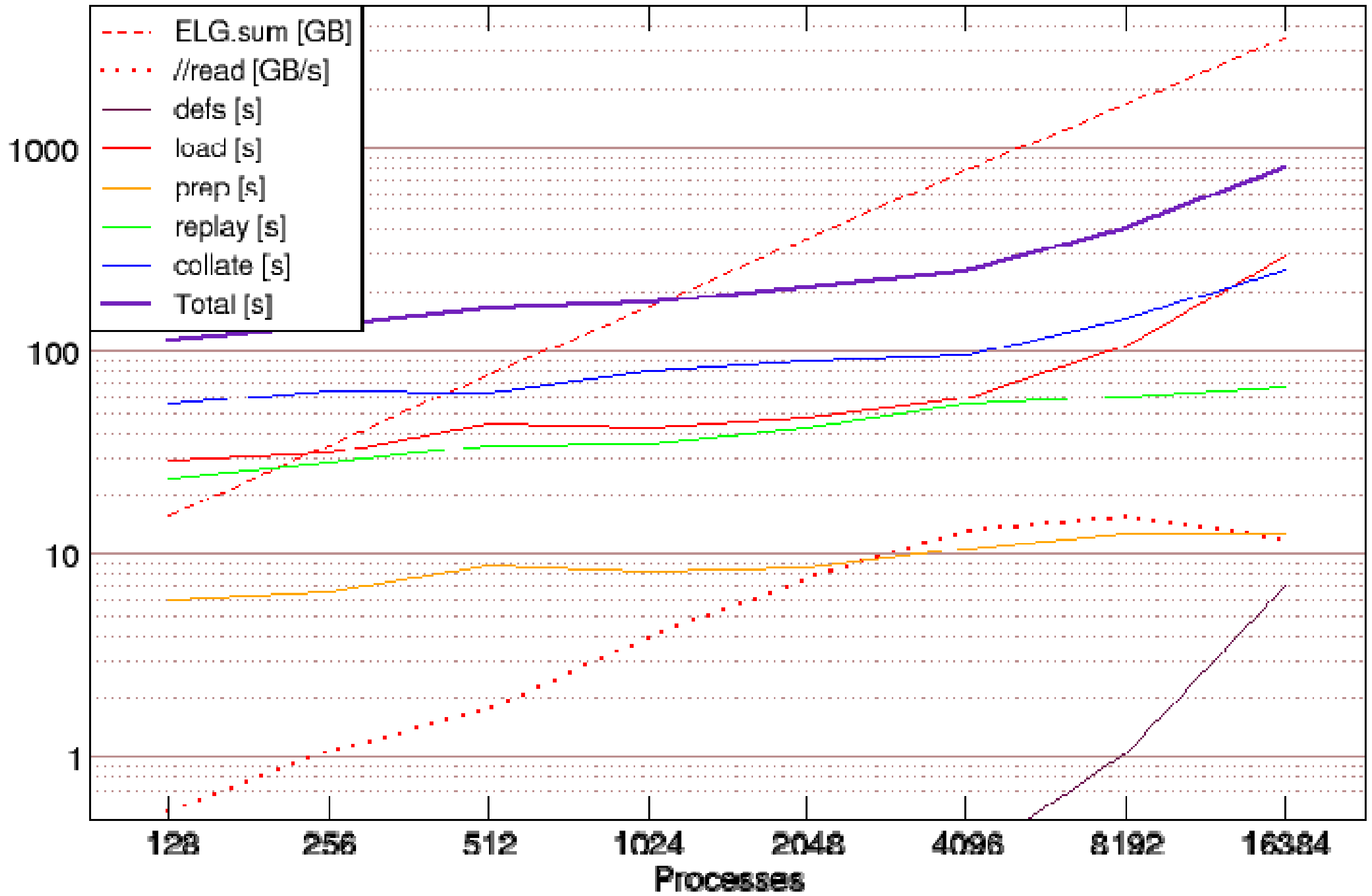
SMG2000 on 16,364 CPUs
- 230 GB trace data
- > 40 * 10^9 events



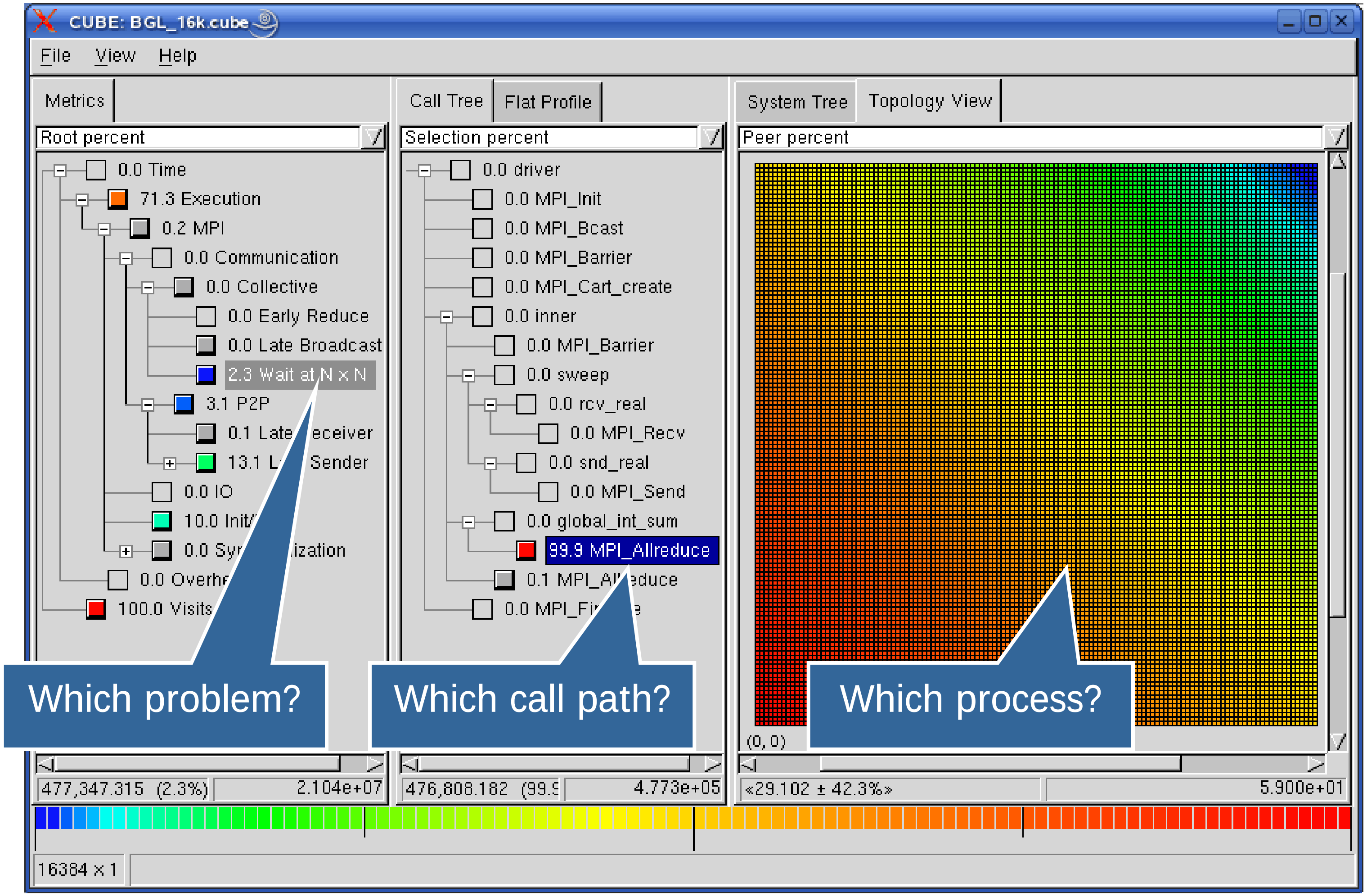
SWEEP3D trace analysis on BG/L



SMG2000 trace analysis on Cray XT3/4

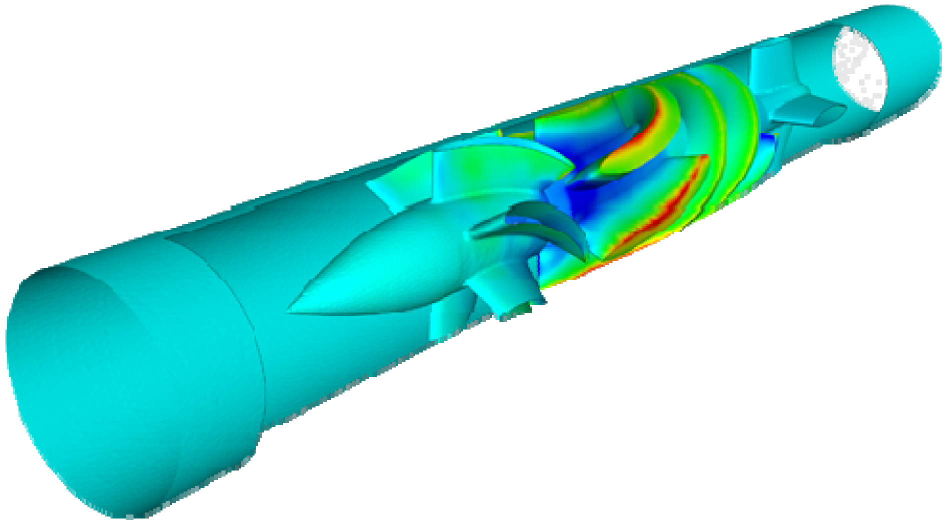


Display of analysis results (SWEEP 3D)

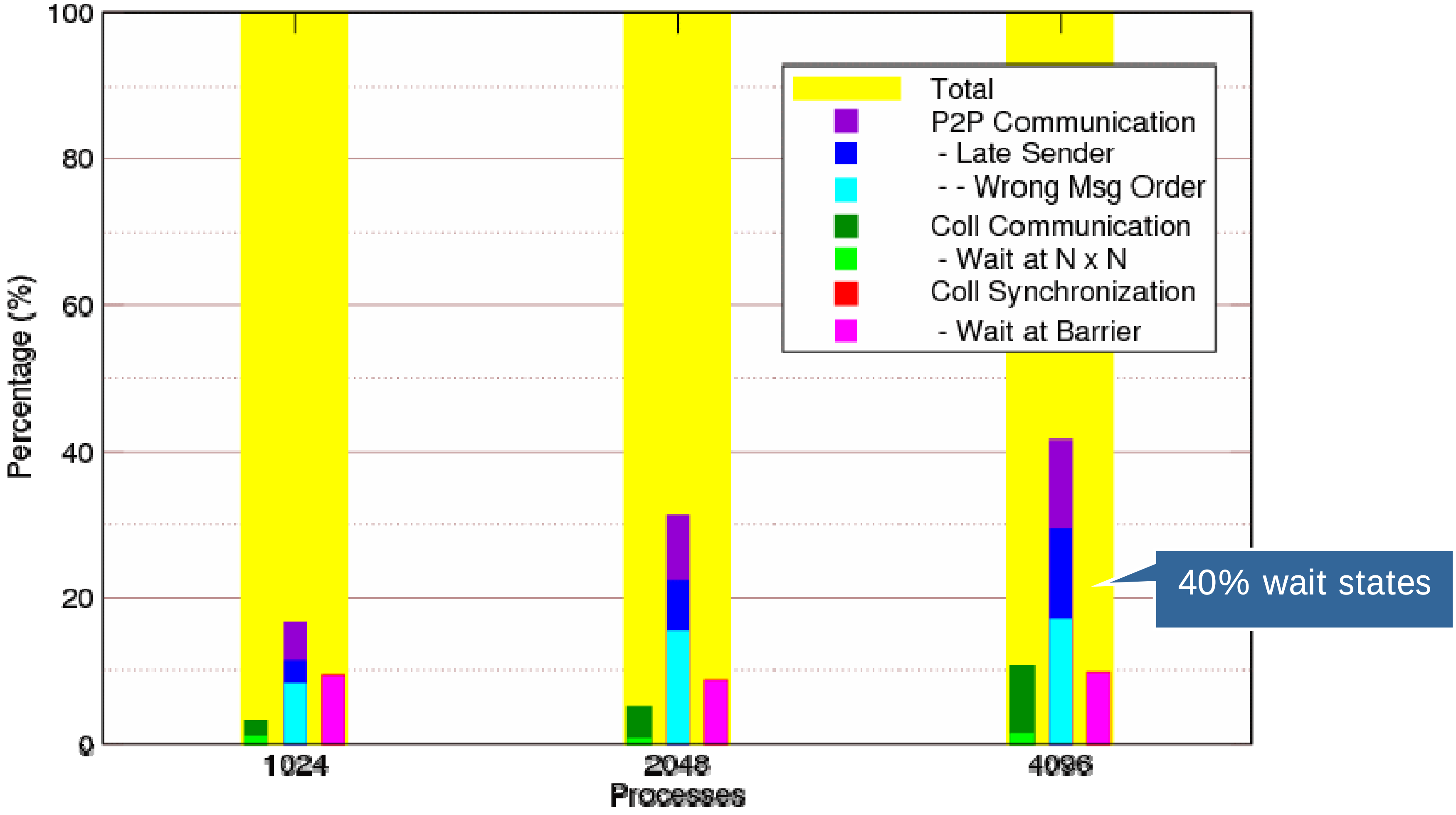


XNS CFD application

- Academic computational fluid dynamics code for simulation of unsteady flows
  - Developed by Computational Analysis of Technical Systems Group, RWTH Aachen University
  - Exploits finite-element techniques, unstructured 3D meshes, iterative solution strategies
  - >40,000 lines of Fortran90
  - Portable parallel implementation based on MPI



Wait states in XNS

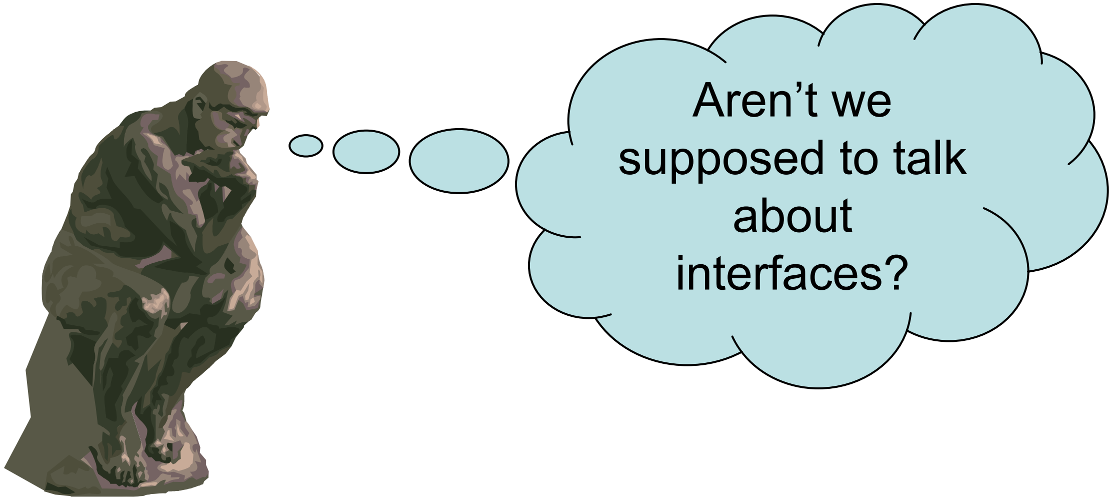


Conclusion

- Wait states addressed by our analysis can be a significant performance problem – especially at larger scales
- Scalability of the analysis can be improved by parallelization
  - Process local trace files in parallel
  - Replay original communication
- Promising results with prototype implementation
  - Analysis scales up to 16,384 processes
    - Previously impractical
  - Potential for further scaling

Future work

- Reduce number of events per process (trace length)
  - Trace selectively
  - Eliminate redundancy
- Find causes of wait states
  - Derive hypotheses from measurements
  - Validate hypotheses using simulation
- Extend (scalable) approach to other programming models



Data formats, interfaces, and components

- Data formats
  - Binary trace format (EPILOG) and archive directory structure
  - XML-based call-path profile format (CUBE)
- Interfaces
  - Serial low-level trace read interface (not beautiful)
  - Parallel low-level trace read interface (planned)
  - Parallel high-level trace read interface (PEARL)
  - Profile read/write interface (CUBE)
  - Profiling interface for OpenMP (POMP)
- Reusable components
  - Tracing and (call-path) profiling library (EPIK)
  - Profile browser (CUBE)
  - Source-to-source instrumenter for OpenMP (OPARI)

Parallel high-level trace read interface

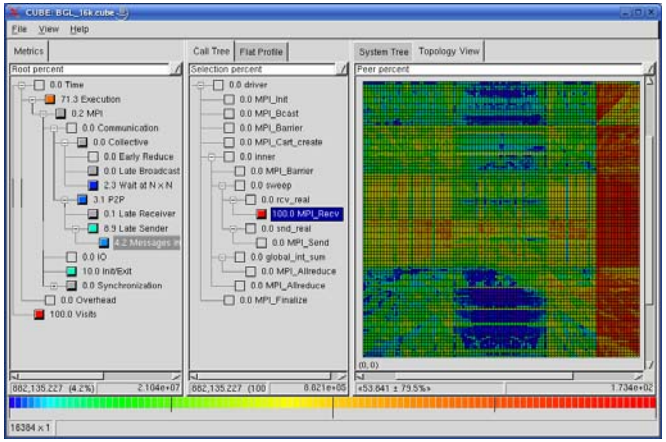
- Provided by separate library written in C++ (PEARL)
  - Parallel version of EARL
- Efficient performance-transparent random access
  - Local traces kept in main memory
  - (Generous) limit for amount of local trace data
  - Different data structures for event storage
    - Linear list, complete call graph (CCG)
- Higher-level abstractions
  - Local execution state
  - Local pointer attributes (can point backward & forward)
- Global abstractions established by parallel replay
  - E.g., repeating message matches SEND with RECV event

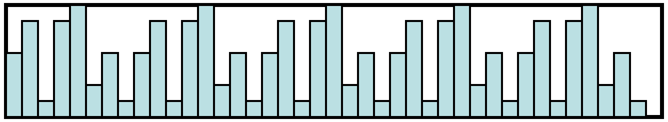
Parallel high-level trace read interface (2)

- Services for cross-process analysis
  - Serialization of events for transfer
  - Remote event
- Two modes of exchanging events
  - Point-to-point & collective
- Current applications
  - Pattern search
  - Time correction utility (in progress)
  - (Simple) statistical trace analysis (in progress)
  - Simulator (in progress)

Call-path profile browser

- CUBE
  - Browser based on tree widgets & topological display
  - Data model and format
  - Operations to manipulate & analyze instances
    - Difference, mean, merge, cut, rank
- New version on the horizon
  - Client server architecture
  - Improved scalability & portability
- Applications
  - SCALASCA trace analysis results & runtime summaries
  - TAU call-path profiles
  - MAMRMOT runtime errors





Virtual Institute
High-Productivity Supercomputing

VI-HPS

Forschungszentrum Jülich

- Central Institute for Applied Mathematics

Forschungszentrum Jülich
in der Helmholtz-Gemeinschaft

RWTH Aachen University

- Center for Computing and Communication

RWTH AACHEN UNIVERSITY

Technische Universität Dresden

- Center for Information Services and High Performance Computing

TECHNISCHE UNIVERSITÄT DRESDEN

University of Tennessee

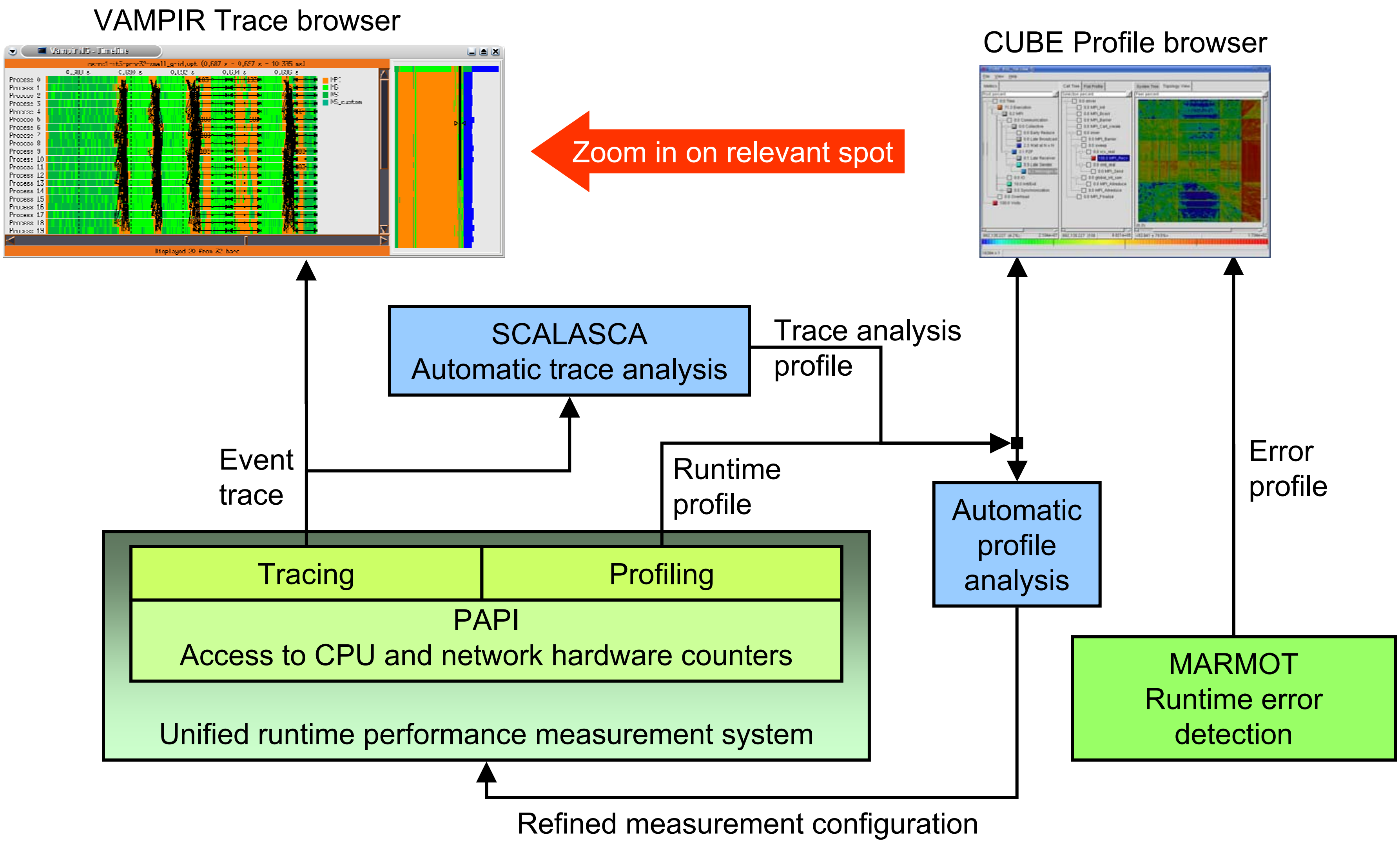
- Innovative Computing Laboratory

THE UNIVERSITY of TENNESSEE UT

Sponsored by HELMHOLTZ ASSOCIATION

http:www.vi-hps.org

Vision: integrated tool environment



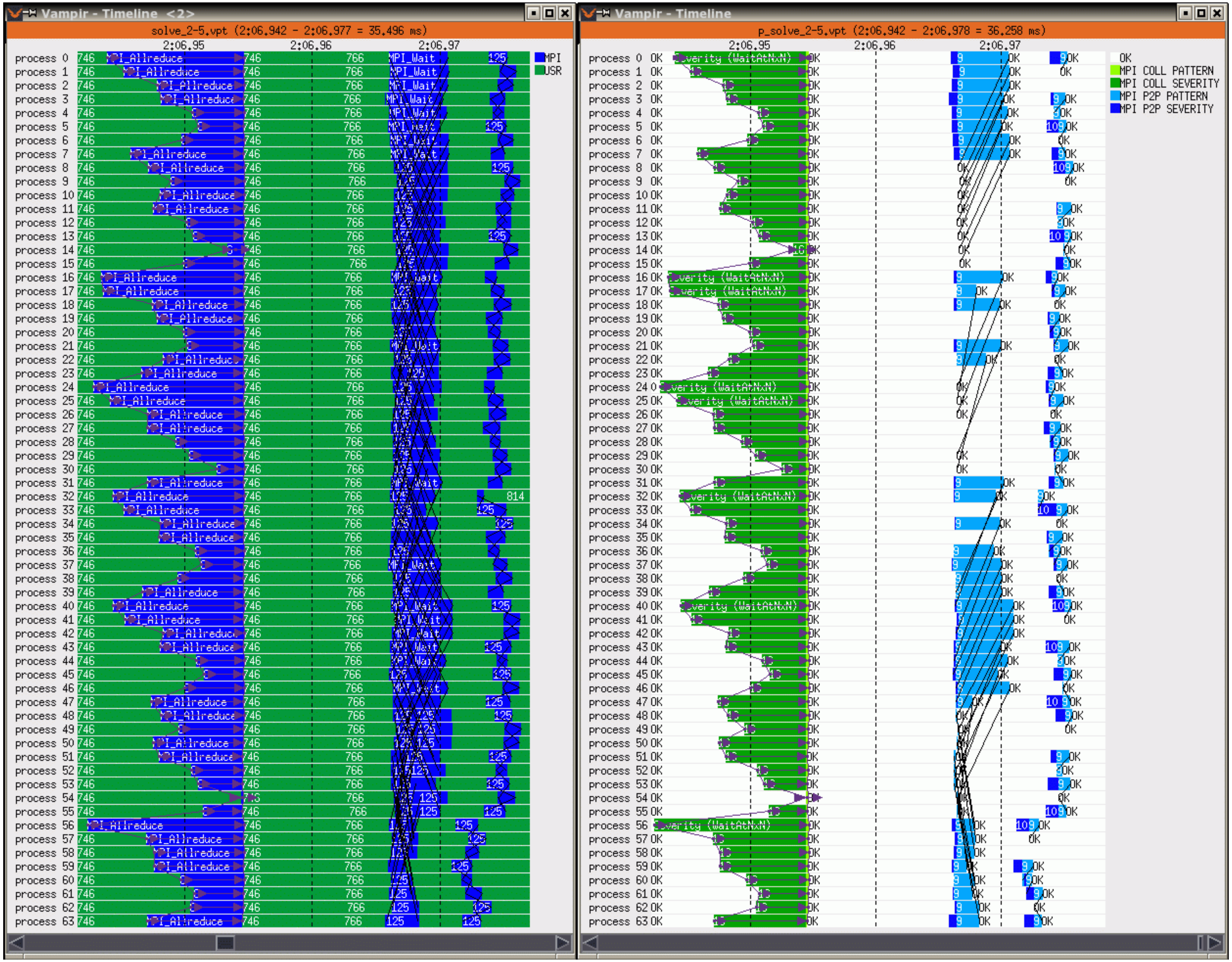
Interfaces

- Trace read / write interface
  - Define common parallel low-level read interface
    - EPILOG & OTF
    - Joint high-level event model
    - Later mapping onto low-level model?
  - Joint format?!
- Profile read / write interface
  - Error data and performance data requirements
- Trace browser ↔ profile browser
  - Profile browser client of trace browser
  - Trace browser client of profile browser?
- Hardware counter read interface
  - Access to network counters / temperature sensors

Conclusion (2)

- Expected benefit of standardization
  - Makes a user’s life easier
  - Makes our life as a community easier
- Challenges
  - Documentation
  - Robustness
  - Funding
  - Licensing
- What SCALASCA has to offer
  - Parallel high-level trace read interface, profile browser
- What SCALASCA needs
  - Integration with time-line browsers
  - Flexible and portable instrumentation technologies

Recent work on integrating automatic with visual trace analysis…



Thank you!

For more information, visit our project home page:

http://www.scalasca.org

scalasca