Mitglied der Helmholtz-Gemeinschaft

[Scalasca] Tool Integrations

Aug 2011 | Bernd Mohr

CScADS Performance Tools Workshop
Lake Tahoe

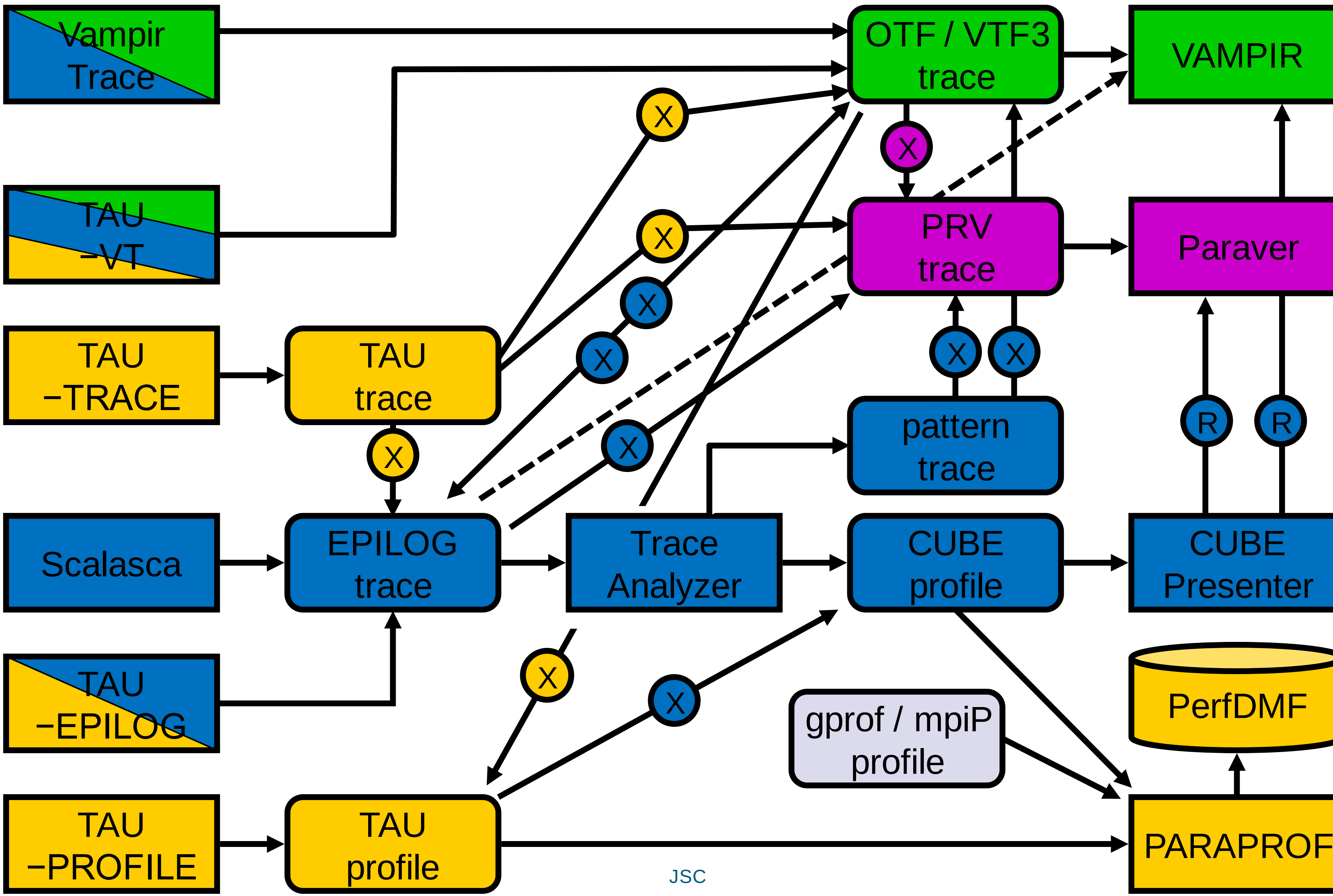
Contents

- Current integration of various direct measurement tools
  - Paraver
  - Scalasca
  - TAU
  - Vampir
- CUBE tool integration API
  - Cube ⇨ Paraver / Vampir
  - Cube ⇨ generic tool
- UNITE
- EU funded integration projects
  - Score-P

Direct measurement tools

- **Extrae / Paraver**
  - Very flexible (as programmable) trace visualizer
  - Barcelona Supercomputing Center
  - http://www.bsc.es/paraver
- **Scalasca**
  - Scalable callpath profiler and trace analyzer
  - Jülich Supercomputing Centre and GRS Aachen
  - http://www.scalasca.org
- **TAU Performance System ®**
  - Very portable and versatile profile and tracing toolset
  - University of Oregon
  - http://tau.uoregon.edu
- **VampirTrace / Vampir**
  - Trace measurement and visualization
  - Technical University of Dresden
  - http://www.tu-dresden.de/zih/vampirtrace and http://www.vampir.eu

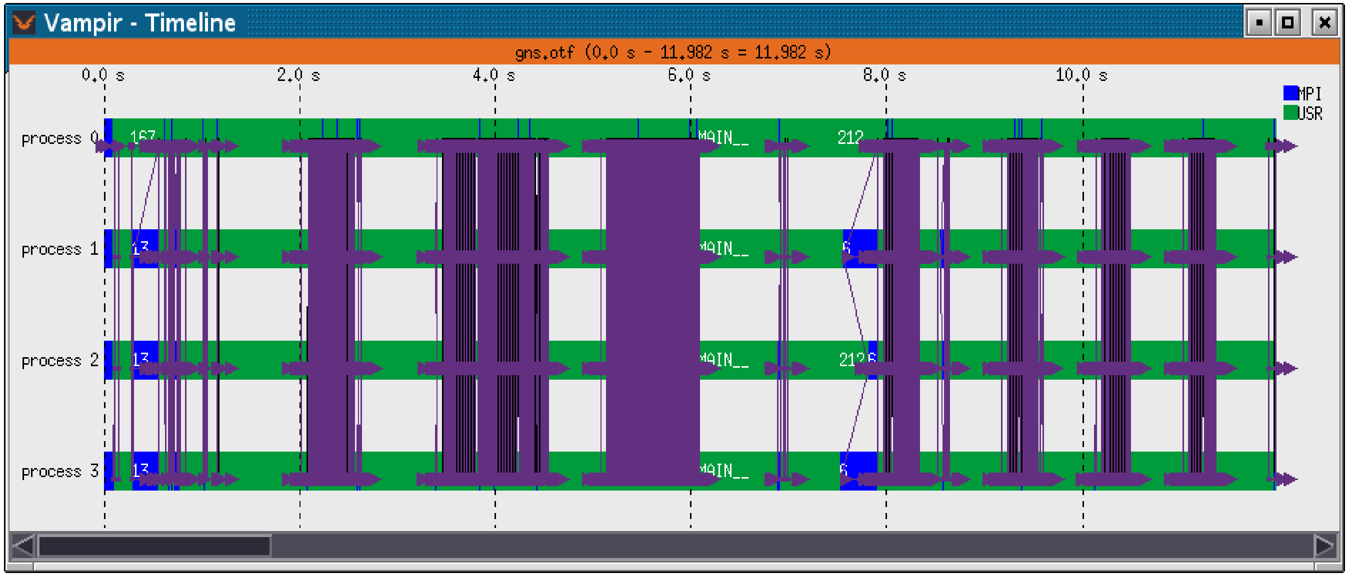
Scalasca ⇔ TAU ⇔ VAMPIR ⇔ Paraver



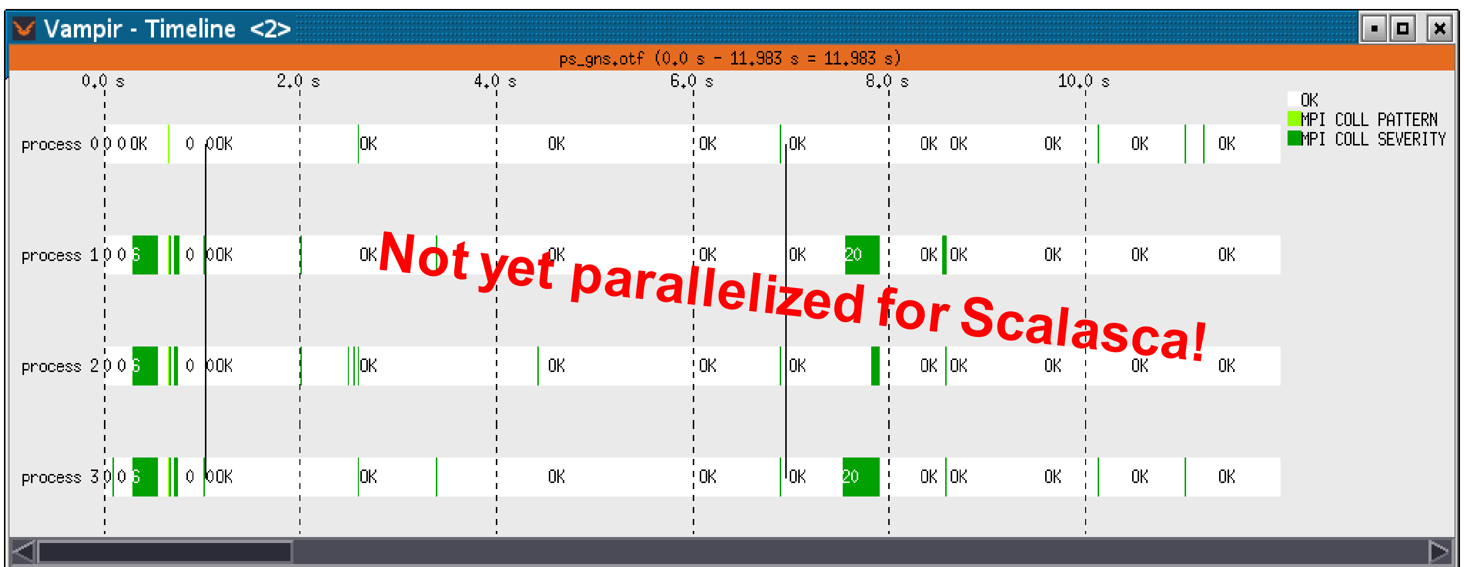
Integration Paths

- **based on component usage**
  - all tools use PAPI for portable HW counter measurement
  - Scalasca, TAU, VampirTrace
    - Use OPARI for portable OpenMP instrumentation
    - Use PDT/tauinst for source code instrumentation
    - Use DynInst for binary instrumentation
  - TAU can be configured to use measurement system of Scalasca or Vampir as backend
- **based on data exchange**
  - Vampir (7.2+) / VampirServer (2.3+) can read Scalasca's EPILOG traces
  - TAU paraprof can read Scalasca's CUBE profiles
  - Large variety of profile and trace format converters

VAMPIR ⇔ KOJAK via Pattern Traces

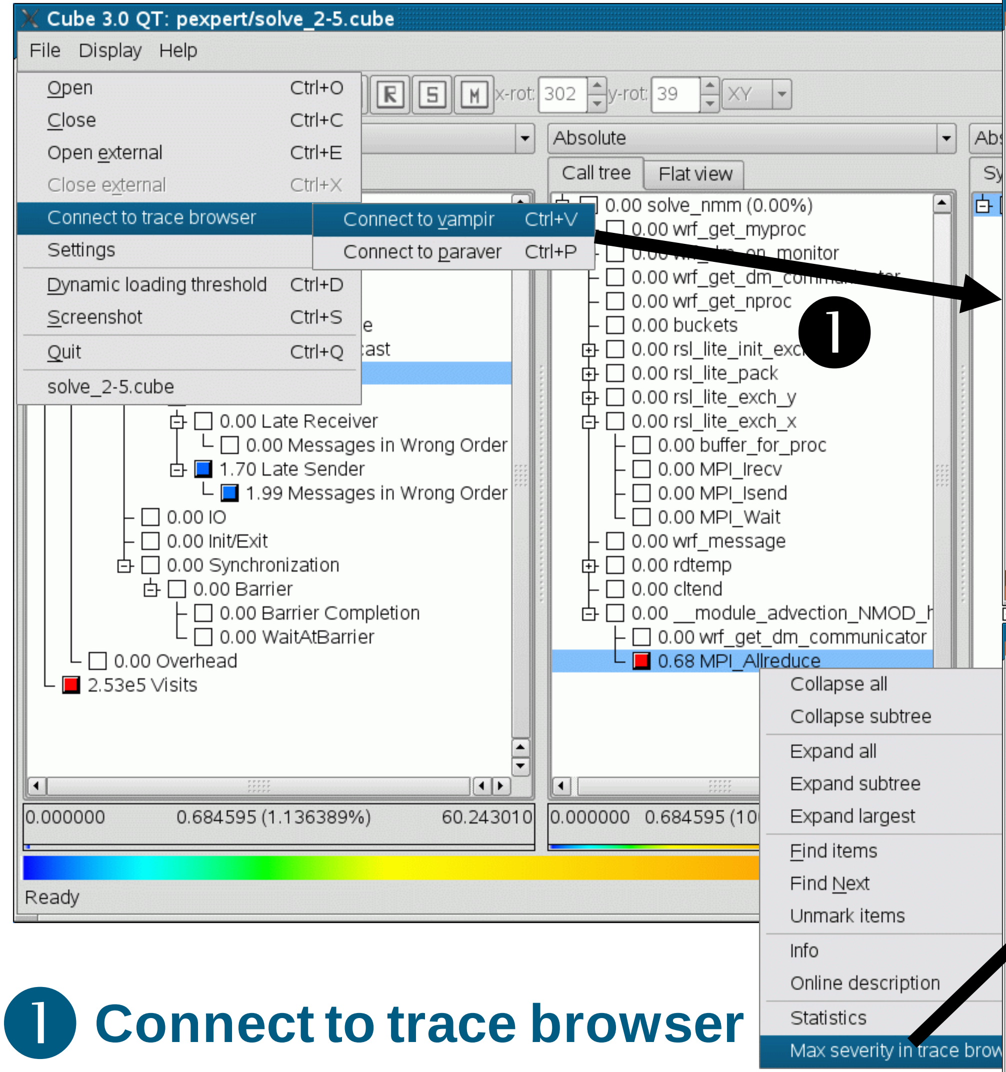


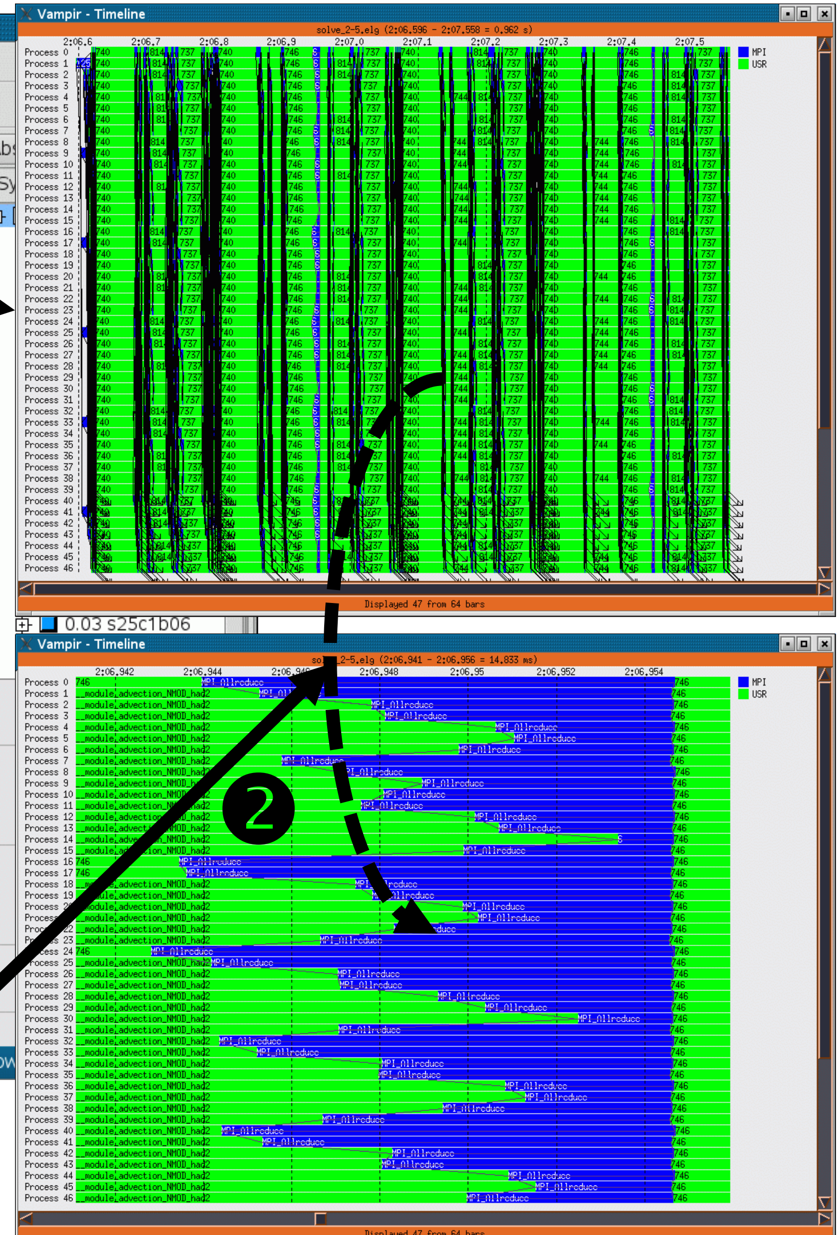
Original Vampir event trace



Pattern trace generated by KOJAK analysis **highlighting problematic areas**

Scalasca ⇨ Vampir Integration





❶ **Connect to trace browser**

❷ **Max severity in trace browser**

CUBE Tool Integration API

- Current hard-coded interactions with trace browsers
  - Vampir via D-BUS interface
    - 2-way communication (+), complex implementation (-)
  - Paraver via configuration file loaded via USR1 signal
    - 1-way communication (-), simple implementation (+)

- Current work
  - Design (and implementation) of a CUBE generic tool integration API
    - Small but well-defined set of interaction points (callbacks) and context information (parameters)
    - Tool-specific implementation of interface as shared library
  - **Better ideas? Comments? Experiences?**

UNITE

- **UN**iform **I**ntegrated **T**ool **E**nvironment

- **Goal:**
  - Provide portable common access to parallel performance tools
  - Lower bar for inexperienced users and admins

- **Basic idea:**
  - Based on “module” command (www.modules.org)
  - Standardize module names and structure (e.g. help)
  - Activate by “**module load UNITE**”

Definitions and Standard Names

- **Package** ::= product, tool, or component which
  - Is available / can be used / can be installed as separate entity
  - Two basic sorts of packages: Tools, Utils
  - Typically comes in multiple versions
  - Example: vampir, scalasca, marmot, …
- **Version**
  - <MajorVersion> . <MinorVersion> [.<Plevel>] [(rc|b)<Number>]
  - Example: 2.1b2
- **Specialization** ::= Optional constraints
  - Which limit the applicability of a package and/or version
  - Currently mainly needed on Linux installations
  - Specified as: –<MpiLibrary>–<Compiler>–<Precision>
  - Unnecessary constraints are left out
  - Example: –openmpi–32bit

Installation Space Layout: Module Files

- Install required UNITE components together at **system-specific** installation path **UNITE_ROOT**

```
${UNITE_ROOT}/
    modulefiles/                  # UNITE module files
        tools/
            <package>/
                <version>-<spezialization>
        utils/
            <package>/
                <version>-<spezialization>
        scripts/                  # for basic scripts
        templates/                # for “generic” module files
    doc/                          # for overall UNITE docu
```

Installation Space Layout: Package Files

- Actual package are installed also under ${UNITE_ROOT}/packages
  [Note: if not feasable or to include historic installations, create symbolic-link trees to real installation directories]

```
${UNITE_ROOT}/
   packages/
      <package>/
         <version>-<spezialization>/
            <package-specific-sublayout>
```

Example: "module help scalasca" Output

```
% module help scalasca

-- Module Specific Help for 'scalasca/1.0-mpibull2-intel-64bit‘ --

Scalasca:
Scalable Performance Analysis of Large-Scale Parallel Applications
Version 1.0 (for BullMPI 2, Intel Compiler, 64bit)

Basic usage:
1. Instrument application with skin = "scalasca -instrument"
2. Collect & analyze execution measurement with scan = "scalasca -
   analyze"
3. Examine analysis results with square = "scalasca -examine"

For more information:
- See ${SCALASCA_ROOT}/doc/manuals/quickref.pdf
  or type "scalasca -h"
- http://www.scalasca.org
- mailto:scalasca@fz-juelich.de
```

UNITE Tools Package

- UNITE website: http://apps.fz-juelich.de/unite/
  - **Common** usage and installation **documentation**
  - **Download, build and install**
    a set of performance and validation tools **in one package:**
    - UNITE package installer and module package
    - OTF-1.6.5 (⇨ **1.9**)
    - pdtoolkit-3.15 (⇨ **3.16**)
    - cube-3.3 (⇨ **3.3.2**)
    - Scalasca-1.3.1 (⇨ **1.3.3**)
    - Vampirtrace-5.8.2 (⇨ **5.11**)
    - UniMCI-1.0.1
    - Marmot-2.4
    - Vampir-5.x or 7.x
    - VampirServer-1.x, 2.x
  - **Updated version with latest tool versions available real soon now!**

UNITE Tools Package II

- Extensively tested on
  - Itanium/IA32/x86_64 platforms with various MPI libraries (MPICH1, MPICH2, OpenMPI, Intel MPI, LAM, BullMPI, Parastation MPI, SGI MPT, ...)
  - AIX and Solaris clusters
- Already in use on Bull Nova and production machines of JSC, ZIH, RWTH, HLRN, …

- Future work:
  - Integration of other tools (Paraver, TAU, …)
  - More platforms (Cray XT, IBM BlueGene, NEC)

Funded Integration Projects

JÜLICH FORSCHUNGSZENTRUM

- **SILC (01/2009 to 12/2011)**
  - Unified measurement system (Score-P) for Vampir, Scalasca, Periscope
- **PRIMA (08/2009 to 08/2012)**
  - Integration of TAU and Scalasca
- **LMAC (08/2011 to 07/2013)**
  - Evolution of Score-P
  - Analysis of performance dynamics
- **H4H (10/2010 to 09/2013)**
  - Hybrid programming for heterogeneous platforms
- **HOPSA (02/2011 to 01/2013)**
  - Integration of system and application monitoring

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

DEPARTMENT OF ENERGY UNITED STATES OF AMERICA

GEFÖRDERT VOM Bundesministerium für Bildung und Forschung

ITEA 2 INFORMATION TECHNOLOGY FOR EUROPEAN ADVANCEMENT

SEVENTH FRAMEWORK PROGRAMME

MINISTRY OF EDUCATION AND SCIENCE OF THE RUSSIAN FEDERATION

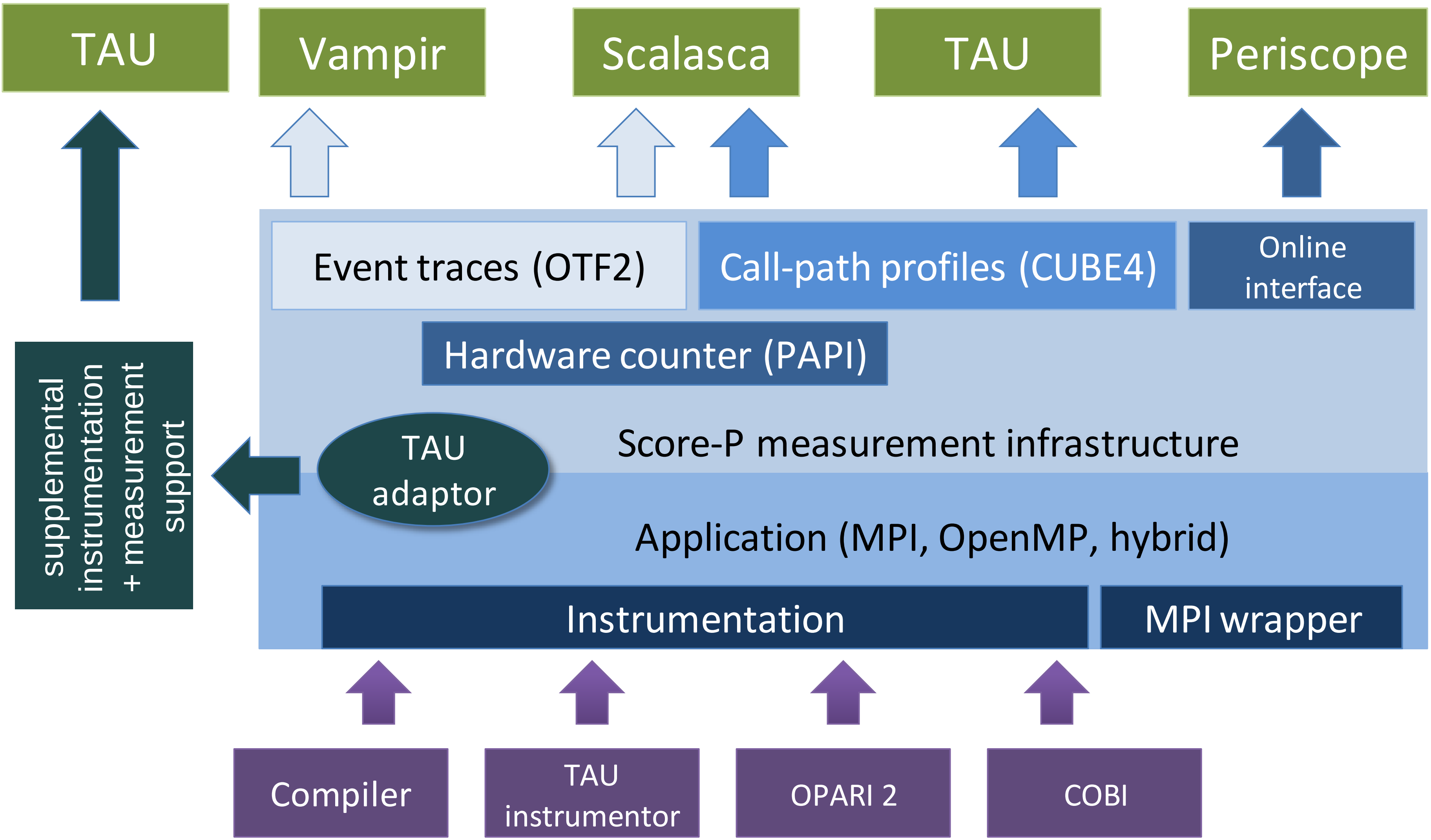
Score-P Objectives

- Mainly funded by SILC, PRIMA, LMAC projects
- Make common part of Periscope, Scalasca, TAU, and Vampir a community effort
  - **Score-P measurement system**
- Save manpower by sharing resources
- Invest this manpower in analysis functionality
  - Allow tools to differentiate faster according to their specific strengths
  - Increased benefit for users
- Avoid the pitfalls of earlier community efforts
  - Start with small group of partners
  - Build on extensive history of collaboration

Score-P Design Goals

- **Functional requirements**
  - Performance data: profiles, traces
  - Initially direct instrumentation, later also sampling
  - Offline and online access
  - Metrics: time, communication metrics and hardware counters
  - Initially MPI 2 and OpenMP 3, later also CUDA and OpenCL
- **Non-functional requirements**
  - Portability: all major HPC platforms
  - Scalability: petascale
  - Low measurement overhead
  - Easy installation through UNITE framework
  - Robustness
  - Open source: New BSD license

Score-P Architecture

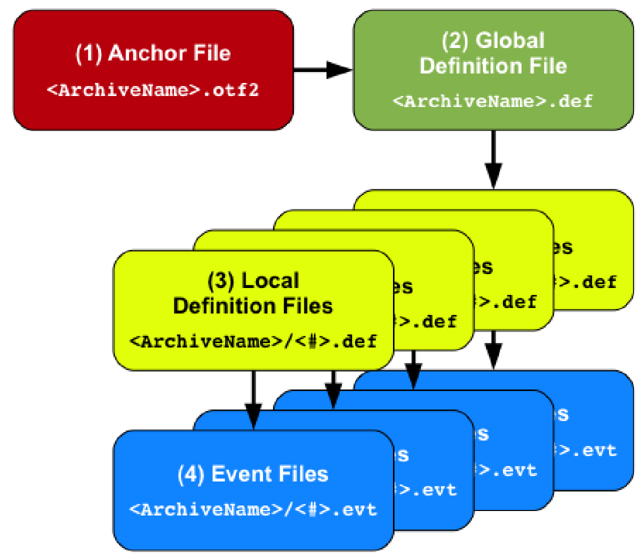


Score-P Partners

- Forschungszentrum Jülich, Germany
- German Research School for Simulation Sciences, Aachen, Germany
- Gesellschaft für numerische Simulation mbH Braunschweig, Germany
- RWTH Aachen, Germany
- Technische Universität Dresden, Germany
- Technische Universität München, Germany
- University of Oregon, Eugene, USA

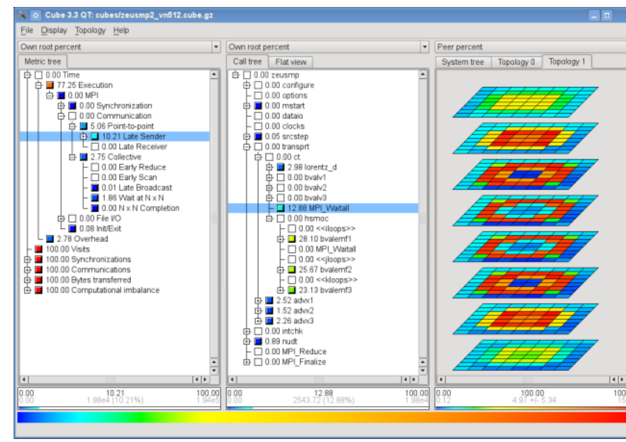
OTF-2 Tracing Format

- Successor to OTF and EPILOG
- Same basic structure as OTF, EPILOG, or other formats
- Design goals
  - High scalability
  - Low overhead (storage space and processing time)
  - Good read/write performance
    - Reduced number of files during initial writing via SIONlib
  - Compatibility reader for OTF and Epilog formats
  - Extensibility



CUBE-4 Profiling Format

- Latest version of a family of profiling formats
  - Still under development, to be released soon
- Representation of three-dimensional performance space
  - Metric, call path, process or thread
- File organization
  - Metadata stored as XML file
  - Metric values stored in binary format
    - Two files per metric:
      data + index for storage-efficient sparse representation



- Optimized for
  - High write bandwidth
  - Fast interactive analysis through incremental loading

Score-P Status and Future Plans

- Currently being extensively tested
- Release of beta version at SC11
- Extensions
  - Heterogeneous computing (H4H project)
  - Time-series profiling (HOPSA & LMAC projects)
  - Sampling (LMAC project)