



Parallel Matrix Computing on Peer to Peer Platforms.

Serge G. Petiton
petiton@lifl.fr

Outline

Goals and hypothesis

Middlewares and platforms

Algorithms and Experimentations

Programming paradigm and framework

Perspectives and Conclusion

Goals and hypothesis

Questions :

Is Scientific Parallel Computing adaptable to *large scale* P2P Platforms?

Or, is it possible to define a new programming paradigm for scientific computing on large scale P2P platforms? Other than only task-farming programming.

What algorithms for classical linear algebra problems?

Is the class of well-adapted methods large enough?

What language and framework for the **end-users**?

What evaluation and economic model?

Questions :

Is Scientific Parallel Computing adaptable to large scale P2P Platforms?

Or, is it possible to define a new programming paradigm for scientific computing on large scale P2P platforms? Other than only task-farming programming.

What algorithms for classical linear algebra problems?

Is the class of well-adapted methods large or limited?

What language and framework for the **end-user**?

What evaluation and economic models?

Linear algebra on such platforms.

Theoretical evaluations, simulations, **experimentations** and emulations.

Framework based on a Component oriented graph **language** for the end-user.

Large scale P2P computation

*P2P scientific computing is not an alternative to supercomputers or large supercomputing centers. The idea is to use existing networks of computers when there are idles, to increase the available resources for large scientific applications and to target **new** end-users.*

Large number of computers (several thousand of PCs)

- Heterogeneous
- Properties unknown during programming by the end-user

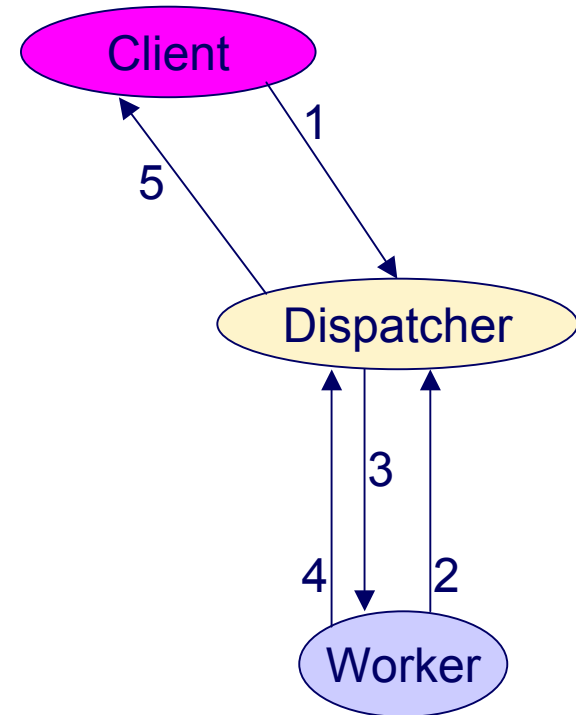
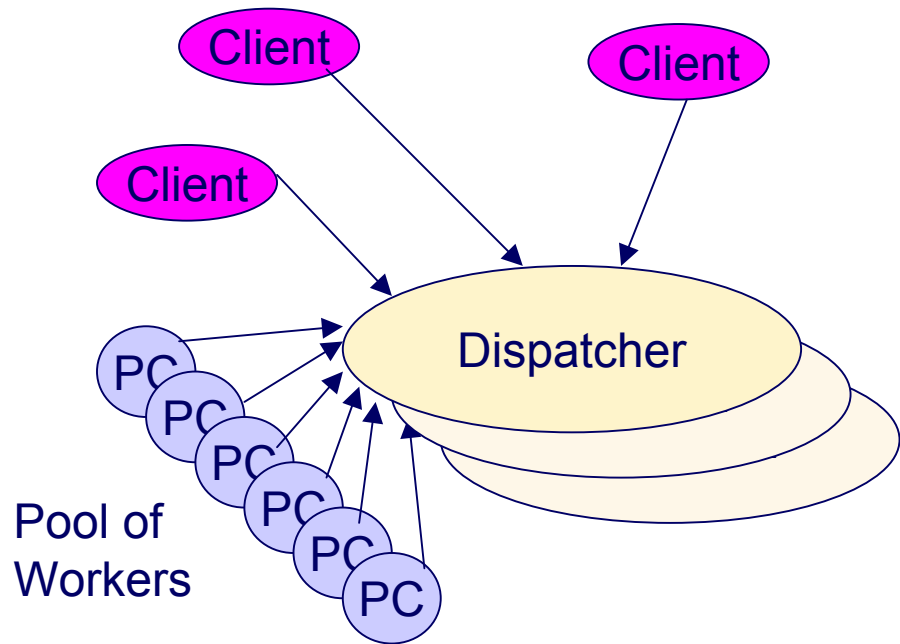
Large scale P2P computing, hypothesis

- Each peer can :
 - compute
 - Send and receive data
 - Participate to the system coordination*
- Several thousand of peers targeted :
 - Between 4Gflops and 250Mflops per peer
 - Different Memory size, between 1Gbytes and 128Mbytes
 - Different software on each peer (BLAS and LAPACK available)
- Fault tolerant system (including MPI communications)
 - Indeterminism
 - MPICH-V as a Fault-Tolerant MPI ; with multicasts

Middlewares and Platforms

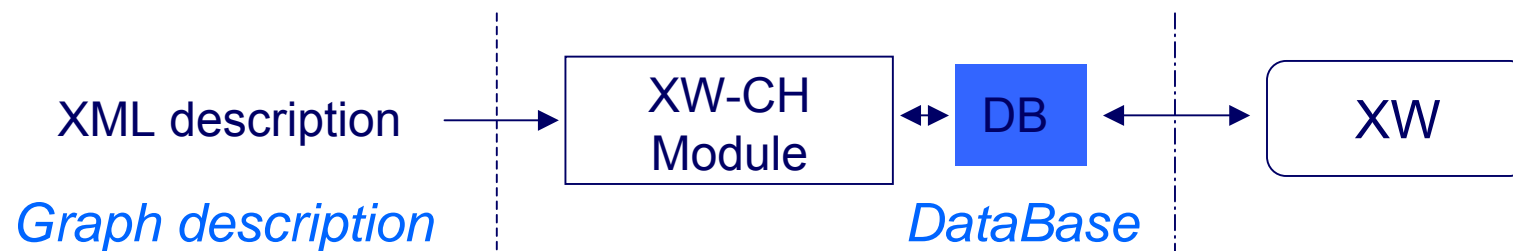
XtremWeb

- P2P experimental platform
 - Dispatchers, Workers, Clients



XtremWeb-CH

- A decentralized version of XtremWeb (university of Geneva, CH).
- It consist of a scheduling module, in the top of XtremWeb, which inserts the tasks on the fly through an XML description.



- Workers can communicate directly and send their results directly to clients and thus limit the bandwidth consumption.
- NO ANY MORE FAULT TOLERANT SYSTEM!
- Important system overhead

Platforms

Two configurations for XtremWeb:

- A LAN based with **128** PCs, largely non-dedicated (Polytech-Lille, LIFL, INRIA).
- A WAN based with **128PCs+133PCs** distributed on two geographic sites (Polytech-Lille/LIFL and Paris XI university/INRIA), also non-dedicated.

One XtremWeb-CH testbed of **128** PCs is also deployed.

Platforms

Polytech'Lille/LIFL/INRIA Platform (128 PCs)

- 28 PIII, 450MHz,
- 81 Intel Celerons 600Mhz to 2.4 GHz,
- 15 AMD Duran, 750MHz
- 4 PIV, 2.4 GHz.

Memory size from
128MBytes to 512MBytes

Paris 11 Univ./LRI/INRIA (133 PCs)

- 101 AMD Athlon and K7, 7 bi proc., 600MHz to 2.8GHz,
- 10 PIV, 2GHz,
- 12 PIII, 500 MHz,
- 9 PII, 400 MHz.

Memory Size from
128Mbytes to 1Gbytes

Ethernets with heterogeneous speeds from 10Mbits/s to 100Mbits/s

No-dedicated to our experimentations : students and seti@home
are also using these computers

Algorithms, Evaluations and/or Experimentations

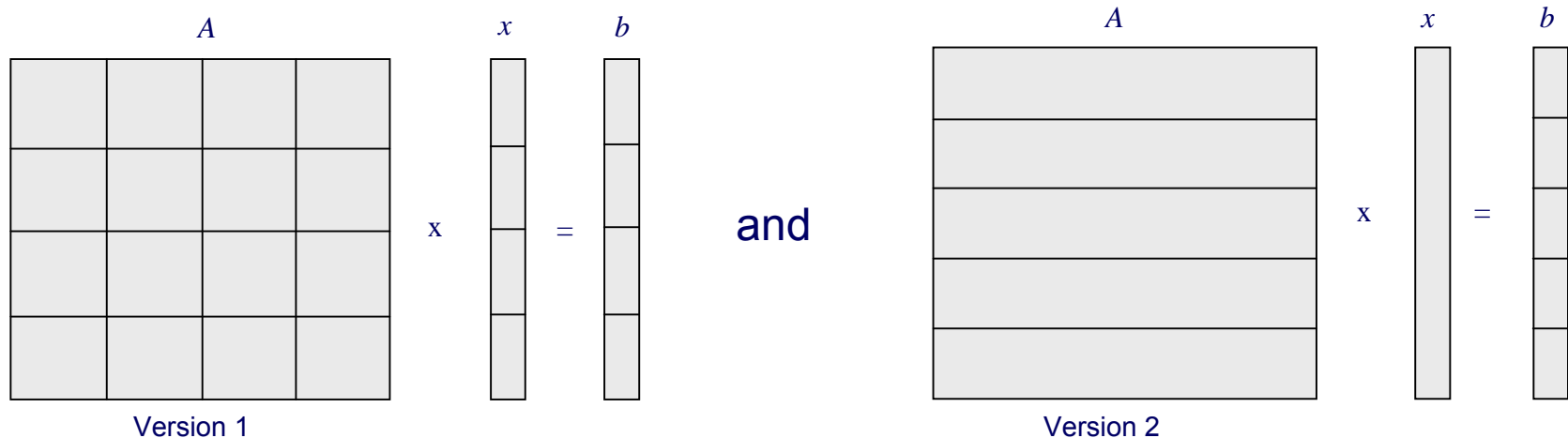
- BLAS2 computation (for iterative methods)
- Eigenvalues and eigenvectors
- Block Gauss-Jordan Method

Algorithms, Evaluations and/or Experimentations

- **BLAS2 computation (for iterative methods)**
- Eigenvalues and eigenvectors
- Block Gauss-Jordan Method

Block based matrix vector multiplication distributed with XtremWeb and XtremWeb-CH

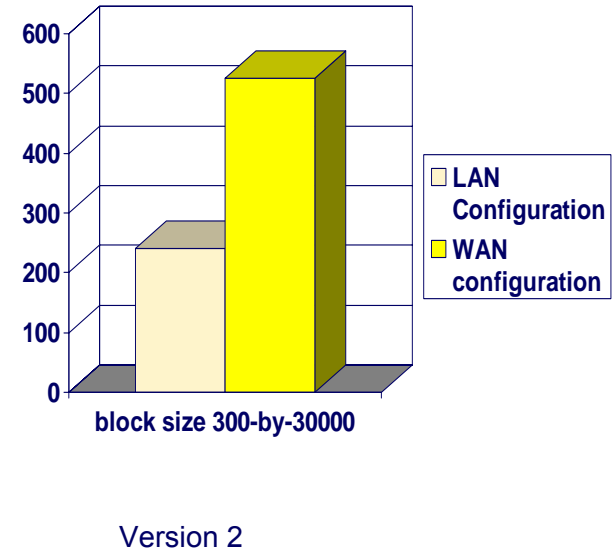
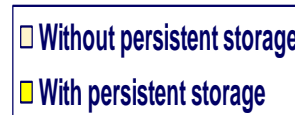
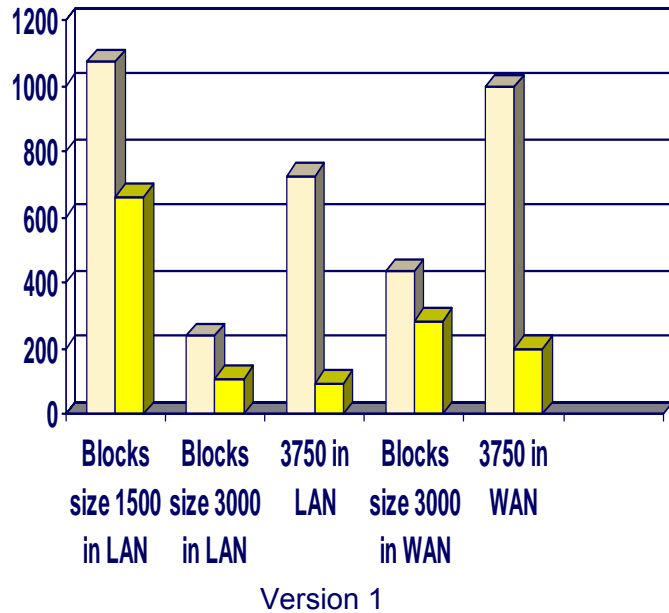
- The matrix vector product is a simple but fundamental operation in matrix computing. It forms the core of iterative methods (Krylov Method for example)
- It is often the primary operation around which supercomputers was benchmarked (ex. CMs).
- We use two algorithms of the block based matrix vector product as follow :



For iterative methods, the matrix A remains unchanged, this enables us to pre-deploy data and use “**persistent data storage**”.

Results

XtremWeb



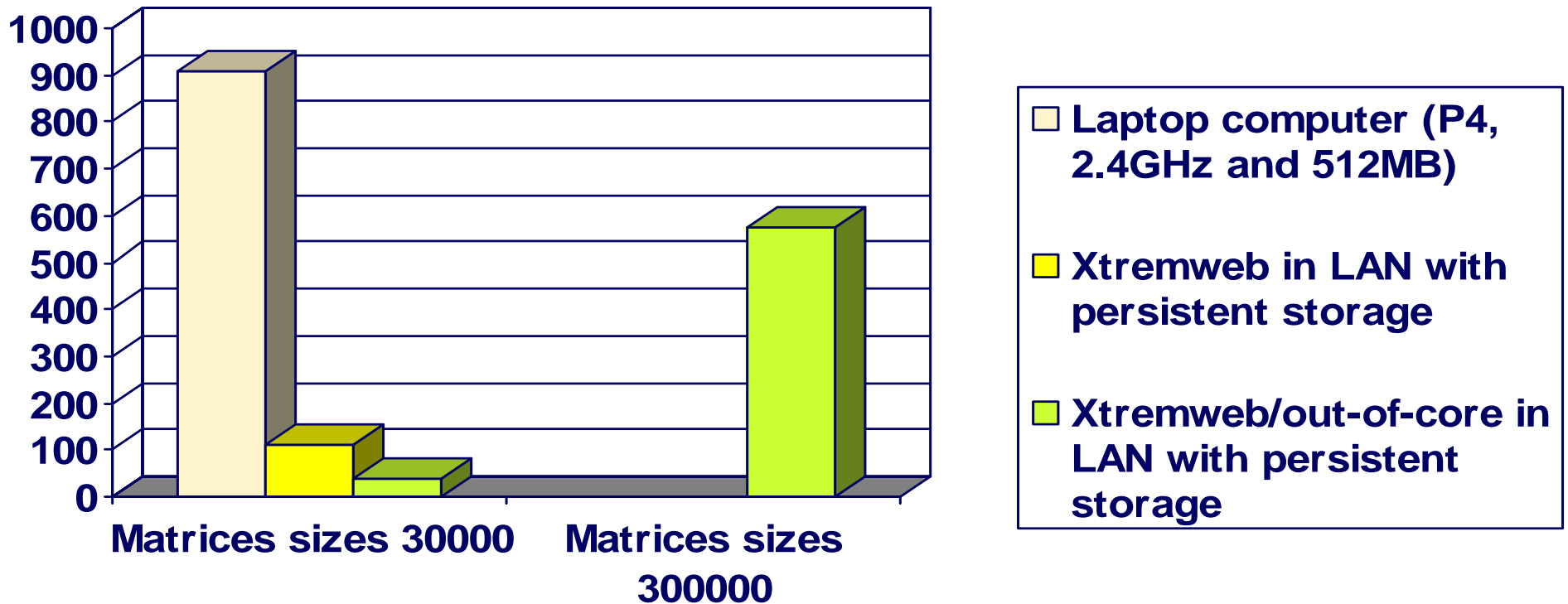
Matrix size = 30000, dense (7.2 G bytes)

We present here the average computing times (in seconds) for each execution.

For XtremWeb-CH, the average computing time, in LAN with blocks size 3000, is 642 seconds.

Out-of-core in Grid

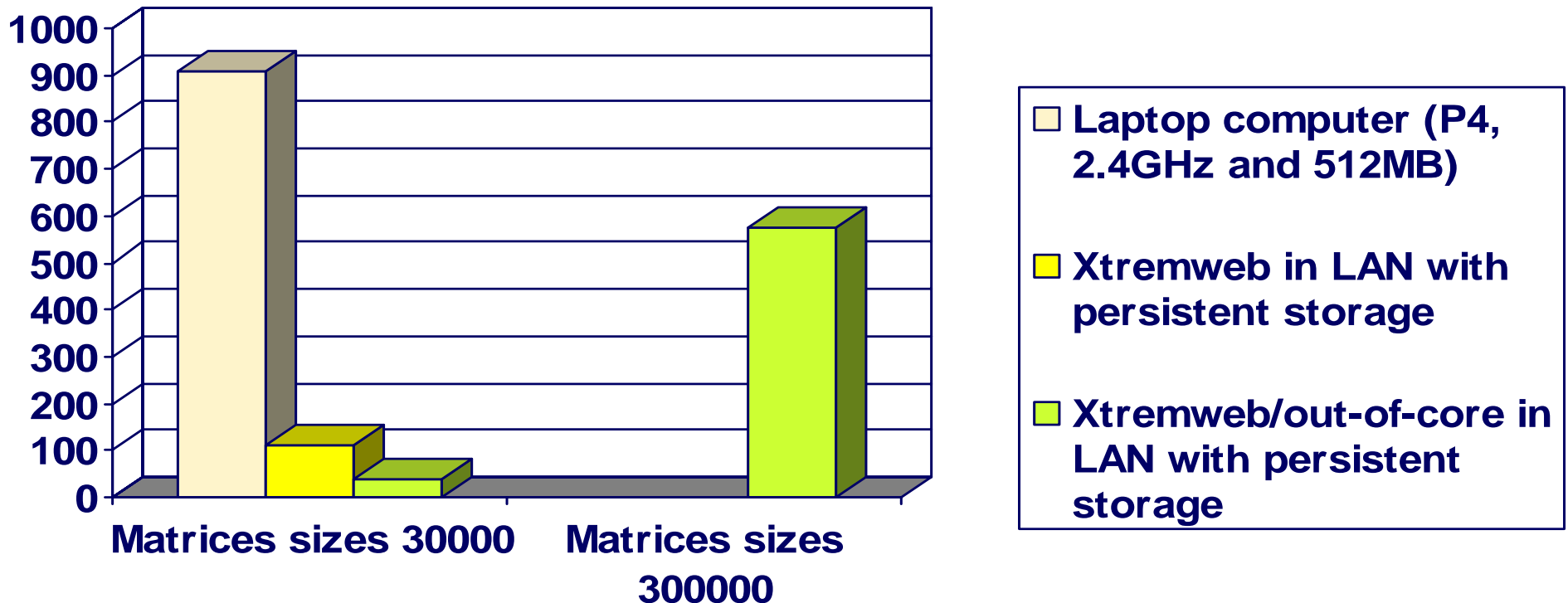
- The out-of-core technique solves large problems on standard computers. We will use it for increasing the size of problems on peers.
- We combine persistent storage to reduce communications and out-of-core to cover them.



We increase the size of matrix up to 300000 (100 times the initial matrix, close to 1 TeraBytes), the average computing time is 575 seconds

Out-of-core in Grid

- The out-of-core technique solves large problems on standard computers. We will use it for increasing the size of problems on peers.
- We combine persistent storage to reduce communications and out-of-core to cover them.



We increase the size of matrix up to 300000 (100 times the initial matrix, close to 1 TeraBytes), the average computing time is 575 seconds (400 Megaflops only!)

First remarks

- We need smart scheduler, at least with data persistency strategies.
- The communications are too long compare to the computation for this example.
- Out of core in one computer can be faster that with dozen of comparable computers using a P2P middleware such as XW.
- The future of iterative linear algebra method on P2P platforms is uncertain.
- Nevertheless, it depends of the evaluation and economic model we chose. The computers are used when idle and are already connected.
- We can have huge matrices with out-of-core and P2P middlewares
- Same conclusion approximately for matrix product.

Algorithms, Evaluations and/or Experimentations

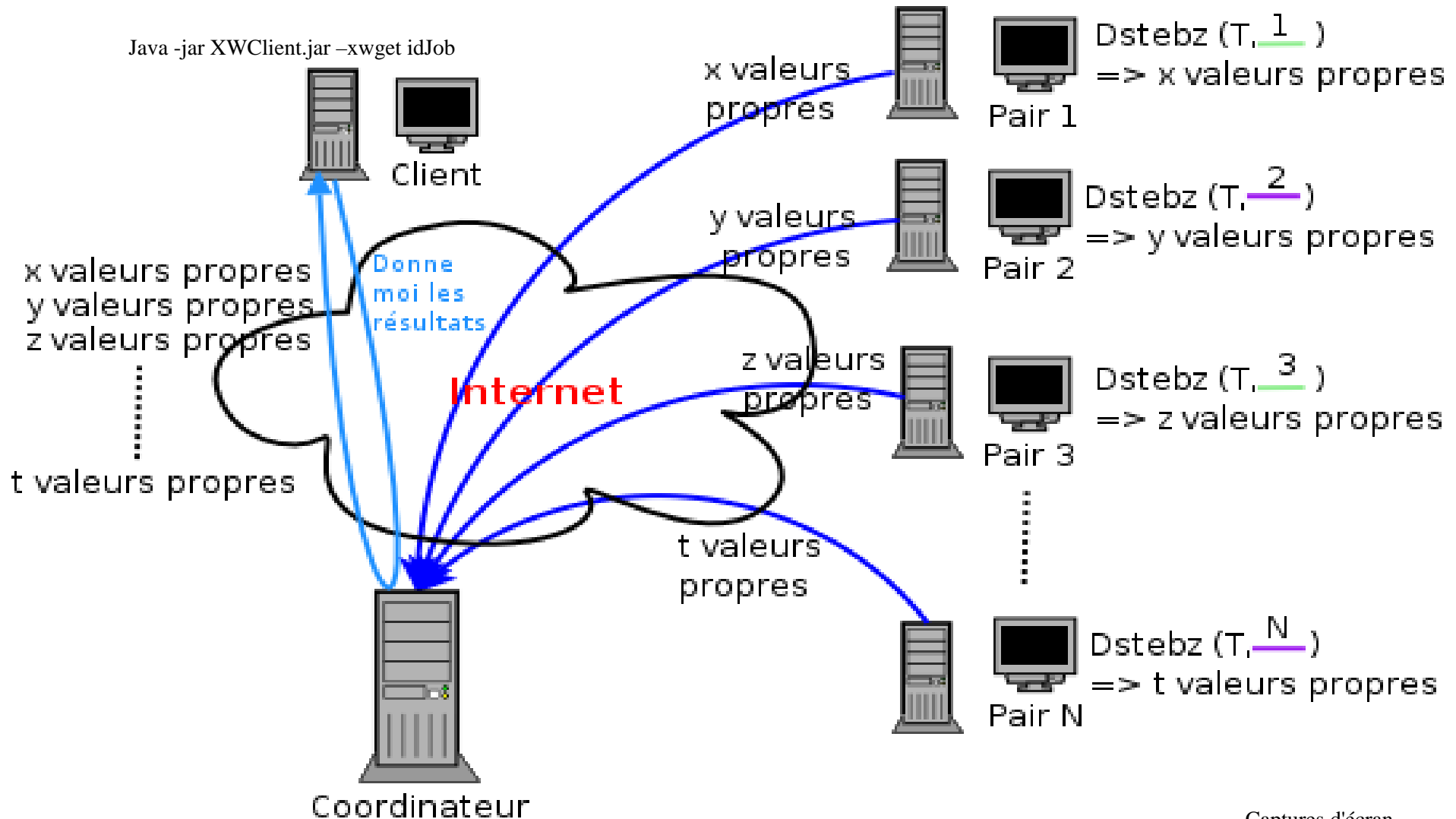
- BLAS2 computation (for iterative methods)
- **Eigenvalues and eigenvectors**
- Block Gauss-Jordan Method

Householder/Givens + Bisection Methods

- A Real Symmetric Matrix
- Phase 1
 - Translate to an Hessenberg Form Matrix
 - - Householder, or
 - - Givens.
 - $T = Q^t A Q$, Q orthogonal, T symmetrical in this case
 - LAPACK: DSBTRD
- Phase 2: bisection
 - $\text{recurrence}[T, x] = \text{number of eigenvalues larger than } x$
 - Gershgorin Domain permit to know where is the spectra approx.
 - LAPACK: DSTEBZ

$$\begin{pmatrix} \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ \bullet & \bullet & \bullet & \bullet & \bullet & \bullet \\ 0 & \bullet & \bullet & \bullet & \bullet & \bullet \\ \vdots & \ddots & & & & \\ \vdots & & \ddots & & & \\ 0 & \dots & \dots & 0 & \bullet & \bullet \end{pmatrix}$$

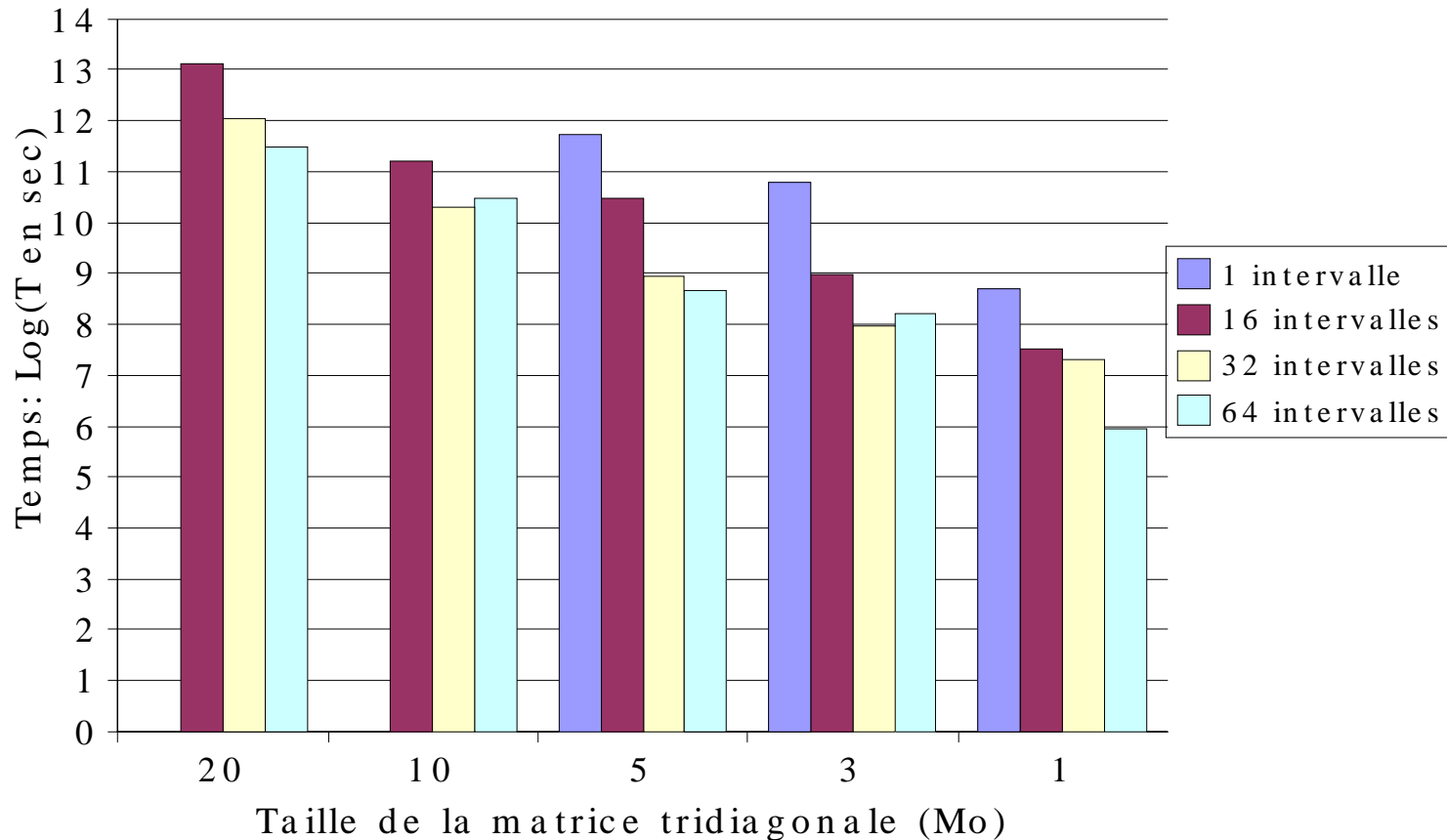
How to compute the eigenvalues of a symmetric real tridiagonal matrix?



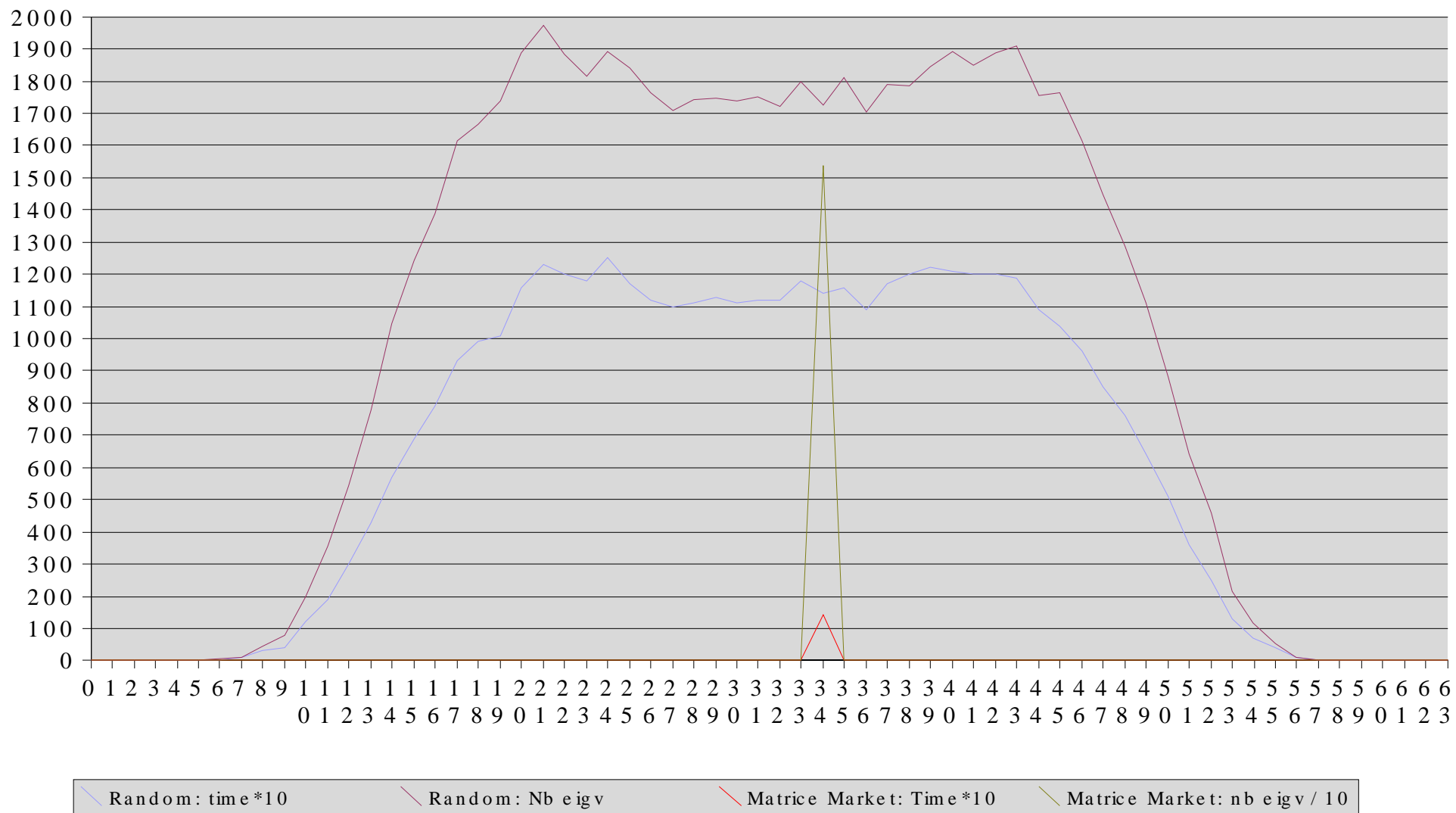
First results, LRI/INRIA Platform

- Dense, random tridiagonal real matrice
- Time evaluate from the XW web pages

Temps écoulé entre :
le début d'activité du premier pair et
la fin d'activité du dernier pair



- Bisection in 64 intervals for the MatrixMarket bcsstk25(n=15439) and random matrices
- Time in seconds



Optimization

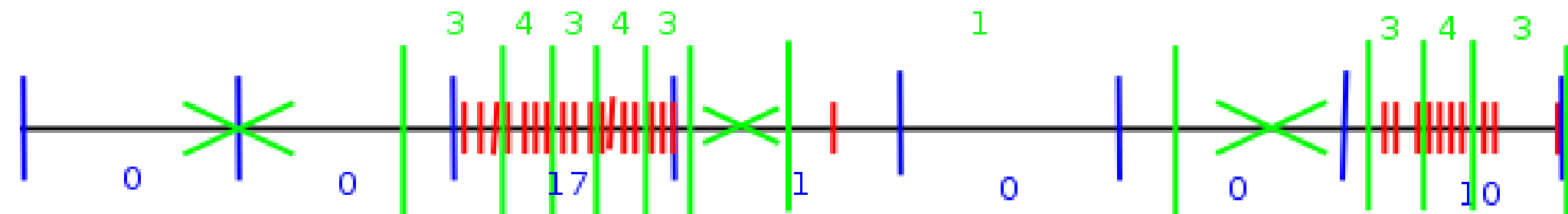
- Compute the intervals with respects to the eigenvalue distribution

DLAEBZ(IJOB=1, T, [b_inf; b_sup])

recurrence[T, b_inf] = number of eigenvalues > b_inf

recurrence[T, b_sup] = number of eigenvalues > b_sup

=> Number of eigenvalue in [b_inf; b_sup]



28 valeurs propres dans l'intervalle

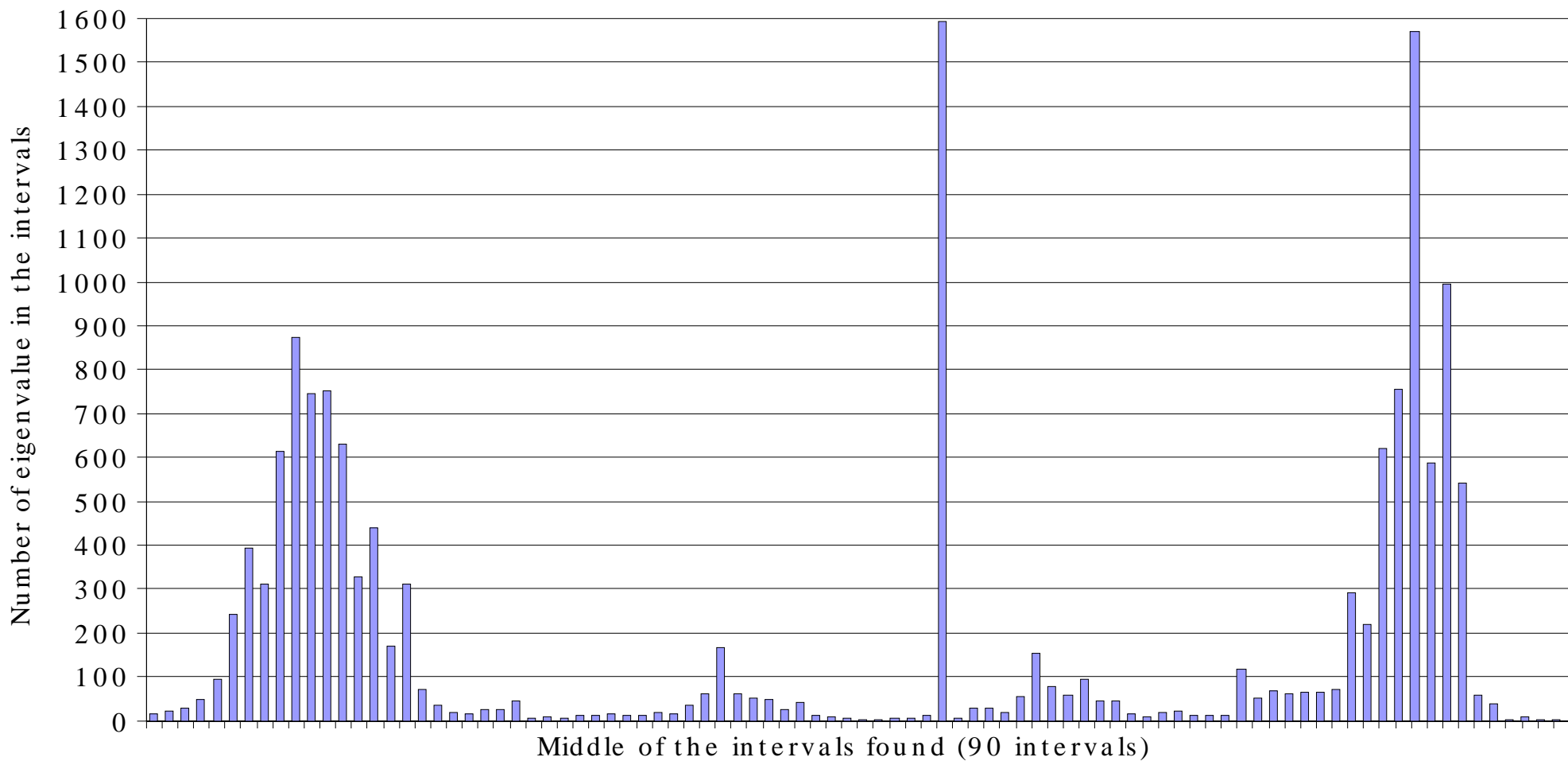
Ancienne méthode: partage en 7 sous intervalles

(Objectif : 4 valeur/intervalle?)

Nouvelle méthode: 4 valeurs propres maximum par sous intervalles

New Eigenvalue Distribution into peers

bcsstk25 - maximum threshold of eigenvalues = 1600



Task farming programming paradigm well-adapted to compute eigenvalue of tridiagonal symmetric real matrices

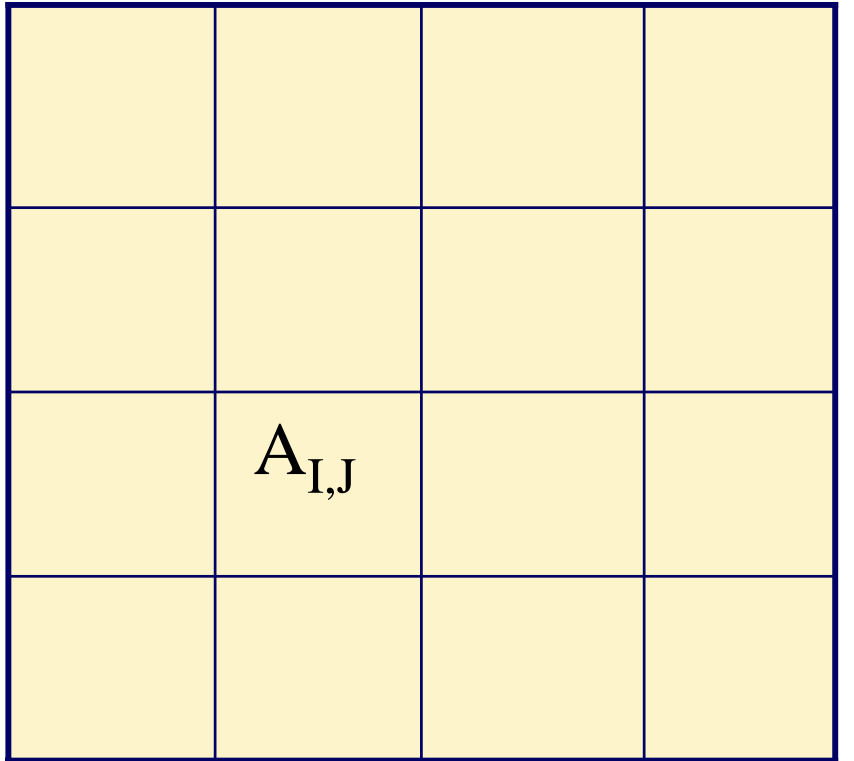
For non symmetric matrices, we have to use the Givens method to tridiagonalize the matrices – with redundant computing on each peer.

We can after compute the eigenvectors on each peer where was computed previously the associated eigenvalues.

The first results seems very interesting. Too many parameters are to be analysed to present complete results today.

Algorithms, Evaluations and/or Experimentations

- BLAS2 computation (for iterative methods)
- Eigenvalues and eigenvectors
- **Block Gauss-Jordan Method (direct method)**



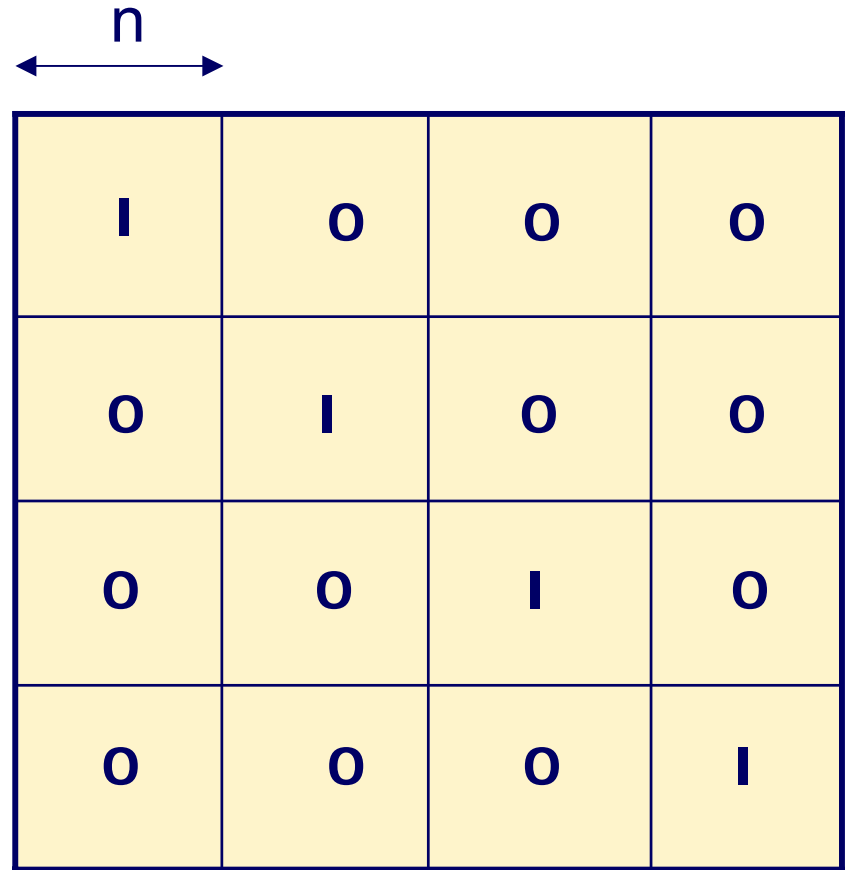
p

$$A B = B A = I$$

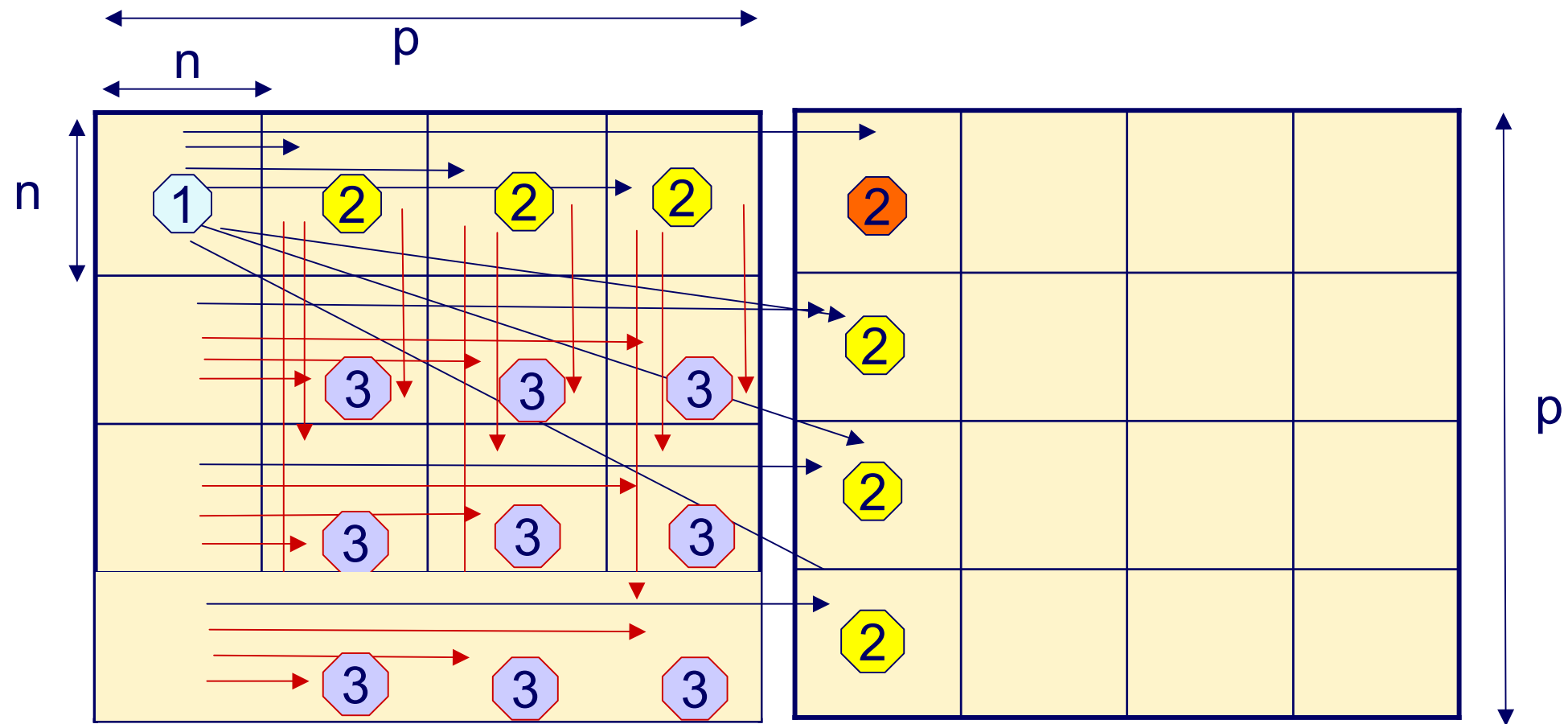
Block Gauss-Jordan

Matrix size = $N = p n$

B =



To invert a matrix
 $2N^3$ operations
 Challenge : $N = 10^6$

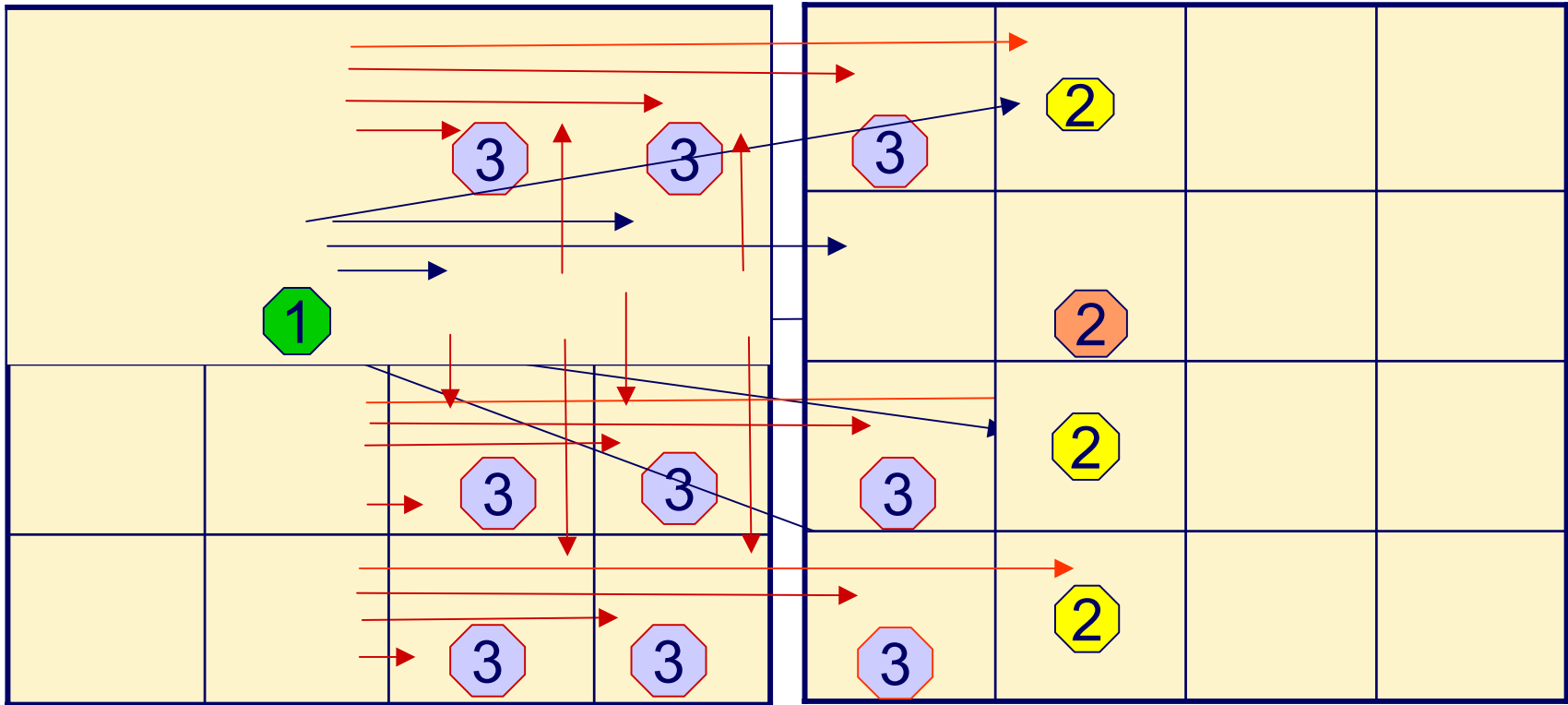


1 Element Gauss-Jordan, LAPACK, $cx = 2n^3 + O(n^2)$

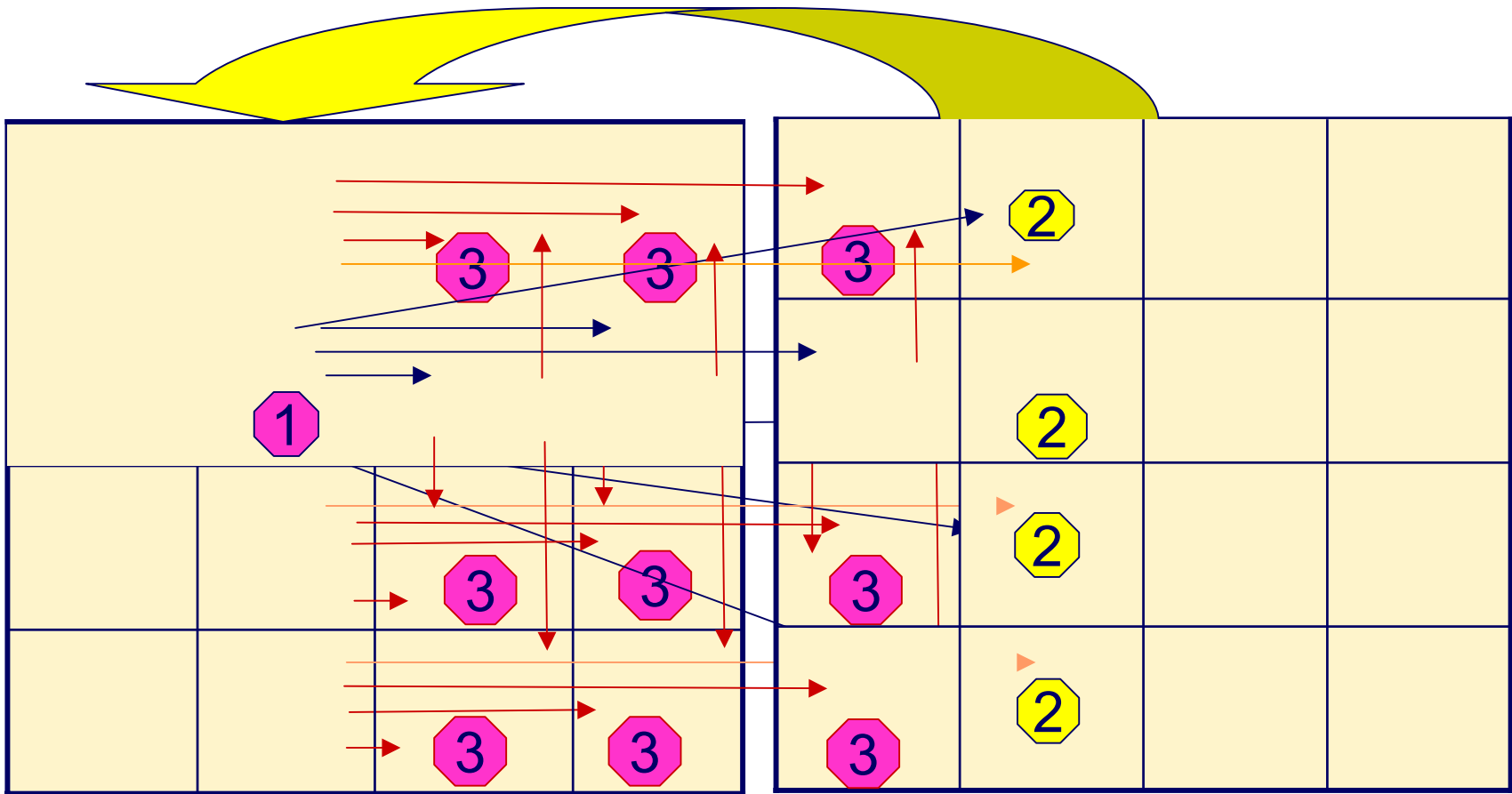
2 $A = +/- A B$; BLAS3, $cx = 2 n^3 - n^2$, 2 $A = B$

3 $A = A - B C$; BLAS3, $cx = 2n^3$

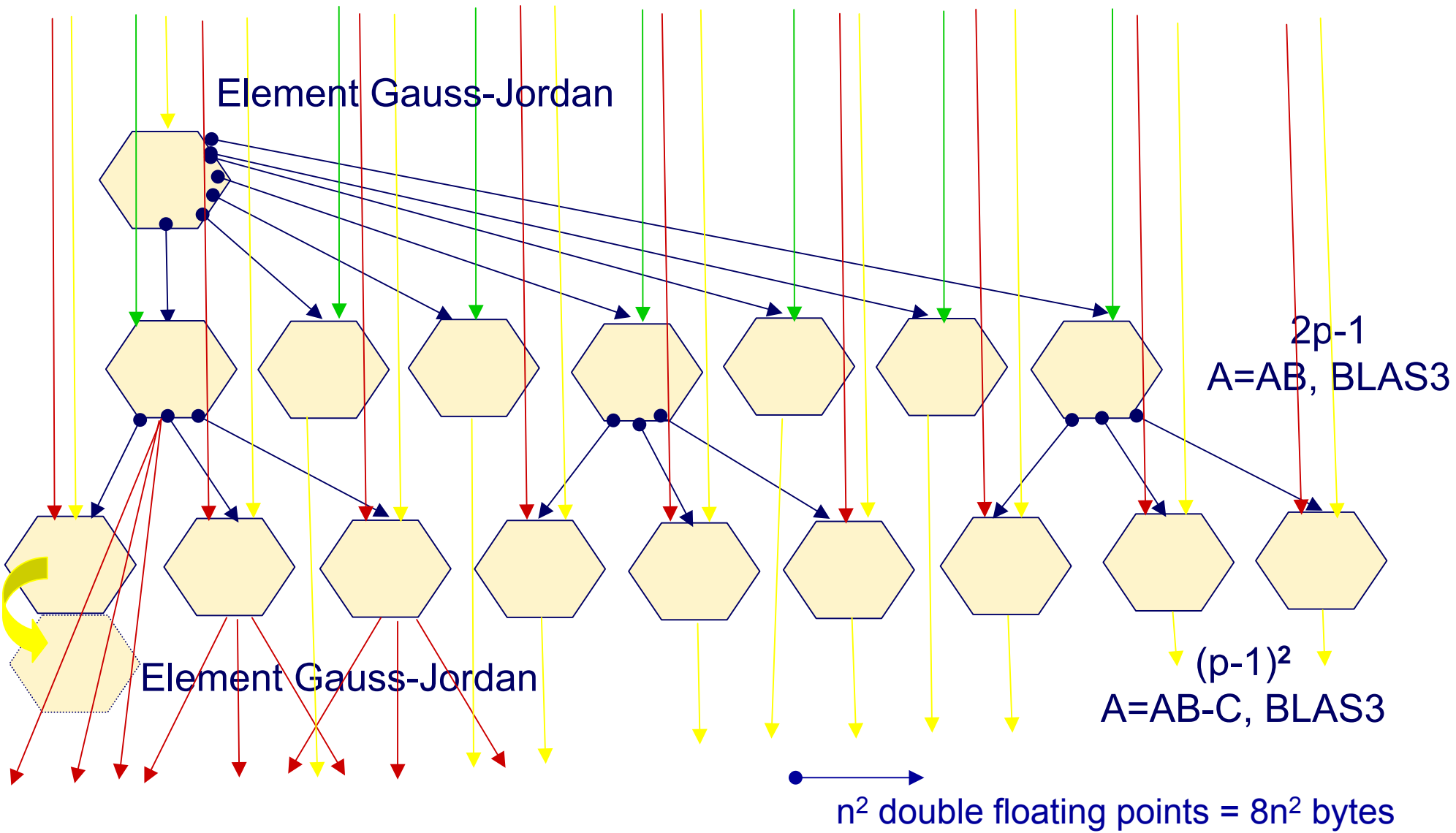
→ n^2 64 bit floating point numbers



Each computing task : 1 up to 3 blocks
 maximum $n < (\text{memory size of one pair}) / 3$
 Up to $(p-1)^2$ peers

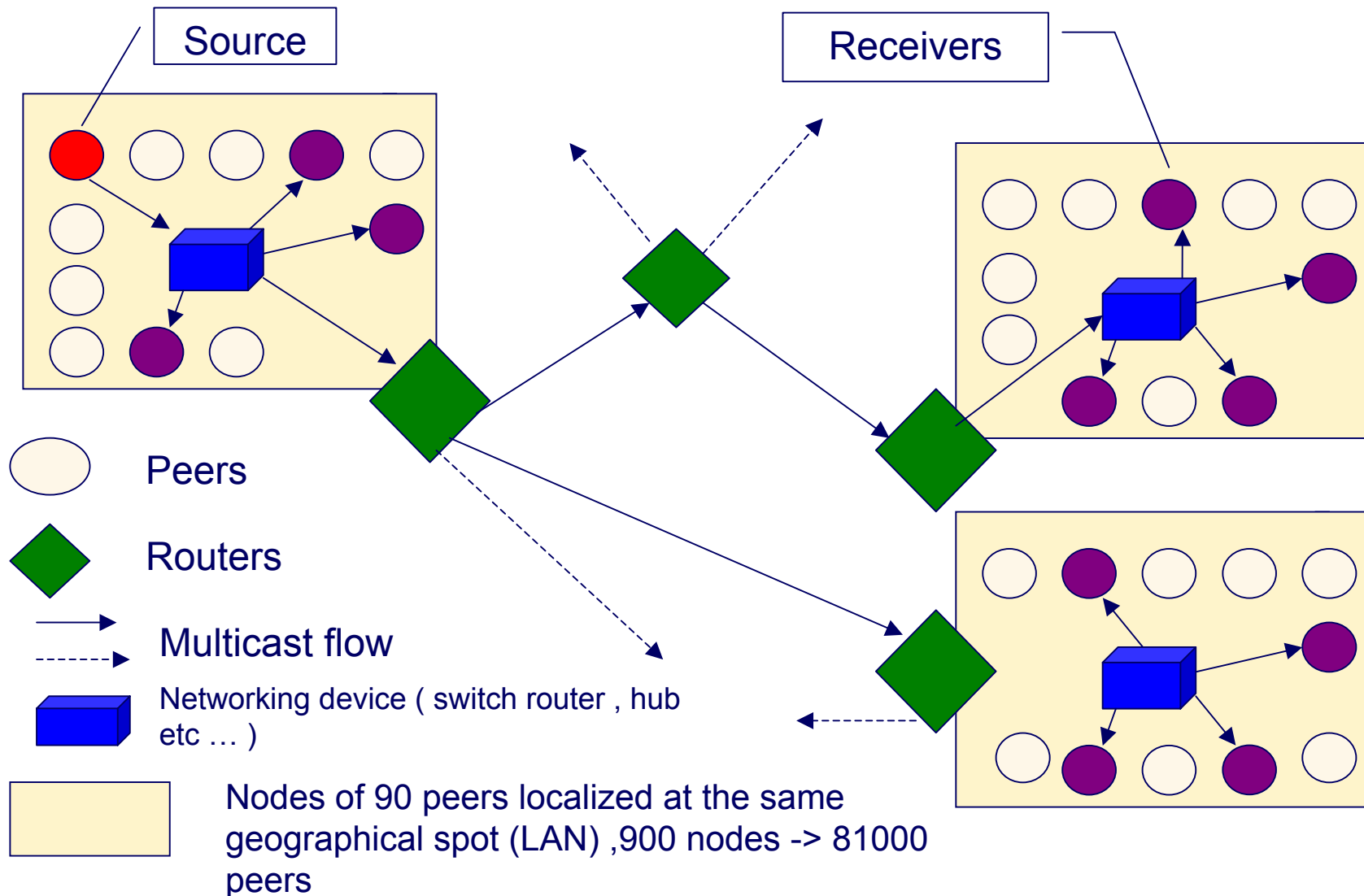


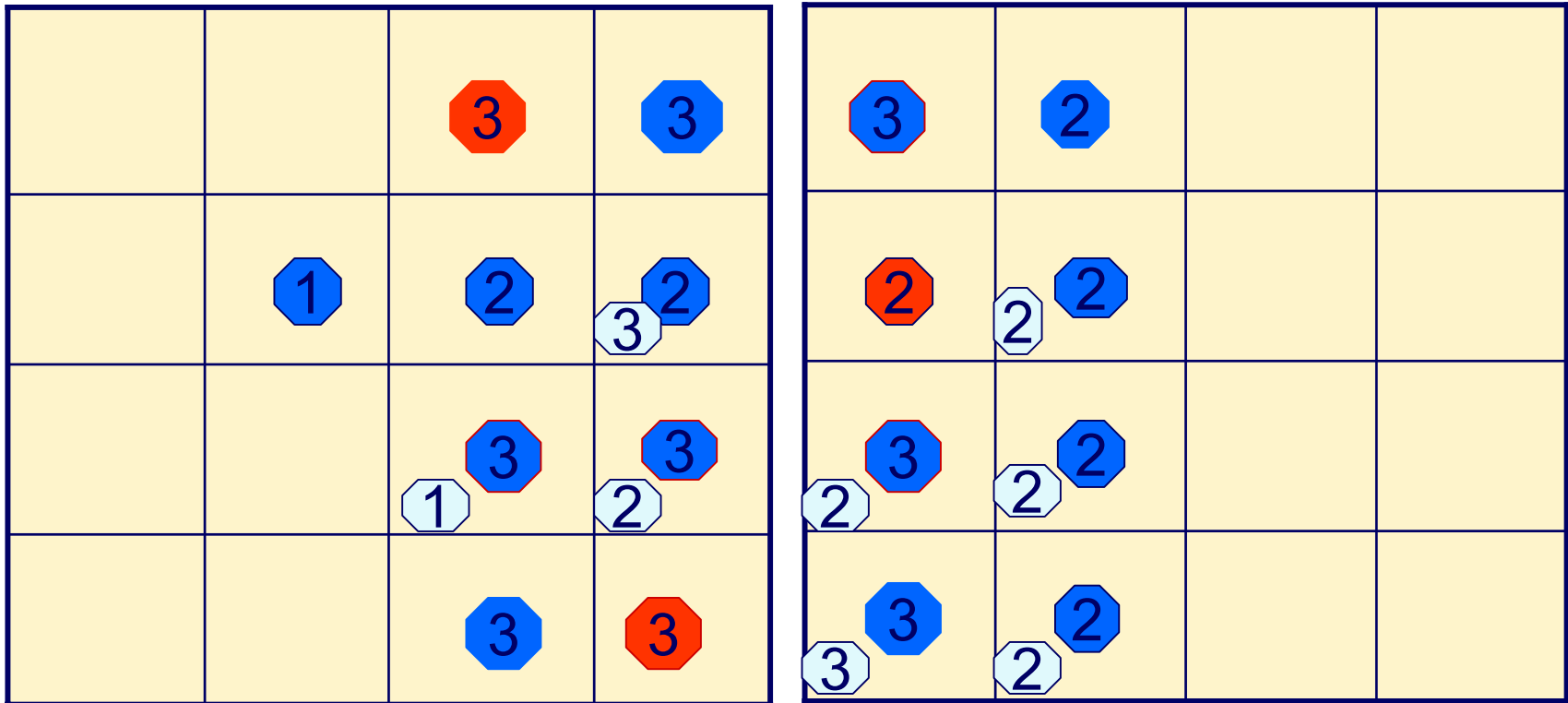
- Computation of « new » blocks on peer which minimize communications
- « update » of block at step k , on peer who updated the block a step $k-1$
- data send to dedicate peer ASAP



One step of the Block Gauss-Jordan method ; p=4

Network topology and flow of multicast data among peers of the same multicast group





Nevertheless, peers are not stables. Then, we can have in parallel computing from several steps of the method.

We have to use an inter and intra steps dependency graph.

Teraflops

With memory size
of 256 Mbytes

2.1

2

2.4

1

0.3

0.2

0.02

N=270000, p=90, n=3000

8100 peers :

1Kwords, efficiency < 0.5 %

8Kwords, efficiency de 5%

N=900000, p=300, n=3000

90000 peers

1Kwords, efficiency of 0.5 %

8Kword, efficiency of 5%

2.4j

19j

0.27

n=3000, 8Kwords/s

n=3000, 1Kwords/s

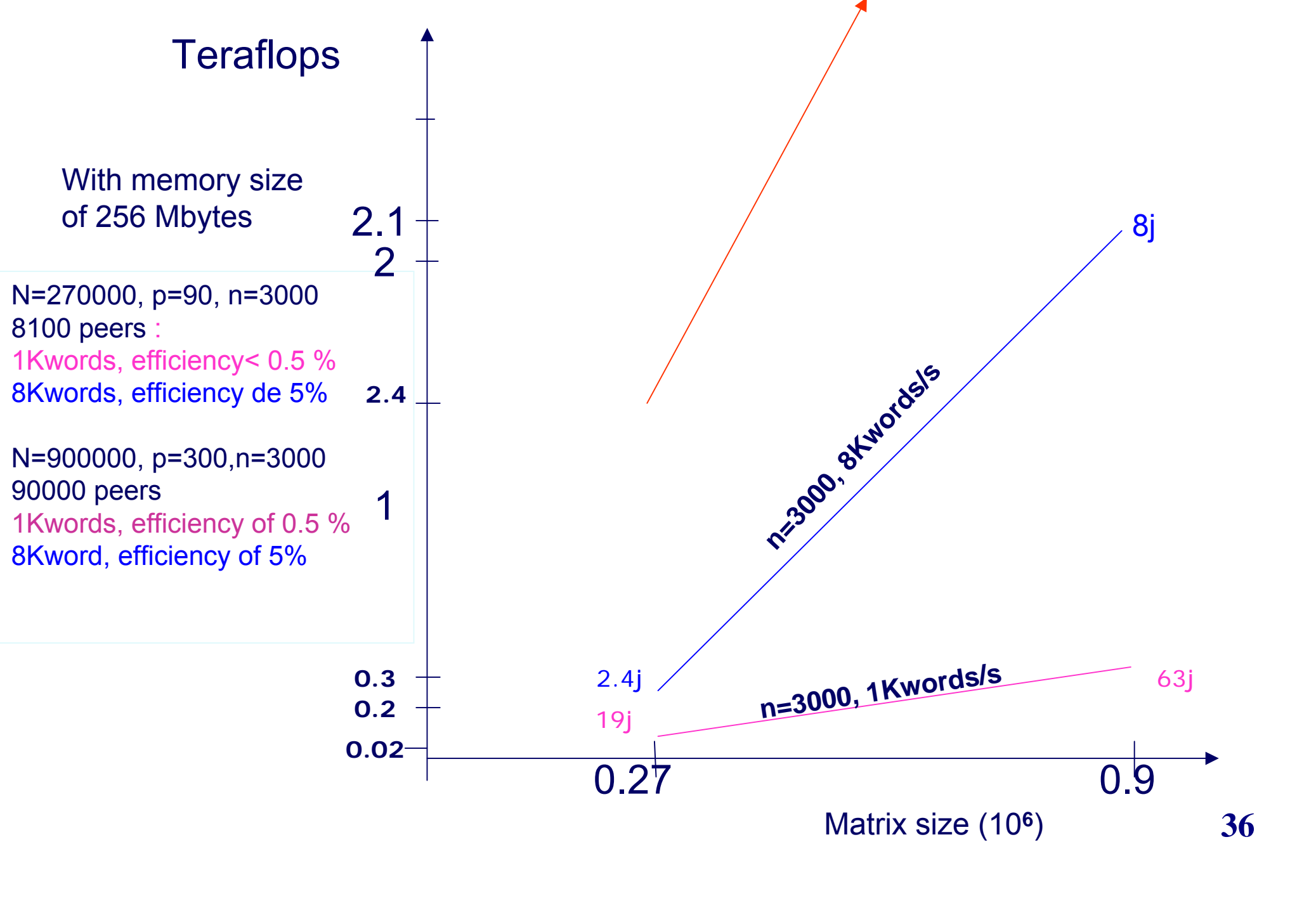
8j

63j

0.9

Matrix size (10^6)

36



Teraflops

15

27h

N=270000, p=90, n=3000

8100 peers :

1Mwords/s, efficiency of 30%

N=900000, p=300, n=3000

90000 peers

1Mword/s, efficiency of 30%

n=3000, 64Mbits/s

1.5

8h

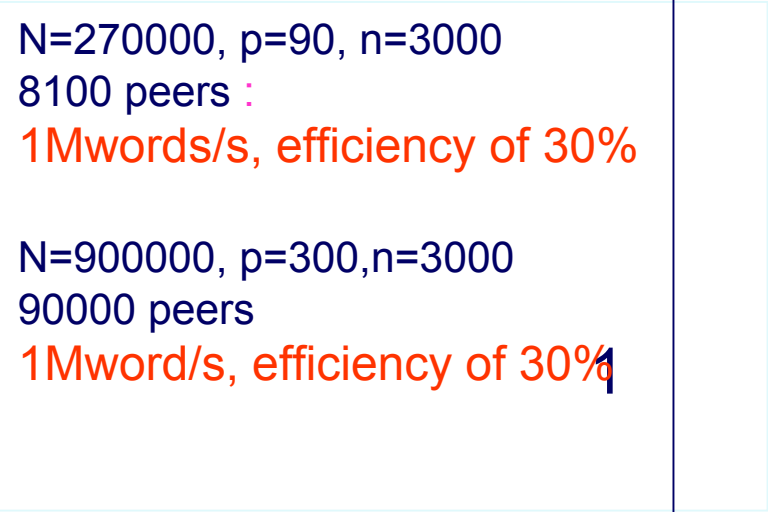
0.27

0.9

Sept. 27, 2004

Tsukuba University Matrix size (10^6)

37



Evaluation

Computation time is very short compare to communications

We have to evaluate first the communications.

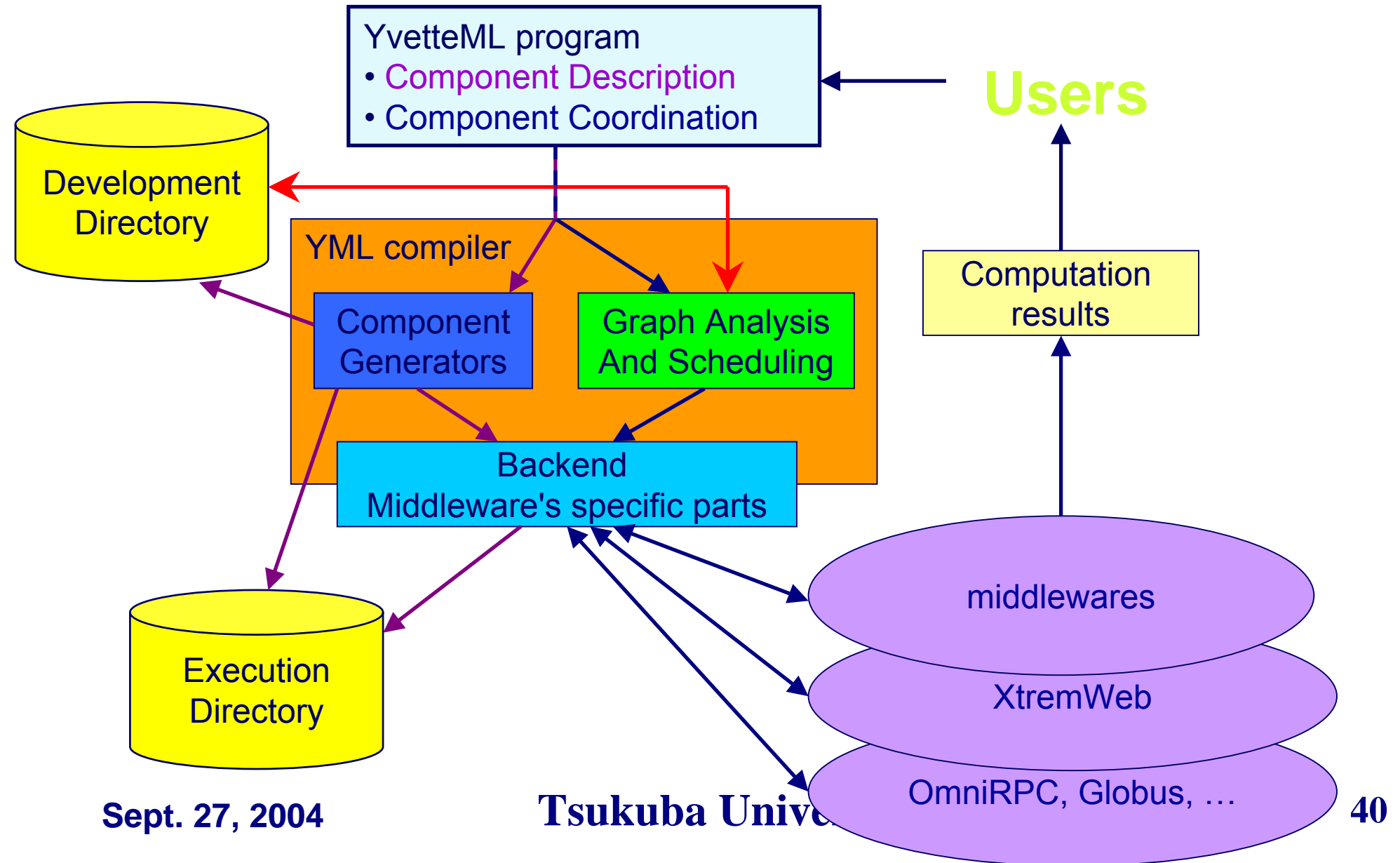
We have to find new performance evaluation model

Then, we can use these remarks to :

- highly compress data
- use more stable methods (Householder instead of gauss for example)
- control floating point computation.

Programming Paradigm and Framework

YML Framework



YvetteML Language

- Coordination language

- A component model


- Abstract components

- Graph components

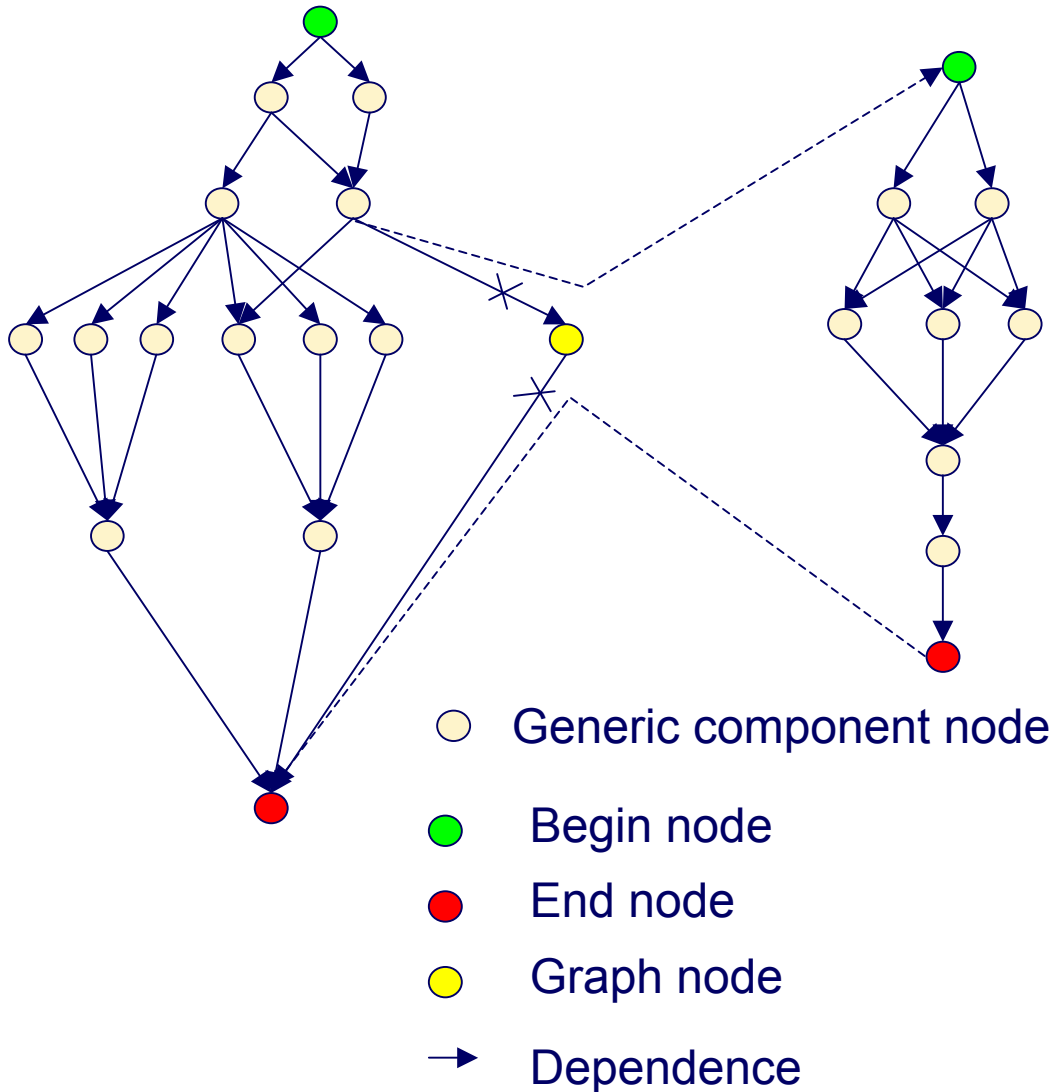
- Implementation components

- A graph description language

- Nodes  Tasks/components

- Edges  Dependencies

Graph example



```

par
  compute tache1(..);
  signal(e1);
//
  compute tache2(..); migrate matrix(..);
  signal(e2);
//
  wait(e1 and e2);
  Par
    compute tache3(..);
    signal(e3);
  //
    compute tache4(..);
    signal(e4);
  //
    compute tache5(..); control robot(..);
    signal(e5); visualize mesh(...);
  end par
//
  wait(e3 and e4 and e5);
  compute tache6(..);
  compute tache7(..);
end par
  
```

```

<?xml version="1.0" encoding="ISO-8859-1"?>
<component type="Graph">
  <name>GaussJordan</name>
  <description>
This component invert a bloc matrix using Gauss-Jordan
algorithm.
  </description>
  <parameters>
    <parameter name="Minput"
      type="BLOC_MATRIX_REAL" access="in"/>
    <parameter name="Moutput"
      type="BLOC_MATRIX_REAL" access="ou"/>
    <parameter name="bloc_count" type="INTEGER"
      access="in"/>
    <parameter name="size" type="INTEGER"
      access="in"/>
  </parameters>

  <graph>
/* Declarations */
const twop := 2 * bloc_count,
      pp1 := bloc_count+ 1,
      pm1 := bloc_count - 1,
      m := size/ bloc_count ;
domain a[1..bloc_count, 1..twop] of
MATRIX_MATRIX_DOUBLE;
bind a[1..bloc_count,1..bloc_count] to Minput;
bind a[1..bloc_count,pp1..twop] to Moutput;
event cal[1..bloc_count, 1..bloc_count, 1..twop];

```

```

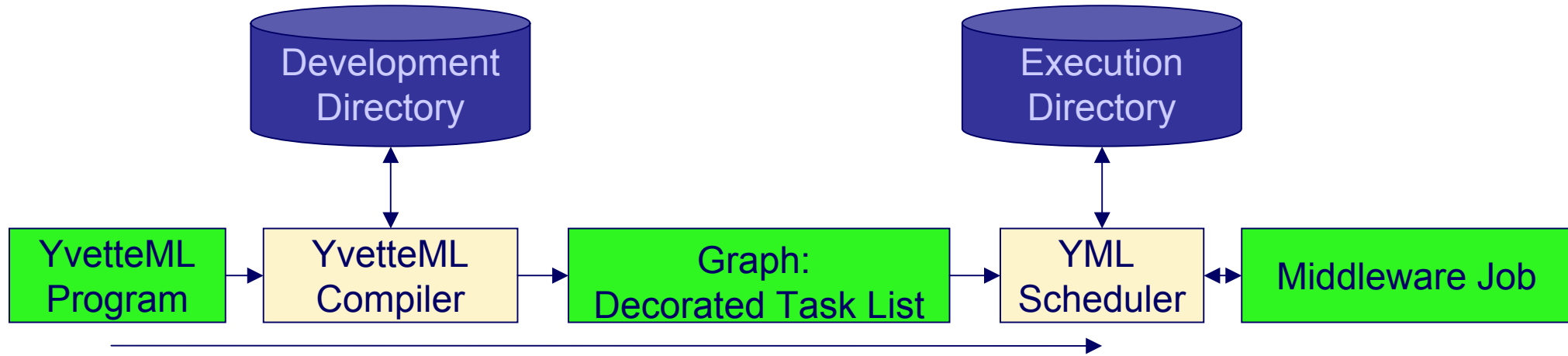
/* Calcul */
compute invmatcp(a[1,1], a[1,pp1], m);

par

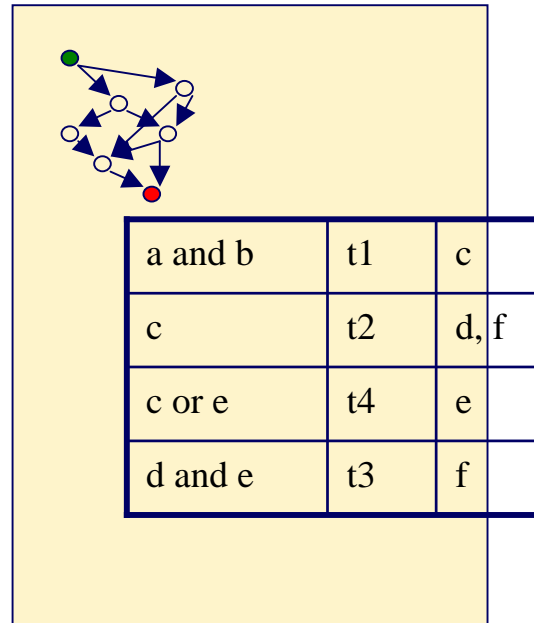
  par (j := 2, bloc_count) do
    compute diad3(a[1,j], a[1,1], m);
    signal(cal[1,1,j], cal[1,1,j]);
  end par do
//
  par (i := 2, bloc_count) (j := 2, bloc_count) do
    wait(cal[1, 1, j]);
    if (i eq 2 and j eq 2) then
      compute triad3inv(a[2,2], a[2,2 + bloc_count], a[2,1],
        a[1,2], m);
      signal(cal[2, 2, 2], cal[2, 2, 2 + bloc_count]);
    else
      compute triad3(a[i,j], a[i,1], a[1,j], m);
      signal(cal[1, i, j]);
    endif
  end par do
//
  par (i := 2, bloc_count) do
    compute mdiad3(a[i,1+bloc_count], a[1,1], m);
    signal(cal[1, i, pp1]);
  end par do
//
.....

```

Application Execution



```
Par
  par (l := 1,
      blockSize) do
    compute
    MatProd(Mat1[l],
            Mat[i-1], Mat[i-1]);
  ...
  ...
```



Conclusion and Perspectives

- Some linear algebra methods are not well-adpated to P2P computing (but it was the same for the CMs for example).
- Others would be well-adapted with smart scheduler and powerful multicast.
- Scientific Computing on Large scale P2P Platforms have to find a programming paradigm and evaluation and economical models.
- New end-users would have acces to large amount of memory and large computing power. Some who never had acces to supercomputers before.

In collaboration with

- Lamine Aouad
- Laurent Choy
- Olivier Delannoy
- Benoit Hudzia
-

Sept. 27, 2004

In collaboration with

- Lamine Aouad
- Laurent Choy
- Olivier Delannoy
- Benoit Hudzia

A Peer to Peer Computing Framework : Design and Performance Evaluation of YML, Olivier Delannoy and Serge Petiton, HeteroPar ' 2004, Cork, Irland, July 2004.

Reliable MultiCast using Fault Tolerant MPI on the GRID Environment, Benoit Hudzia and Serge Petiton, GRIDnet ' 04, San-Jose, USA, Oct. 2004

Experimentations and Programming Paradigms for Matrix Computing on P2P Grids. Lamine Aouad and Serge Petiton, GRID' 2004 at SC' 04, Pittsburgh, USA, Nov. 2004.

Sept. 27, 2004

In collaboration with

- Lamine Aouad
- Laurent Choy
- Olivier Delannoy
- Benoit Hudzia

A Peer to Peer Computing Framework : Design and Performance Evaluation of YML, Olivier Delannoy and Serge Petiton, HeteroPar' 2004, Cork, Irland, July 2004.

Reliable MultiCast using Fault Tolerant MPI on the GRID Environment, Benoit Hudzia and Serge Petiton, GRIDnet' 04, San-Jose, USA, Oct. 2004.

Experimentations and Programming Paradigms for Matrix Computing on P2P Grids. Lamine Aouad and Serge Petiton, GRID' 2004 at SC' 04, Pittsburgh, USA, Nov. 2004.

MERCI

Sept. 27, 2004