Solving symmetric eigenproblems on grid and global computing environment

Laurent Choy, Serge G.Petiton and Mitsuhisa Sato

11/09/2006

INSTITUT NATIONAL
DE RECHERCHE
EN INFORMATIQUE
ET EN AUTOMATIQUE

INRIA FUTURS

HPCS Lab.

Outline

- Context
- Why the explicit restarted Lanczos method?
- How we adapt it to a global computing environment
- Experimental platform
- Tests and first results
- Conclusion and outlooks

Context of Grid and global computing environment

- For most of parallel paradigms, we know that supercomputers are more efficient
- It is not an opposition but a study of another way of computing
- Characteristics of Global Computing
  - HUGE amount of resources connected to Internet
  - Often free access
  - But peers are unstable, not trustful
  - But a peer may have few memory, moderate disk space, not efficient CPU
  - But Internet has a low bandwidth and a high latency
- Some constraints can hardly be solved at the same time
  - A fine/average-grain parallelism or a coarse-grain parallelism ?
- Few works already done for linear algebra

Outline

- Context
- Why the explicit restarted Lanczos method?
- How we adapt it to a global computing environment
- Experimental platform
- Tests and first results
- Conclusion and outlooks

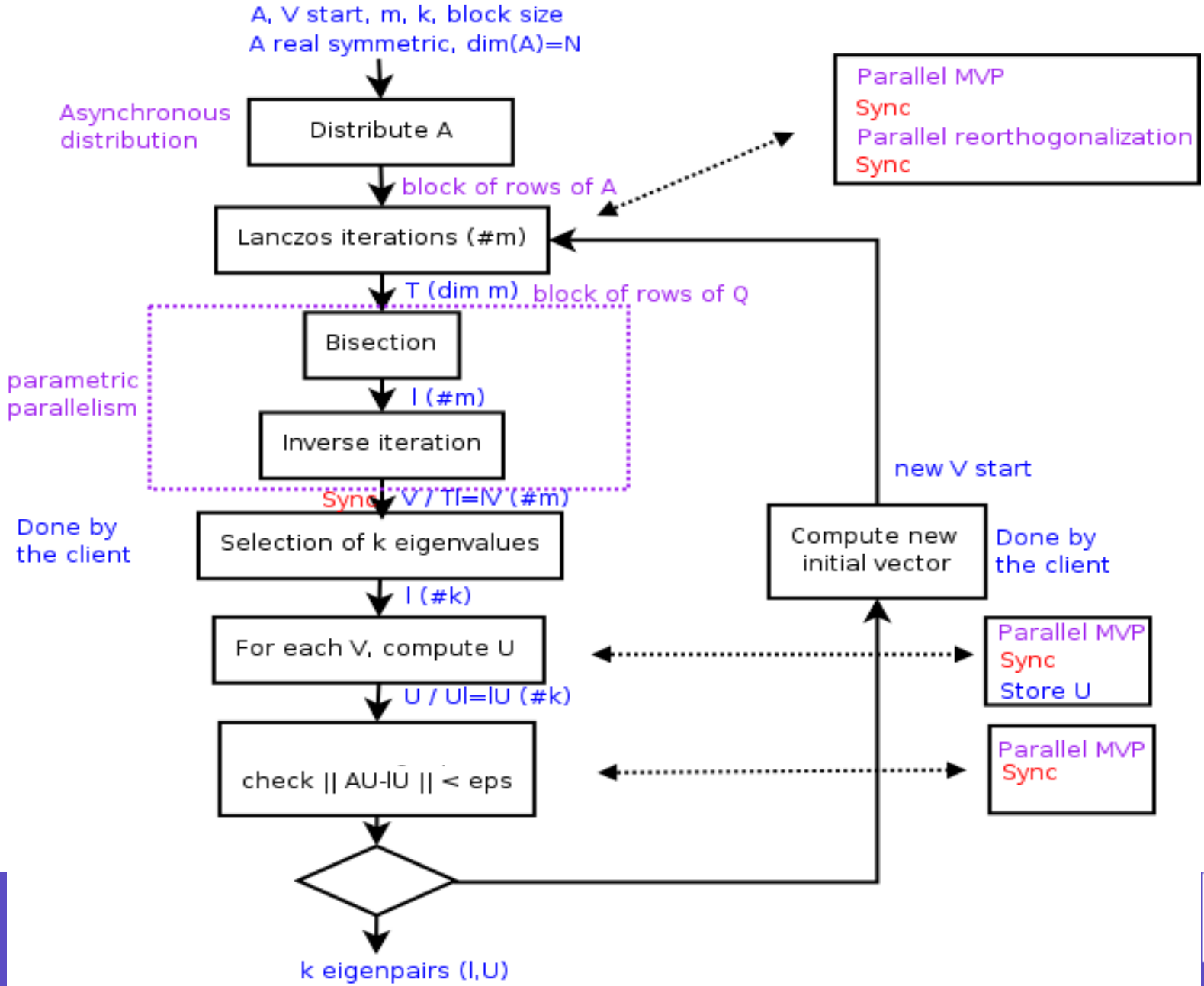
Why the explicit restarted lanczos method ?

- A is accessed by means of MVP (Q, the matrix of the bases too)
  - A and Q are shared among peers (resp. block of rows, cyclic distribution of rows).
  - the MVP is easily parallelized
- Generally, few eigenpairs are required
- Advantage of a restarted scheme
  - Working on a small subspace (m<<N)
  - Little main memory is required on peers (and on the client)
  - Q is not too big
  - Systematic full-reorthogonalization not too time consuming (easy to parallelize)
- Challenges to adapt this method:
  - we deal with 2 different paradigms of parallelism
  - We perform a kind of out-of-core technic (on the client and peers' sides)
    - A and Q are stored in files (peers), the eigenvectors of A and T too (client)
    - Data is gradually loaded when needed
    - Max memory (double precision numbers): N*m (peer), N*nb_peer (client)
  - Data persistency

Outline

- Context
- Why the explicit restarted Lanczos method?
- How we adapt it to a global computing environment
- Experimental platform
- Tests and first results
- Conclusion and outlooks

How we adapt it to a global environment

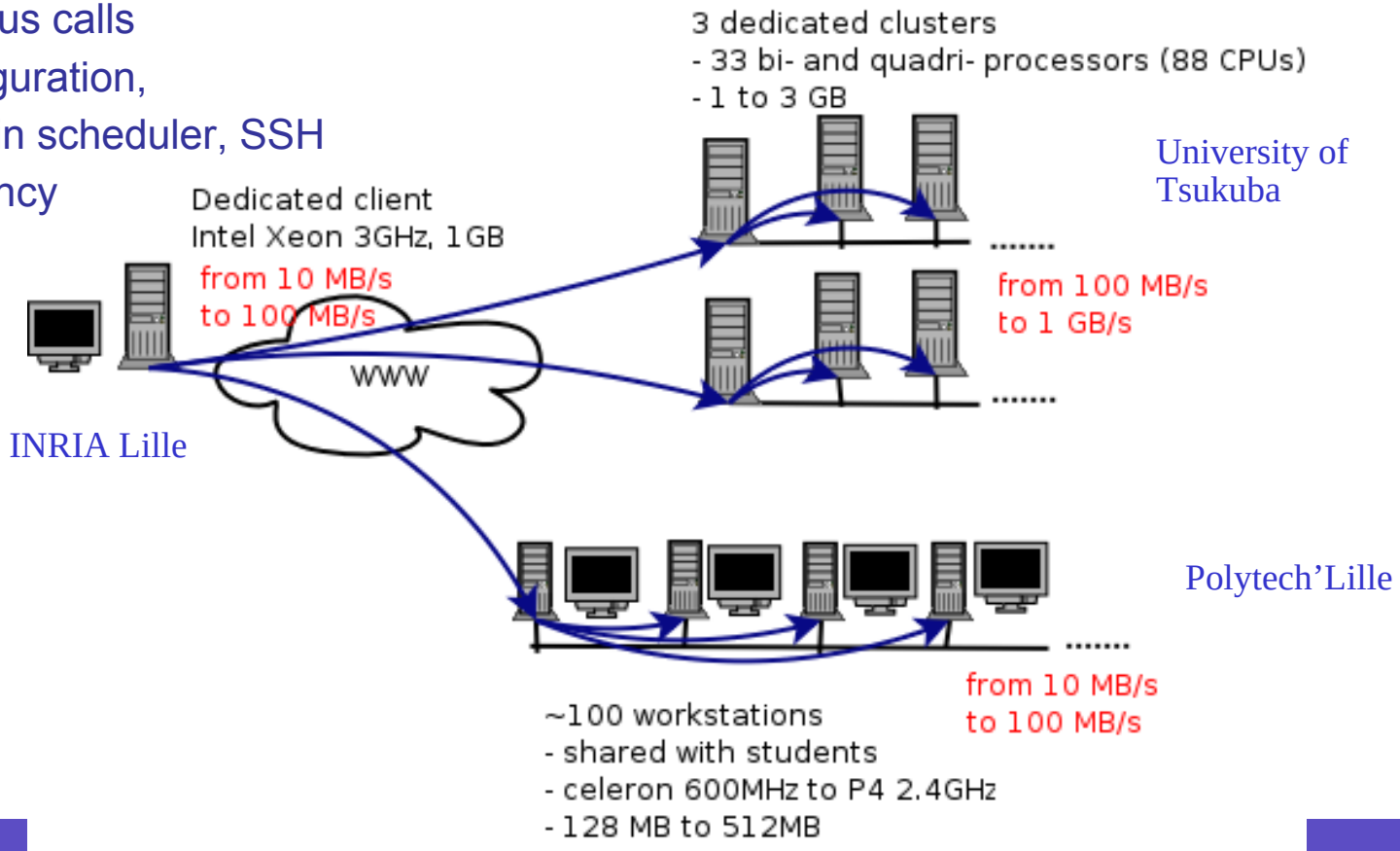


Outline

- Context
- Why the explicit restarted Lanczos method?
- How we adapt it to a global computing environment
- Experimental platform
- Tests and first results
- Conclusion and outlooks

Experimental platform

- OmniRPC (HPCS lab, University of Tsukuba, Japan)
  - RPC programming paradigm
  - User-friendly, C API (easy to use numerical librairies like CBLAS)
  - Asynchroneous calls
  - Cluster configuration,
    - Round robin scheduler, SSH
  - data persistency

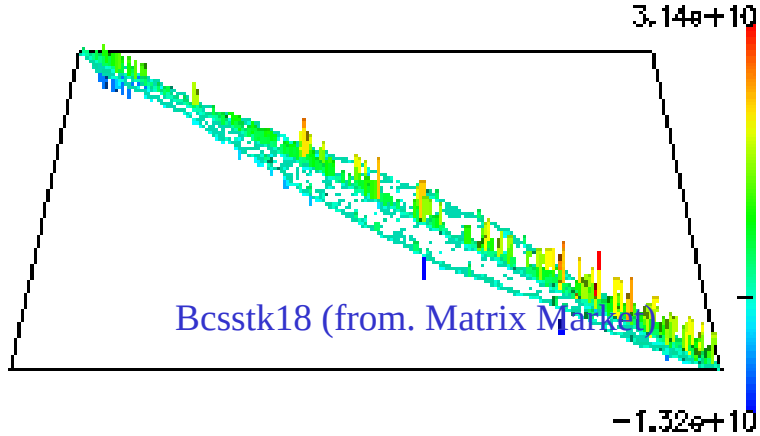


Outline

- Context
- Why the explicit restarted Lanczos method?
- How we adapt it to a global computing environment
- Experimental platform
- Tests and first results
- Conclusion and outlooks

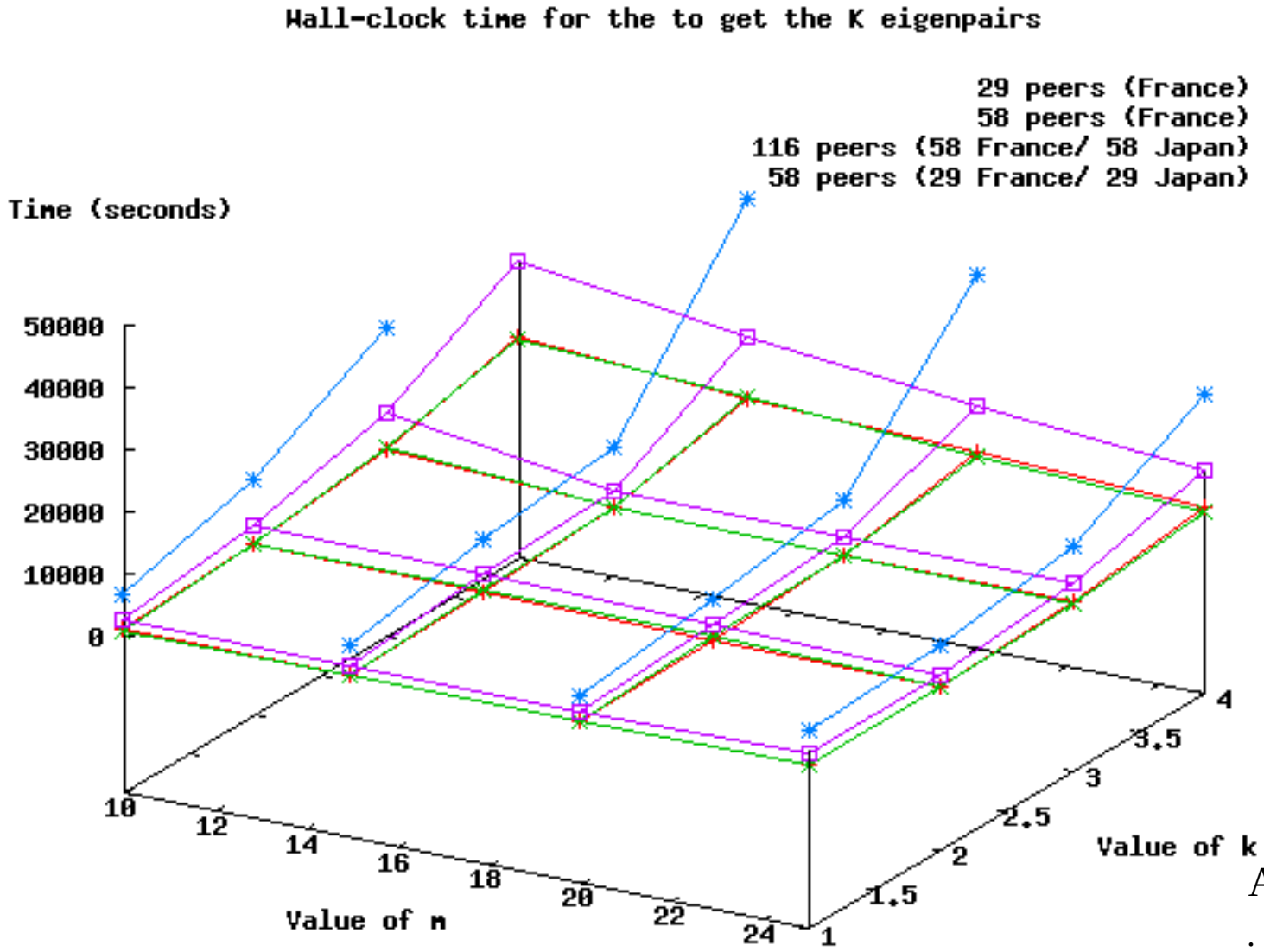
Tests and results (1/4)

- Numerical settings
  - Bcsstk18 (N=11948, 80519 entries, real symmetric positive definite, avg nonzeros per row/column=12, cond number=65)
  - For our tests, we do a paving of Bcsstk18
  - Folowing slides:
    - N=47792
    - M=10-15-20-25, K=1-2-3-4



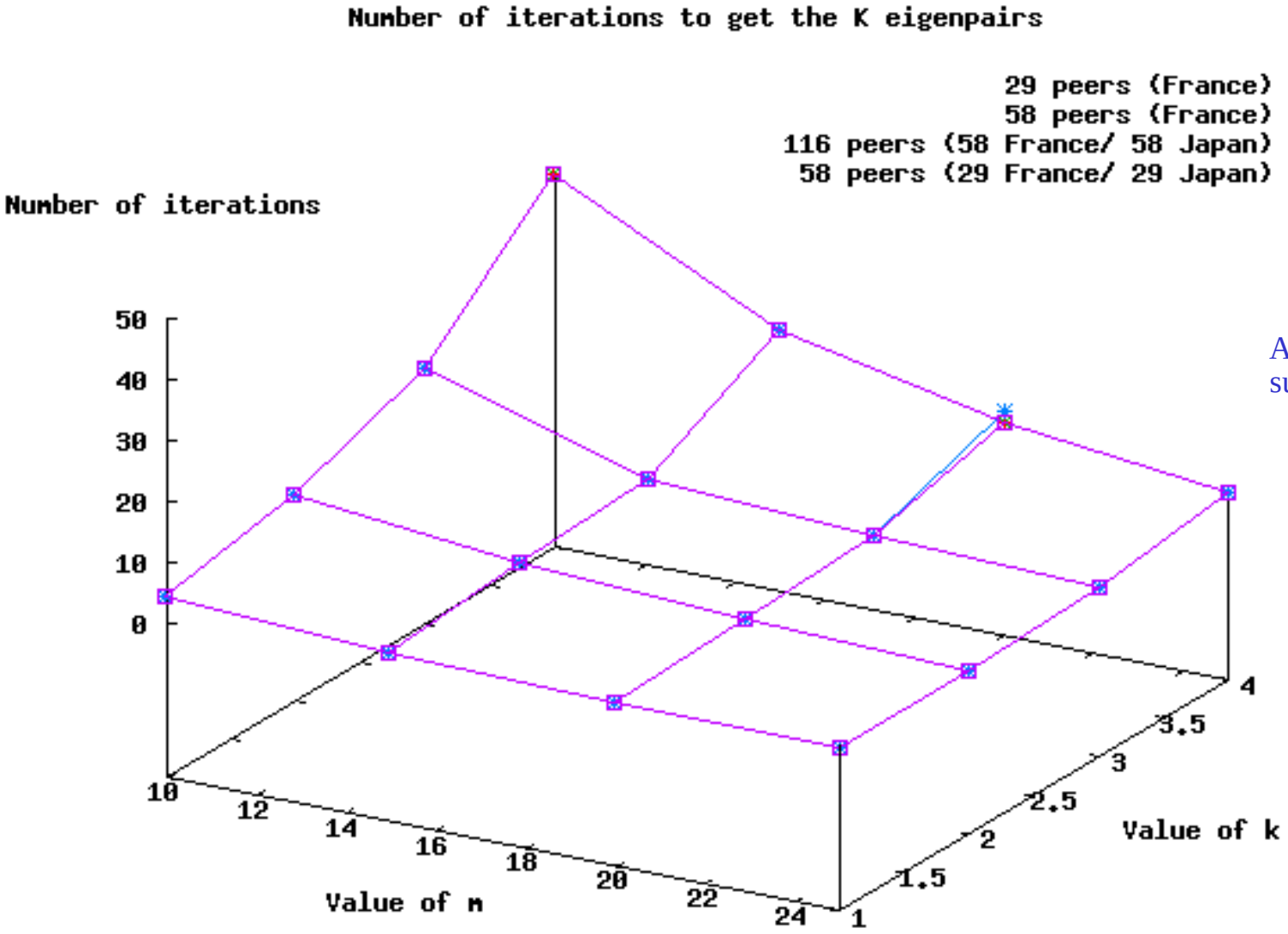
Bcsstk18 (from. Matrix Market)

- Experimental settings
  - Client at Lille
  - Number of peers: 29 (Lille), 58 (Lille), 58 (29 Lille+29 Tsukuba), 116 (58 Lille+58 Tsukuba)
- We present here the first results



Average times:

- $t_{116(F+JP)} \sim 4t_{58(F)}$
- $t_{116(F+JP)} \sim 2t_{58(F+JP)}$
- $t_{58(F+JP)} \sim 2t_{58(F)}$



All configurations are superimposed (except for one run)

	Lanczos-Main ratio				MVP-Lanczos ratio				Reorthog-Lanczos ratio			
(m,k)	29 p	58 p	58 p (F+JP)	116 p	29 p	58 p	58 p (F+JP)	116 p	29 p	58 p	58 p (F+JP)	116 p
10 1	90	90	89	90	88	88	79	91	9	10	17	6
10 2	87	87	85	87	89	88	79	81	9	10	16	7
10 3	86	86	83	86	89	90	79	91	9	8	16	6
10 4	86	86	81		89	88	78		9	9	17	
15 1	93	93	92	93	84	86	74	88	13	12	22	10
15 2	90	90	89	90	84	86	74	87	14	12	22	10
15 3	87	87	85	87	87	86	73	88	11	12	23	9
15 4	86	87	85	87	84	85	71	88	14	13	25	9
20 1	95	95	94	94	83	80	69	85	15	18	27	12
20 2	92	92	91	92	80	86	69	86	17	12	28	12
20 3	89	89	88	89	82	83	68	86	16	15	28	12
20 4	87	88	85	88	82	79	67	85	16	18	29	12
25 1	89	95	95	95	77	78	63	83	21	20	34	15
25 2	93	93	92	93	76	77	63	83	21	20	33	14
25 3	91	91	90	91	79	76	64	83	18	22	32	15
25 4	88	88	87	89	78	77	64	83	20	22	33	14

	Eigenvectors-Main ratio				Check-Main ratio			
(m,k)	29 p	58 p	58 p (F+JP)	116 p	29 p	58 p	58 p (F+JP)	116 p
10 1	1	1	2	<1	8	7	6	8
10 2	2	2	4	1	9	9	8	9
10 3	3	3	6	2	10	10	8	10
10 4	4	4	9		9	8	7	
15 1	1	1	1	<1	5	5	4	5
15 2	2	2	3	1	7	7	6	7
15 3	2	2	5	2	9	9	7	9
15 4	4	4	8	2	8	8	7	9
20 1	<1	1	1	<1	3	3	3	4
20 2	2	1	3	1	5	5	4	5
20 3	3	2	5	2	7	7	5	7
20 4	3	4	7	3	8	8	6	7
25 1	1	1	1	<1	2	2	2	3
25 2	2	2	3	1	4	4	3	4
25 3	2	3	4	2	5	5	4	6
25 4	4	4	6	2	7	7	5	8

Conclusion and future works

- What this program does
  - It computes k eigenpairs of a real symmetric matrix (it is the goal!)
  - It harnesses heterogeneous peers
    - On several different geographic sites, using heterogeneous networks
  - It uses data persistency
  - It performs an out-of-core technic
- What it should (will) do
  - Harnessing volatile peers (by using YML)
- We have to finish current tests and use bigger matrices
  - And see how interesting global computing can be for linear algebra method