Runtime Correctness Checking for Emerging Programming Paradigms

Joachim Protze (protze@itc.rwth-aachen.de), Christian Terboven, Matthias S. Müller, Serge Petiton, Nahid Emad, Hitoshi Murai and Taisuke Boku

RWTH Aachen University, Germany
University of Tsukuba / RIKEN, Japan
Maison de la Simulation, France

Motivation

- Increasing concurrency in HPC needs new concepts of programming
- MPI + X is one candidate
  - Highlights the need for multilevel parallelism
  - Potentially even more levels of parallelism
- Multiple PGAS approaches were developed
  - Did any of these gain acceptance?
  - Why?

Key to success might be tool support

Portable runtime correctness checking

- Can we reuse existing tools and apply them to new programming paradigms?
- What are the common challenges?
- What are the limitations?
- How big is the effort to integrate analysis for pragma-based PGAS approach XMP into existing MPI tool MUST?
- Data race and deadlock are major threats in parallel
  - Can we define an abstract interface, that provides sufficient information to analyse arbitrary parallel paradigms?

Defect classification in parallel programs

- Defect parallel programs can
  - violate the programming standard / specification
  - result in program failures or wrong results

- How can we classify the set $A_P = \{A_1, .., A_n\}$ of possible defects for a paradigm P?
  - Design issues: D
  - Defects that can be detected statically S
  - Defects that can be detected at runtime R
  - Other defects: O

- Questions addressed in MYX:
  - Identify members of D, S, R and O
  - How to minimize D?
  - Ho to minimize O?
  - How can we improve the detection and analysis of members of S and R?

Defect classification cont.

- Tools can detect defects statically, or at runtime
  - Some defects can only be detected statically:
    - Using an integer instead of MPI_SOURCE_ANY
  - At runtime some defects are detected with local information
    - MPI-Send to rank 10, but only 8 processes
  - Some defects need distributed or global knowledge:
    - MPI-Send of 4 doubles, MPI-Recv of 4 integers
  - Some defects might need compile time information at runtime:
    - Sending an array of integers, declaring it as doubles

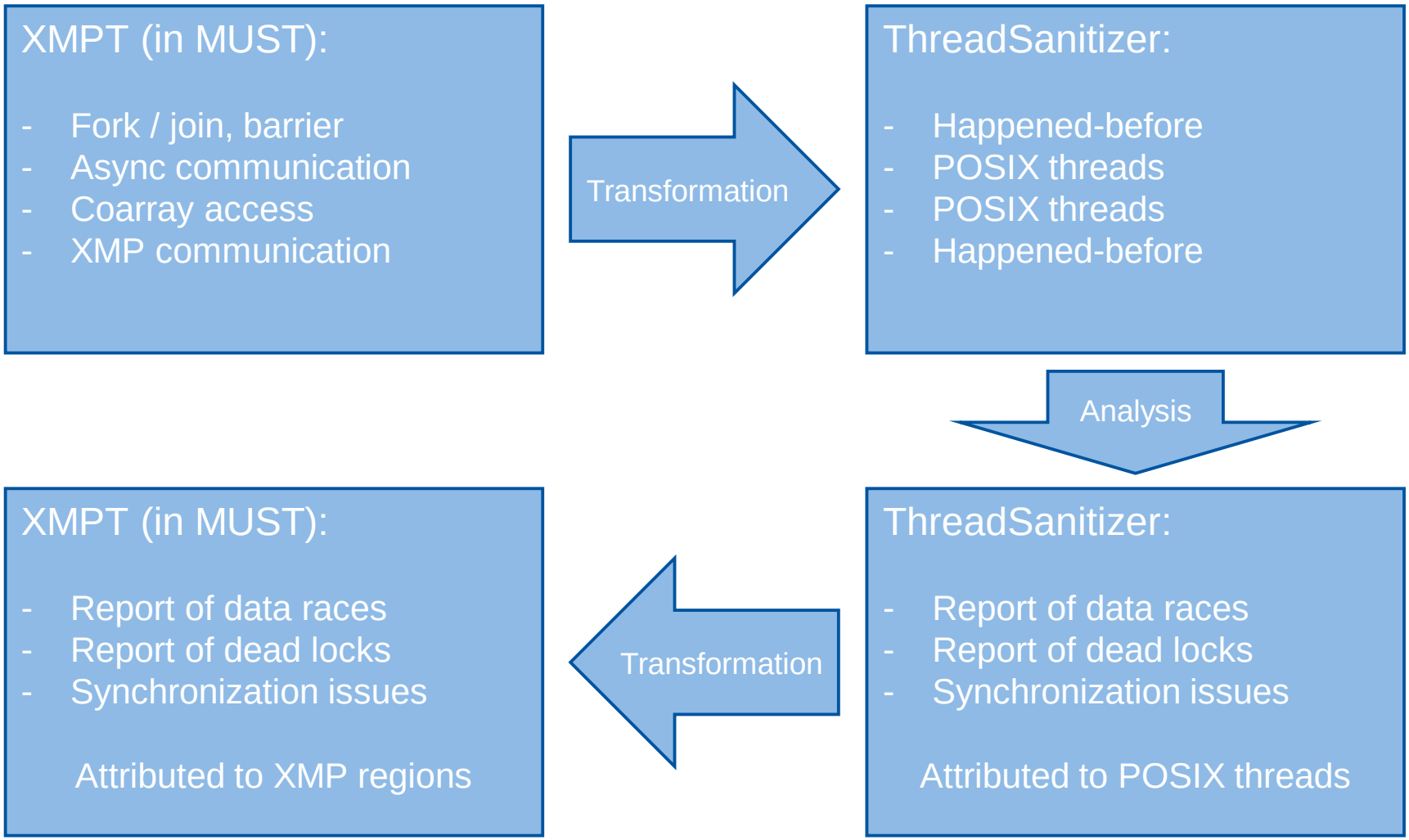
Tools interfaces

- PMPI
  - Tools interface for MPI
  - MPI spec describes wrapping of MPI functions
- OMPT
  - Tools interface for OpenMP
  - Latest OpenMP spec describes events, tool gets notification about encountered events
- XMPT
  - Tools interface for XMP, follows the specification of OMPT

XMPT

- What events are needed?
  - Events for the begin and end of XMP regions
- What information is needed?
  - Essentially, all information possibly provided to the XMP pragma
  - To allow stateless implementation of the tool, the runtime stores a tool data with scopes
  - XMP already provided functions to derive information from handles
- Is this information sufficient for performance analysis?

Example: Data race detection for XMP



Summary

- Correctness tools are important
- Tools are always steps behind the development of new languages
- Tools interface is important for easier porting of tools
- Why not use a language, that prevents the issues?
  - How many HPC codes are written in RUST?

Thank you for your attention.