ELP

Effektive Laufzeitunterstützung für zukünftige Programmierstandards

TECHNISCHE UNIVERSITÄT DRESDEN

RWTH AACHEN UNIVERSITY

science + computing
an atos company

allinea

intel

Bundesministerium
für Bildung
und Forschung

DLR

Speaker: Tim Cramer, RWTH Aachen University

Agenda

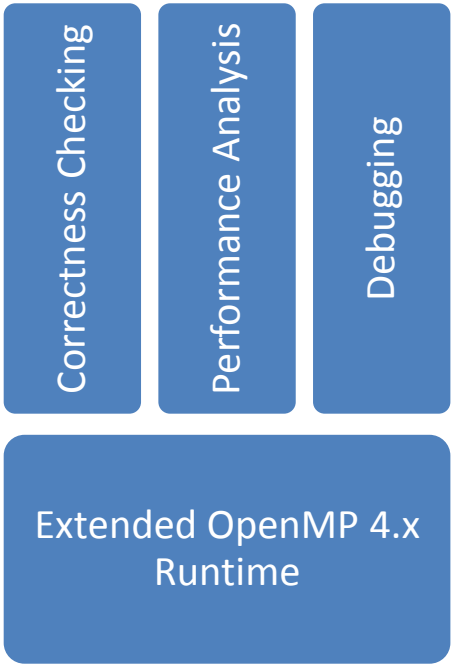
- **ELP Project Goals**
- **ELP Achievements**
- **Remaining Steps**

ELP Project Goals

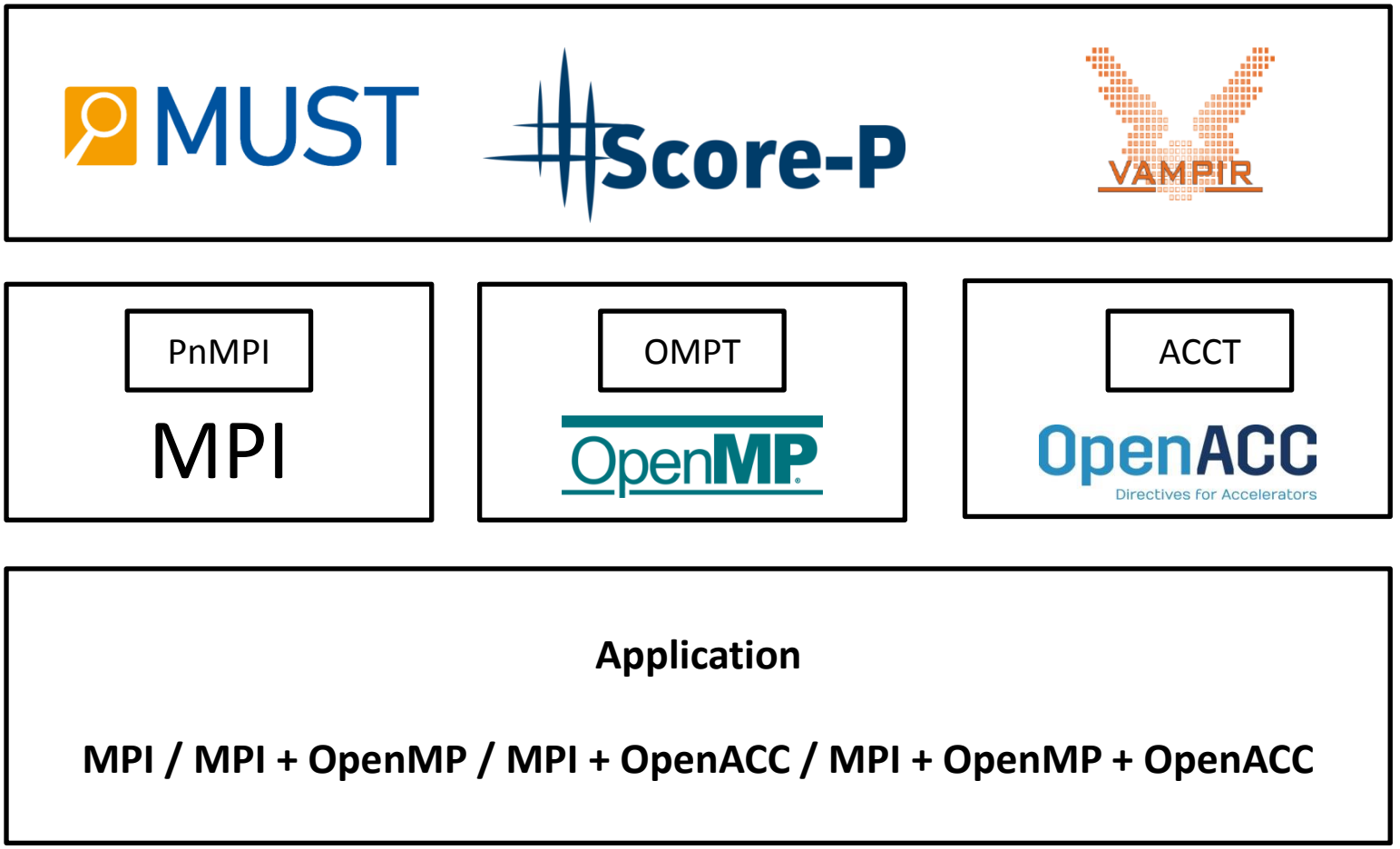
- **Goals of ELP: Improve programmer productivity**
  - By influencing the programming standards
  - By extending compiler / runtime / tools support
  - Reducing workflow complexity
    - Tight tool integration in the workflow
- **Make the correctness analysis part of a standard workflow**
- **Focus on new programming features (e.g., offloading)**

The ELP Approach for Programmer Productivity

- **Support for directive-based standards**
  - Tools support for OpenACC applications
  - Focus on OpenMP 4.x
    - OpenMP standardization work
    - Prototype implementation of proposed extensions, i.e. OMPT in LLVM/Intel OpenMP runtime
  - Tools and infrastructure are available as open source
- **Performance analysis with Score-P and Vampir**
  - Support for offloading directives
  - Device activity analysis
  - Enhanced analysis of measured accelerator data
- **Correctness checking with MUST**
  - Support for hybrid programs in MUST: MPI + OpenMP
  - New kinds of races, support of new APIs
  - Hybrid deadlocks



Tools Landscape



MUST and Score-P

- **MUST was limited on MPI correctness analysis:**
  - Type mis-matches, overlapping of buffer usage, …
  - Deadlocks resulting from MPI calls

- **ELP: support for hybrid MPI + OpenMP programs**
  - Pure OpenMP 4.x: data races between host and accelerator, deadlocks, …
  - Hybrid: MPI deadlocks involving threads, data races involving data transfer, ...
- **Score-P: Scalable performance measurement infrastructure for parallel codes**
  - Measurement infrastructure for profiling, event tracing and online analysis of applications
- **ELP: Support for OpenMP 4.0 and OpenACC programs**
  - Standardized tools interfaces: OMPT and ACCT
  - Recording of
    - OpenMP target events
    - OpenMP events on the target device (offloaded via the target construct)
    - OpenACC events

Agenda

- ELP Project Goals
- **ELP Achievements**
- Remaining Steps

Project Achievements: Overview

- **OpenACC Profiling interface is part of OpenACC 2.5**
  - Supporting the development of OpenACC
- **Contributions to OMPT implementation (LLVM runtime)**
  - Close collaboration with RICE and UOREGON and INTEL
- **Proposal for an OMPT extension to support OpenMP target constructs**
  - Including a prototype implementation in the Intel/LLVM OpenMP runtime
  - Extension will be part of OpenMP 5.0
- **Score-P support for OpenMP 4.x target constructs and OpenACC directives**
  - OpenMP target support implemented in Score-P prototype
  - OpenACC support in Score-P 3.0 release
- **MUST + GTI infrastructure is ready for hybrid parallel programs**
  - Set of new checks: Correct use of OpenMP barriers, simple hybrid deadlocks, race detection between host and accelerator
- **11 peer-reviewed papers published**

Project Achievements: OpenACC Tools Interface

- **OpenACC 2.5 standard released in November 2015**
- **Technical report has been incorporated in the official specification**
  - Chapter 5: Profiling Interface
  - Defines a set of OpenACC runtime events
- **Dissemination**
  - Paper published at ICPP'15:
    "OpenACC Programs Examined: A Performance Analysis Approach"
  - Presented at SC'15, HPCwire article:
    „OpenACC Reviews Latest Developments and Future Plans"
- **OpenACC support is part of the Score-P 3.0 release**
  - Tested with the PGI 15.x and 16.x compilers (NVIDIA GPUs)
  - Validated with OpenACC benchmarks of the SPEC ACCEL suite
  - Already used in GPU-Hackathon at TU Dresden (March 2016) as well as by OpenACC program developers at TU Dresden

Project Achievements: OMPT (1/3)

- **Contribution to future OpenMP specification (OpenMP 5.0)**
  - Proposal for an extensions of the OpenMP Tools (OMPT) Interface [1]
  - Contribution to the revised technical report for the OMPT Interface [2] and the integration to official technical report 4 (preview on OpenMP 5.0) [3]
  - Adding an interface for buffer traces (collaboration with RICE)
    - Asynchronous buffer handling
    - Approach similar to CUDA performance tools interface (CUPTI)
    - Reference implementation available [4]

[1] Tim Cramer, Robert Dietrich, Christian Terboven, Matthias S. Müller, Wolfgang E. Nagel: **Performance Analysis for Target Devices with the OpenMP Tools Interface**. In Proceedings of 20th IEEE International Workshop on High-level Parallel Programming Models and Supportive Environments (HIPS), pages 216-224. 2015, Hyderabad, May 2015.

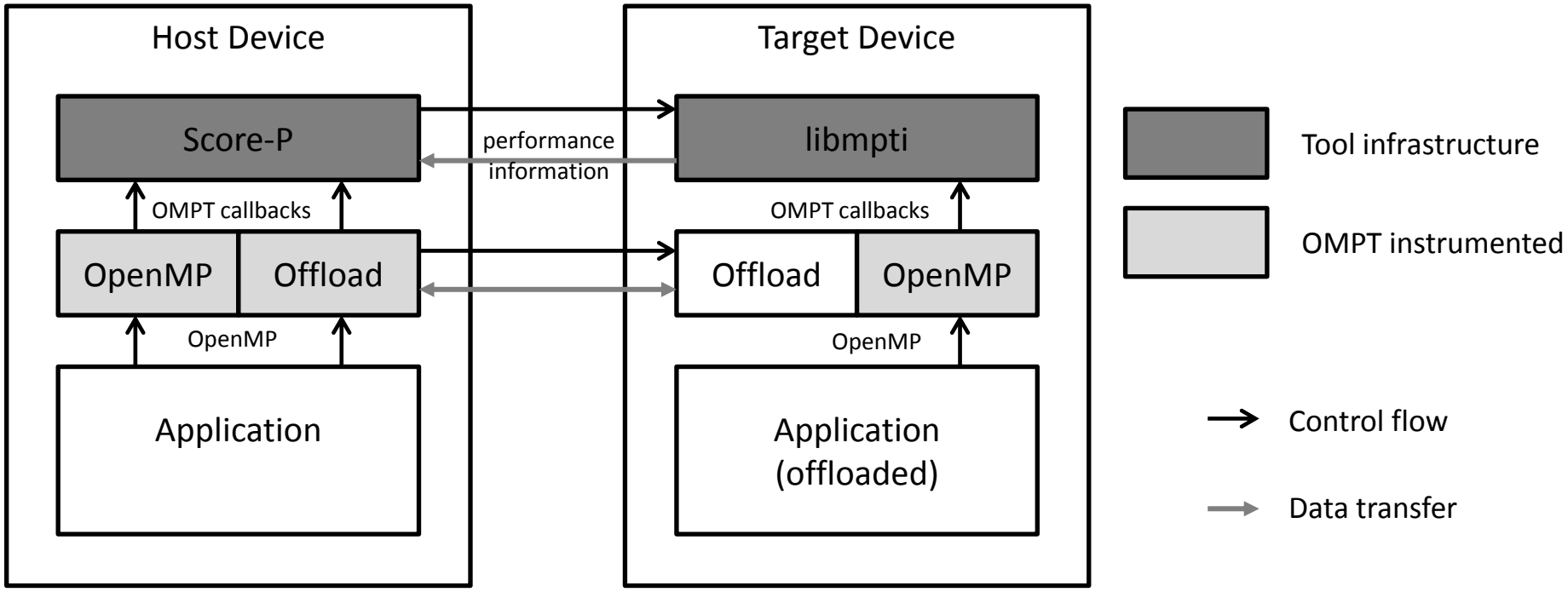
[2] Alexandre E. Eichenberger, John M. Mellor-Crummey, Martin Schulz, Nawal Copty, Jim Cownie, Tim Cramer, Robert Dietrich, Xu Liu, Eugene Loh, and Daniel Lorenz et al. **OMPT: An OpenMP Tools Application Programming Interface for Performance Analysis.** Revised OpenMP Technical Report 2, https://github.com/OpenMPToolsInterface/OMPT-Technical-Report, August 2016.

[3] http://www.openmp.org

[4] https://github.com/OpenMPToolsInterface/LLVM-openmp, October 2015

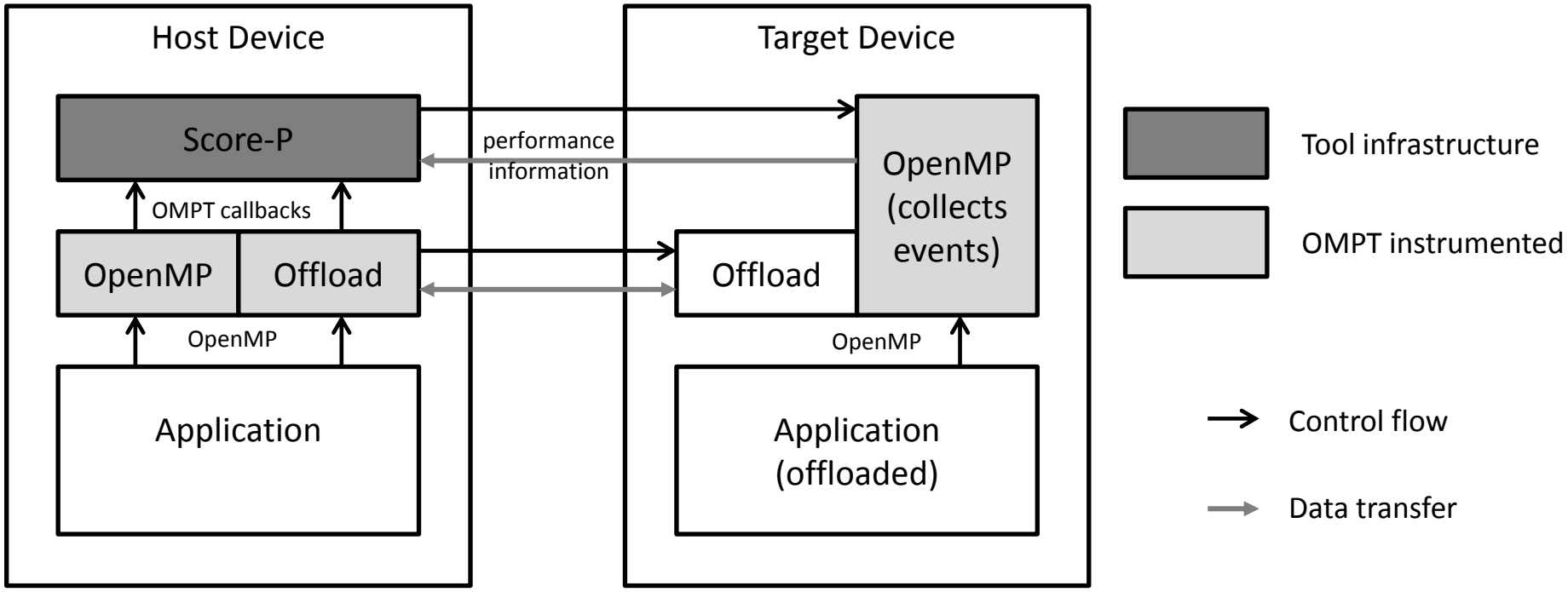
Project Achievements: OMPT (2/3)

- **OMPT extension in Score-P**
  - Low overhead
    - 1.5 % (host-sided events only)
    - 12.9 % (host- and device-sided events)
  - Additional library (e.g., libmpti) was necessary to collect data on device

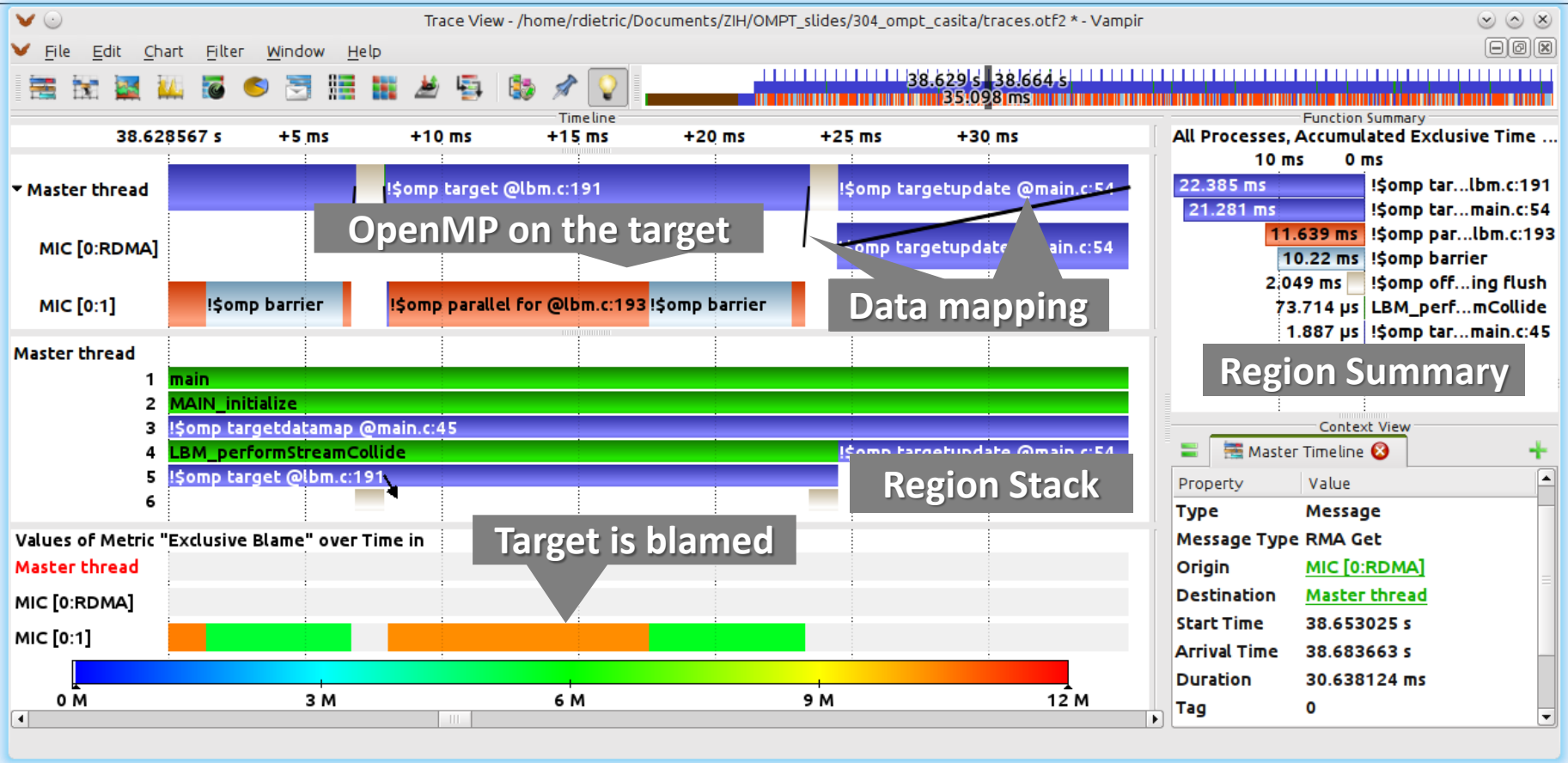


Project Achievements: OMPT (3/3)

- **OMPT extension in Score-P**
  - With buffering API
    - No additional (vendor/hardware-dependent) library required anymore
    - Device-sided events are collected within the runtime
  - Ongoing Work: Complete integration in Score-P



Vampir: Performance Data Visualization



Vampir Visualization of OpenMP 4.x offloading on Intel Xeon Phi

The trace has been generated with Score-P prototype implementation. The screenshot shows an interval in the execution of SPEC ACCEL benchmark 304.olbm (OpenMP 4.x version).

Project Achievements: Hybrid Correctness Checking

- **OMPT: Integration into MUST‘s event system**
  - Including 4.x target support
- **Memory access tracer**
  - MUST uses binary instrumentation (with Intel PIN) to detect races between host / accelerator
- **Implemented correctness checks in MUST, utilizing OMPT and PIN:**
  - Wrong threading level
    - With more than one thread *MPI_Init* is used instead of *MPI_Init_thread*
    - *MPI_THREAD_SINGLE* is used with multiple threads
    - MPI-Thread-Level < MPI_THREAD_MULTIPLE when multiple threads issue MPI-Calls concurrently
  - Multiple threads of a team passing different barriers
  - Usage of uninitialized locks
  - Deadlock using a single lock
  - Deadlock using multiple locks
  - Races between host / accelerator (clang compiler only)

Agenda

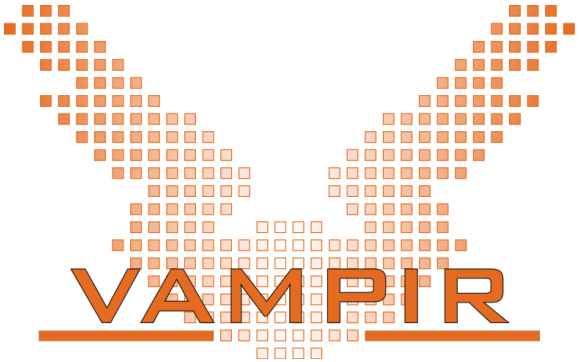
- ELP Project Goals
- ELP Achievements
- **Remaining Steps**

Remaining Steps

- **MUST**
  - Code clean-up (+documentation)
  - Provide code on webpage
- **Score-P**
  - Enhance stability and functionality of the OMPT implementation
  - Advance standardization of OMPT to enable a convenient implementation in Score-P
  - Integrate OMPT device activity buffering API
- **Support development of OpenMP 5.0 to improve OMPT**

Conclusion

- **Contributions for standard-compliant tools interfaces**
  - OpenMP
  - OpenACC
- **Tools development**
  - Performance analysis: Score-P, Vampir
  - Correctness checking: MUST



The ELP-Team Thanks You!

http://www.vi-hps.org/projects/elp/

Tim Cramer
cramer@itc.rwth-aachen.de