Overview of MYX – results and perspectives

MYX == MUST correctness checking for YML and XMP programs

SPPEXA Workshop on Parallel Programming Models – Productivity and Applications for Exascale and Beyond

Versailles, France, Mar. 21, 2019

Presenter: Matthias S. Müller

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, Dr. Christian Terboven

Partner from Japan

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

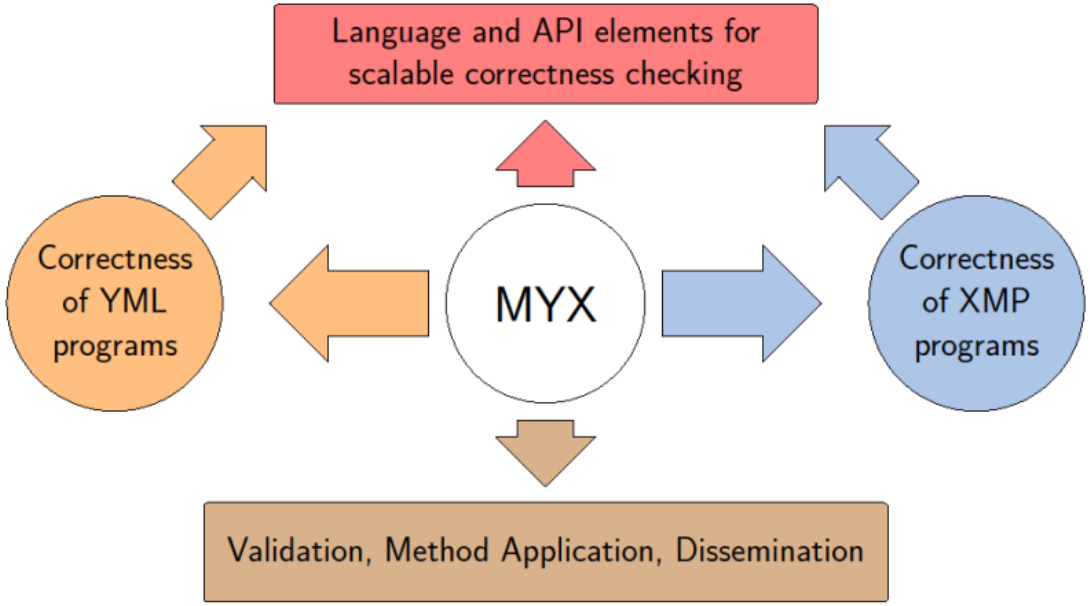
Partner from France

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

Research Challenges and Project Results

- The more parallelism expressed, the higher the chance of errors being made
- Time of programming error search and fix: productivity loss!
  - Automatic correctness checking may be used to avoid that

- MYX objectives are
  - enable productivity improvements by means of scalable correctness checking
  - of YML- and XMP-programs
    - XMP: PGAS, with both global-view and local-view
    - YML: graph of components language
  - guide the development of future programming models



Do we need a safety net in HPC?

Source: David Madison, gettyimages

Challenges in runtime correctness analysis for parallel programming

- Point-to-point communication
  - Message matching needs distributed information (send+recv) — MPI ✓
- Collective communication
  - Scalability of collective communication — MPI, XMP ✓
- One-sided communication
  - Analysis of data races needs VC in distributed communication — MPI, XMP ✓
  - No scalable solution yet ✗
- Multi-paradigm parallelism (MPI+X / XMP+X)
  - Analysis of conflicting, concurrent communication / API calls MPI+OpenMP ✓
  - Concrete message matching in MT is undecidable just by observation ✗
  - Data races with memory access in base language ✗
  - Potential conflict with parallelism in base languages ✗

Approach taken in MYX for Correctness Checking

- Automatic runtime correctness checking avoids the state explosion problem of model checking

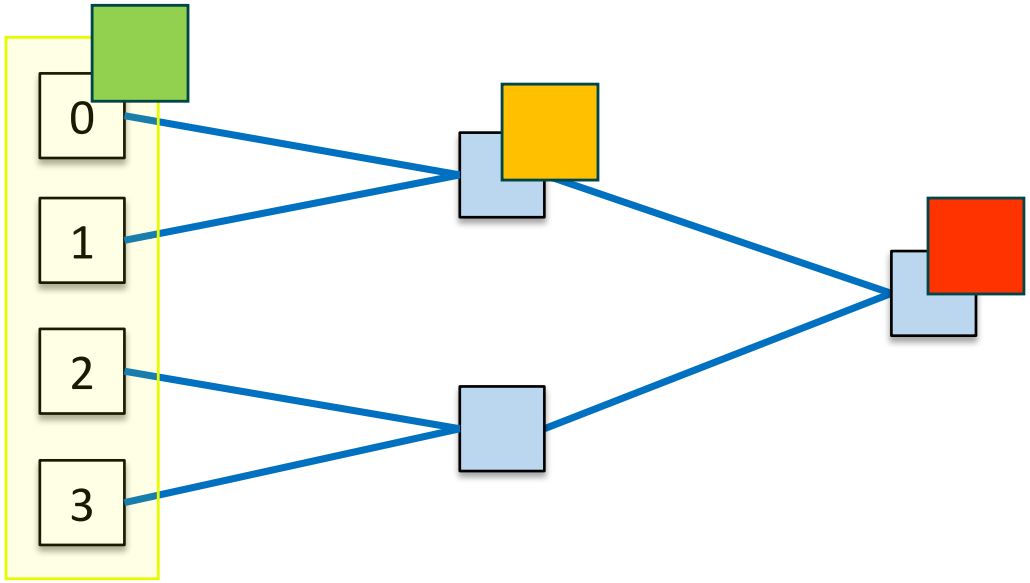
- MUST: scalable correctness checking of MPI, OpenMP and hybrid programs
  - PnMPI used to intercept MPI calls

- Current use case example for MPI
  - MPI lacks type safety in the functions to send and receive messages
  - This may lead to invalid data, if receiver expects different data from what sender sent
  - Runtime analysis can find actual and potential instances of this problem

Distributed Agent-based Runtime Correctness Analysis

- Correctness analysis is done by specialized agents
- Locally whenever applicable
- Distributed for scalability
- Centralized only for global knowledge

MUST

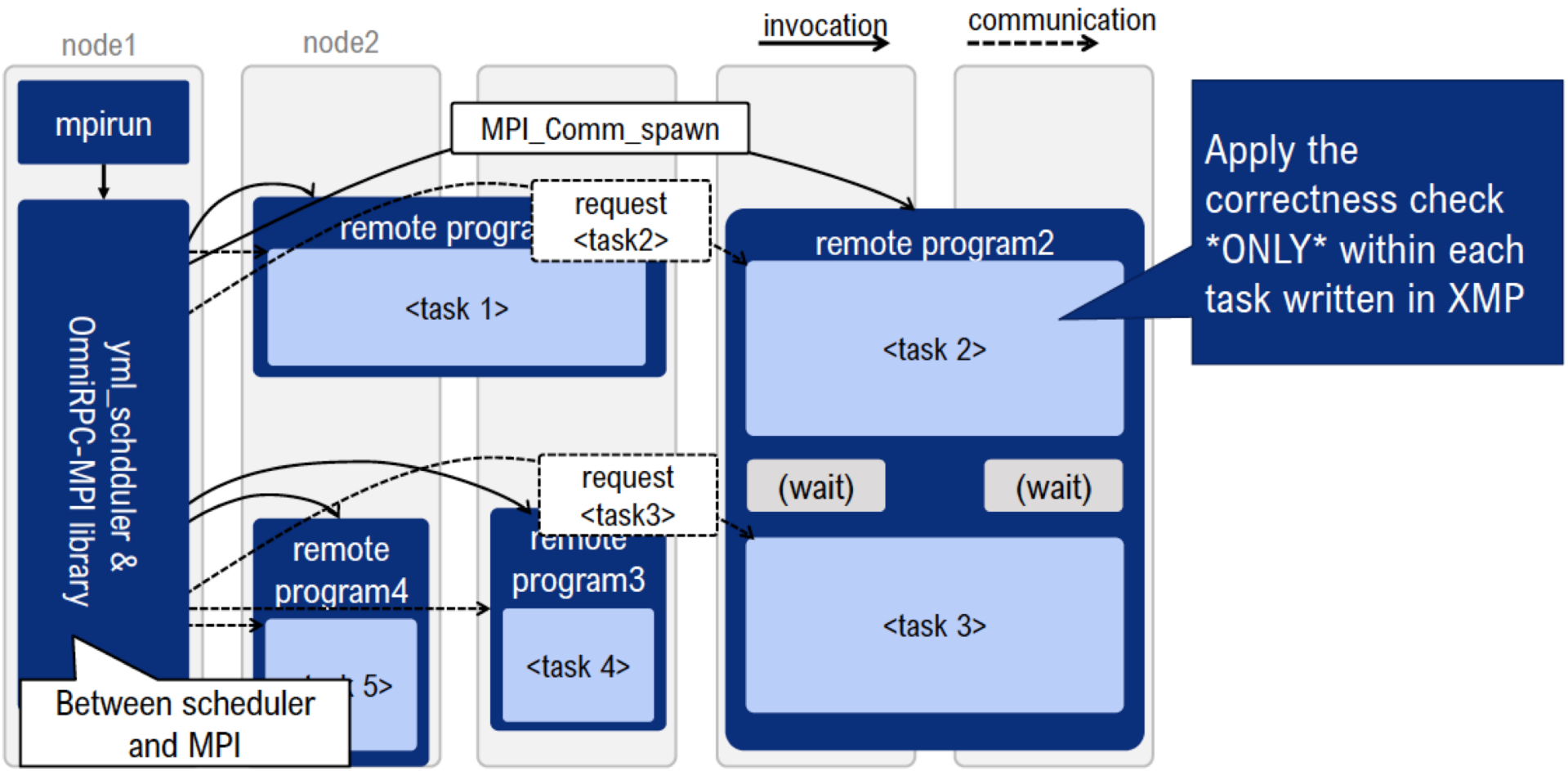


Challenge I: Multi-Paradigm Programming

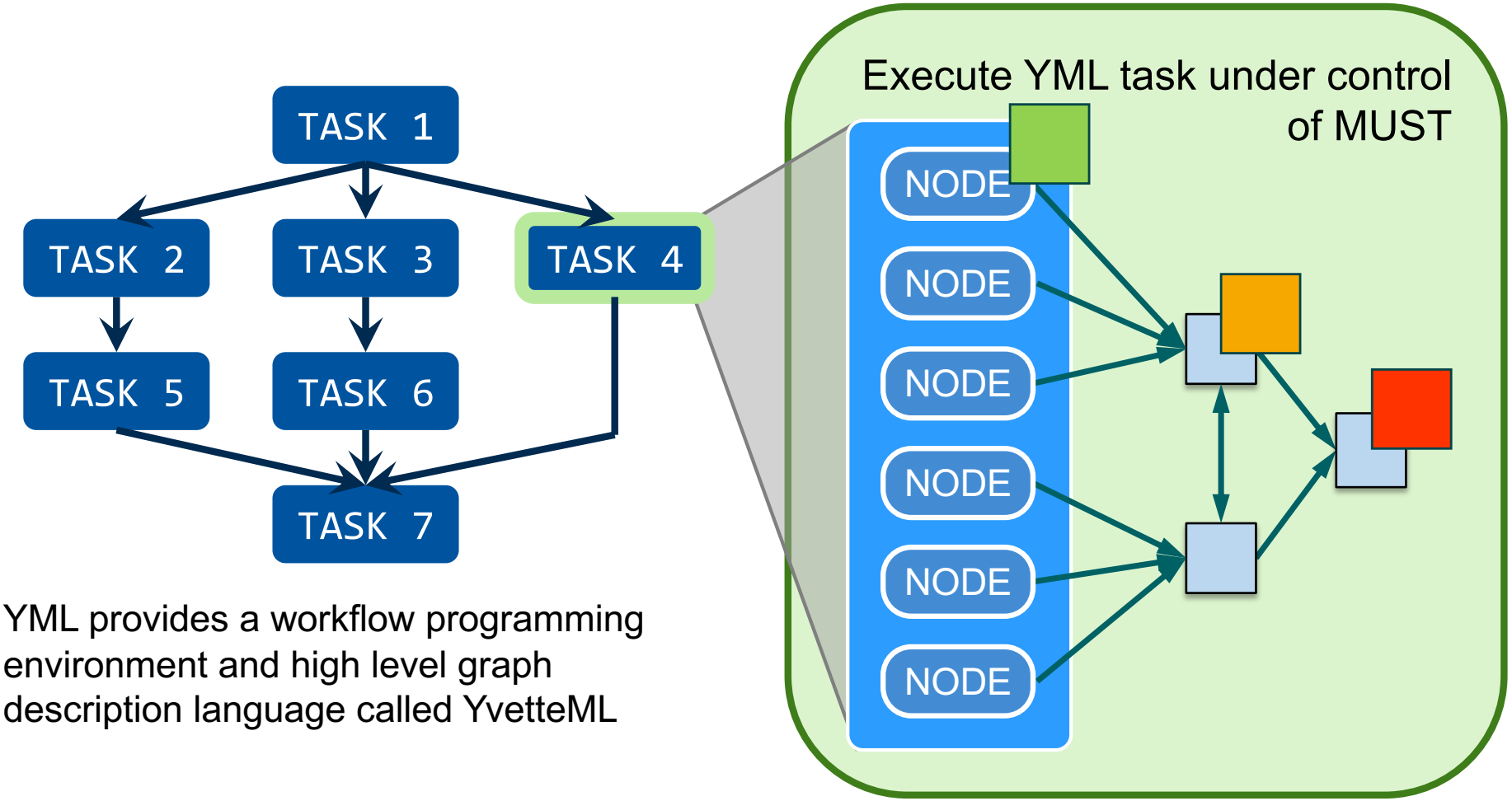
Mixing YMP and XMP

IT Center | RWTH AACHEN UNIVERSITY

Correctness checking for YML+XMP programs

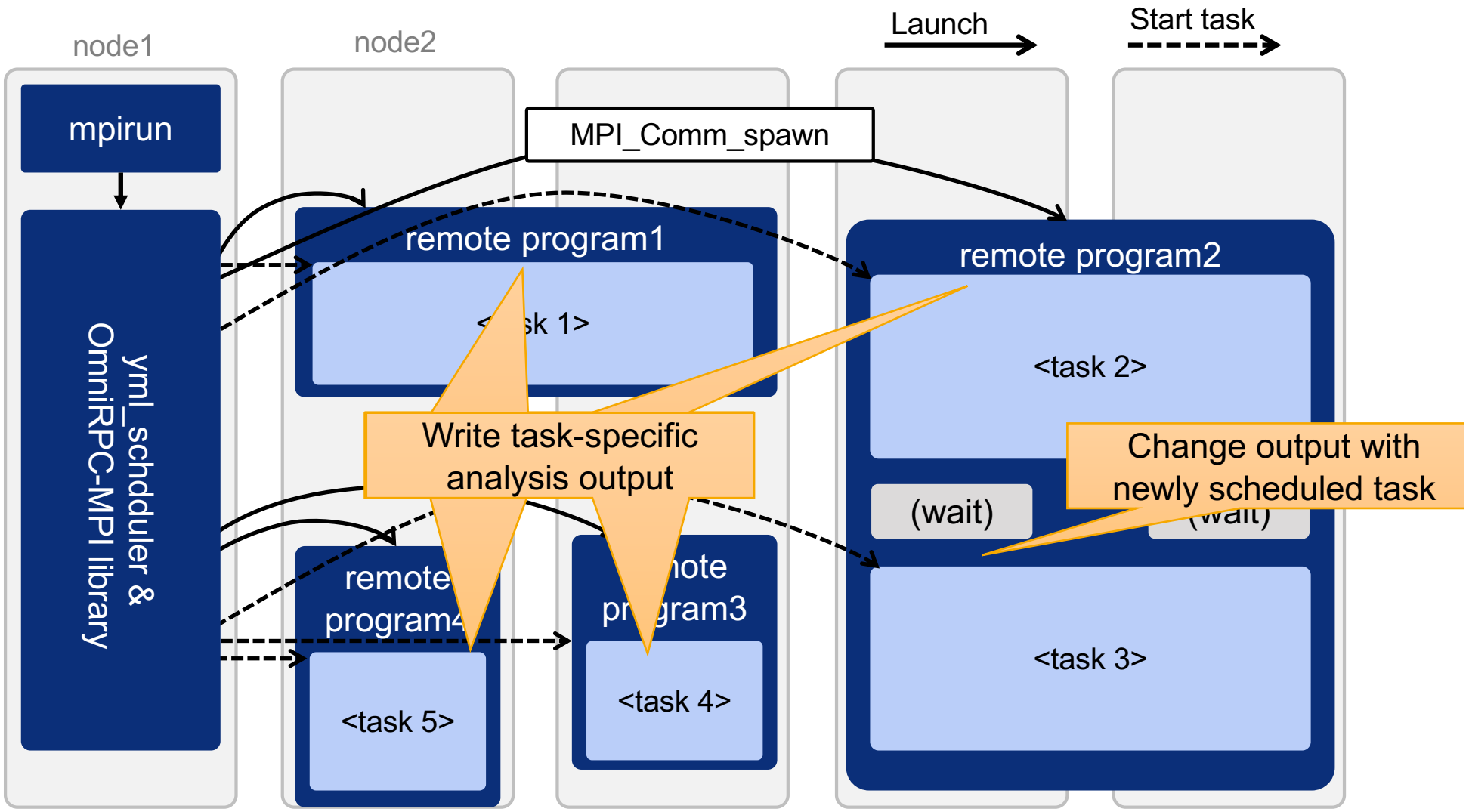


MUST analysis of YML tasks



YML provides a workflow programming environment and high level graph description language called YvetteML

New interface in MUST intended to dynamically change output



YML+XMP+MUST execution by Miwako Tsuji

MPI_Pcontrol interface for MUST

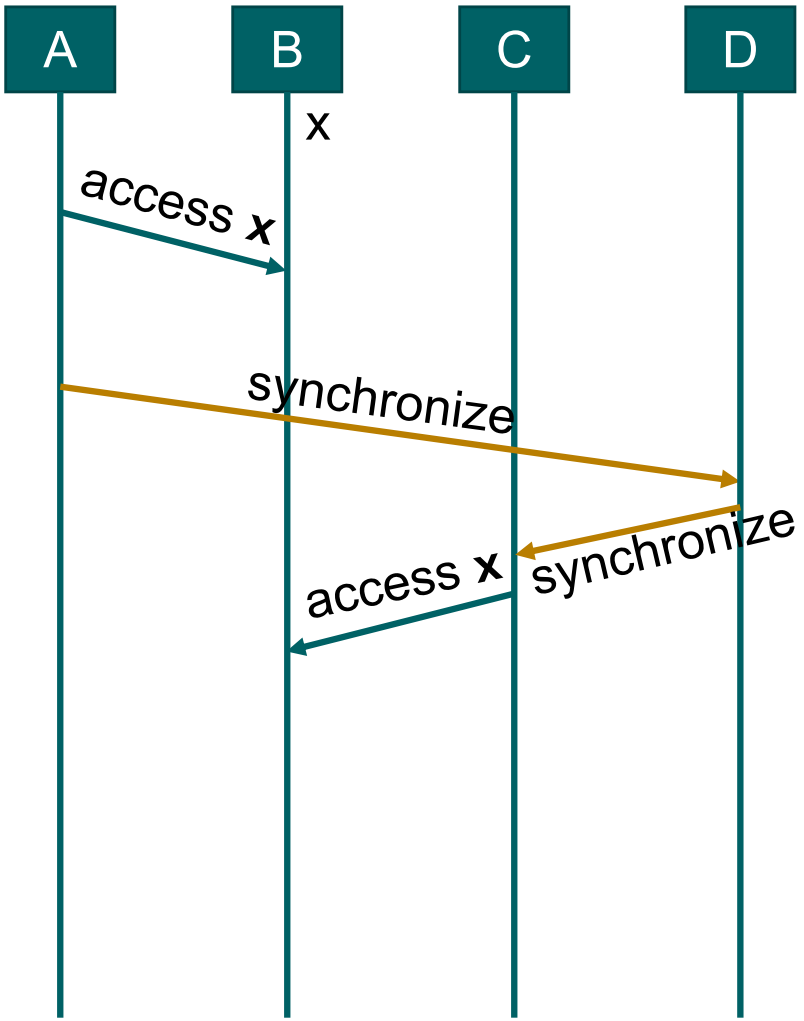
- Initial idea: provide a way for YML to specify new output file for each task
  - Multiple tasks in a single “program” execution
  - ➢ New file for each task
- Technical challenge: ensure that analysis result goes into the right file
  - ➢ MPI_Pcontrol call is “collective” for the application
- Semantical benefit: Allows to perform additional analysis
  - ➢ (Non-blocking) communication should not be open across this point
- Future work: what additional parallelism assertions can we define?
  - Assert parallel barrier
  - Assert no open communication
  - Assert no parallel handle

Challenge: One sided communication

Supporting XMP as a PGAS language

Challenges of Data Race Detection for One-Sided Communication

- Process B owns „**x**“
  - analyze accesses there
  - A and C send information about access to B
- Did access by A *happen before* access by C?
  - Identify the synchronization path
- Treatment of access that is made directly from base language
  - Thread sanitizer
- Use vector clocks?
  - Not scalable (as we have seen for MPI one-sided)



Applicability for XMP Coarray

- XMP Coarray do not allow local Coarray access beyond API usage
  - No conflict of API and non-API memory access possible
  - Full memory access tracking not necessary

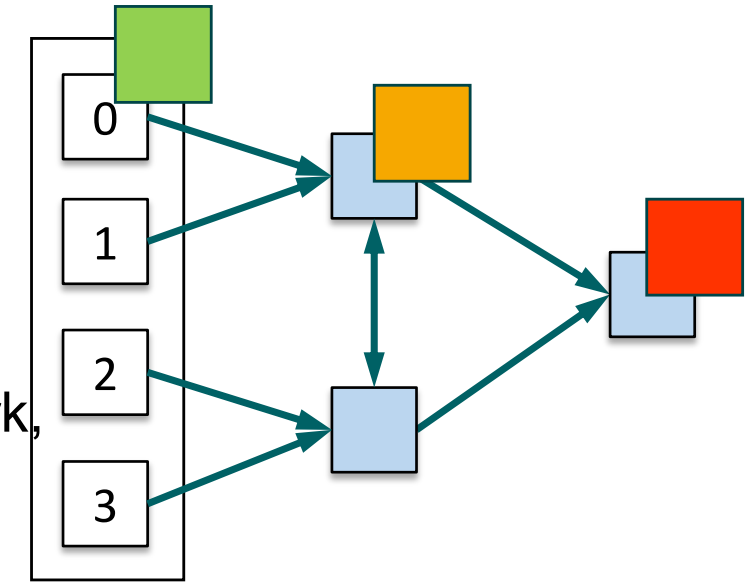
- Analysis based on:
  - Tracking API memory accesses
  - Tracking API synchronization information

- Information is provided by the XMPT interface
  - Callbacks for coarray memory access
  - Callbacks for XMP synchronization

Due to the API and semantics of XMP the runtime analysis can be implemented much more efficiently (compared to MPI)

Execute the Data Race Detection in the TBON

- In XMP we have no conflicts with base-language accesses
  - No need to track and correlate with loads and stores of the application
- Idea: integrate with distributed DL detection
  - Already tracks synchronization semantics
- MUST operates as a distributed agent network, organized in a tree-based overlay network (TBON)
- Benefit of approach:
  - Less memory consumption on application threads
  - Fewer synchronization points with application threads



Challenge: Monitoring application execution in applications using multiple-paradigms

Common tools interfaces for OpenMP and XMP

IT Center | RWTH AACHEN UNIVERSITY

History of Tools Interfaces

- **1996: PMPI**
  - **Profiling tools interface in MPI 1.0**

- 2002: POMP (OpenMP profiling) proposal - rejected
- 2004: DMPL (OpenMP debugging) proposal - rejected
- 2005: An OpenMP runtime API for profiling (Sun proposal) - rejected
- **2012: MPI_T query interface in MPI 3.0**
- 2013: first draft of OMPT+OMPD (OpenMP profiling + debugging)
- 2015 - : Pushing OMPT to support runtime correctness
- **2018: OMPT and OMPD became part of OpenMP 5.0 standard**

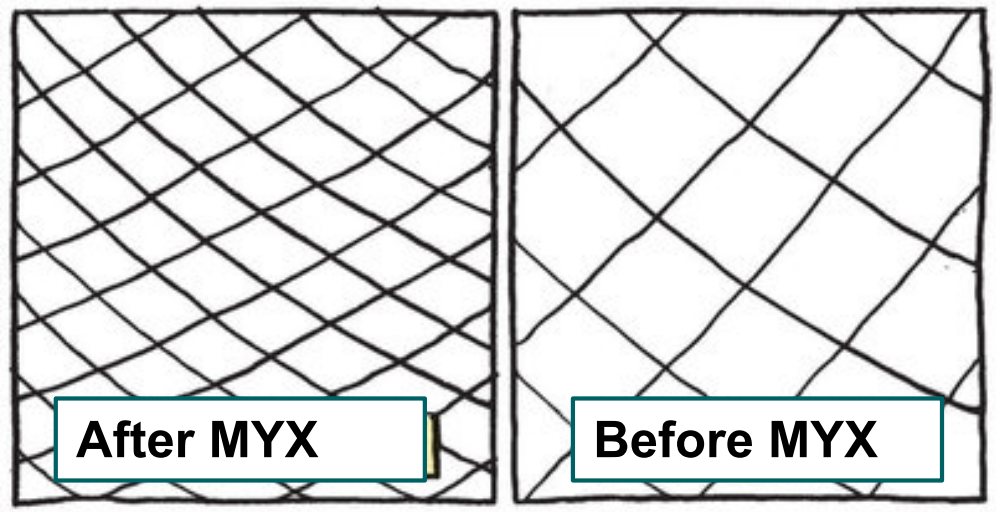
- 2016 -: Work on XMPT interface in MYX
- 2017-: discussion for MPI_T event interface to be included in 4.0
- XXXX-YYY: work on MPI_T to be included in MPI 5.0
- 2019: XMPT will be proposed for inclusion to XMP specification

Summary and outlook

- Improved programming models and environments are important for Exascale and beyond.

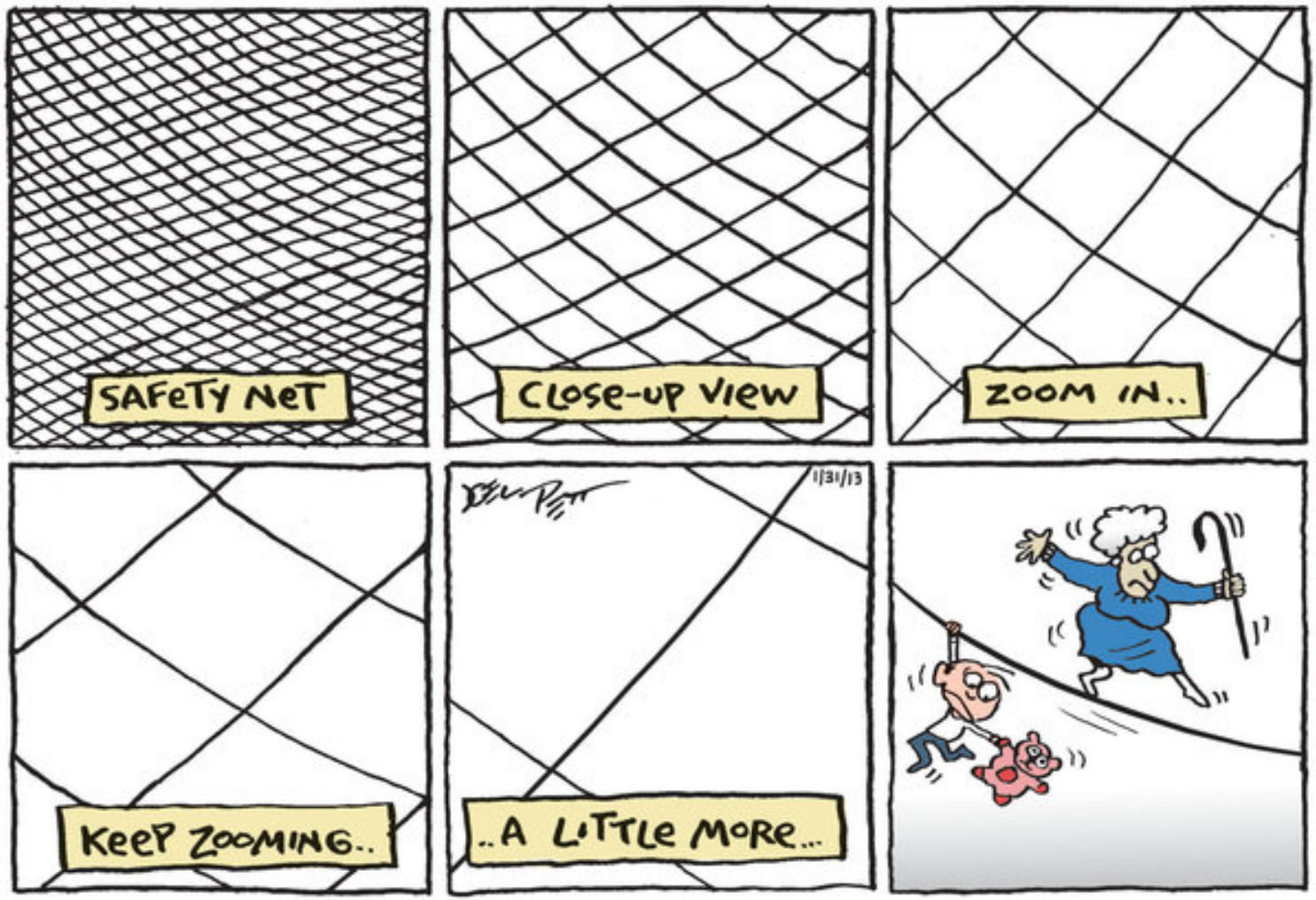
- **Project goals and achievements of MYX**
  - **Applied correctness checking to XMP and YML**
  - **Integration of message passing, distributed shared memory and workflow into an integrated correctness model**
  - **Transfer of tools interface from OpenMP to XMP**
  - **Improve existing parallel programming paradigms**

Safety net before and after the MYX project …



Source: Joel Pett in the Lexington Herald-Leader.
http://2.bp.blogspot.com/-ERKbUp4-7XI/URBmWgfkpAI/AAAAAAAAuHs/SU3bsu7Aop4/s1600/safety+net.jpg

Safety net: the current approach is good, but they are still holes …



Source: Joel Pett in the Lexington Herald-Leader.
http://2.bp.blogspot.com/-ERKbUp4-7XI/URBmWgfkpAI/AAAAAAAAuHs/SU3bsu7Aop4/s1600/safety+net.jpg

MUST publications performed during

- *T. Hilbrich, M. Weber, J. Protze, B. R. de Supinski, and W. E. Nagel*. **Runtime correctness analysis of MPI-3 nonblocking collectives.** (EuroMPI 2016)
- *J. Protze, J. Hahnfeld, D. H. Ahn, M. Schulz, and M. S. Müller*. **OpenMP Tools Interface: Synchronization Information for Data Race Detection**. (IWOMP '17)
- *J. Protze, C. Terboven, M. S. Müller, S. G. Petiton, N. Emad, H. Murai, and T. Boku*. **Runtime Correctness Checking for Emerging Programming Paradigms**. (CORRECTNESS@SC 2017)
- *A. Hück, J.-P. Lehr, S. Kreutzer, J. Protze, C. Terboven, C. Bischof, and M. S. Müller*. **Compiler-aided type tracking for correctness checking of mpi applications**. (CORRECTNESS@SC 2018)
- *J. Protze, M. Schulz, D. H. Ahn, and M. S. Müller*. **Thread-local concurrency: a technique to handle data race detection at programming model abstraction.** (HPDC 2018)