マルチSPMD環境に向けたXMP/YMLの活用

Miwako TSUJI　AICS

2014.10.24
第2回XcalableMPワークショップ

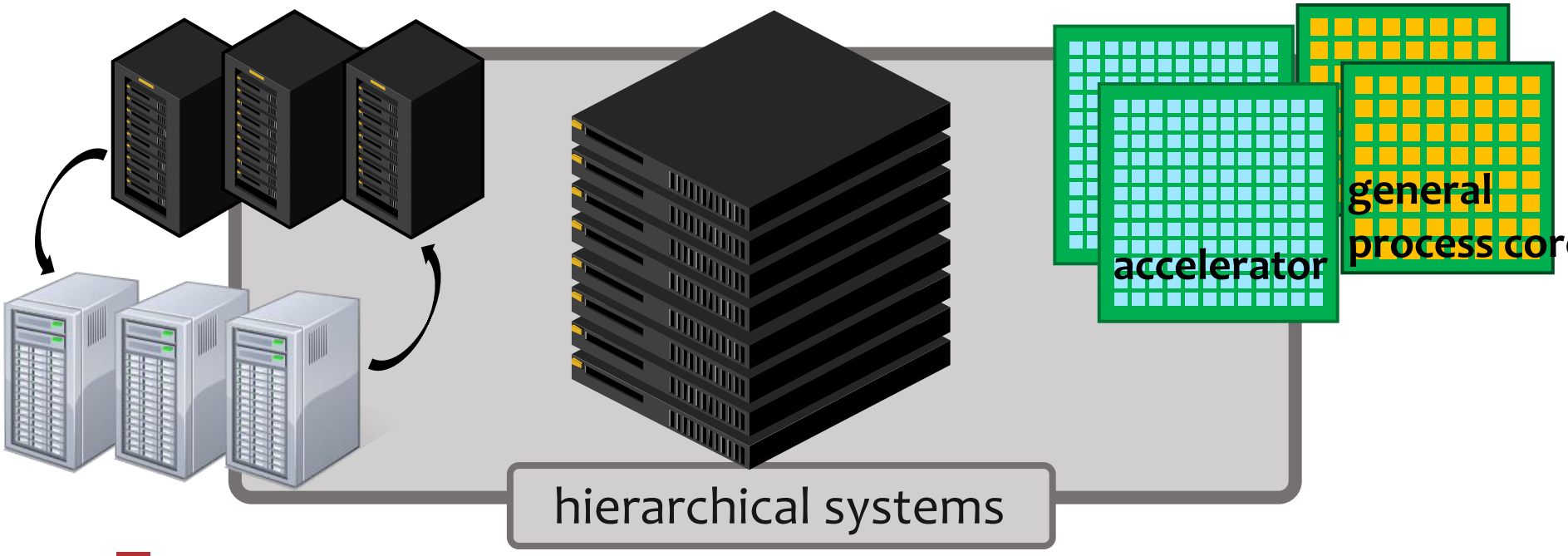
JST-ANR "Framework and Programming for Post Petascale Computing (FP3C)" project

- 日仏共同研究
- 2010.09〜2014.03
- 多岐にわたる研究分野とそれらのインテグレーション
  - プログラミングモデル
    とプログラミング
    言語設計
  - ランタイムライブラリ
  - アクセラレータ
  - アルゴリズムと
    数値計算ライブラリ

概　要

- プログラミングモデル・プログラミング言語設計
- 上記とランタイムライブラリの統合
- 上記と数値計算ライブラリの統合
- FP2C（Framework for Post-Petascale Computing)
  - 階層的かつヘテロな巨大システムにおいてアプリケーションを構築し，実行するためのソフトウエア
  - YML + XMP(-dev) + StarPU → integrated
    - developed in Japan and in France
- Experimental Results on K-computer and others

FP2C (YML/XMP-dev/StarPU)



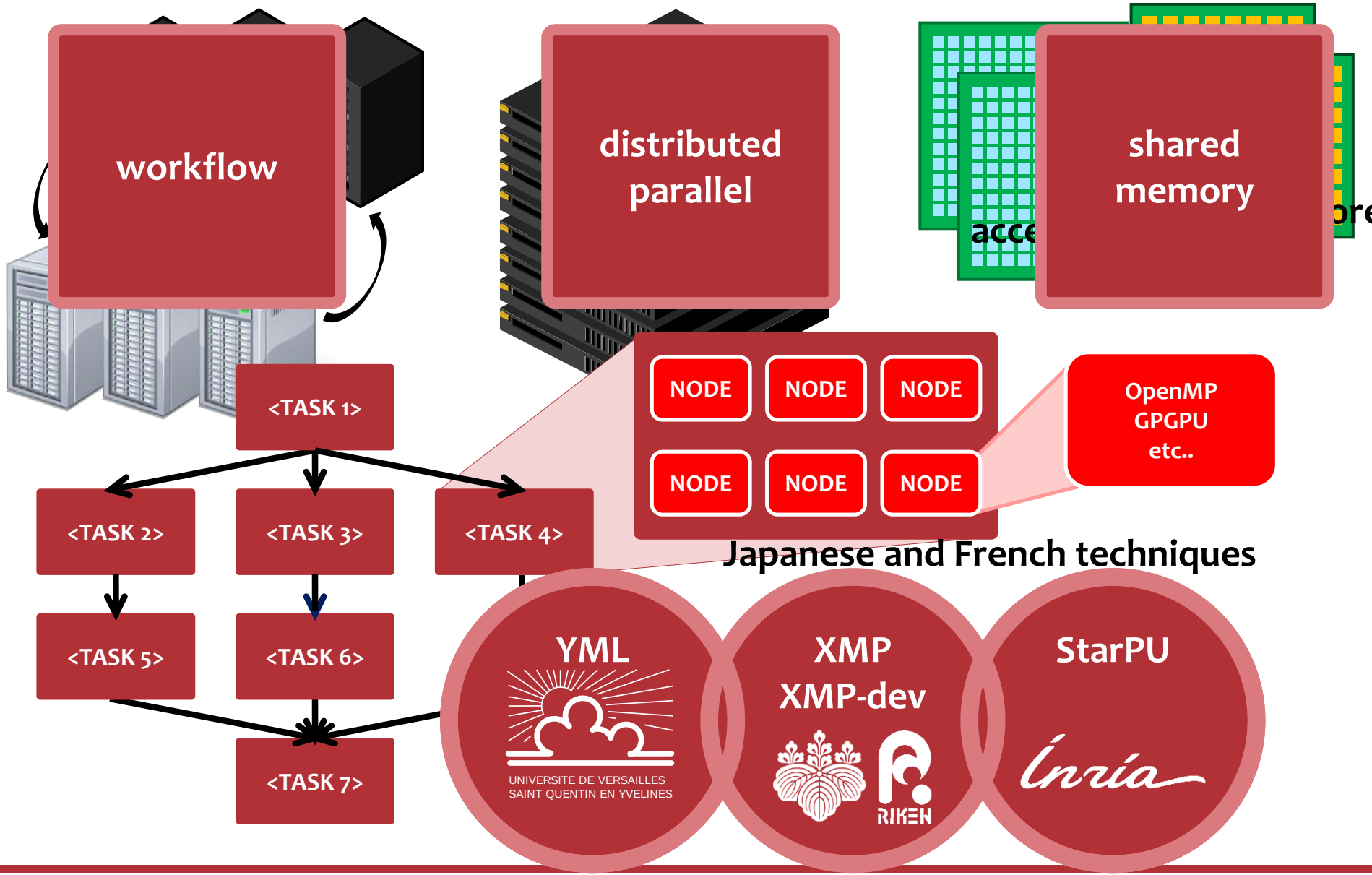
- Model

  Multi-programming methodologies

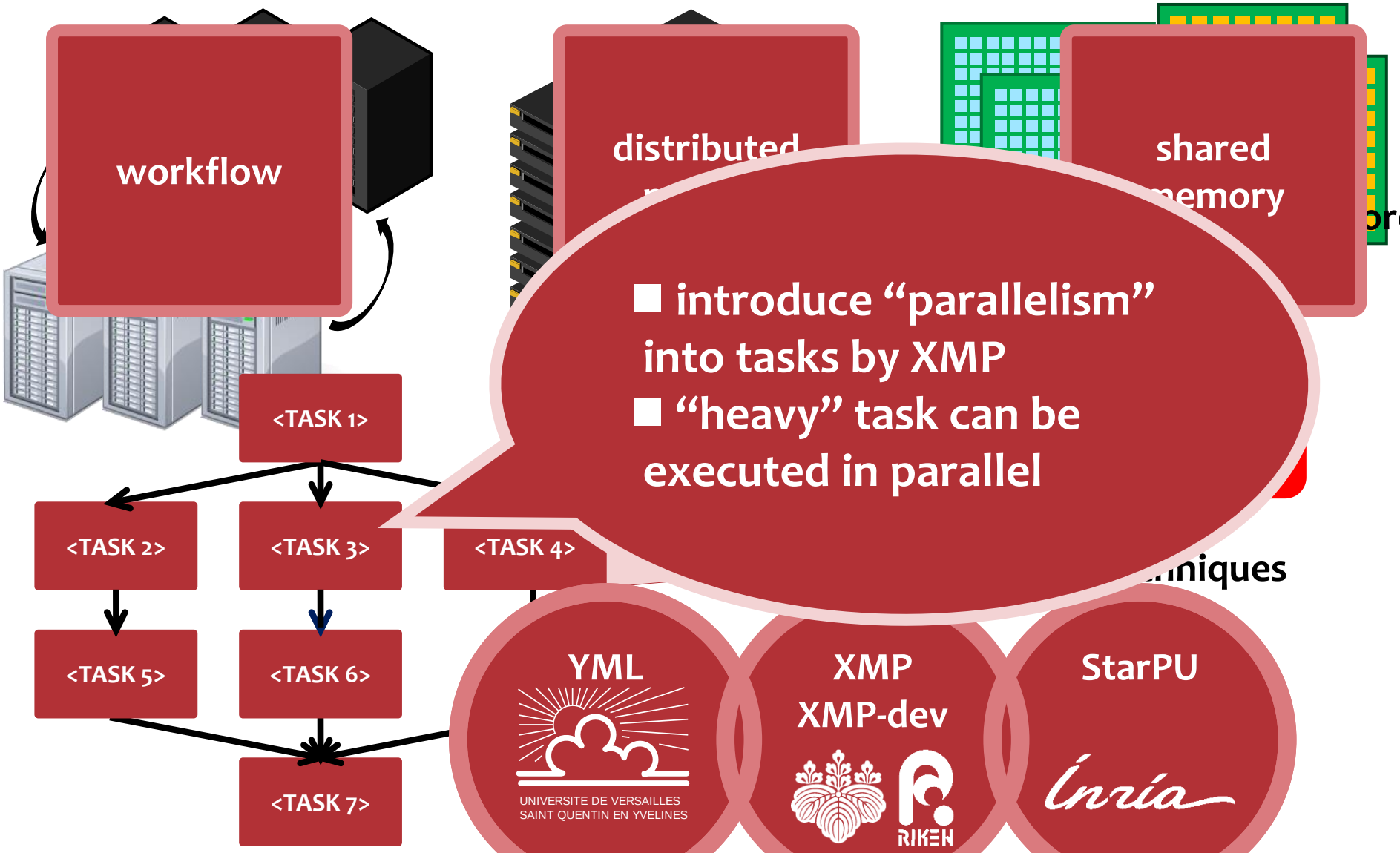
  across multi-architectural levels

- FP2C (Framework for Post-Petascale Computing), a tool to develop and execute applications based on the model

FP2C (YML/XMP-dev/StarPU)

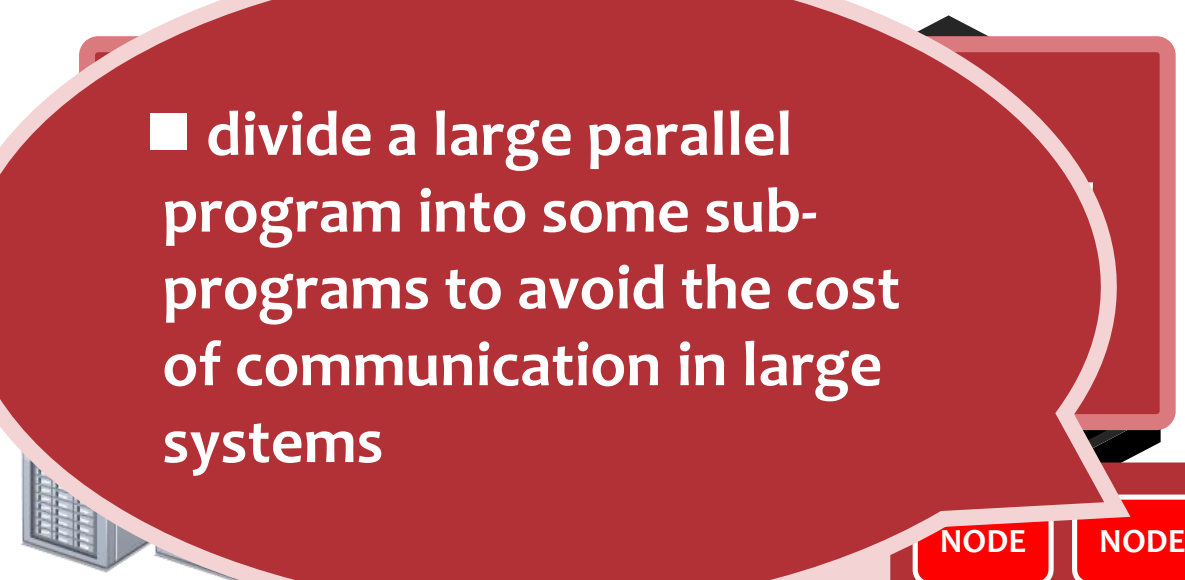


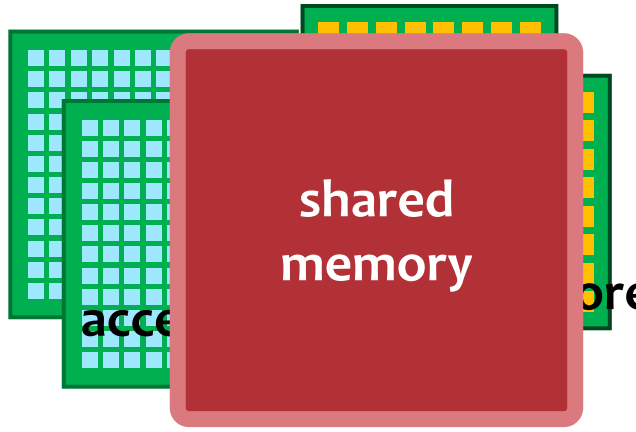
FP2C (YML/XMP-dev/StarPU)



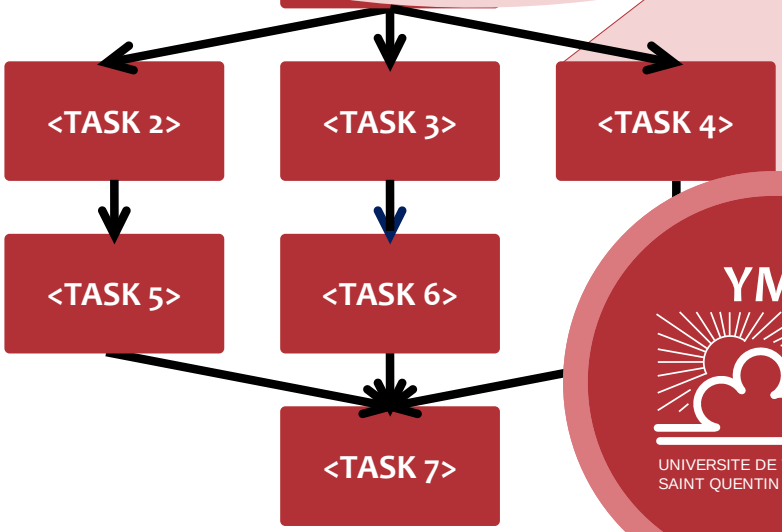
FP2C (YML/XMP-dev/StarPU)

- divide a large parallel program into some sub-programs to avoid the cost of communication in large systems





OpenMP
GPGPU
etc..



Japanese and French techniques

YML
UNIVERSITE DE VERSAILLES
SAINT QUENTIN EN YVELINES

XMP
XMP-dev
RIKEN

StarPU
Inria

FP2C (YML/XMP-dev/StarPU)

- use both of CPU and GPU cores
- XMP-dev/StarPU by the collaboration of U. Tsukuba and Inria

shared memory

NODE NODE NODE

OpenMP
GPGPU
etc..

<TASK 2> <TASK 3> <TASK 4>

<TASK 5> <TASK 6>

<TASK 7>

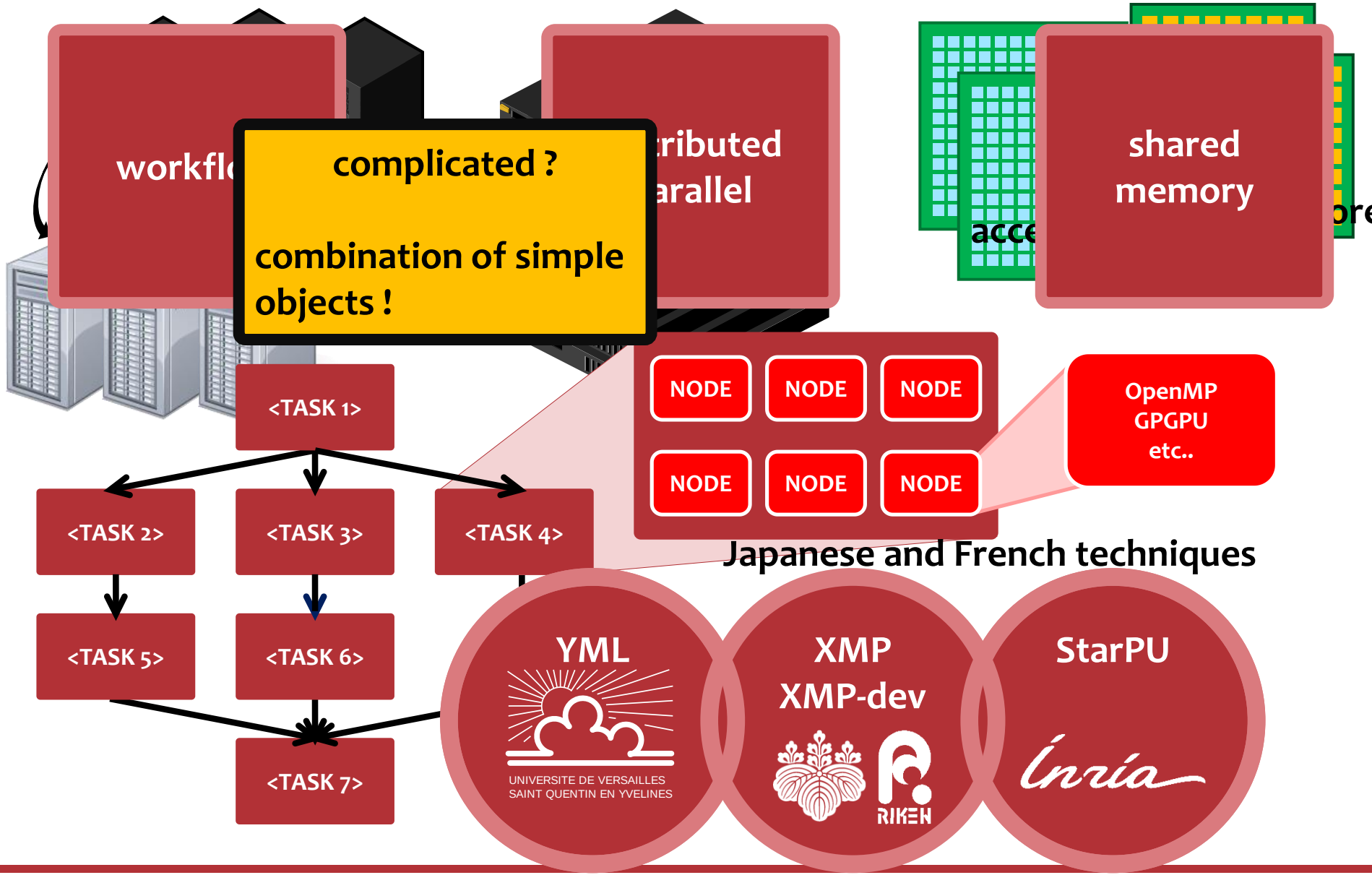
Japanese and French techniques

YML
UNIVERSITE DE VERSAILLES SAINT QUENTIN EN YVELINES

XMP
XMP-dev
RIKEN

StarPU
Inria

FP2C (YML/XMP-dev/StarPU)



Application Development with FP2C (YML/XMP-dev/StarPU)

- Task development
  - Define interface (input/output) of a task
  - Define procedure of a task
    - C++, XMP, XMP-dev/StarPU, XMP for Fortran, MPI

      (The original YML supported only C++, parallel programming was not supported)

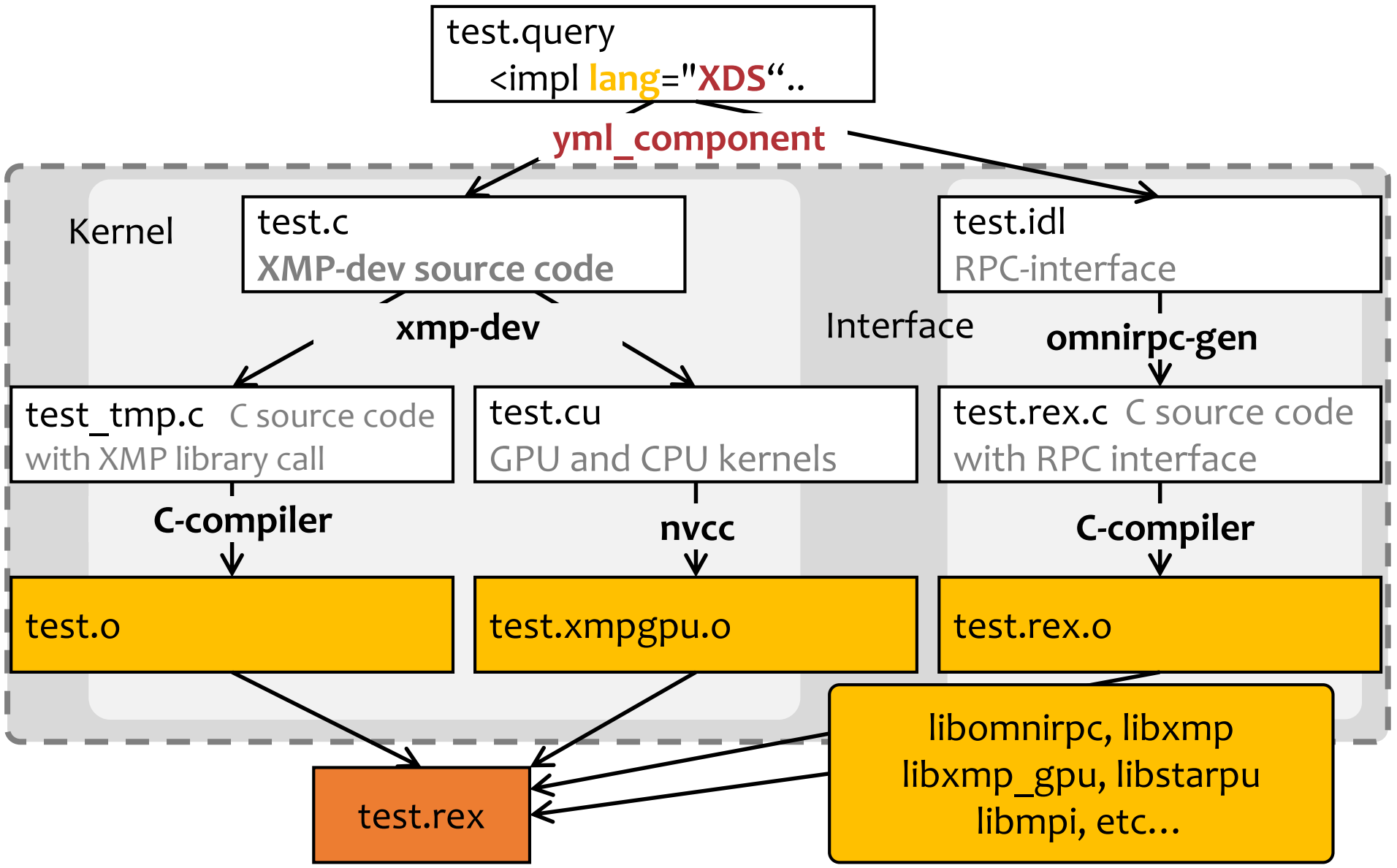
- Workflow development
  - Define dependency between tasks
  - Compile the definition into directed acyclic graph
    - yml_compiler

Application Development with FP2C (YML/XMP-dev/StarPU)

```
<?xml version="1.0"?>
<component type="impl" name="sample" abstract="sample">
<impl lang="XDS" nodes="CPU:(16)" libs="starpu">
<templates>
<template name="t"
format="block" size="256"/>
</templates>
<distribute>
<param template="t"
name="A(256)" align="[i]:(i)"/>
<param template="t"
name="B(256)" align="[i]:(i)"/>
</distribute>
```

```
<source>
<![CDATA[
int i;
#pragma xmp acc replicate (A,B)
{
#pragma xmp acc replicate_sync in (A,B)
#pragma xmp loop (i) on t(i) acc
 for(i=0;i<256;i++){
   B[i] = A[i]*A[i];
 }
#pragma xmp acc replicate_sync out(B)
}
]]
</source>
```

Task (Remote Program) Generator



Application Development with FP2C (YML/XMP-dev/StarPU)

Workflow Description in YvetteML

```
par
  A[i][j] is initialized at random
  B[i][j] is initialized as an unit matrix
endpar

par
  par(k:=0;count-1)
  do
    if (k neq 0) then
      wait(prodDiffA[k][k][k-1]);
    endif
    compute inversion(A[k][k],B[k][k]);
    notify(bInversed[k][k]);
    if (k neq count-1) then
      par (i:=k+1; count-1)
      do
        wait(bInversed[k][k]);
        compute prodMat(B[k][k],A[k][i]);
        notify(prodA[k][i]);
      enddo
    endif
    wait(bInversed[k][k]);
    par(i:=0;count-1)
    do
      if(i neq k) then
        compute mProdMat(A[i][k],B[k][k],B[i][k]);
        notify(mProdB[k][i][k]);
      endif
      if(k gt i) then
        compute prodMat(B[k][k],B[k][i]);
        notify(prodB[k][i]);
      endif
    enddo
    par(i:= 0;count-1)
    do
      if (i neq k) then
        if (k neq count - 1) then
          par (j:=k + 1;count-1)
          do
            wait(prodA[k][j]);
            compute prodDiff(A[i][k],A[k][j],A[i][j]);
            notify(prodDiffA[i][j][k]);
          enddo
        endif
        if (k neq 0) then
          par(j:=0;k-1)
          do
            wait(prodB[k][j]);
            compute prodDiff(A[i][k],B[k][j],B[i][j]);
          enddo
        endif
      endif
    enddo
  enddo
endpar
```

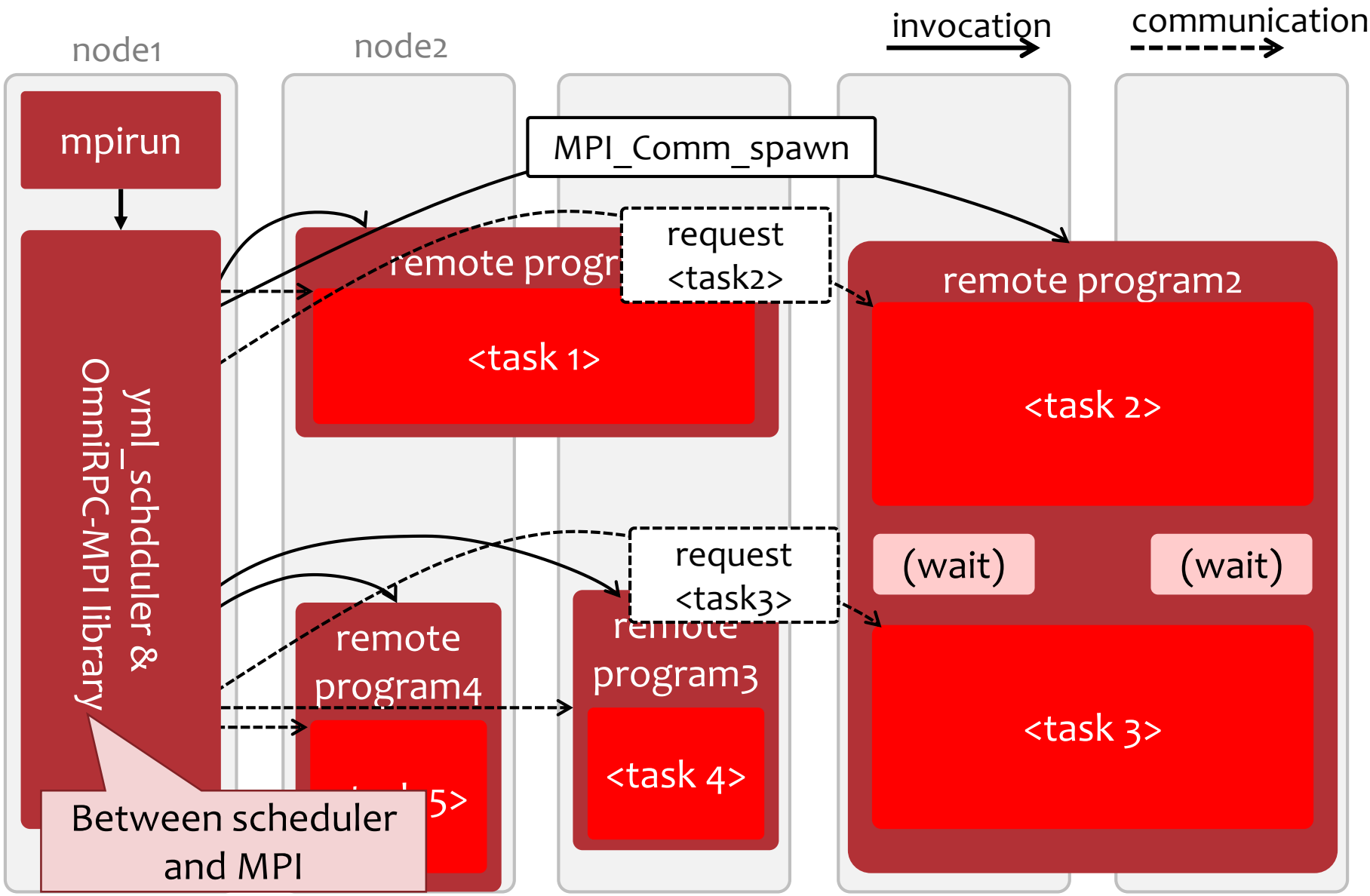
call task

notify-wait (dependency)

Parallel Execution

YvetteML: simple workflow language

Application Execution with FP2C (YML/XMP-dev/StarPU)

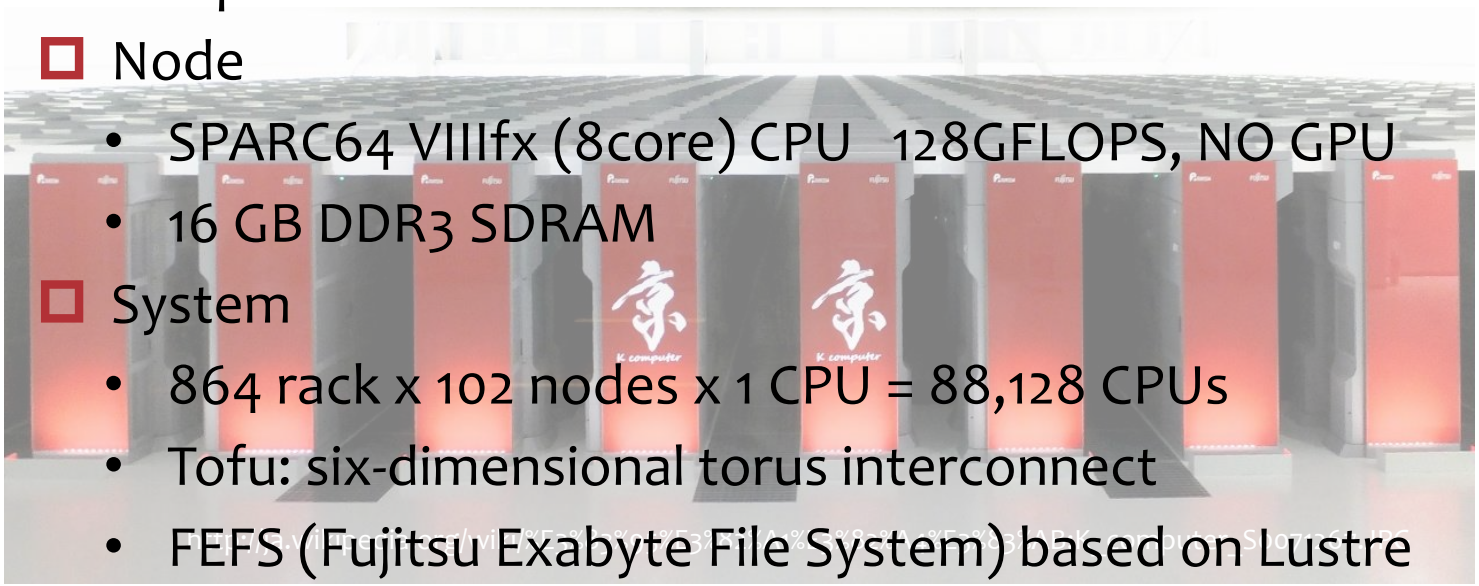


Experiments

- Block Gauss Jordan on K-computer
  - YML/XMP
- Block dgemm on GPU cluster
  - YML/XMP-dev/StarPU
- MIRAM on K-computer
  - YML/XMP + various parallel numerical libraries

Experiments (1) BGJ on K-Computer

- K-Computer
  - Node
    - SPARC64 VIIIfx (8core) CPU 128GFLOPS, NO GPU
    - 16 GB DDR3 SDRAM
  - System
    - 864 rack x 102 nodes x 1 CPU = 88,128 CPUs
    - Tofu: six-dimensional torus interconnect
    - FEFS (Fujitsu Exabyte File System) based on Lustre



- Block Gauss Jordan B=A^{^-1}
  - Compute the inversion of a matrix by computing the inversion of a block and updating other blocks repeatedly

Experiments (1) BGJ on K-Computer

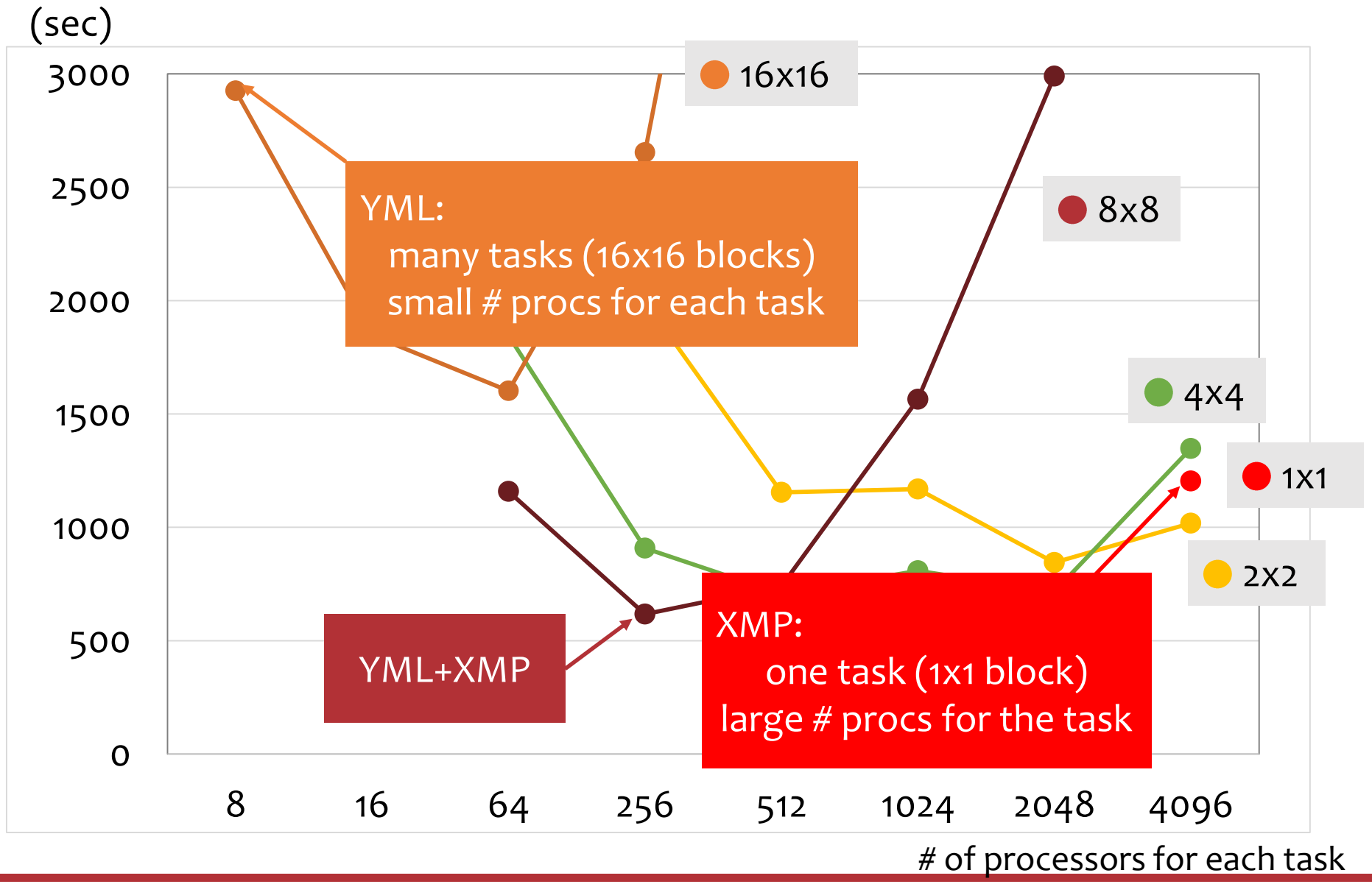
- Investigate different levels of hierarchical parallelism
  - the total size of matrix is fixed, but the number of blocks is varied
  - the total number of processes for a workflow is fixed, but the number of processes for each task is varied.

    ↓
  - "1x1 blocks & all processes for a task" ≒ distributed parallel program
  - A small # of processes for a task ≒ traditional workflow (the original YML)
- 32,768 x 32, 768 matrix

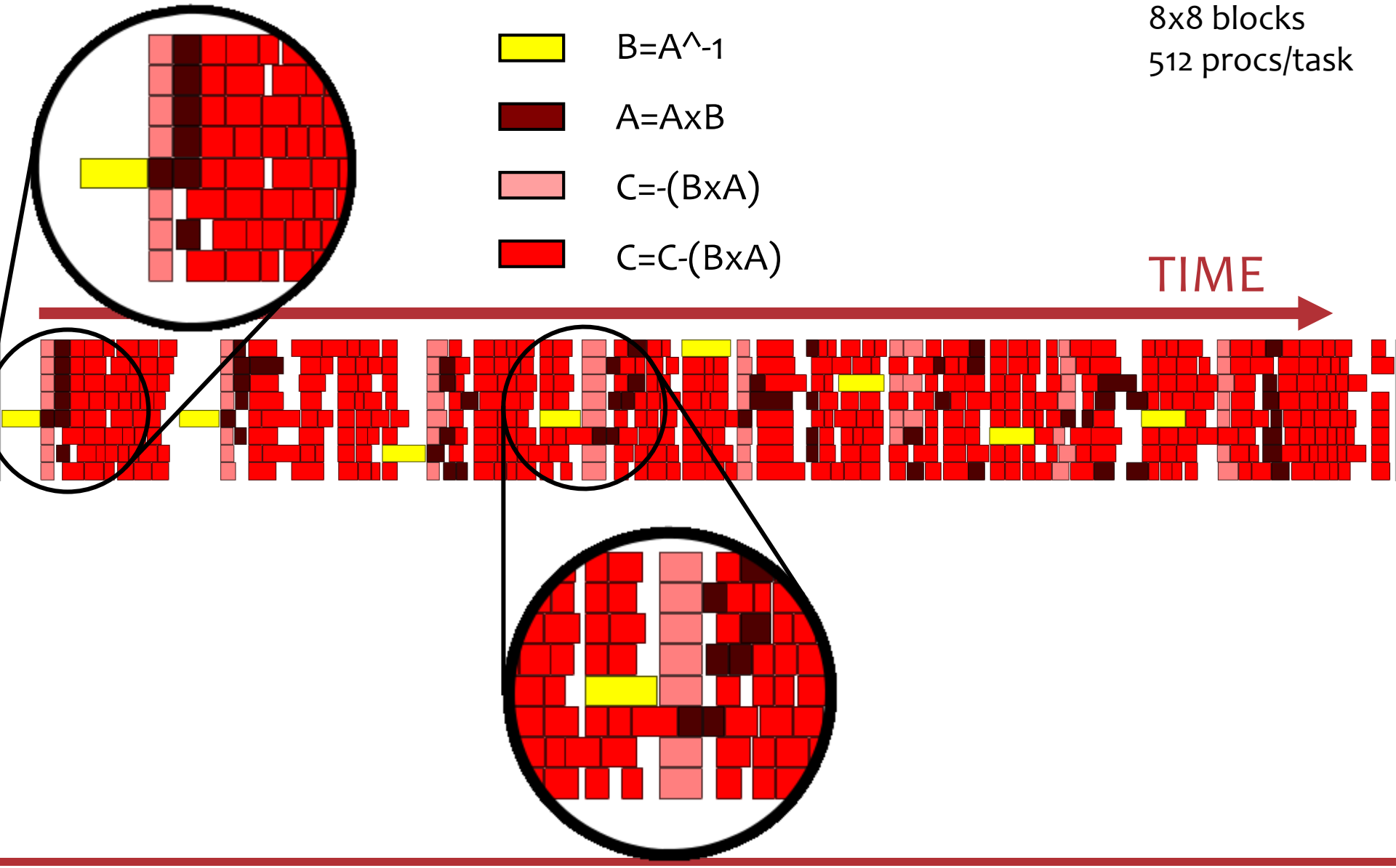
blocks	1x1	2x2	4x4	8x8	16x16
block size	32768	16384	8192	4096	2048

- 4096 processes for a workflow
  - 8~4096 processes for a task
  - (If 512 processes for a task, at most 8 tasks can be executed at the same time)

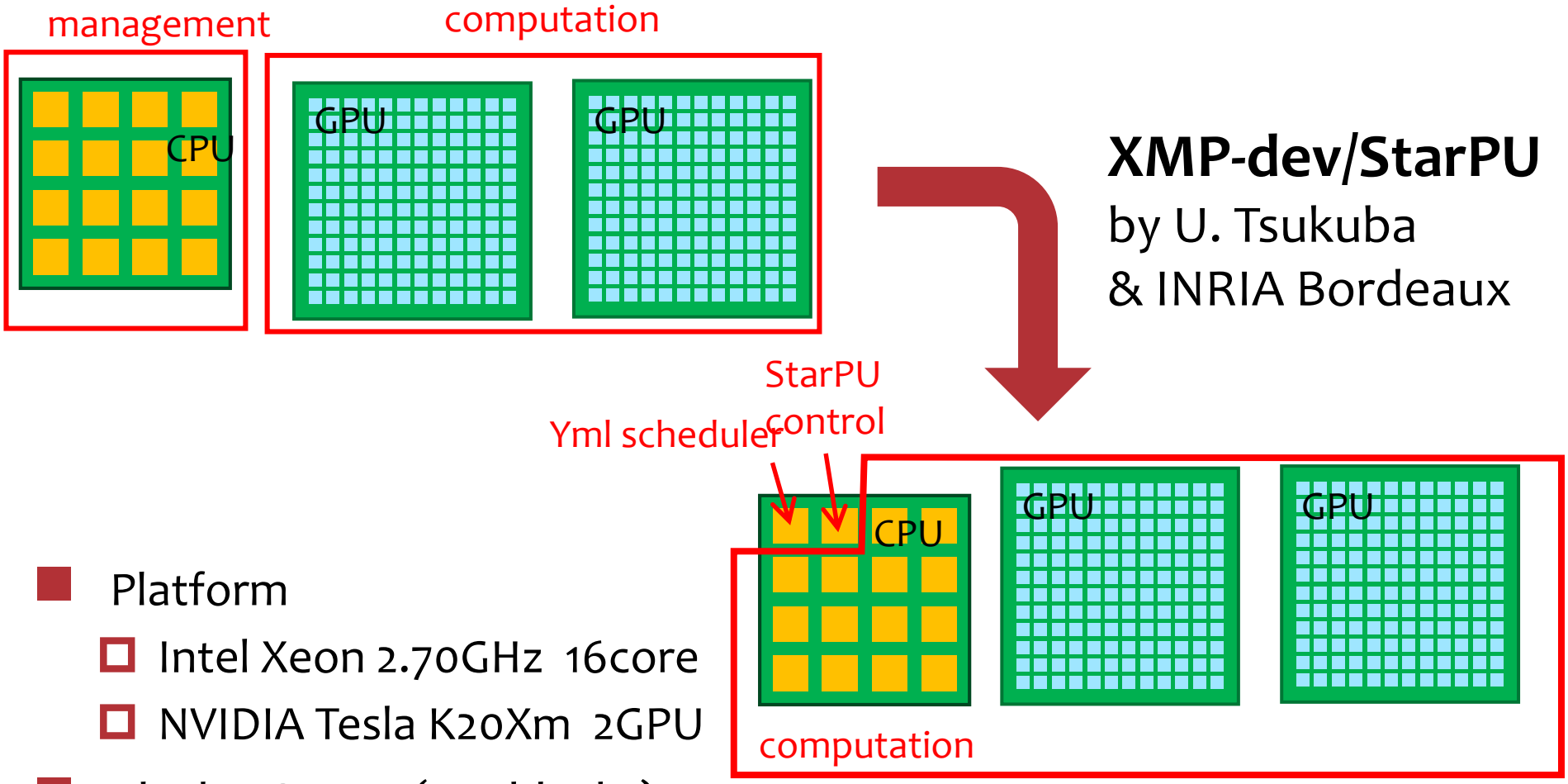
Experiments (1) BGJ on K-Computer



Block Gauss Jordan Execution Time Line

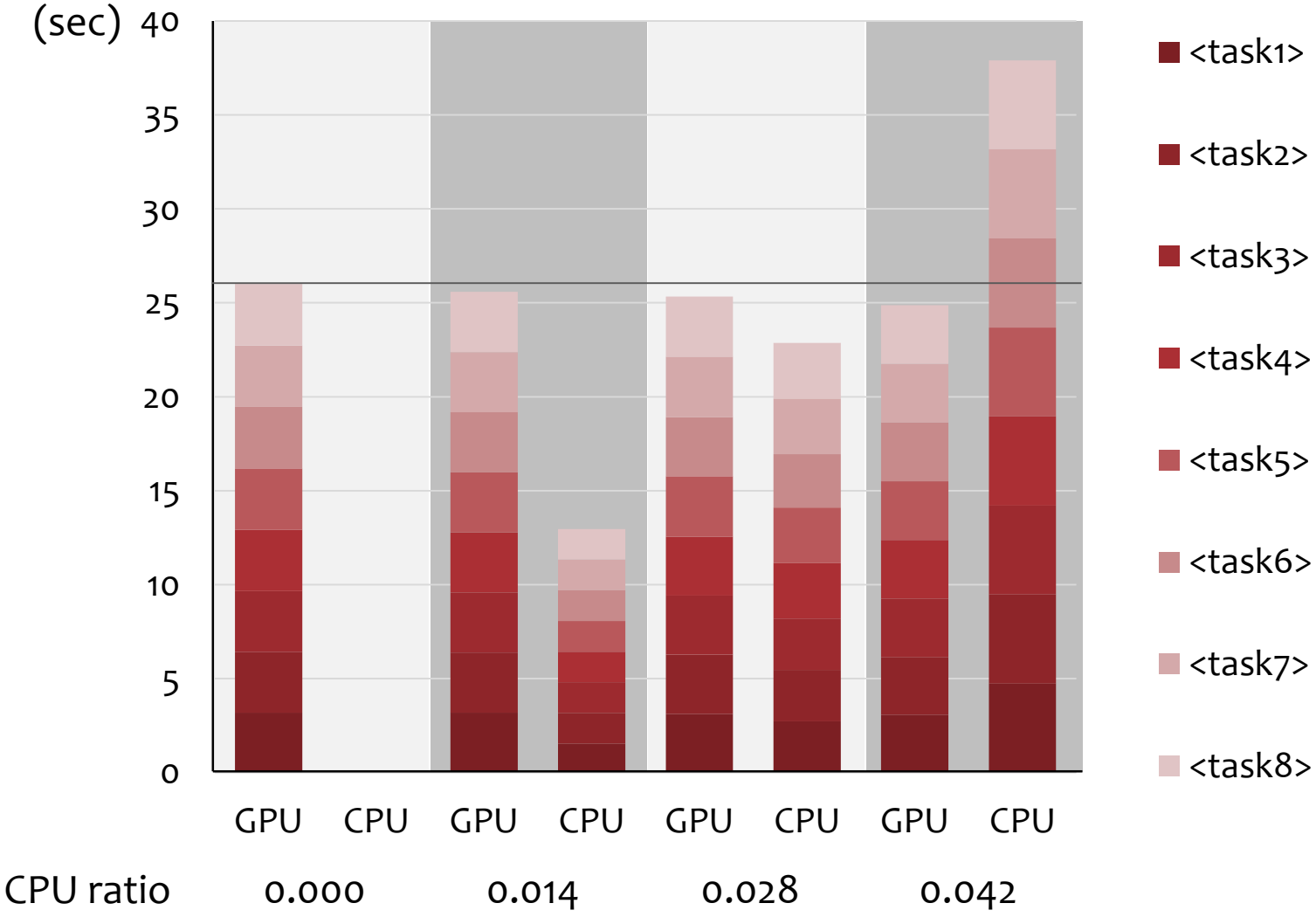


Experiments (2) YML+XMP-dev+StarPU



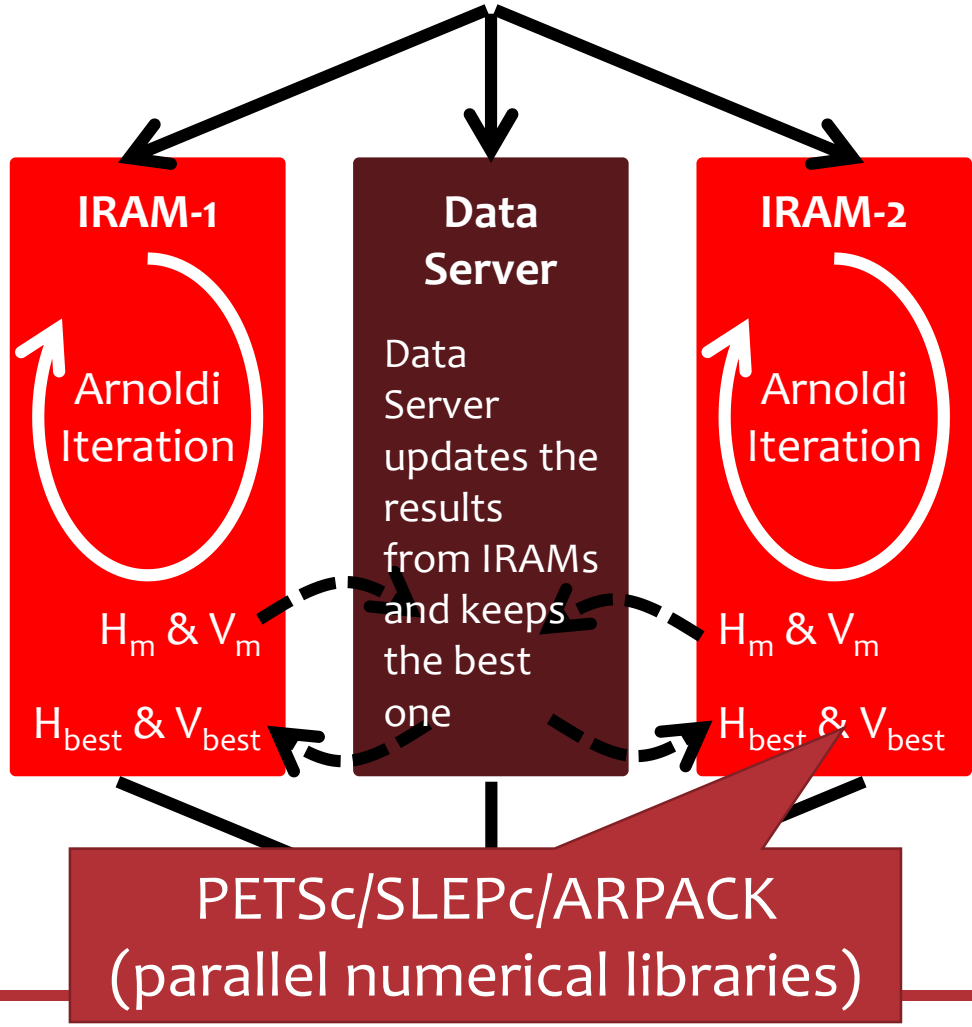
- Platform
  - Intel Xeon 2.70GHz 16core
  - NVIDIA Tesla K20Xm 2GPU
- Block DGEMM (2x2 blocks)
  - 10000 x 10000 matrix (-> 5000x5000 block)

Experiments (2) YML+XMP-dev+StarPU

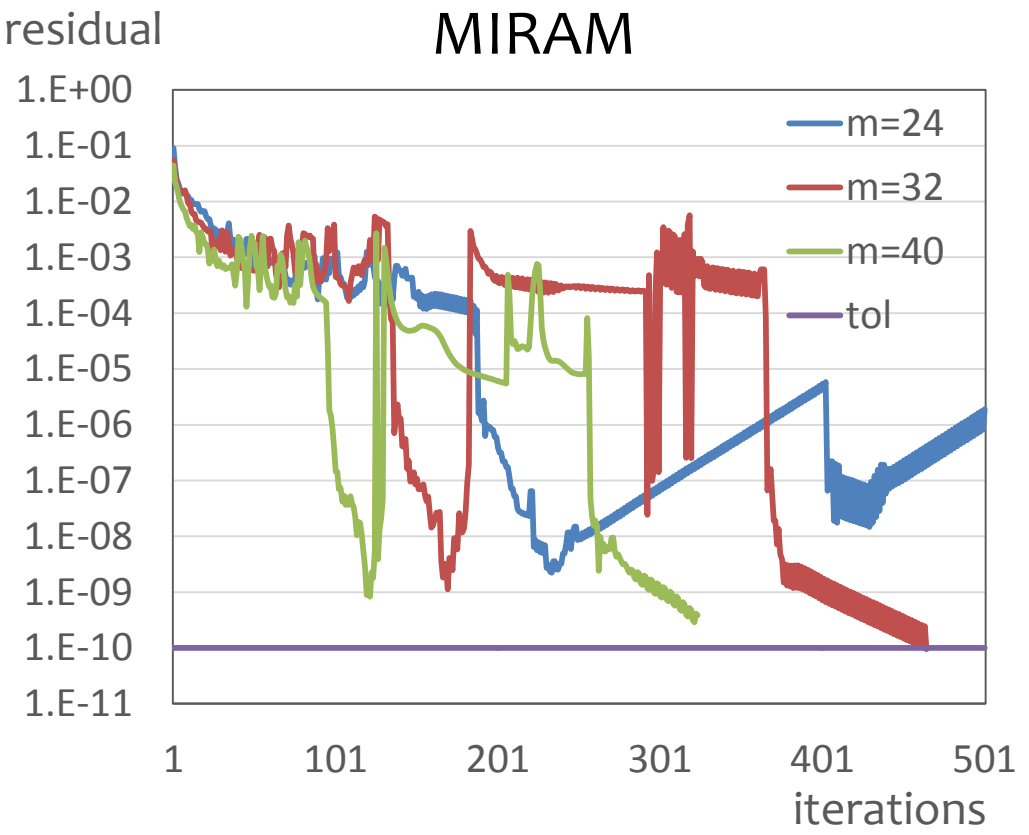


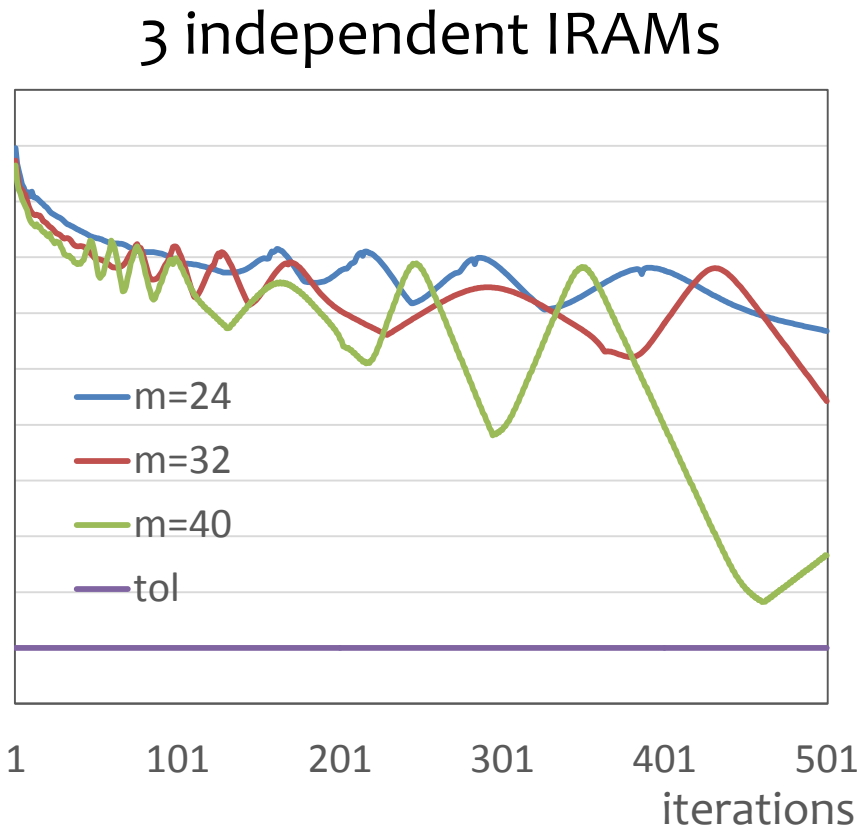
MIRAM Multiple Implicitly Restarted Arnodi Method

- IRAM (Implicitly Restarted Arnodi Method)
  - Iterative methods to obtain eigen pair of a matrix
- MIRAM
  - hybrid iterative method
  - invokes several IRAMs with different parameters
  - exchanges information between IRAMs to speedup convergence
- Schenk/nlpkkt240 (the largest matrix in the UF Sparse Matrix Collection)
  - rows x cols 27,993,600^2
  - # of non-zeros 760,648,352
- **K-computer**



MIRAM: Speedup Convergence

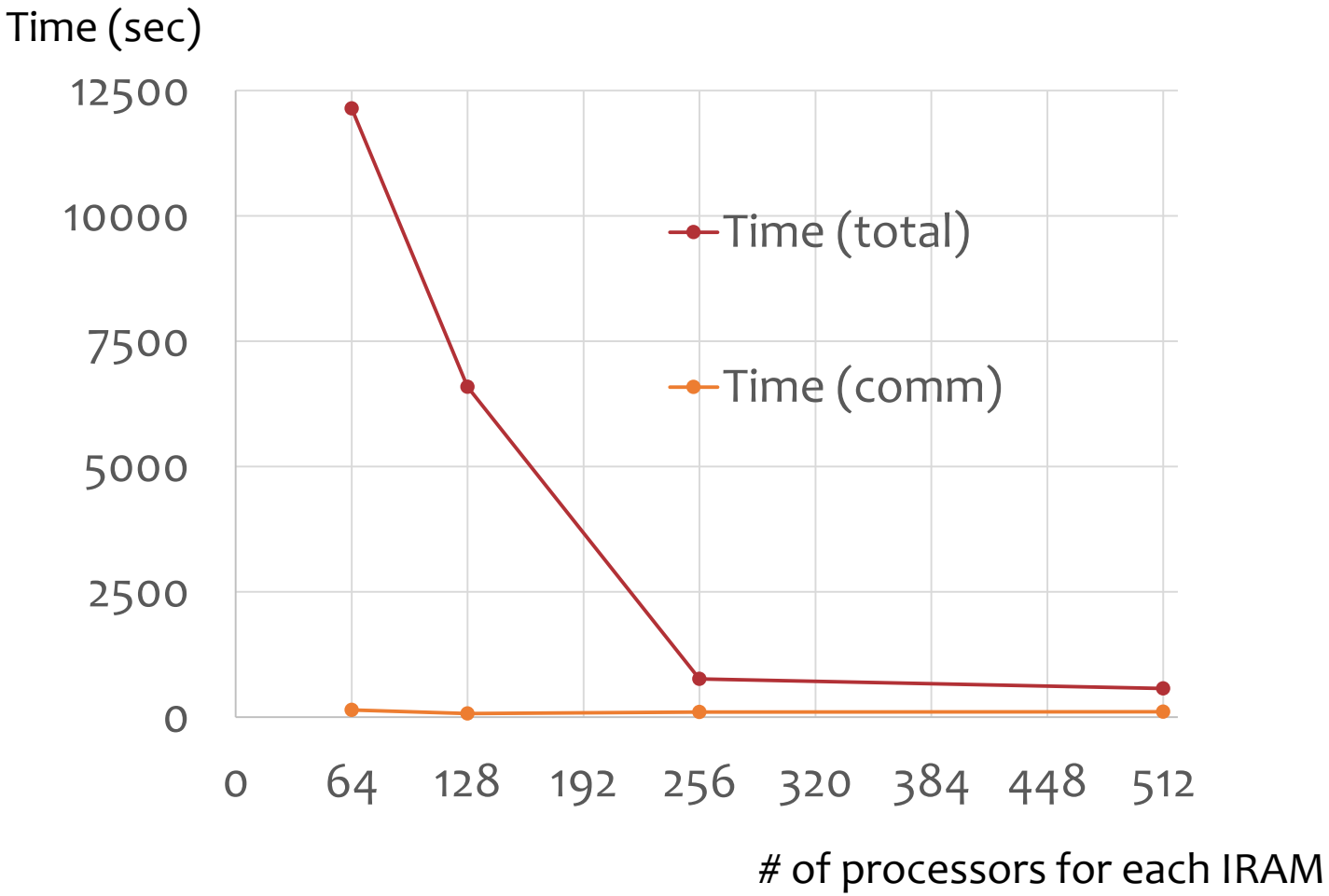




We can reduce the number of iterations!

MIRAM: Speedup Execution Time

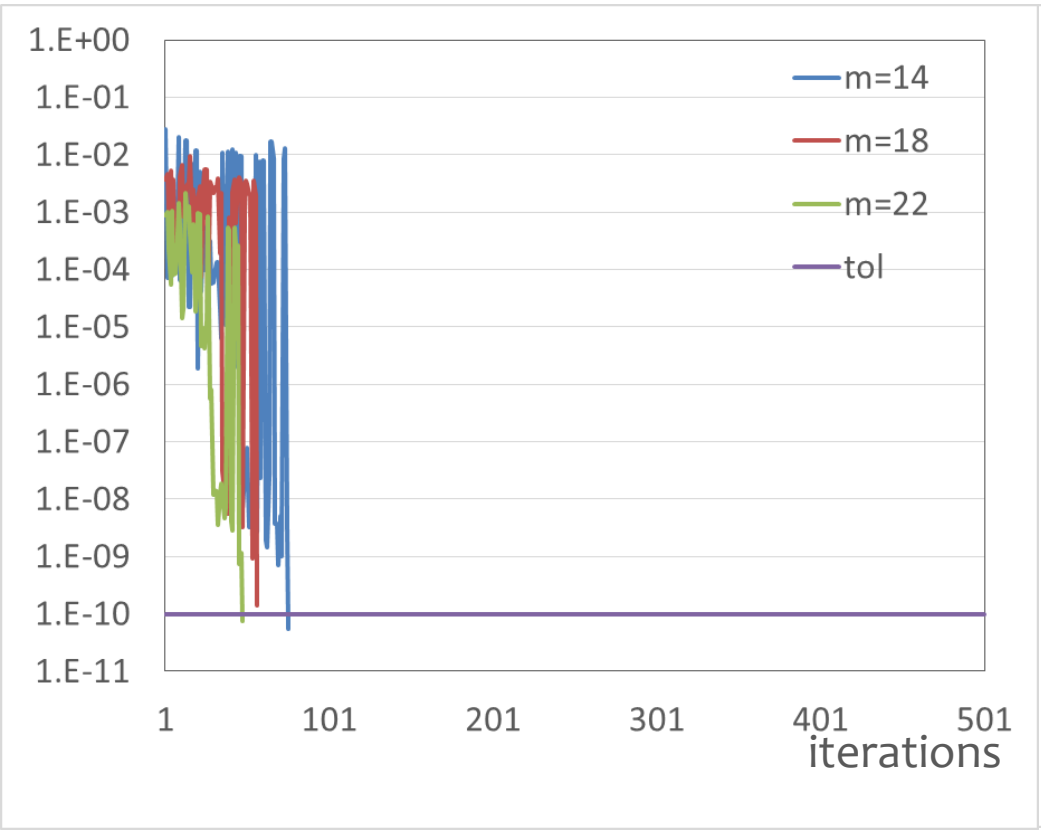
64, 128, 256, 512 cores for each IRAM on K-Computer



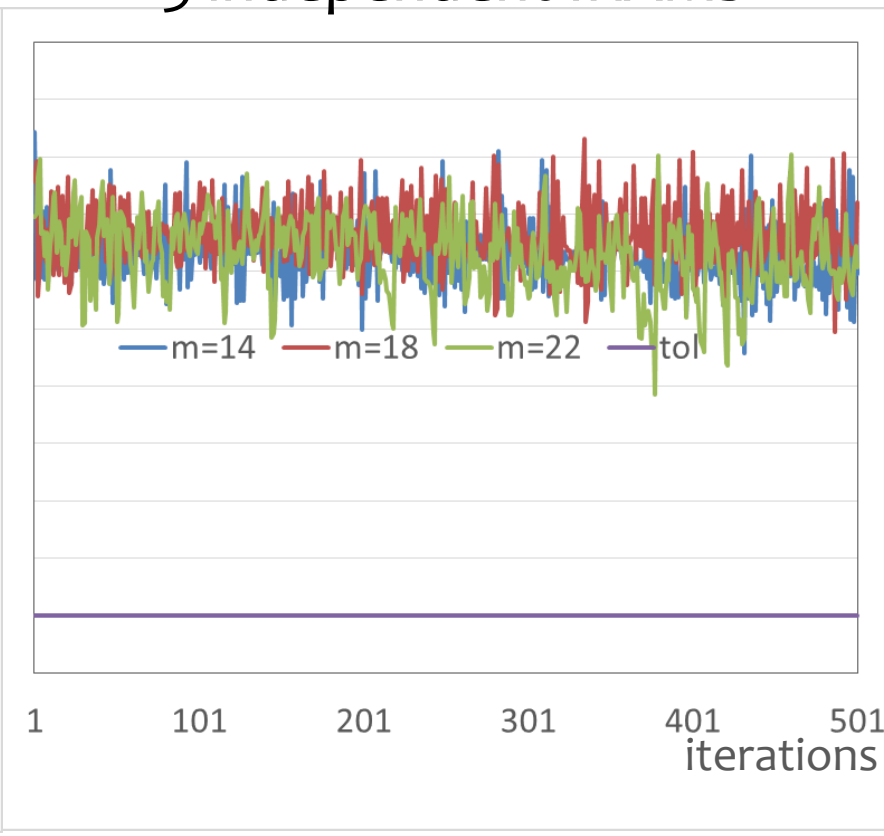
MIRAM: Speedup Convergence

af23560 (23560 x 23560)

MIRAM



3 independent IRAMs



CONCLUSION

- FP2C
  - multi-programming methodologies across hierarchical and heterogeneous architectures
  - YML + XMP(-dev) + StarPU
  - Combination of Japanese and French techniques
- YML+XMP >> YML, XMP on K-computer
- Uniform approach for heterogeneous process units
  - Within a task, heterogeneous process units can be used in a uniform approach by XMP-dev/StarPU
- 2 kinds of speedup based on 2 different parallel models for MIRAM with FP2C
  - speedup convergence based on workflow programming model
  - speedup each iteration based on distributed parallel programming model