RWTH AACHEN UNIVERSITY

Maison de la Simulation

SPPEXA
An SPPEXA Project funded by DFG, ANR & JST

筑波大学
University of Tsukuba

MYX

MUST Correctness Checking for YML and XMP Programs

Motivation

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

guide

Aspects of MYX

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

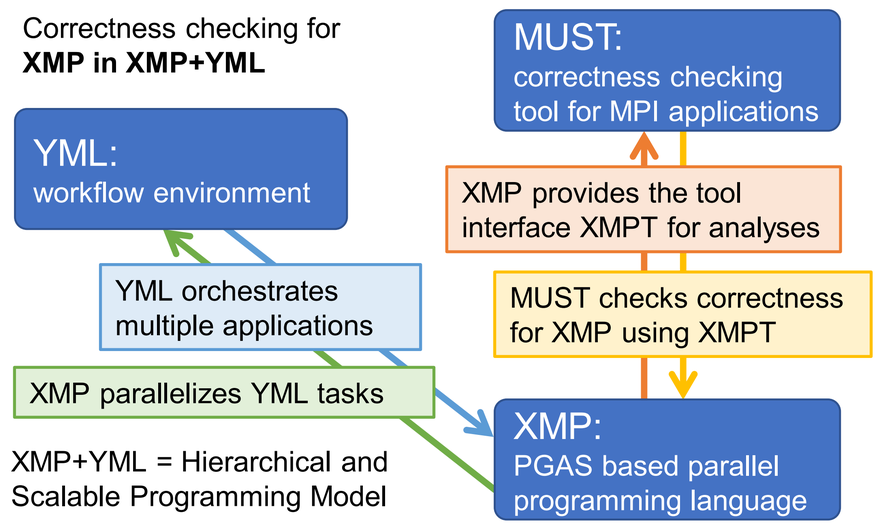
XMP, YML and MUST

YvetteML (YML): graph of components language to express parallelism at high level

- Any parallel program can be encapsulated as a component for YML
- Strong support to design reusable components

MUST: scalable runtime correctness checking for parallel programs

- Scalability achieved by distributed analysis in a tree-based overlay network
- MUST analysis is applied on a per component basis to YML programs



Tools Interface for XMP

XMP: PGAS, distributed memory with global-view and local-view

- XMPT: XMP tools interface modeled after the OpenMP tools interface
- XMPT events: The XMP runtime notifies an interested tool about any encountered directive
- Definition of XMPT will finally be included in XMP specification

```
#pragma xmp nodes p[*]
int main(void)
{
#pragma xmp task on p[0:3]          --> task-begin-event
   {
#pragma xmp barrier                 --> task-end-event, barrier-events
   }
#pragma xmp barrier                 --> barrier-events
#pragma xmp task on p[0]            --> task-begin-event
   {
     printf("PASS\n");              --> task-end-event
   }
   return 0;
}
```

Fig.1: XMPT events provided for an example XMP program

XMPT events are designed to be used by correctness as well as performance analysis.

Correctness analysis:

- enable productivity improvements in programming for Exascale by means of scalable correctness checking of XMP-programs
- one-sided communication, global data access
- analyze the semantics expressed by the XMP control flow directives and identify semantic issues

Performance analysis:

- a tracing tool like ScoreP can log event information and use the data to visualize the performance of execution.
- events provide source code reference
- tool can bind own objects on context

Consortium

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

Correctness Checking

Parallel Programs can exhibit a wide range of errors

- from simple mistakes (e.g. invalid API arguments) to complex errors (deadlocks)

Runtime error detection is most practical and improves programmer productivity

- deployment can be transparent to the user
- no exponential analysis time of model checking

MUST correctness checker

- MPI profiling interface: PnMPI
- OpenMP tools interface: OMPT
- capable of tracing any MPI communication from any middleware runtime

Extension in the scope of MYX

- support for PGAS and workflow model by means of supporting one-sided MPI communication and using XMPT as source of information for runtime correctness analysis

Challenge Addressed by XMP+YML

Increase core counts challenge purely message-based parallel models Hybrid and PGAS models overcome scalability limits, such as message rates

- XMP: PGAS, distributed memory with global-view and local-view
- YML: graph of components language, allows for the expression of parallelism at highest level

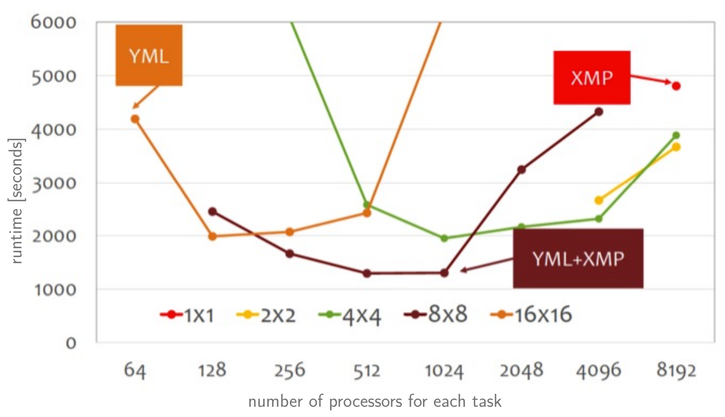


Fig.2: Combination of models increases scalability over the use of a single model for BGJ code (developed with TOTAL) on K-Computer (32 k × 32 k matrix size)

The more parallelism is 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.

Prof. Dr. Matthias S. Müller, Joachim Protze
Dr. Christian Terboven, Simon Schwitanski
RWTH Aachen University, IT Center and Chair for High-Performance Computing, Germany
mueller@itc.rwth-aachen.de

Prof. Dr. Serge G. Petiton, Prof. Dr. Nahid Emad
Maison de la Simulation and
LILLE 1 University, France
serge.petiton@univ-lille1.fr

Prof. Dr. Taisuke Boku, Dr. Hitoshi Murai
Dr. Miwako Tsuji
Centers for Computational Sciences at University of Tsukuba and RIKEN, Japan
taisuke@cs.tsukuba.ac.jp