High scale computation with YML: from application to execution

T.DUFAUD, N. EMAD, S. PETITON

France-Japan-Germany trilateral workshop : Convergence of HPC and Data Science for Future Extreme Scale Intelligent Applications 6-8 Nov 2019 Tokyo (Japan)

FGJ workshop: HPC and AI - 2019/11/07

Plan

1. Graph modeling with YML
2. Framework and latest developments
3. Related application and some results

Graph modeling with YML

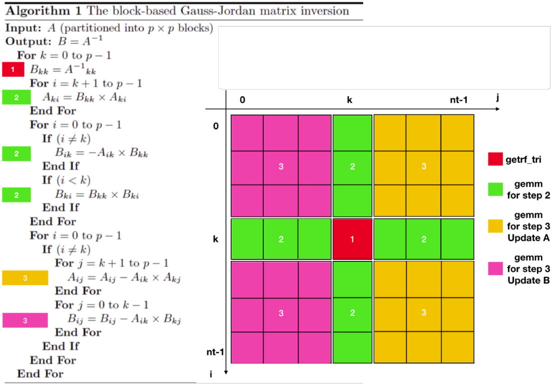
Needs of parallel programming environment

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

BGJ as an example

- **Block Gauss-Jordan Inversion**
- **Coarse grain:** algorithm written with YML
- **Fine grain:** linear algebra library written with XMP



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 $p-1$
1   $B_{kk} = A^{-1}{}_{kk}$
    For $i = k+1$ to $p-1$
2     $A_{ki} = B_{kk} \times A_{ki}$
    End For
    For $i = 0$ to $p-1$
      If $(i \neq k)$
2       $B_{ik} = -A_{ik} \times B_{kk}$
      End If
      If $(i < k)$
2       $B_{ki} = B_{kk} \times B_{ki}$
      End If
    End For
    For $i = 0$ to $p-1$
      If $(i \neq k)$
        For $j = k+1$ to $p-1$
3         $A_{ij} = A_{ij} - A_{ik} \times A_{kj}$
        End For
        For $j = 0$ to $k-1$
3         $B_{ij} = B_{ij} - A_{ik} \times B_{kj}$
        End For
      End If
    End For
  End For
```

🟥 getrf_tri

🟩 gemm for step 2

🟨 gemm for step 3 Update A

🟪 gemm for step 3 Update B

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

User's point of view

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.

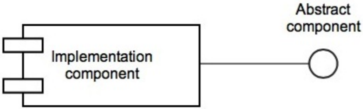


Figure: A service

Component approach

- modularity
- reusability

User's point of view

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.

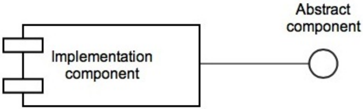


Figure: A service

Component approach

- modularity
- reusability

Workflow programming

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

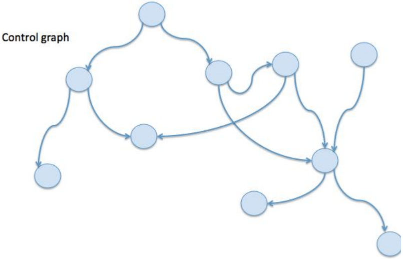


Figure: Graph of tasks

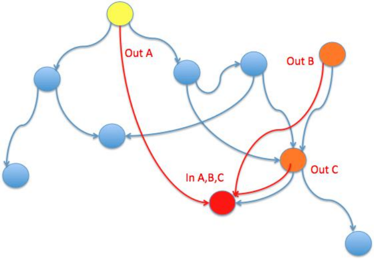


Figure: Deducing dataflow

BGJ: 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
```

Framework and latest developments

Architecture overview

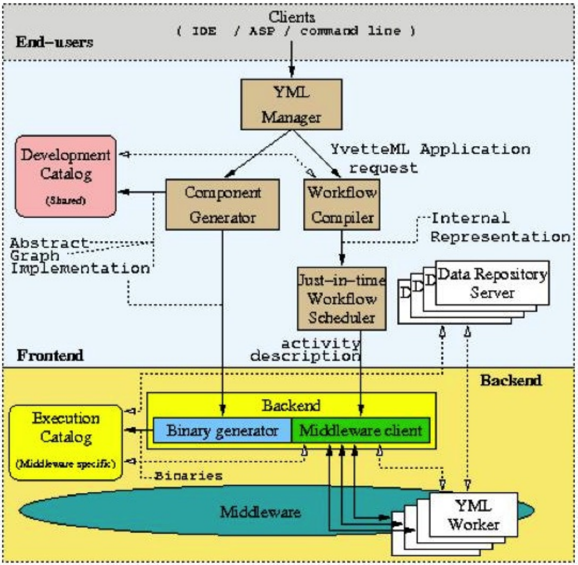


Figure: YML Architecture

Backend and Middleware coupling through yml_scheduler

Note on backend interface

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

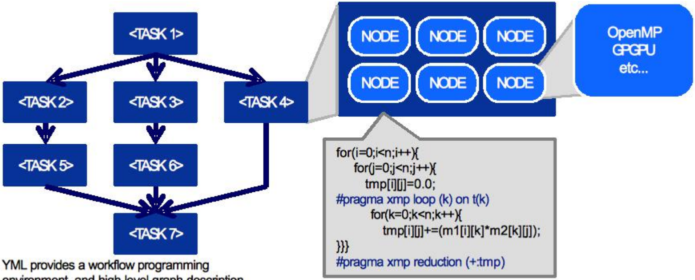
yml_scheduler

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

Note on Multilevel parallelism with YML

Multilevel programming paradigm

- **High level**: communication inter nodes/group of nodes **(YML)**
- **Low level**: group of nodes / cores **(XMP)**



```
for(i=0;i<n;i++){
   for(j=0;j<n;j++){
      tmp[i][j]=0.0;
#pragma xmp loop (k) on t(k)
         for(k=0;k<n;k++){
            tmp[i][j]+=(m1[i][k]*m2[k][j]);
}}}
#pragma xmp reduction (+:tmp)
```

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

Each task is a parallel program over several nodes.
XMP language can be used to descript parallel program easily!

Figure: Multilevel programming

YML-XMP + MUST

MYX project (SPPEXA)

- MUST: correctness checking tool for MPI
- correctness checking for XMP with MUST using XMPT
- YML + XMP

Correctness checking

- Correctness checking in multi-SPMD programming model context
- SPPEXA project U. Aachen, U. Tsukuba, Riken R-CCS, U. Lille, U. Paris-Saclais / Versailles
- **M. Tsuji's Talk**

YML in virtual machines

YML distribution

- **A virtual machine for YML since 2015** (JDev CNRS)
- A docker image for YML since 2018
- **Containerization with docker (2019-)** with J. Gurhem (U. Lille, CNRS) and M. Mancip (MDLS, CNRS)

Containerization

- **Pipework**: setup infiniband interface ib0
- **Docker Swarm**: Network of containers
- Host network
- **Docker image**: OS + YML + MPI

Network of containers for YML with OmniRPC-MPI backend

- **pros**
  - reproducibility
  - ease of deployment
- **cons**
  - docker image is 1.5 GB (can be reduced to 800 MB)
  - average **overhead** on parallel block linear algebra : **3%**

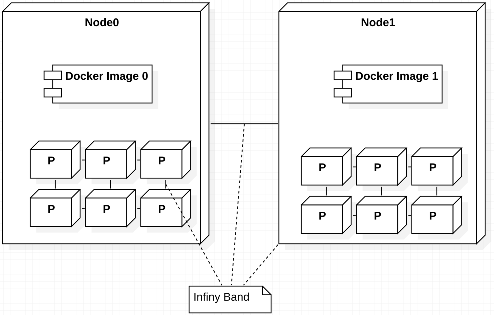


Figure: Docker Images on Nodes (group of processors that communicate using MPI)

Related application and some results

Numerical Algorithms

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

- solution by RAM by a component with possibly many implementation
  - expression of paralleslism
  - use of different libraries (Petsc/Slepc, Scalapack)
- Restarted Krylov Method: How to choose the good size of subspaces and restarting ?
- Restart combine multiple instances of RAM 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)

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

People involved

- U. Lille 1, CNRS (FRANCE): S. Petiton, M. Hugues (TOTAL)
- U. Paris-Saclay / Versailles (FRANCE): N.Emad, M. Dandouna
- Riken RCCS (JAPAN): M. Tsuji, M. Sato
- LBNL (USA): L. Drummond

MERAM on Grid'5000

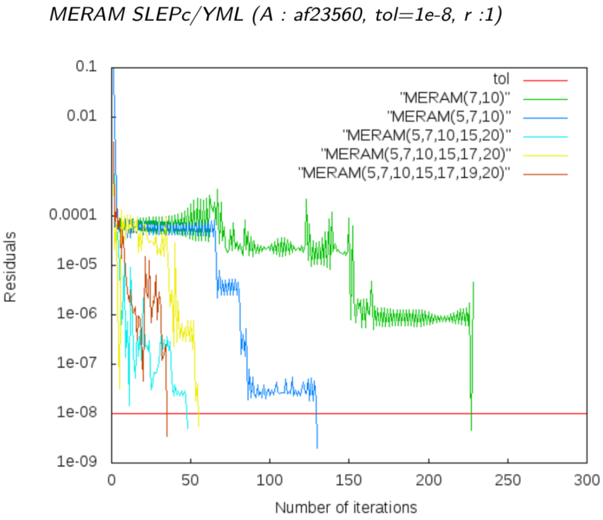


Figure: MERAM on Grid'5000

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

MERAM on Carver

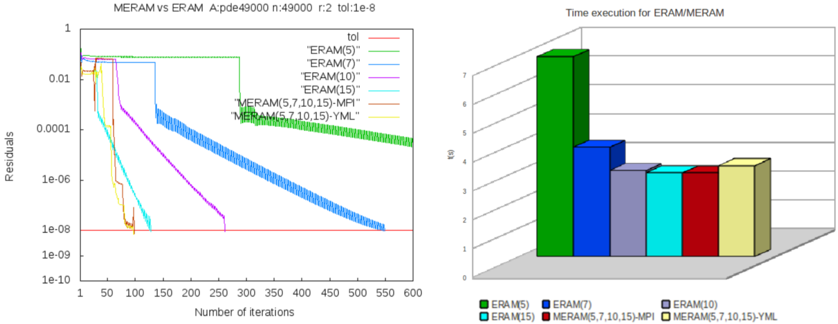
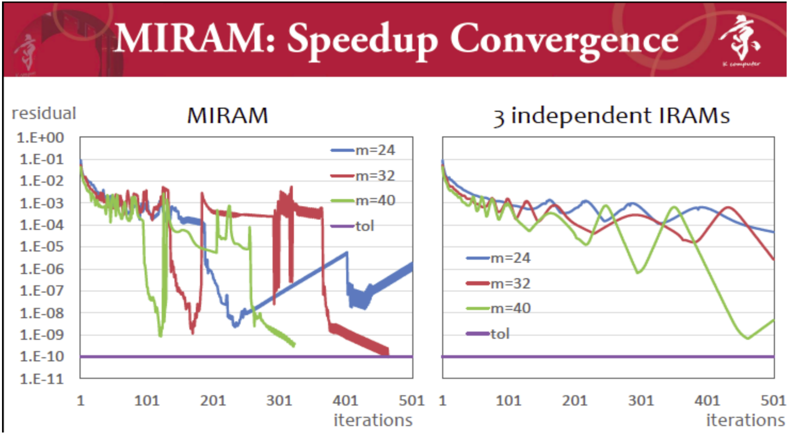


Figure: MERAM on Carver

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

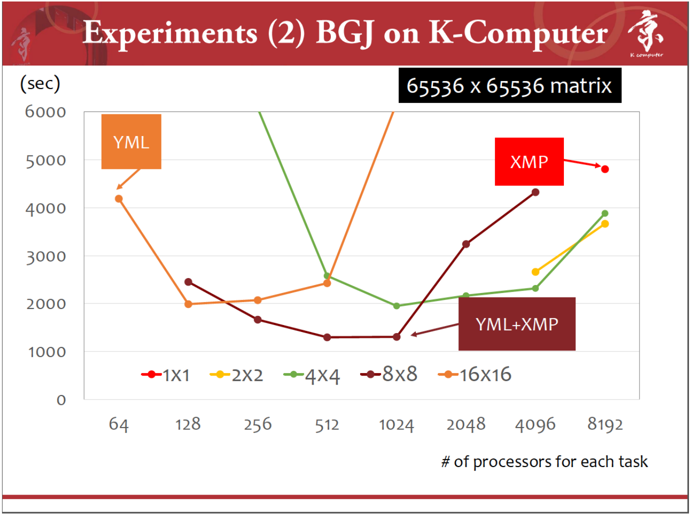
MIRAM on K Computer



MIRAM: Speedup Convergence

Figure: MIRAM with YML on K Computer **(by M. Tsuji)** - Matrix: Schenk/nlpkkt240 n=27,993,600 ; $k = 10$, $tol = 1E - 10$

Multilevel, YML-XMP approach



Experiments (2) BGJ on K-Computer

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.)

Concluding remarks

Target algorithm

- Current algorithms: BGJ, MERAM, MIRAM
- YML-XMP for direct solution method (J. Gurhem, S. Petiton)
- New algorithms: Padé Rayleigh Ritz (PRR), Multiple PRR (2019 internship)

Design and expertise

- Internships on languages integration and interoperability in YML (Master CHPS 2014, 2016)
- Yvette language specification and YML-XMP + MUST (MYX Project (ANR-15-SPPE-0003))
- Execution on containers (First Docker tests for OmniRPC backend)
- Data managment in YML (First FGJ Workshop, ESPM2@SC18)

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) (on demand)
- with tutorial: add, sort, BGJ

Thanks for your attention!

**France-Japan-Germany trilateral workshop:
Convergence of HPC and Data Science for
Future Extreme Scale Intelligent Applications**

November 7th 2019
MFJ, Tokyo

References (Selection)

- **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