Solving large sparse linear systems in a grid environment using Java

Raphaël COUTURIER[1] Fabienne JÉZÉQUEL[2]

[1]Laboratoire d'Informatique de l'Université de Franche-Comté
IUT Belfort-Montbéliard
France

[2]Laboratoire d'Informatique de Paris 6
Université Pierre et Marie Curie - Paris 6
France

11th IEEE International Workshop on Parallel and Distributed Scientific and Engineering Computing
April 19-23, 2010
Atlanta, USA

Our objectives

Our aim is to solve large sparse linear systems in a grid environment using the Java language and the MPJ library.

MPJ is an implementation of the Java bindings for the MPI standard `http://mpj-express.org`.

We compare

- a Java code using MPJ
- a C code using MPI
- a code using the PETSc library.

The PETSc library has been written in C and employs the MPI standard for all message-passing communication.
`http://www-unix.mcs.anl.gov/petsc`

The GMRES method to solve linear systems

Saad & Schultz, 1986

Let $Ax = b$ with $A \in \mathbb{R}^{n \times n}$, $x, b \in \mathbb{R}^n$ and x_0 be an initial approximation.

GMRES(A, b, x_0, m)

Let $r_0 = b - Ax_0$. In m steps, we compute the Krylov subspace

$$\mathcal{K}_m(A, r_0) = span\{r_0, Ar_0, ..., A^{m-1} r_0\}.$$

x_m is the unique vector of $x_0 + \mathcal{K}_m(A, r_0)$ which minimizes $\|b - Ax_m\|_2$.

Because the storage and computational requirements become prohibitive as m increases, the GMRES method is restarted.

RestartedGMRES(A, b, x_0, m)

REPEAT

GMRES(A, b, x_0, m)

$x_0 \leftarrow x_m$

UNTIL convergence

Decomposition of the linear system (1/2)

The matrix A is split into as many horizontal rectangle matrices as processors.

The right-hand-side b is split into as many sub-vectors as processors.

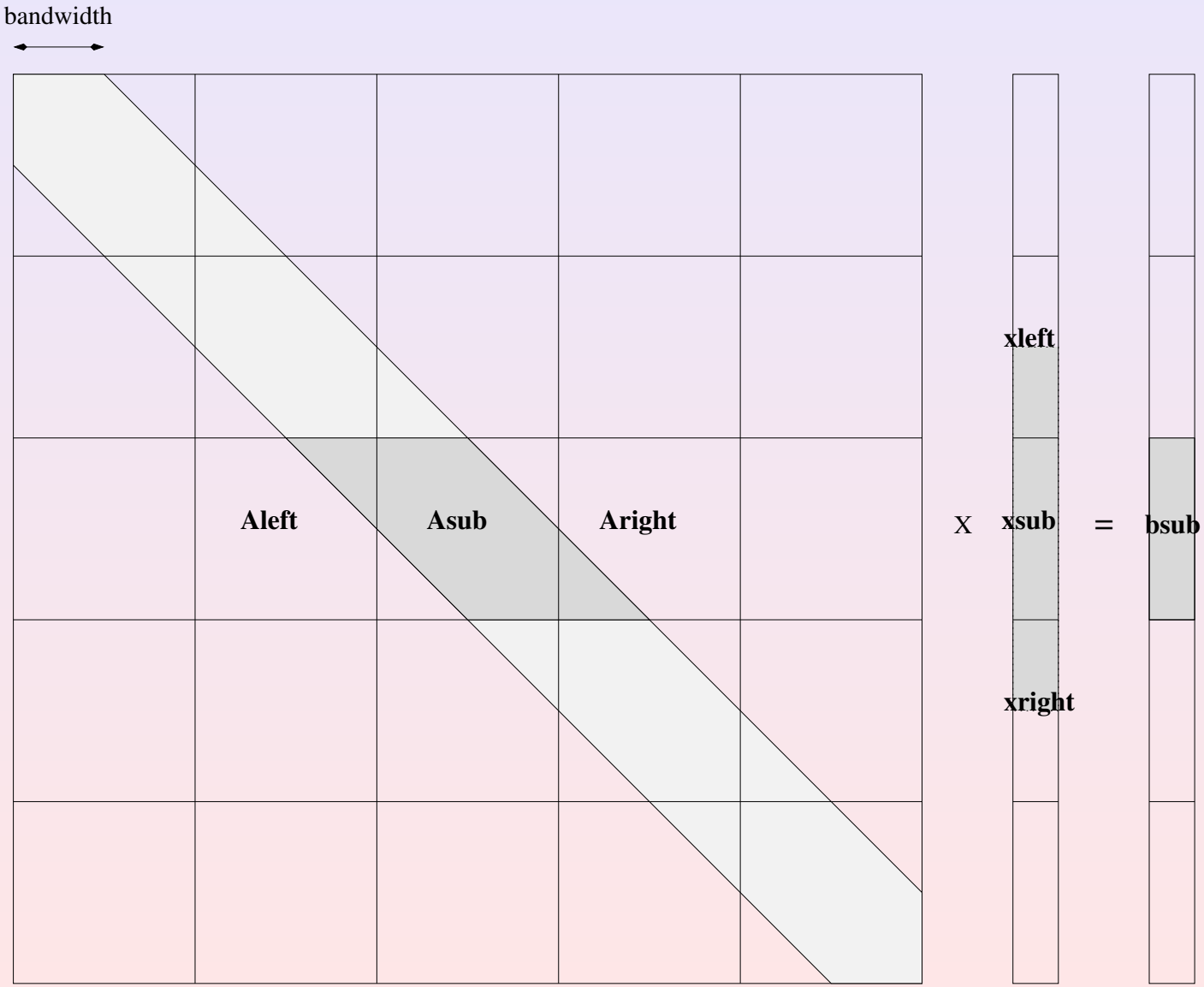
Our parallel version of GMRES takes into account the sparsity of the matrix.

Our aim is to solve a very large system: a typical value for n is 10^8.

To avoid the storage of the entire vector x on all the processors, x is split into sub-vectors too.

Decomposition of the linear system (2/2)

Each processor has to solve $A_{sub} \times x_{sub} = b_{sub} - A_{left} \times x_{left} - A_{right} \times x_{right}$.



Generation of matrices

Because we aim at comparing performance in solving very large sparse linear systems without long data file transfers, our linear systems are generated at run time.

Each processor computes specific part of the matrix, so that each rectangle matrix is automatically distributed on the processors.

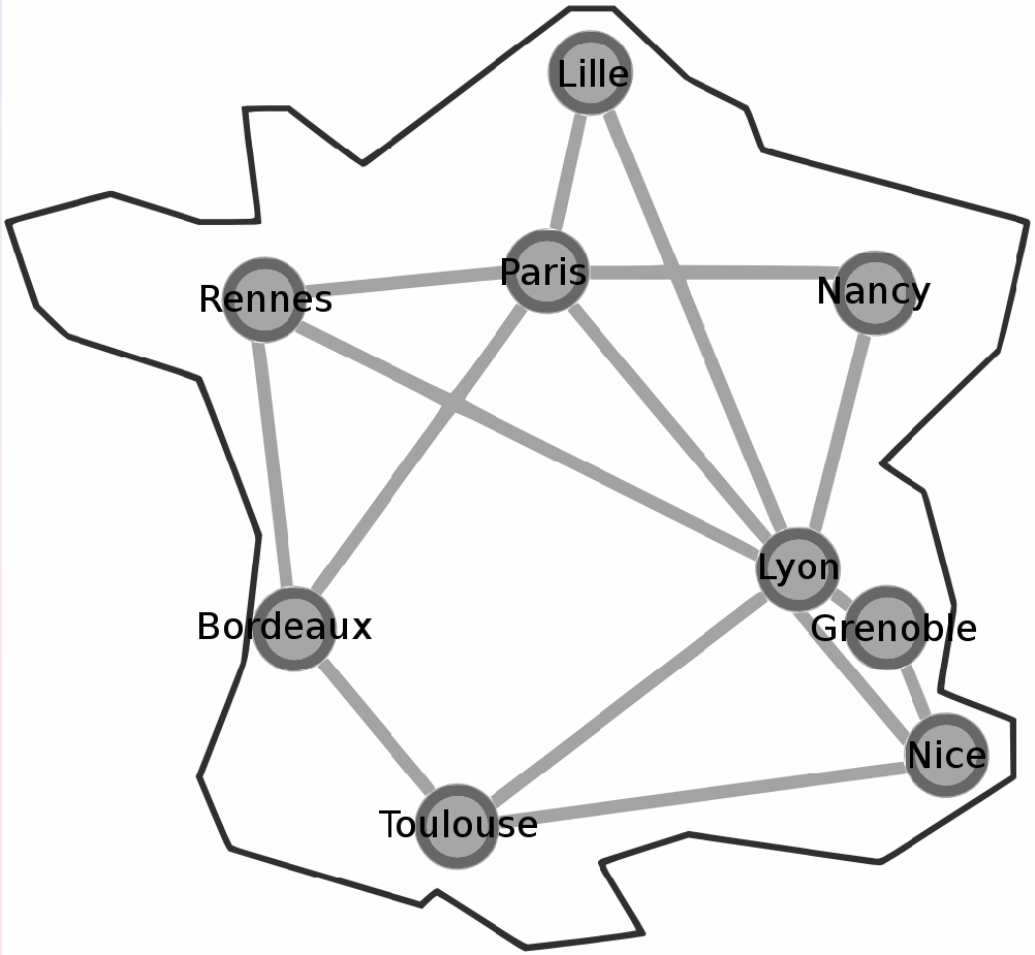
Each generated matrix consists of several non-empty diagonals.

Random values are used to compute the matrix entries.

The same random number generator has been used for C and Java.

Context of the experiments: GRID'5000

GRID'5000 (http://www.grid5000.fr) is an experimental grid platform that features a total of 5000 cores geographically distributed on different sites: 9 in France and 1 in Brazil.



Context of the experiments: deployment on GRID'5000

Because clusters in GRID'5000 use different operating systems and libraries, it is possible to deploy a common Linux image on the nodes reserved for an experiment.

- ☺ We can install our operating system with all our libraries.
- ☺ Our operating system does not offer services that may slow down network performance.
- ☹ Deployment is quite long when using many distant sites.
- ☹ When we deploy many nodes, some of them may be not deployed.

Experiments in a local context

Serial run times measured on the *bordereau* cluster

size	bandwidth	solver	time	iter.	error
5.10^6	5.10^2	Java (deploy.)	39.11	47	1.3e-8
		Java (no deploy.)	43.14		
		C	34.31		
		PETSc	35.97	52	3.6e-8
5.10^6	5.10^3	Java (deploy.)	35.30	43	7.5e-9
		Java (no deploy.)	35.35		
		C	34.08		
		PETSc	35.29	49	3.6e-8

- Comparable run times with the four kinds of computation.
- If the bandwidth increases, the run time is similar or decreases. This trend is particular to our serial executions. In parallel environments, if the bandwidth increases, the communication volume also increases and so does the execution time.

Experiments in a local context

Results obtained on 64 machines of the *bordereau* cluster

size	bandwidth	solver	time	iter.	error
8.10^8	8.10^4	Java (deploy.)	142.8	43	1.9e-7
		Java (no deploy.)	141.0		
		C	129.7		
		PETSc	129.1	49	2.5e-7
8.10^8	8.10^5	Java (deploy.)	231.7	45	2.5e-7
		Java (no deploy.)	201.2		
		C	232.1		
		PETSc	226.4	53	4.1e-7

- The impact of the deployment on performance can be positive or negative, depending on the matrix.
- Except in one case, the C code leads to better performance than the Java code (with or without deployment).
- The C code and the PETSc code lead to comparable performance.
- The code in Java or C performs less iterations and leads to a better error than the PETSc code.

Experiments in a distant context

Java code without deployment

190 machines located in Lille, Nancy and Sophia (7 clusters involved)

size	bandwidth	time	iter.	error
8.10^8	8.10^4	73.1	43	1.9e-7
8.10^8	8.10^5	208.6	45	2.5e-7
10^9	10^5	90.5	43	1.4e-7
10^9	10^6	259.2	44	2.1e-7

- For the matrix of size 8.10^8 and bandwidth 8.10^4, the performance on 190 processors in a distant context is twice better than on 64 processors in a local context.
- For the matrix of size 8.10^8 and bandwidth 8.10^5, there is no such significant difference between the two run times.

If the bandwidth increases, the communication volume also increases and the impact of communications on run time is greater in a distant context than in a local context.

Experiments in a distant context

Java code without deployment
60 machines located in Rennes and Nancy (2 clusters involved)

size	bandwidth	time	iter.	error
10^9	10^5	123.1	43	1.4e-7
10^9	10^6	263.2	44	2.1e-7
2.10^9	2.10^5	276.7	43	1.2e-7
2.10^9	2.10^6	522.8	44	2.0e-7

For the matrices of size 10^9, the run times on 60 processors are longer than on 190 processors:

- 36 % longer for the matrix of bandwidth 10^5
- 1.5 % longer for the matrix of bandwidth 10^6.

If the bandwidth $\nearrow$, the communication volume $\nearrow$.
The impact on run time is greater in a context involving 7 clusters on 3 sites than in a context involving only 2 distant clusters.

The run times of systems of size 2.10^9 are about twice that of systems of size 10^9 having the same size/bandwidth ratio.

Conclusion

The efficiency of our linear GMRES parallel solver in Java is comparable :

- to the same solver written in C
- to the PETSc library.

Our GMRES implementation in Java allows us to solve a sparse system of size two billions using two geographically distant clusters in a few minutes.

Java could be more often used for high performance computing.
Its portability is an undeniable advantage on heterogeneous grids.

Perspectives

- Sparse matrix stored in the CSR format
  Performance of alternative storage formats (BCSR, BCSD)?
- Performance of Java implementation of other linear solvers or preconditioners?
- In our experiments, one process per machine has been run.
  Implementation taking into account the number of processors per machine and the multi-core architecture of the processors?
- Implementation of sparse linear solvers on hybrid architectures consisting of both CPUs and GPUs?