J Supercomput (2015) 71:1345–1356
DOI 10.1007/s11227-014-1367-7

A scalable multisplitting algorithm to solve large sparse linear systems

Raphaël Couturier · Ziane Khodja Lilia

Published online: 25 December 2014

Abstract In this paper, we revisit the Krylov multisplitting algorithm presented in Huang and O'Leary (Linear Algebra Appl 194:9–29, 1993) which uses a sequential method to minimize the Krylov iterations computed by a multisplitting algorithm. Our new algorithm is based on a parallel multisplitting algorithm with few blocks of large size using a parallel GMRES method inside each block and on a parallel Krylov minimization to improve the convergence. Some large-scale experiments with a 3D Poisson problem are presented with up to 8,192 cores. They show the obtained improvements compared to a classical GMRES both in terms of number of iterations and in terms of execution times.

Keywords Large sparse linear systems · Multisplitting algorithm · 3D Poisson problem

1 Introduction

Iterative methods are used to solve large sparse linear systems of equations of the form $Ax = b$ because they are easier to parallelize than direct ones. Many iterative methods have been proposed and adapted by different researchers. For example, the GMRES method and the conjugate gradient method are very well known and used [12]. Both

R. Couturier (✉)
Femto-ST Institute, University of Franche Comte, Besançon, France
e-mail: raphael.couturier@univ-fcomte.fr

Z. K. Lilia
Inria Bordeaux Sud-Ouest, Talence, France
e-mail: lilia.ziane@inria.fr

methods are based on the Krylov subspace which consists in forming a basis of a sequence of successive matrix powers times the initial residual.

When solving large linear systems with many cores, iterative methods often suffer from scalability problems. This is due to their need for collective communications to perform matrix-vector products and reduction operations. Preconditioners can be used to increase the convergence of iterative solvers. However, most of the good preconditioners are not scalable when thousands of cores are used.

Traditional parallel iterative solvers are based on fine-grain computations that frequently require data exchanges between computing nodes and have global synchronizations that penalize the scalability [14]. Particularly, they are more penalized on large-scale architectures or on distributed platforms composed of distant clusters interconnected by a high-latency network. It is therefore, imperative to develop coarse-grain-based algorithms to reduce the communications in the parallel iterative solvers. Two possible solutions consists either in using asynchronous iterative methods [2] or in using multisplitting algorithms. In this paper, we will reconsider the use of a multisplitting method. In opposition to traditional multisplitting method that suffer from slow convergence, as proposed in [9], the use of a minimization process can drastically improve the convergence.

1.1 Contributions

In this work, we develop a new parallel two-stage algorithm for large-scale clusters. Our objective is to create a mix between Krylov-based iterative methods and the multisplitting method to improve scalability. In fact, Krylov subspace methods are well-known for their good convergence compared to other iterative methods. So, our main contribution is to use the multisplitting method which splits the problem to solve into different blocks to reduce the large amount of communications and, to implement both inner and outer iterations as Krylov subspace iterations to improve the convergence of the multisplitting algorithm.

The present paper is organized as follows. First, Sect. 2 presents some related works and the principle of multisplitting methods. Then, in Sect. 3 the algorithm of our Krylov multisplitting method, based on inner-outer iterations, is presented. Finally, in Sect. 4, the parallel experiments on Hector architecture show the performances of the Krylov multisplitting algorithm compared to the classical GMRES algorithm to solve a 3D Poisson problem.

2 Related works and presentation of the multisplitting method

A general framework to study parallel multisplitting methods has been presented in [11] by O'Leary and White. Convergence conditions are given for the most general cases. Many authors have improved multisplitting algorithms by proposing, for example, an asynchronous version [5] or convergence conditions [1,3] or other two-stage algorithms [5,7].

In [9], the authors have proposed a parallel multisplitting algorithm in which all the tasks except one are devoted to solve a sub-block of the splitting and to send their local solutions to the first task which is in charge of combining the vectors at each iteration. These vectors form a Krylov basis for which the first task minimizes the error function over the basis to increase the convergence, then the other tasks receive the updated solution until the convergence of the global system.

In [6], the authors have developed practical implementations of multisplitting algorithms to solve large-scale linear systems. Inner solvers could be based on sequential direct method with the LU method or sequential iterative one with GMRES.

In [4], the authors have designed a parallel multisplitting algorithm in which large blocks are solved using a GMRES solver. The authors have performed large-scale experiments up-to 32,768 cores and they conclude that an asynchronous multisplitting algorithm could be more efficient than traditional solvers on an exascale architecture with hundreds of thousands of cores.

So, compared to these works, we propose in this paper a practical multisplitting method based on parallel iterative blocks which gives better results than classical GMRES method for the 3D Poisson problem we considered.

The key idea of a multisplitting method to solve a large system of linear equations $Ax = b$ is defined as follows. The first step consists in partitioning the matrix A in L several ways

$$A = M_\ell - N_\ell, \tag{1}$$

where for all $\ell \in \{1, \ldots, L\}$ M_ℓ are non-singular matrices. Then the linear system is solved by an iteration based on the obtained splittings as follows

$$x^{k+1} = \sum_{\ell=1}^{L} E_\ell M_\ell^{-1}(N_\ell x^k + b), \quad k = 1, 2, 3, \ldots \tag{2}$$

where E_ℓ are non-negative and diagonal weighting matrices and their sum is an identity matrix I. The convergence of such a method is dependent on the condition

$$\rho\left(\sum_{\ell=1}^{L} E_\ell M_\ell^{-1} N_\ell\right) < 1. \tag{3}$$

where ρ is the spectral radius of the square matrix.

The advantage of the multisplitting method is that at each iteration k there are L different linear sub-systems

$$v_\ell^k = M_\ell^{-1} N_\ell x_\ell^{k-1} + M_\ell^{-1} b, \quad \ell \in \{1, \ldots, L\}, \tag{4}$$

to be solved independently by a direct or an iterative method, where v_ℓ is the solution of the local sub-system. Thus the computations of $\{v_\ell\}_{1 \leq \ell \leq L}$ may be performed in parallel by a set of processors. A multisplitting method using an iterative method as an inner solver is called an inner-outer iterative method or a two-stage method. The results v_ℓ obtained from the different splittings (4) are combined to compute solution x of the linear system using the diagonal weighing matrices

$$x^k = \sum_{\ell=1}^{L} E_\ell v_\ell^k, \tag{5}$$

In the case where the diagonal weighting matrices E_ℓ have only zero and one factors (i.e. v_ℓ are disjoint vectors), the multisplitting method is non-overlapping and corresponds to the block Jacobi method.

3 A two-stage method with a minimization

Let $Ax = b$ be a given large and sparse linear system of n equations where $A \in \mathbb{R}^{n\times n}$ is a sparse square and non-singular matrix, $x \in \mathbb{R}^n$ is the solution vector and $b \in \mathbb{R}^n$ is the right-hand side vector. We use a multisplitting method to solve the linear system on a large computing platform to reduce communications. Let the computing platform be composed of L blocks of processors physically adjacent or geographically distant. In this work, we apply the block Jacobi multisplitting method to the linear system as follows

$$\begin{cases} A = [A_1, \dots, A_L] \\ x = [X_1, \dots, X_L] \\ b = [B_1, \dots, B_L] \end{cases} \tag{6}$$

where for $\ell \in \{1, \dots, L\}$, A_ℓ is a rectangular block of size $n_\ell \times n$ and X_ℓ and B_ℓ are sub-vectors of size n_ℓ each, such that $\sum_\ell n_\ell = n$.

The splitting is performed row-by-row without overlapping in such a way that successive rows of sparse matrix A and both vectors x and b are assigned to a block of processors.

So, the multisplitting format of the linear system is defined as follows

$$\forall \ell \in \{1, \dots, L\},\ A_{\ell\ell} X_\ell + \sum_{\substack{m=1 \\ m\neq\ell}}^{L} A_{\ell m} X_m = B_\ell, \tag{7}$$

where $A_{\ell m}$ is a sub-block of size $n_\ell \times n_m$ of the rectangular matrix A_ℓ, $X_m \neq X_\ell$ is a sub-vector of size n_m of the solution vector x and $\sum_{m\neq\ell} n_m + n_\ell = n$, for all $m \in \{1, \dots, L\}$.

Our multisplitting method proceeds by iteration to solve the linear system in such a way that each sub-system

$$\begin{cases} A_{\ell\ell} X_\ell = Y_\ell, \text{ such that} \\ Y_\ell = B_\ell - \displaystyle\sum_{\substack{m=1 \\ m\neq\ell}}^{L} A_{\ell m} X_m, \end{cases} \tag{8}$$

is solved independently by a *block of processors* and communications are required to update the right-hand side vectors Y_ℓ, such that the vectors X_m represent the data dependencies between the blocks. In this work, we use the parallel restarted GMRES method [13] as an inner iteration method to solve sub-systems (8).

GMRES is one of the most used Krylov iterative methods to solve sparse linear systems by minimizing the residuals over an orthonormal basis of a Krylov subspace.

It should be noted that the convergence of the inner iterative solver for the different sub-systems (8) does not necessarily involve the convergence of the multisplitting algorithm. It strongly depends on the properties of the global sparse linear system to be solved [2,11]. Furthermore, the splitting of the linear system among several blocks of processors increases the spectral radius of the iteration matrix, thereby slowing the convergence. In fact, the larger the number of splittings is, the larger the spectral radius is. In this paper, our work is based on the work presented in [9] to increase the convergence and improve the scalability of the multisplitting methods.

Krylov subspace methods are well-known for their good convergence compared to other iterative methods.

To accelerate the convergence, we implemented the outer iteration of our multisplitting solver as a Krylov iterative method which minimizes some error function over a Krylov subspace [12]. The Krylov subspace that we used is spanned by a basis composed of successive solutions issued from solving the L splittings (8)

$$S = \{x^1, x^2, \ldots, x^s\}, \quad s \leq n, \tag{9}$$

where for $j \in \{1, \ldots, s\}$, $x^j = [X_1^j, \ldots, X_L^j]$ is a solution of the global linear system.

The advantage of such a Krylov subspace is that we neither need an orthonormal basis nor any synchronization between the different blocks to orthogonalize the generated basis. This avoids to perform other synchronizations between the blocks of processors.

The multisplitting method is periodically restarted every s iterations with a new initial guess $\tilde{x} = S\alpha$ which minimizes an error function, in our case it minimizes the residuals $\|b - Ax\|_2$ over the Krylov subspace spanned by vectors of S. So α is defined as the solution of the large overdetermined linear system.

$$R\alpha = b, \tag{10}$$

where $R = AS$ is a dense rectangular matrix of size $n \times s$ and $s \ll n$. This leads us to solve a system of normal equations

$$R^{\mathrm{T}} R\alpha = R^{\mathrm{T}} b, \tag{11}$$

which is associated with the least squares problem

$$\text{minimize } \|b - R\alpha\|_2, \tag{12}$$

where R^{T} denotes the transpose of matrix R. Since R (i.e. AS) and b are split among L blocks, the symmetric positive definite system (11) is solved in parallel. Thus, an iterative method would be more appropriate than a direct one to solve this system. We use the parallel conjugate gradient method for the normal equations CGNR [10,12].

Algorithm 1 A two-stage linear solver with inner iteration GMRES method

I1nput: A_ℓ (sparse sub-matrix), B_ℓ (right-hand side sub-vector)
Output: X_ℓ (solution sub-vector)

```
1: Load $A_\ell$, $B_\ell$
2: Set the initial guess $x^0$
3: Set the minimizer $\tilde{x}^0 = x^0$
4: for $k = 1, 2, 3, \dots$ until the global convergence do
5:     Restart with $x^0 = \tilde{x}^{k-1}$:
6:     for $j = 1, 2, \dots, s$ do
7:         Inner iteration solver: InnerSolver($x^0$, $j$)
8:         Construct basis $S$: add column vector $X_\ell^j$ to the matrix $S_\ell^k$
9:         Exchange local values of $X_\ell^j$ with the neighboring blocks
10:        Compute dense matrix $R$: $R_\ell^{k,j} = \sum_{i=1}^{L} A_{\ell i} X_i^j$
11:    end for
12:    Minimization $\|b - R\alpha\|_2$: UpdateMinimizer($S_\ell$, $R$, $b$, $k$)
13:    Local solution of linear system $Ax = b$: $X_\ell^k = \tilde{X}_\ell^k$
14:    Exchange the local minimizer $\tilde{X}_\ell^k$ with the neighboring blocks
15: end for

16: function InnerSolver($x^0$, $j$)
17:    Compute local right-hand side $Y_\ell = B_\ell - \sum_{\substack{m=1 \\ m \neq \ell}}^{L} A_{\ell m} X_m^0$
18:    Solving local splitting $A_{\ell\ell} X_\ell^j = Y_\ell$ using parallel GMRES method, such that $X_\ell^0$ is the initial guess
19:    return $X_\ell^j$
20: end function

21: function UpdateMinimizer($S_\ell$, $R$, $b$, $k$)
22:    Solving normal equations $(R^k)^T R^k \alpha^k = (R^k)^T b$ in parallel by $L$ blocks using parallel CGNR method
23:    Compute local minimizer $\tilde{X}_\ell^k = S_\ell^k \alpha^k$
24:    return $\tilde{X}_\ell^k$
25: end function
```

The main key points of our Krylov multisplitting method to solve a large sparse linear system are given in Algorithm 1. This algorithm is based on a two-stage method with a minimization using restarted GMRES iterative method as an inner solver. It is executed in parallel by each block of processors. Matrices and vectors with the subscript ℓ represent the local data for block ℓ, where $\ell \in \{1, \dots, L\}$. The two-stage solver uses two different parallel iterative algorithms: the GMRES method to solve each splitting (8) on a block of processors, and the CGNR method, executed periodically in parallel by all blocks to minimize the function error (12) over the Krylov subspace spanned by S. The algorithm requires two global synchronizations between L blocks. The first one is performed line 12 in Algorithm 1 to exchange local values of vector solution x (i.e. the minimizer $\tilde{x}$) required to restart the multisplitting solver. The second one is needed to construct the matrix R. We chose to perform this latter synchronization s times in every outer iteration k (line 7 in Algorithm 1). This is a straightforward way to compute the sparse matrix-dense matrix multiplication $R = AS$. We implemented all synchronizations using message passing collective communications of MPI library.

4 Experiments

To illustrate the interest of our Krylov multisplitting algorithm, we have compared its performances with those of a classical block Jacobi multisplitting method and those of the GMRES method which is a commonly used method in many situations.

We have chosen to focus on only one problem which is very simple to implement: a 3-dimensional Poisson problem.

$$\begin{cases} \nabla u = f & \text{in } \omega \\ u = 0 & \text{on } \Gamma = \partial\omega \end{cases} \tag{13}$$

After discretization, with a finite difference scheme, a seven-point stencil is used. It is well-known that the spectral radius of matrices representing such problems are very close to 1. Moreover, the larger the number of discretization points is, the closer to 1 the spectral radius is. Hence, to solve a matrix obtained for a 3D Poisson problem, the number of iterations is high. Using a preconditioner, it is possible to reduce the number of iterations but preconditioners are not scalable when using many cores.

We have performed some experiments on an infiniband cluster of three Intel Xeon quad-core E5620 CPUs of 2.40 GHz and 12 GB of memory. For the GMRES code (alone and in both multisplitting versions) the restart parameter is fixed to 16. The precision of the GMRES version is fixed to 1e−6. For the multisplitting versions, there are two precisions, one for the external solver which is fixed to 1e−6 and another one for the inner solver (GMRES) which is fixed to 1e−10. It should be noted that a high precision is used but we also fixed a maximum number of iterations for each internal step. In practice, we limit the number of iterations in the internal step to 10. So an internal iteration is finished when the precision is reached or when the maximum internal number of iterations is reached. The precision and the maximum number of iterations of CGNR method used by our Krylov multisplitting algorithm are fixed to 1e−25 and 20 respectively. The size of the Krylov subspace basis S is fixed to 10 vectors.

Figures 1 and 2 show the scalability performances of GMRES, classical multisplitting and Krylov multisplitting methods: strong and weak scaling are presented, respectively. We can remark from these figures that the performances of our Krylov multisplitting method are better than those of GMRES and classical multisplitting methods. In the experiments conducted in this work, our method is approximately, twice faster than the GMRES method and about nine times faster than the classical multisplitting method. Our multisplitting method uses a minimization step over a Krylov subspace which reduces the number of iterations and accelerates the convergence. We can also remark that the performances of the classical block Jacobi multisplitting method are the worst compared with those of the other two methods. This is why in the following experiments we compare the performances of our Krylov multisplitting method with only those of the GMRES method.

In the following, we present some experiments we could achieve out on the Hector architecture, a UK high-end computing resource, funded by the UK Research Councils [8]. This is a Cray XE6 supercomputer, equipped with two 16-core AMD Opteron 2.3 GHz and 32 GB of memory. Machines are interconnected with a 3D torus. The

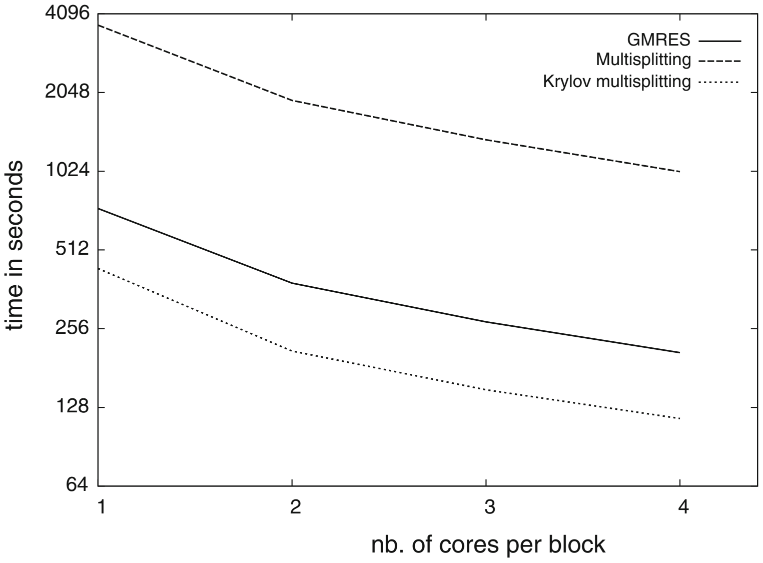


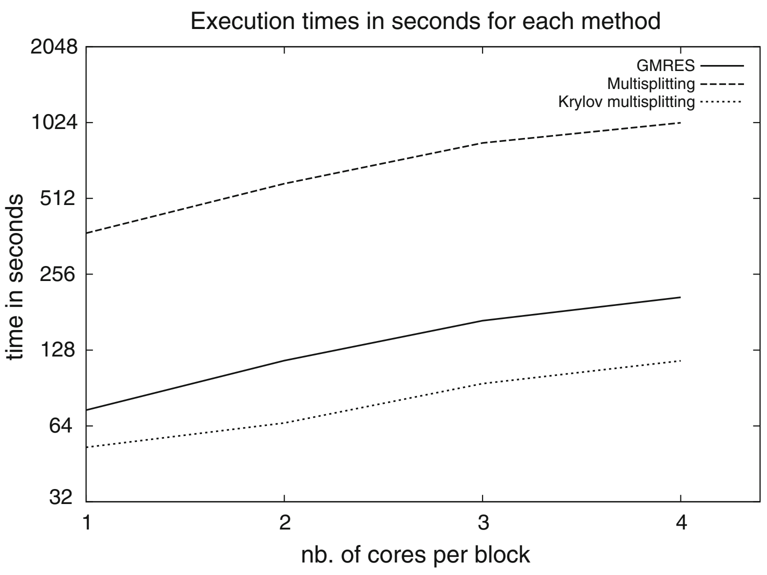
Fig. 1 Strong scaling with 3 blocks of 4 cores each to solve a 3D Poisson problem of size 150^3 components

different parameters used by the GMRES and the Krylov multisplitting codes are as those previously mentioned.

Table 1 shows the result of the experiments. The first column shows the size of the 3D Poisson problem. The size is chosen to have approximately 50,000 components per core. The second column represents the number of cores used. Between brackets, one can find the decomposition used for the Krylov multisplitting. The third column and the sixth column respectively show the execution time for the GMRES and the Krylov multisplitting codes. The fourth and the seventh column describe the number of iterations. For the multisplitting code, the total number of inner iterations is represented between brackets.

From these experiments, it can be observed that the multisplitting version is always faster than the GMRES version. The acceleration gain of the multisplitting version ranges between 4 and 6. It can be noticed that the number of iterations is drastically reduced with the multisplitting version even it is not negligible. Moreover, with 8,192 cores, we can see that using 4 blocks of cores gives a better performance than simply using 2 blocks. In fact, we can notice that the precision with two blocks is slightly better but in both cases the precision is under the specified threshold.

In Fig. 3, the number of iterations per second is reported for both GMRES and the multisplitting methods. It should be noted that we took only the inner number of iterations (i.e. the GMRES iterations) for the multisplitting method. Iterations of CGNR are not taken into account. From this figure, it can be seen that the number of iterations per second is higher with GMRES but it is not so different with the multisplitting method. For the case with 8,192 cores, the number of iterations per second with 4 blocks is approximately, equal to 115. So, it is not different from GMRES.



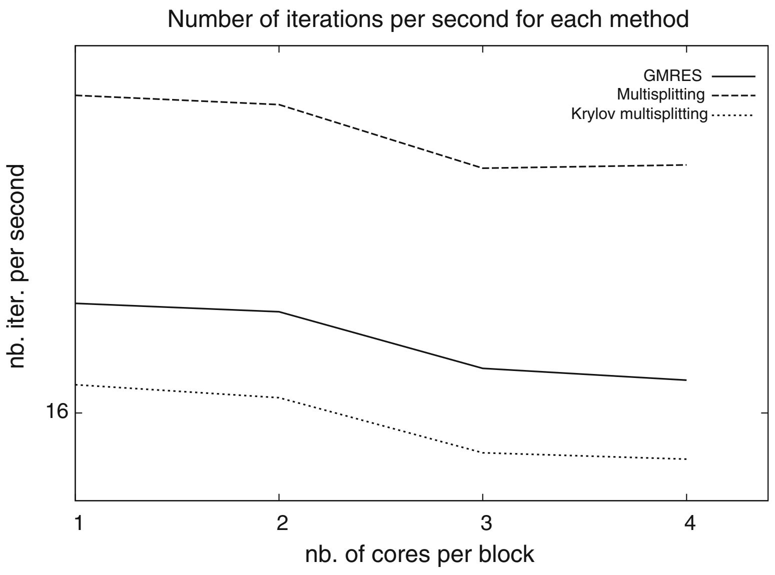


Fig. 2 Weak scaling with 3 blocks of 4 cores each to solve a 3D Poisson problem with approximately 280K components per core

4.1 Final remarks

It should be noted, on the one hand, that the development of a complete new code usable with any kind of problem is a really long and fastidious task if one is working from scratch. On the other hand, using an existing tool for the inner solver is also

Table 1 Results

Pb size	Nb. cores	GMRES			Krylov multisplitting			Ratio
		Time (s)	nb Iter.	Δ	Time (s)	nb Iter.	Δ	
468^3	2,048 (2 × 1,024)	299.7	41,028	5.02e−8	48.4	691 (6,146)	8.24e−08	6.19
590^3	4,096 (2 × 2,048)	433.1	55,494	4.92e−7	74.1	1,101 (8,211)	6.62e−08	5.84
743^3	8,192 (2 × 4,096)	704.4	87,822	4.80e−07	151.2	3,061 (14,914)	5.87e−08	4.65
743^3	8,192 (4 × 2,048)	704.4	87,822	4.80e−07	110.3	1,531 (12,721)	1.47e−07	6.39

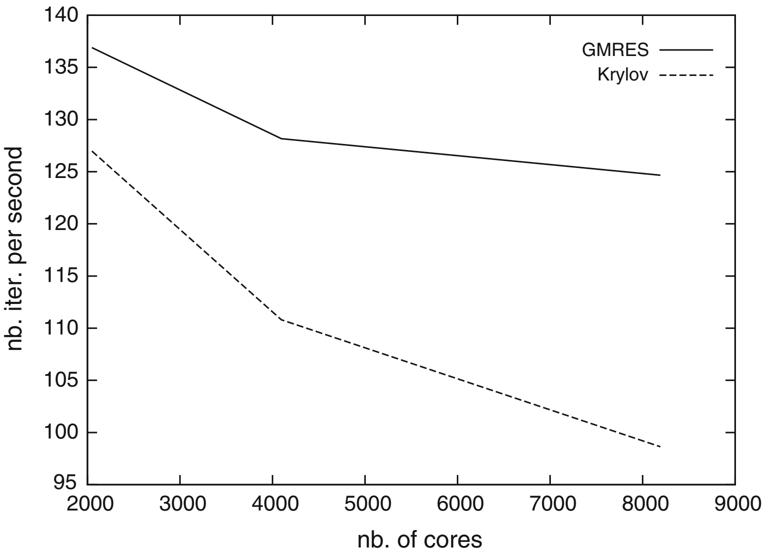


Fig. 3 Number of iterations per second with the same parameters as in Table 1 (weak scaling) with only blocks of cores

quite difficult because it requires to establish a link between the inner solver and the outer one. We plan to do that later with engineers working specifically on that point. Moreover, we think that it is very important to analyze the convergence of this method compared to other methods. In this work, we have focused on the description of this method and its performances with a typical application. Many other investigations are required for this method as explained in the next section.

5 Conclusion and perspectives

We have implemented a Krylov multisplitting method to solve sparse linear systems on large-scale computing platforms. We have developed a synchronous two-stage method based on the block Jacobi multisplitting which uses GMRES iterative method as an inner iteration. Our contribution in this paper is twofold. First, we provide a multi block decomposition that allows us to choose the appropriate size of the blocks according to the architectures of the supercomputer. Second, we have implemented the outer iteration of the multisplitting method as a Krylov subspace method which minimizes some error function. This increases the convergence and improves the scalability of the multisplitting method.

We have tested our multisplitting method to solve the sparse linear system issued from the discretization of a 3D Poisson problem. We have compared its performances to the classical GMRES method on a supercomputer composed of 2,048 up-to 8,192 cores. The experimental results showed that the multisplitting method is about 4–6 times faster than the GMRES method for different sizes of the problem split into 2 or 4 blocks when using the multisplitting method. Indeed, the GMRES method has difficulties to scale with many cores while the Krylov multisplitting method allows to hide latency and reduce the inter-block communications.

In future works, we plan to conduct experiments on larger numbers of cores and test the scalability of our Krylov multisplitting method. It would be interesting to validate its performances to solve other linear/nonlinear and symmetric/nonsymmetric problems. Moreover, we intend to develop multisplitting methods based on asynchronous iterations in which communications are overlapped by computations. These methods would be interesting for platforms composed of distant clusters interconnected by a high-latency network. In addition, we intend to investigate the convergence improvements of our method using preconditioning techniques for Krylov iterative methods and multisplitting methods with overlapping blocks.

Acknowledgments The authors would like to thank Mark Bull of the EPCC his fruitful remarks and the facilities of HECToR. This work has been partially supported by the Labex ACTION project (Contract “ANR-11-LABX-01-01”).

References

1. Bahi JM (2000) Asynchronous iterative algorithms for nonexpansive linear systems. J Parallel Distrib Comput 60(1):92–112
2. Bahi JM, Contassot-Vivier S, Couturier R (2008) Parallel iterative algorithms: from sequential to grid computing, Chapman & Hall/CRC numerical analysis and scientific computing. Chapman & Hall/CRC, Boca Roton. ISBN 9781584888086
3. Bai Z-Z, Migallón V, Penadés J, Szyld DB (1999) Block and asynchronous two-stage methods for mildly nonlinear systems. Numer Math 82(1):1–20
4. Brown N, Bull JM, Bethune I (2013) Solving large sparse linear systems using asynchronous multisplitting. Technical report, PRACE White paper number WP84
5. Bru R, Migallón V, Penadés J, Szyld DB (1995) Parallel, synchronous and asynchronous two-stage multisplitting methods. Electron Trans Numer Anal 3:24–38
6. Couturier R, Denis C, Jézéquel F (2008) Gremlins: a large sparse linear solver for grid environment. Parallel Comput 34(6):380–391
7. Frommer A, Szyld DB (1992) H-splittings and two-stage iterative methods. Numer Math 63(1):345–356
8. HECToR: UK National Supercomputing Service. http://www.hector.ac.uk
9. Huang C-M, O’Leary DP (1993) A krylov multisplitting algorithm for solving linear systems of equations. Linear Algebra Appl 194:9–29
10. Li C (2001) An adaptive CGNR algorithm for solving large linear systems. Ann Oper Res 103(1–4):329–338
11. O’Leary DP, White RE (1985) Multi-splittings of matrices and parallel solution of linear systems. SIAM J Algebra Discrete Methods 6(4):630–640

12. Saad Y (1996) Iterative methods for sparse linear systems. PWS Publishing, New York
13. Saad Y, Schultz MH (1986) GMRES: a generalized minimal residual algorithm for solving nonsymmetric linear systems. SIAM J Sci Stat Comput 7(3):856–869
14. Ziane Khodja L, Couturier R, Giersch A, Bahi J (2014) Parallel sparse linear solver with GMRES method using minimization techniques of communications for GPU clusters. J Supercomput 69(1):200–224