# MYX

## MUST Correctness Checking for YML and XMP Programs

SPPEXA Annual Plenary Meeting 2017

**Presenter: Matthias Müller / Joachim Protze**

Project partners:
RWTH Aachen University, Germany
University of Tsukuba, Japan
Maison de la Simulation, France

IT Center
RWTH AACHEN UNIVERSITY

# Consortium

- MYX builds on successful preliminary work and collaboration:
  - FP3C: French-Japanese collaboration on YML and XMP for over 10 years
  - JST-CREST: Japanese Exascale research program supporting XMP
  - MUST: scalable correctness checking tool for MPI (and OpenMP)

## Partner from Germany (project coordinator)

- RWTH Aachen University
  IT Center and Institute for High Performance Computing
- Prof. Dr. Matthias S. Müller, Joachim Protze

## Partner from Japan

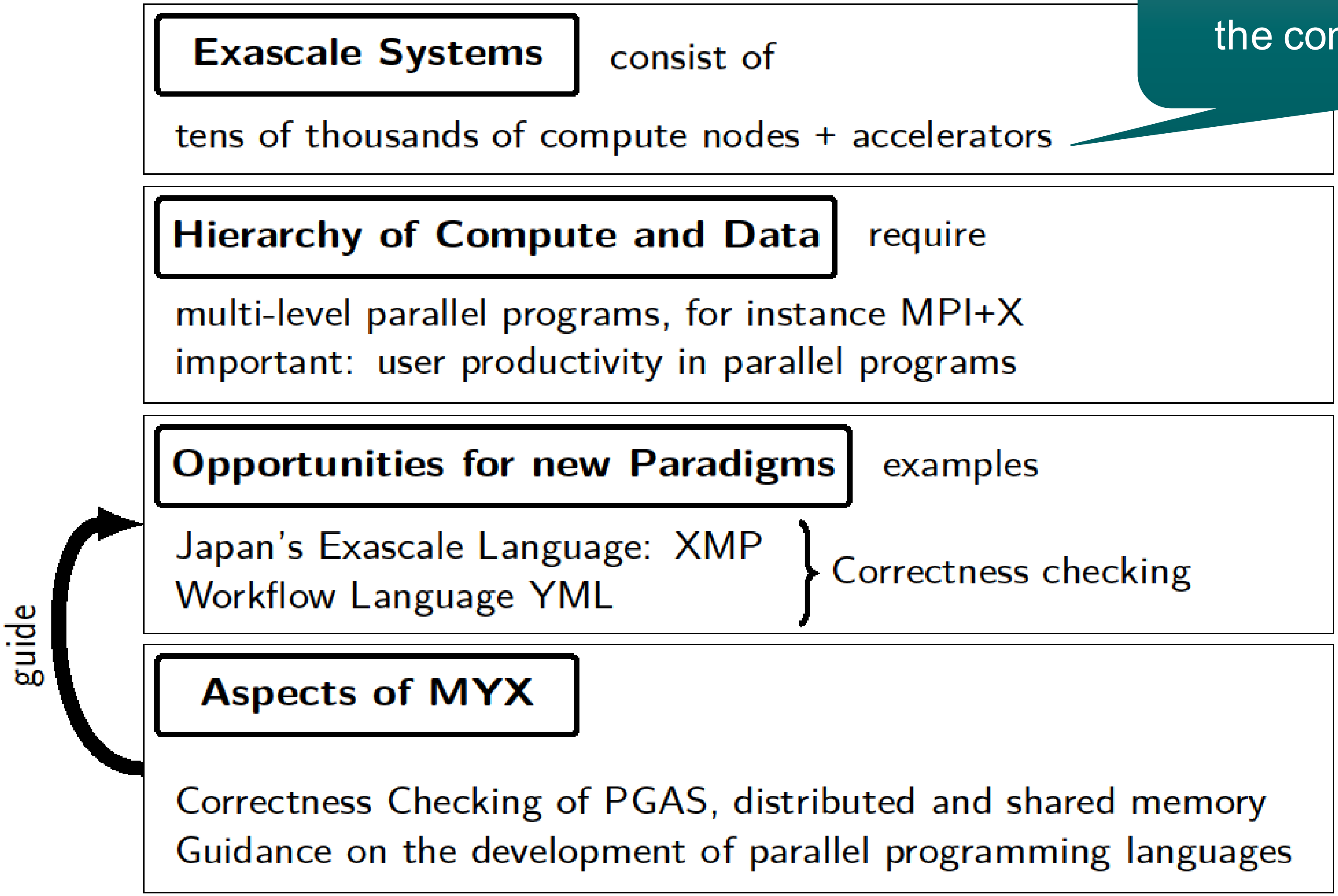
- University of Tsukuba, Center for Computational Sciences, and Advanced Institute of Computational Science, RIKEN
- Prof. Taisuke Boku, Dr. Hiroshi Murai

## Partner from France

- Maison de la Simulation
- Prof. Serge Petiton, Prof. Nahid Emad

# Motivation

Jeff Vetter „We reached the complexity wall“



**Exascale Systems** consist of

tens of thousands of compute nodes + accelerators

**Hierarchy of Compute and Data** require

multi-level parallel programs, for instance MPI+X
important: user productivity in parallel programs

**Opportunities for new Paradigms** examples

Japan's Exascale Language: XMP
Workflow Language YML

} Correctness checking

**Aspects of MYX**

Correctness Checking of PGAS, distributed and shared memory
Guidance on the development of parallel programming languages

# Defect classification in parallel programs

- A programming paradigms follows a certain design philosophy and is created within a specific design space.

- Within a parallel programming paradigm a programmer can write faulty programs containing defects that violate the programming standard or result in program failures.

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

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

## Initial results for defect classification

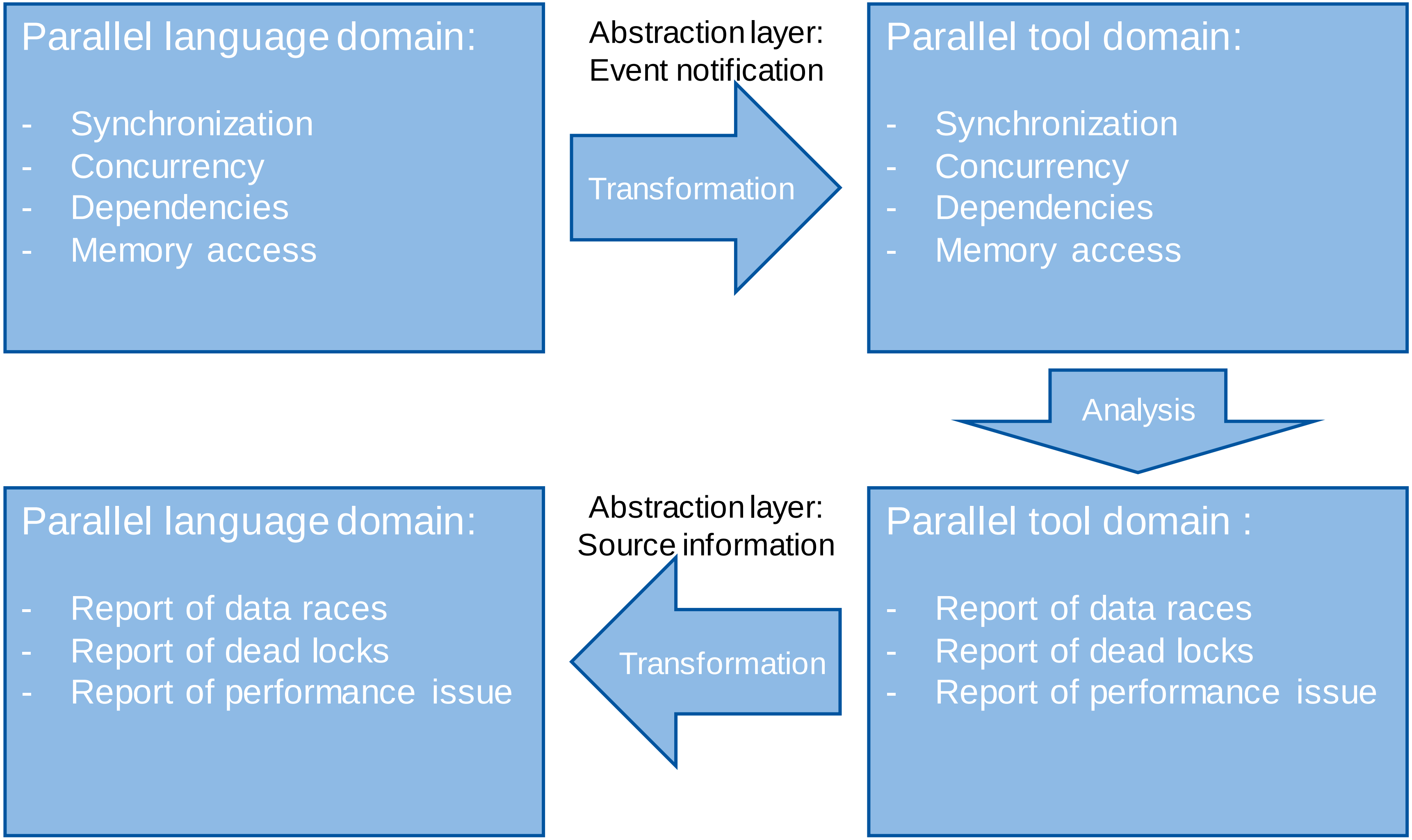
- Degree of non-determinism:
  - Strict rules minimize „design issues“ → detection of issues
  - Loose rules provide more freedom in application / algorithm

- More constraints → issues can be detected at compiletime/runtime
  - Classification of constraints as static, dynamic or global properties

  - Exercised for XMP:
    - Static constraints can be analyzed at compile time
    - Dynamic constraints can be analyzed at runtime
    - Global constraints can be analyzed at runtime with global knowledge

- Memory model:
  - How is synchronization defined?
  - What is the intended behavior for unsynchronized memory access?

# Abstract data / controlflow for correctness tools



Parallel language domain:

- Synchronization
- Concurrency
- Dependencies
- Memory access

Abstraction layer:
Event notification

Transformation

Parallel tool domain:

- Synchronization
- Concurrency
- Dependencies
- Memory access

Analysis

Parallel tool domain :

- Report of data races
- Report of dead locks
- Report of performance issue

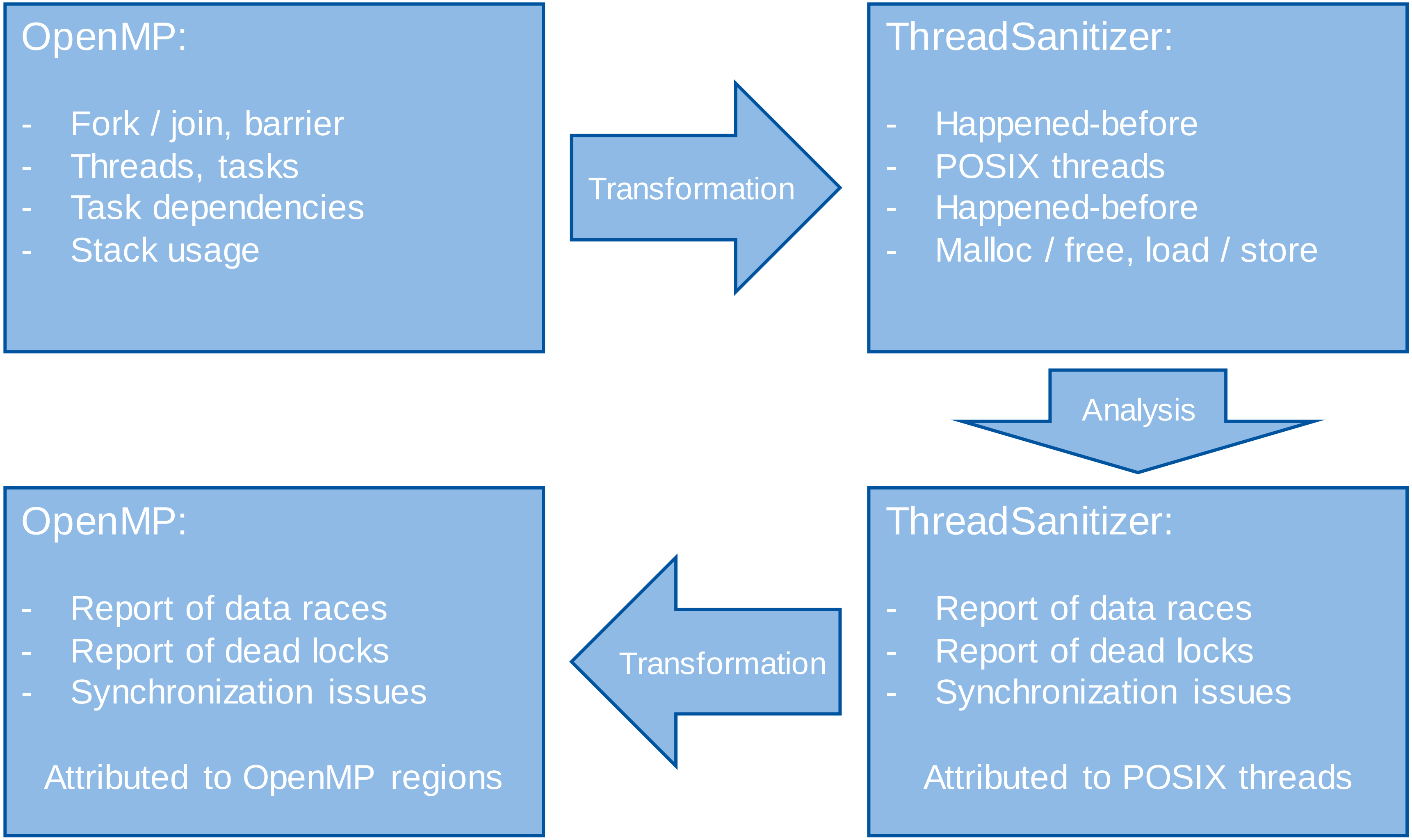
Abstraction layer:
Source information

Transformation

Parallel language domain:

- Report of data races
- Report of dead locks
- Report of performance issue

# Example: Correctness detection for OpenMP



OpenMP:

- Fork / join, barrier
- Threads, tasks
- Task dependencies
- Stack usage

Transformation

ThreadSanitizer:

- Happened-before
- POSIX threads
- Happened-before
- Malloc / free, load / store

Analysis

ThreadSanitizer:

- Report of data races
- Report of dead locks
- Synchronization issues

Attributed to POSIX threads

Transformation

OpenMP:

- Report of data races
- Report of dead locks
- Synchronization issues

Attributed to OpenMP regions

# XMPT: Tools Interface for XMP

- XMP is a pragma based PGAS language

- XMPT: Tools interface modeled like OMPT for OpenMP
  - Events for all XMP pragmas
    - How to express the abstract semantics of XMP spec?

  - Enables dynamic runtime correctness checks for XMP
    - What are the necessary attributes?

  - Enables data race detection

  - Also enables performance analysis for XMP (collaboration with ScoreP team at FZJ)

  - Declares a reliable interface between XMP runtime and correctness tool.

# YML: Data- and Controlflow Description

- YML as a scheduler:
  - Enable YML to start specific tasks under tool control
  - Interface between scheduler and tool:
    - Resource requirements of the tool?
    - Information provided to the tool

- Verify Yvette programs:
  - Mainly static analysis of Yvette programs
    - Races on signals, races on data
    - Mismatch in argument types

- Verify runtime implementation of YML:
  - Challenge: YML spawns MPI processes at runtime

# Workshops in Japan

## First French-Japanese-German Workshop on Programming and Computing for Exascale and beyond

- Wednesday, April 5th, 2017, 9:30 - 18:15
- French Embassy, Tokyo
- Organized by RIKEN AICS and Maison de la Simulation

## SPPEXA Workshop in Japan 2017

- Thursday, April 6th, 2017, 9:00 – 18:00
- Organized by University of Tsukuba

- Attendees for both workshops mainly from MYX, DASH and ESSEX

# Vielen Dank für ihre Aufmerksamkeit