YML: Language, framework and applications

T.DUFAUD

LI-PaRAD/Maison de la Simulation

3rd Workshop on Parallel Programming Models - Productivity and Applications, Aachen

SPPEXA - 2018/03/15

Plan

YML from user's point of view

YML architecture and interaction with middleware

Application

1 YML from user's point of view

2 YML architecture and interaction with middleware

3 Application

YML from user's point of view

YML architecture and interaction with middleware

Application

YML a parallel programming environment

- http://yml.prism.uvsq.fr/
- Component approach for re-usability, and maintainability
- A high-level language to **express coarse grain parallelism**, portability, re-usability, and maintainability
- Multilevel parallel programming for performances (YML-XMP)
  - coarse grain parallelism expressed by a graph of tasks (Yvette)
  - fine grain parallelism expressed in component (XMP)
- **Ease of use**
- **separation** of Computation, Data and Communication

Summary

- Development process using YML
- Remarks on 2-level MPI programming
- **How YML provides information to middleware ?**
- Experiments on various environment.

YML from user's point of view

YML architecture and interaction with middleware

Application

YML from user's point of view

BGJ as an example

YML from user's point of view
YML architecture and interaction with middleware
Application

- **Block Gauss-Jordan Inversion**
- Successive elimination to obtain $B = MA$ diagonal.
- Algorithm is written by block
- **Coarse grain:** algorithm written with YML
- **Fine grain:** linear algebra library written with XMP

$$A_k = E_{k-1}P_{k-1}\dots E_2P_2E_1P_1A_1 = \begin{pmatrix} a^k_{1,1} & & & a^k_{1,k} & \dots & a^k_{1,n} \\ & a^k_{2,2} & & a^k_{2,k} & \dots & a^k_{2,n} \\ & & \ddots & \vdots & & \vdots \\ & & & a^k_{k,k} & \dots & a^k_{k,n} \\ & & & \vdots & & \vdots \\ & & & a^k_{n,k} & \dots & a^k_{n,n} \end{pmatrix}$$

Algorithm 1 The block-based Gauss-Jordan matrix inversion

Input: A (partitioned into $p \times p$ blocks)
Output: $B = A^{-1}$
For $k = 0$ to $p-1$
  $B_{kk} = {A^{-1}}_{kk}$
  For $i = k+1$ to $p-1$ **(1)**
    $A_{ki} = B_{kk} \times A_{ki}$
  End For
  For $i = 0$ to $p-1$ **(2)**
    If $(i \neq k)$
      $B_{ik} = -A_{ik} \times B_{kk}$
    End If
    If $(i < k)$
      $B_{ki} = B_{kk} \times B_{ki}$
    End If
  End For
  For $i = 0$ to $p-1$ **(3)**
    If $(i \neq k)$
      For $j = k+1$ to $p-1$
        $A_{ij} = A_{ij} - A_{ik} \times A_{kj}$
      End For
      For $j = 0$ to $k-1$
        $B_{ij} = B_{ij} - A_{ik} \times B_{kj}$
      End For
    End If
  End For
End For

Figure : Block Gauss-Jordan Algorithm, M. Hugues *et.al.*

YML from user's point of view
YML architecture and interaction with middleware
Application

Developing an application with YML

1. Create tasks defining **components** (using XML language)
2. Write an application with a **graph of tasks with Yvette language**
3. Execute the application in distributed environment

1. Task definition

A task is a service define by:

- **interface**: «abstract» component
  - Input / Output data
- **realization**: «implementation» component
  - C/C++/XMP-C/XMP-FORTRAN
  - libraries etc.

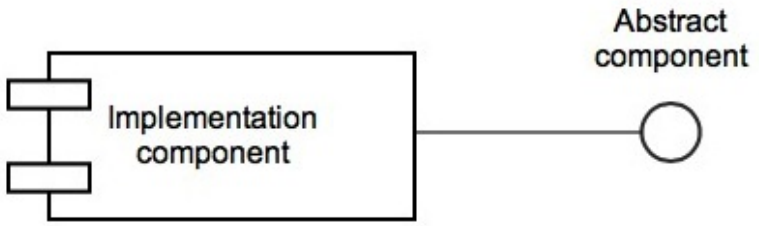


Figure : A service

Component approach

- modularity
- reusability

YML from user's point of view
YML architecture and interaction with middleware
Application

Developing an application with YML

1. Create tasks defining **components** (using XML language)
2. Write an application with a **graph of tasks with Yvette language**
3. Execute the application in distributed environment

1. Task definition

A task is a service define by:

- **interface**: «abstract» component
  - Input / Output data
- **realization**: «implementation» component
  - C/C++/XMP-C/XMP-FORTRAN
  - libraries etc.

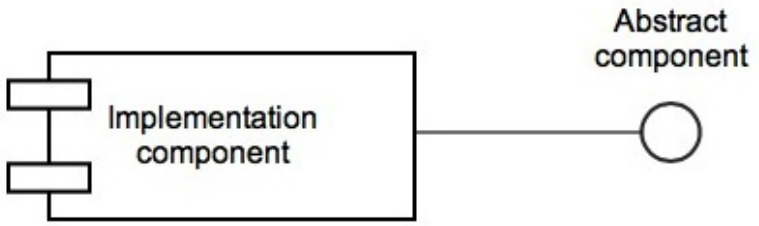


Figure : A service

Component approach

- modularity
- reusability

YML from user's point of view
YML architecture and interaction with middleware
Application

Abstract component

```
 <?xml version="1.0" ?>
<component type="abstract" name="XMP_mProdMat">
<params>
<param name="B0" type="Matrix" mode="in"  />
<param name="A0" type="Matrix" mode="in"  />
<param name="C0" type="Matrix" mode="out" />
</params>
</component>
```

YML from user's point of view
YML architecture and interaction with middleware
Application

Implementation

```
<?xml version="1.0"?>
<component type="impl" name="XMP_mProdMat" abstract="XMP_mProdMat" description="C = -(B x A)">
<impl lang="XMP" nodes="CPU:(2,2)">
  <templates>
    <template name="t" format="block,block" size="64,64"/>
  </templates>
  <distribute>
    <param template="t" name="B0(64,64)" align="[i][j]:(j,i)"/>
    <param template="t" name="A0(64,64)" align="[i][j]:(j,i)"/>
    <param template="t" name="C0(64,64)" align="[i][j]:(j,i)"/>
  </distribute>
<header>
<header>
<![CDATA[
#include<xmp.h>

double A[64][64];
double B[64][64];

#pragma xmp align A[i][j] with t(j,i)
#pragma xmp align B[i][j] with t(j,i)
</header>
<source>
/* XMP CODE HERE */
</source>
<footer />
</impl>
</component>
```

YML from user's point of view
YML architecture and interaction with middleware
Application

Workflow programming

- facilitate the expression of parallelism for user
- close to computational methods (Algorithm)
- high level language (Yvette) -> workflow
- deduce dataflow
- $\Rightarrow$ enable optimization combining two aspects: workflow and dataflow.

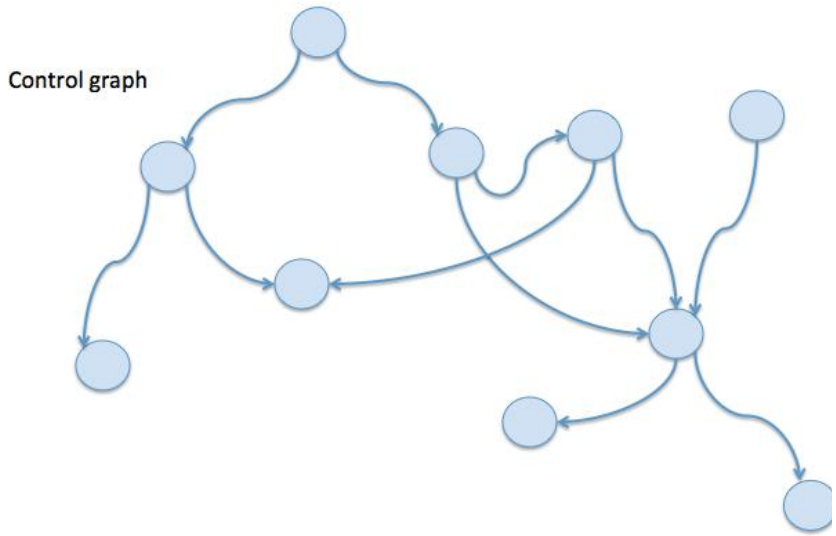


Figure : Graph of tasks

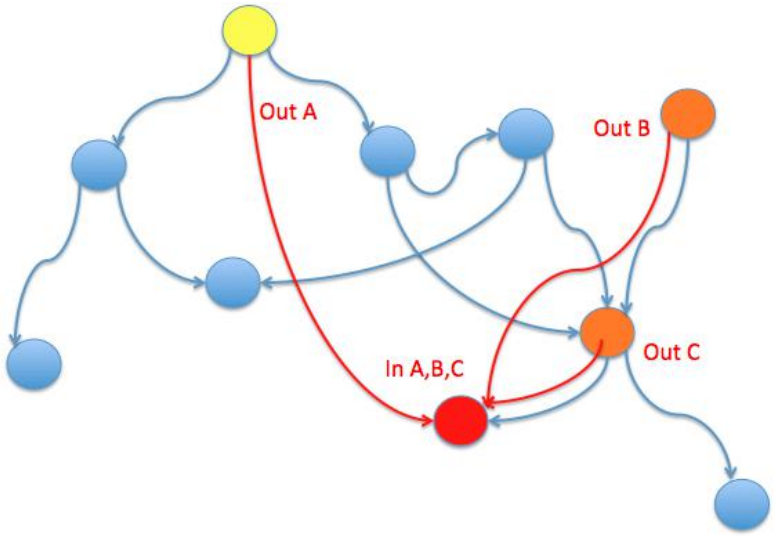


Figure : Deducing dataflow

YML from user's point of view

YML architecture and interaction with middleware

Application

Yvette Language

- **Parallel Section:** par section1 // ... // section N endpar
- **Sequential loop:** seq (i:=begin;end) do ... enddo
- **Parallel loop:** par (i:=begin;end) do ... enddo
- **Conditional structure:** if (condition) then ... else ... endif
- **Synchronization:** wait(event) / notify(event)
- **Component call:** compute NameOfComponent(args,...,...)

Syntaxe

```
<?xml version="1.0"?>
<application name="Gauss-Jordan">
<graph>
blockcount:=4;
par
  par(i:=0;blockcount-1)(j:=0;blockcount-1)
  do
    compute XMP_genMat(A[i][j],i,j);
    compute XMP_copyMat(A[i][j],C[i][j]);
  enddo
//
  par(i:=0;blockcount-1)(j:=0;blockcount-1)
  do
    if (i neq j) then
      compute XMP_fillMatrixZero(B[i][j]);
    endif
  enddo
endpar
```

YML from user's point of view
YML architecture and interaction with middleware
Application

Development workflow

1. Register «abstract» component (name of service, parameters in, out, inout) with yml_component
2. Register «implementation» component (parse code, get information on impl (grid topology for XMP: CPU(:,:), **compile**) with yml_component
3. **compile Application with yml_compiler**
4. **schedule application with yml_scheduler**

YML from user's point of view
YML architecture and interaction with middleware
Application

Multilevel programming paradigm

- **High level**: communication inter nodes/group of nodes
- **Low level**: group of nodes / cores

Multilevel programming

- **High level: YML** - coarse grain parallelism - asynchronous communication
- **Low level: XMP** programming with pragma, PGAS language

Multilevel programming

YML from user's point of view
YML architecture and interaction with middleware
Application

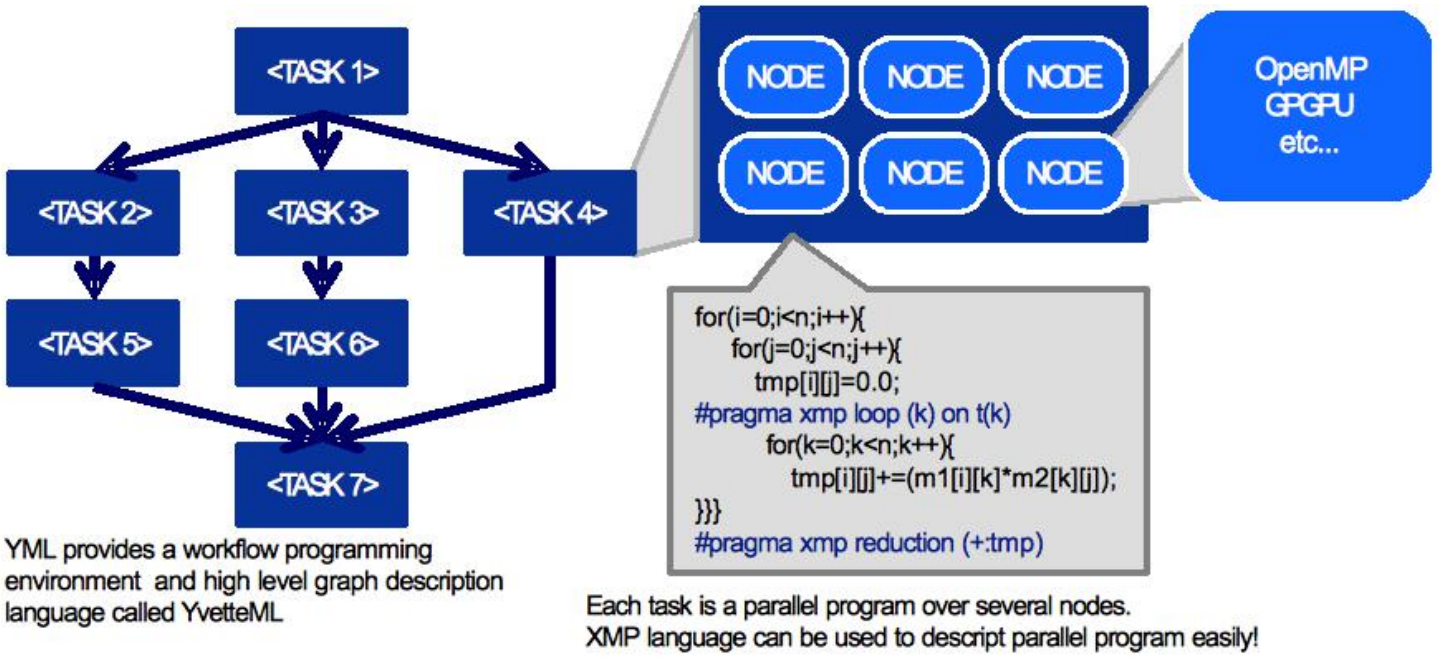


Figure : Multilevel programming

YML from user's point of view
YML architecture and interaction with middleware
Application

YML architecture and interaction with middleware

Architecture overview

YML from user's point of view
YML architecture and interaction with middleware
Application

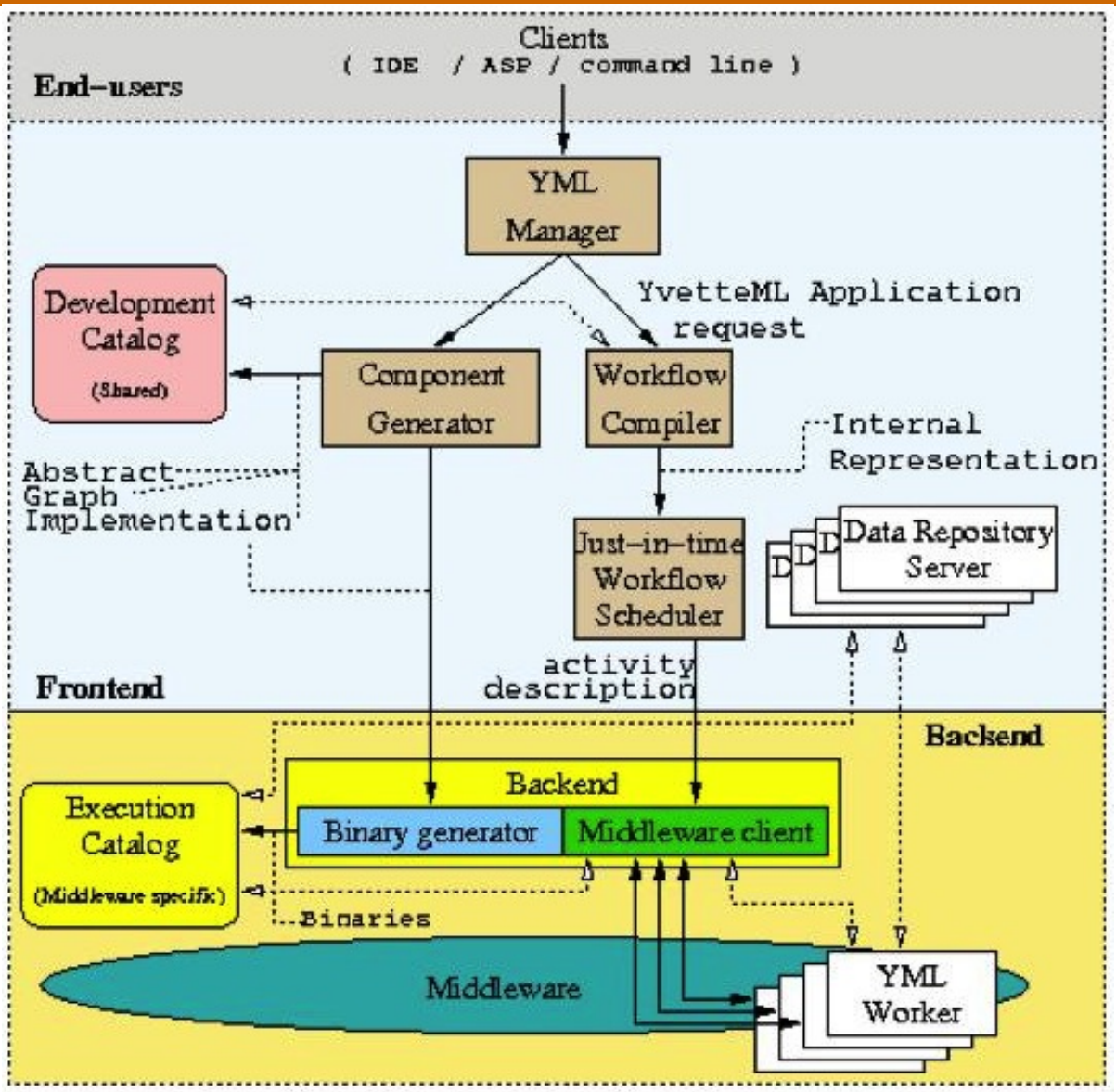


Figure : YML Architecture

Coupling between Backend and Middleware

YML from user's point of view
YML architecture and interaction with middleware
Application

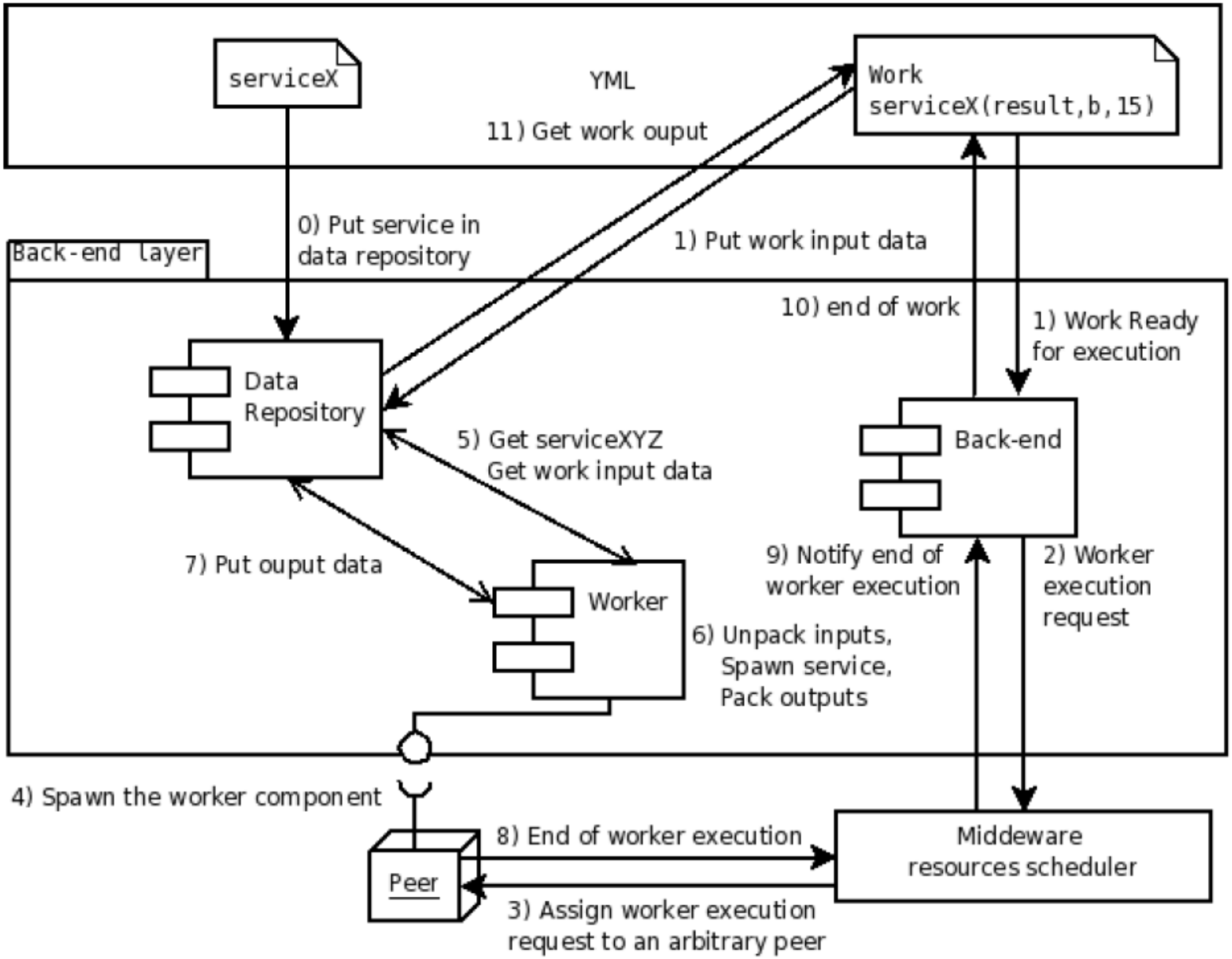


Figure : Backend Model - Structural Point of vue

YML from user's point of view
YML architecture and interaction with middleware
Application

Note on backend interface

- Backend interface provides information from catalogs
- Middleware use this information
- **Separation of Data definition, computation and communication**

yml_scheduler

1. get information on application (graph, binaries)
2. Scheduling loop
   1. Scheduler schedule pending task (with BackendManager)
   2. BackendManager execute task (with Backend)
   3. Backend execute Implementation

YML from user's point of view

YML architecture and interaction with middleware

Application

Application

YML from user's point of view
YML architecture and interaction with middleware
Application

Eigenvalue problem: **M**ultiple **E**xplicit **R**estarted **A**rnoldi **M**ethod (MERAM)

- solution by ERAM by a component with possibly many implementation
  - expression of paralleslism
  - use of different libraries (Petsc/Slepc, Scalapak)
- Restarted Krylov Method: How to choose the good size of subspaces and restarting ?
- Restart combine multiple instances of ERAM with different parameter
  - coarse grain parallelism
  - asynchronous communication

Inverse of a matrix: **B**lock **G**auss **J**ordan

- each component perform parallel linear algebra operation
- coarse grain parallelism expressed by the graph of task (dependencies)

YML from user's point of view
YML architecture and interaction with middleware
Application

Targeted Architecture

- **Grid'5000**: Cluster of clusters distributed over 10 distant sites and > 5000 cores.
- **Carver**: IBM iDataPlex System at NERSC/LBNL (9984 cores, 1120 nodes of 8 cores & 80 nodes of 12 cores).
- **K Computer**: 864 rack x 102 nodes x 1 CPU = 88,128 CPUs. Node: SPARC64 VIIIfx (8core) CPU 128GFLOPS/node

YML from user's point of view
YML architecture and interaction with middleware
Application

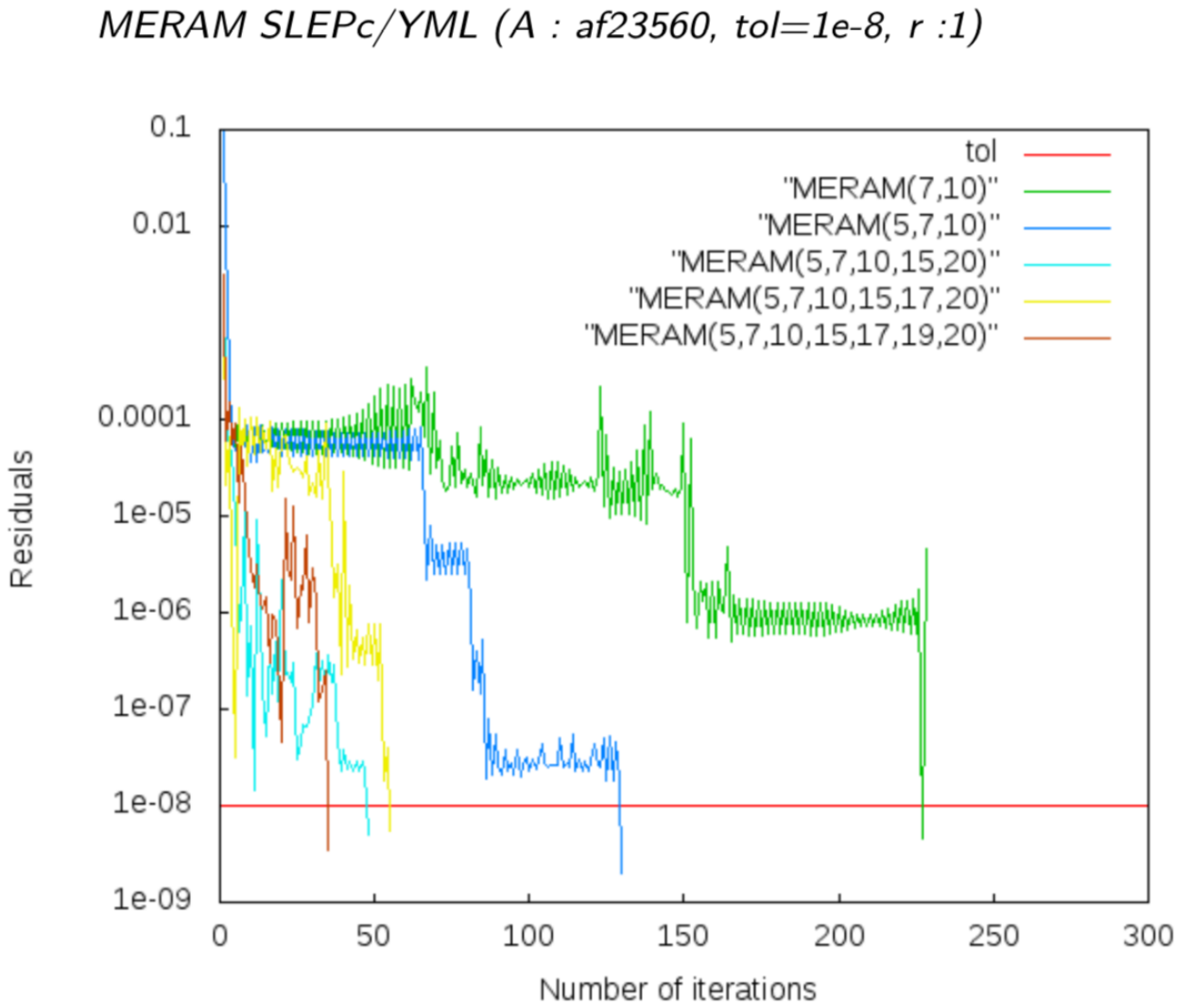
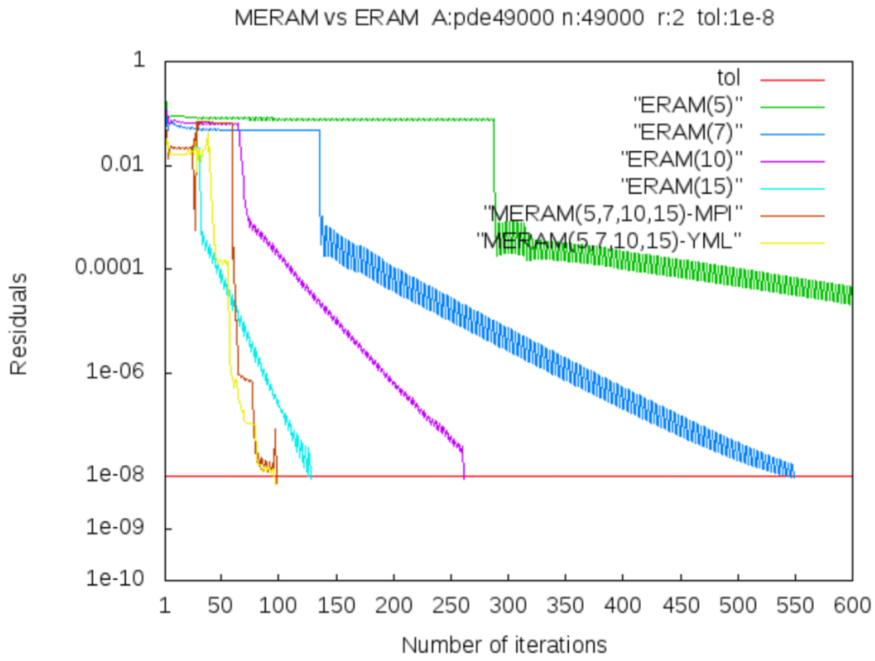


Figure : MERAM on Grid'5000

Results from M. Dandouna (Reusable numerical libraries for large scale distributed system, Ph.D. Thesis, 2012)

MERAM on Carver

YML from user's point of view
YML architecture and interaction with middleware
Application



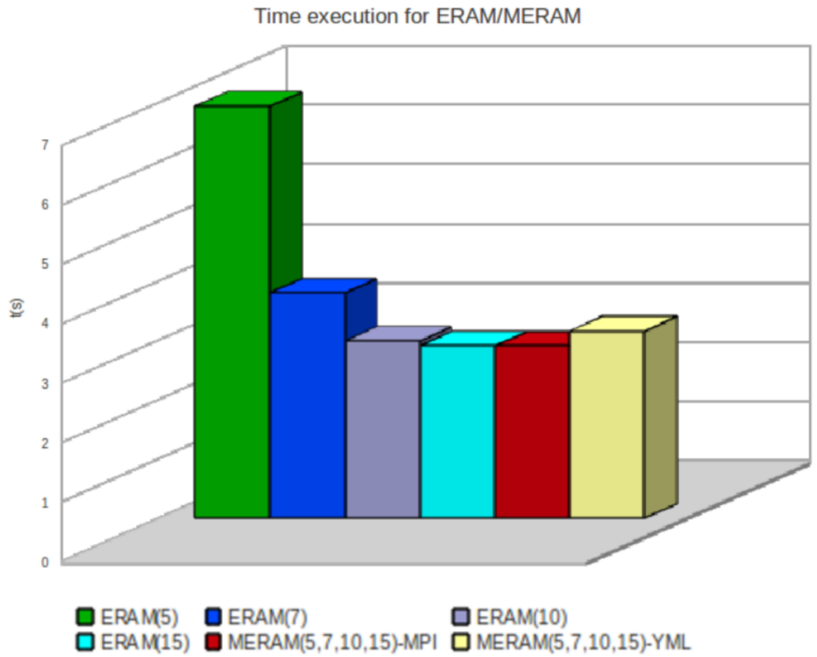


Figure : MERAM on Carver

Results from M. Dandouna (Reusable numerical libraries for large scale distributed system, Ph.D. Thesis, 2012)

YML from user's point of view

YML architecture and interaction with middleware

Application

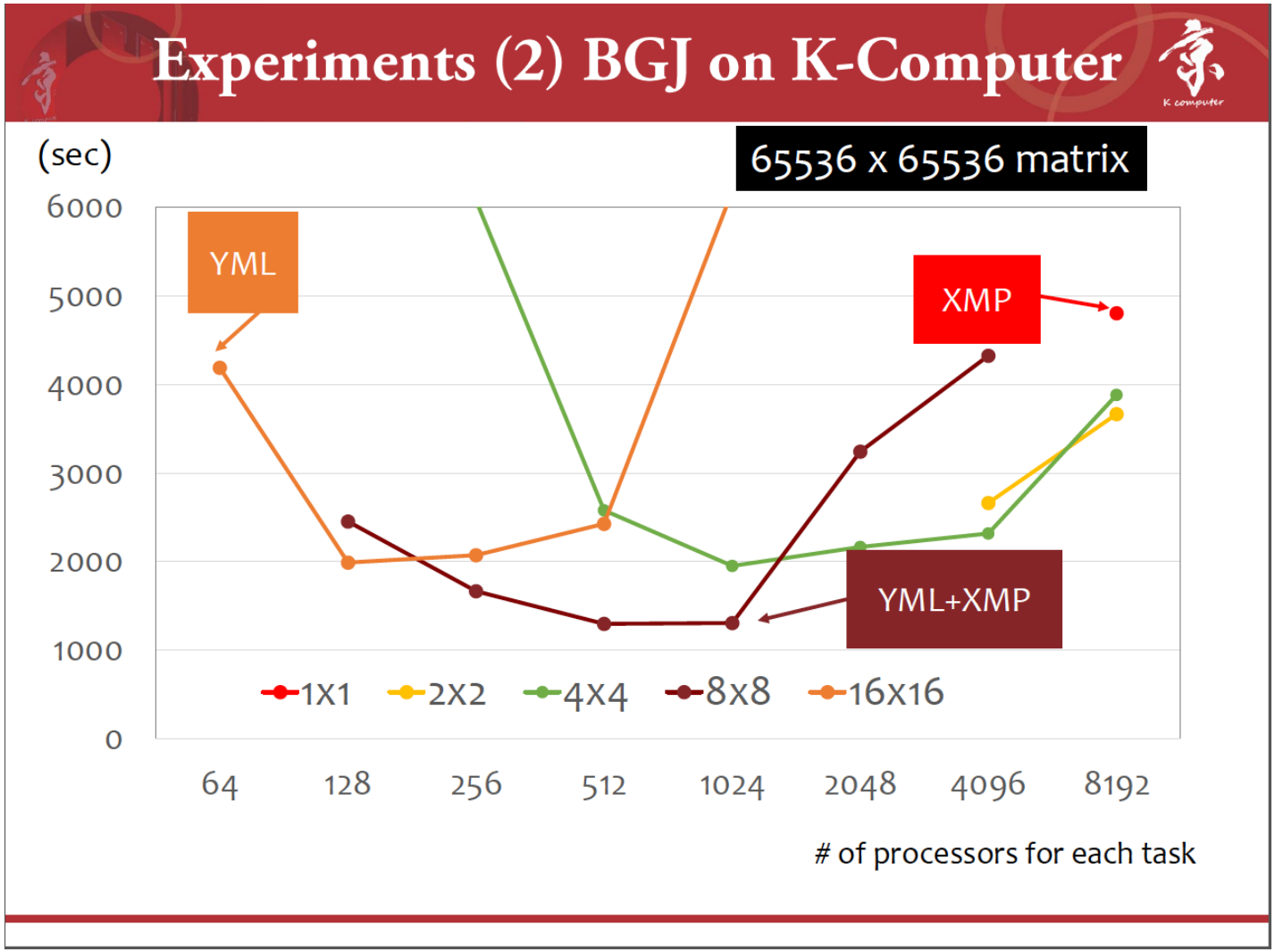


Figure : A^{-1} computation with Gauss-Jordan Block (YML-XMP on K computer (Japan)

Results from Miwako Tsuji (FP3C project, S. Petiton, M. Sato et.al.)

YML from user's point of view

YML architecture and interaction with middleware

Application

Target algorithm

- Current algorithms: BGJ, MERAM
- YML-XMP for direct solution method (see talk J. Gurhem)
- New algorithms: Padé Rayleigh Ritz (PRR), Multiple PRR (2018 internship)

Expertise on current design

- Internships on languages integration and interoperability in YML (Master CHPS 2014, 2016)
- study on data managment in YML (First SPPEXA Workshop)
- Yvette language specification (MYX Project (ANR-15-SPPE-0003))
- creation of a YML development comitee (2018)

Dissemination and support

- yml.prism.uvsq.fr
- **Virtual Machine (Workshop JDev 2015, CNRS) Available on yml.prism.uvsq.fr**
- Various virtual machine environment with container (Docker (Debian, CentOs), Singularity) (Undergraduate student project, 2017-2018)
- with tutorial: add, sort, BGJ

Thanks for your attention!

University of Tsukuba
Center for Computational Sciences
筑波大学　計算科学研究センター

3rd Workshop on Parallel Programming Models - Productivity and Applications

March 15th 2018

RWTH Aachen university, Aachen

References (Selection)

YML from user's point of view
YML architecture and interaction with middleware
Application

- **Algorithms**
  - N. Emad, S. Petiton, and G. Edjlali. Multiple Explicitly Restarted Arnoldi Method for Solving Large Eigen- problems.SIAM Journal on Scientic Computing (SJSC), 27(1) :253-277, 2005
  - L. Shang, S. Petiton, M. Hugues, A new parallel paradigm for block-based Gauss-Jordan algorithm,"Grid and Cooperative Computing, 2009. GCC'09. Eighth International Conference on",193-200,2009,IEEE
- **YML-XMP**
  - S. Petiton, M. Sato, N. Emad, C. Calvin, M. Tsuji and M. Dandouna, Multi level programming Paradigm for Extreme Computing, Published online: 06 June 2014
  - M. Tsuji, M. Dandouna and N. Emad, Multi level parallelism of Multiple implicitly/explicitly restarted Arnoldi methods for post-petascale computing,Proceedings of the 8th IEEE International Conference on P2P Parallel Grid Cloud and Internet Computing (3PGCIC-2013),158–165,2013.10.28-30,University of Technology of Compiegne Compiegne France.
- **YML**
  - N. Emad, O. Delannoy, and M. Dandouna. Numerical library reuse in parallel and distributed platforms. In the proceedings of 9th International Meeting on High Performance Computing for Computational Science, VecPar'10, Lawrence Berkeley National Labratory, California, USA, June, 22-25 2010
  - L. Choy, O. Delannoy, N. Emad and S. Petiton - Federation and abstraction of heterogeneous global computing platforms with the YML framework, in The Third International Workshop on P2P, Parallel, Grid and Internet Computing (3PGIC-2009), March 2009, Japan
  - O. Delannoy, YML: A scientific Workflow for High Performance Computing, Ph.D. Thesis, Septembre 2008, Versailles