Synchronous parallel multisplitting method with convergence acceleration using a local Krylov-based minimization for solving linear systems

Médane A. Tchakorom Raphaël Couturier Jean-Claude Charr
FEMTO-ST Institute, CNRS
Univ. Bourgogne Franche-Comte (UBFC)
Belfort, France
{medane.tchakorom, raphael.couturier, jean-claude.charr}@univ-fcomte.fr

***Abstract*—Computer simulations of physical phenomena, such as heat transfer, often require the solution of linear equations. These linear equations occur in the form Ax = b, where A is a matrix, b is a vector, and x is the vector of unknowns. Iterative methods are the most adapted to solve large linear systems because they can be easily parallelized.**

This paper presents a variant of the multisplitting iterative method with convergence acceleration using the Krylov-based minimization method. This paper particularly focuses on improving the convergence speed of the method with an implementation based on the PETSc (Portable Extensible Toolkit for Scientific Computation) library. This was achieved by reducing the need for synchronization - data exchange - during the minimization process and adding a preconditioner before the multisplitting method. All experiments were performed either over one or two sites of the Grid5000 platform and up to 128 cores were used. The results for solving a 2D Laplacian problem of size 1024^2 components, show a speed up of up to 23X and 86X when respectively compared to the algorithm in [8] and to the general multisplitting implementation.

***Index Terms*—Iterative methods, Krylov methods, Multisplitting, Linear Solvers, PETSc, MPI**

I. Introduction

To simulate physical phenomena, scientists often need to solve systems of equations of the form $Ax = b$, where A is a large, sparse $n \times n$ matrix and both x and b are vectors of size n. Several methods have been developed over the years to solve this type of problem. The means to solve the problem are twofold: direct methods and iterative methods.

Direct methods are mostly inspired from the Gaussian elimination method and compute an exact solution of the linear system. However, they are compute and memory intensive and thus not adapted for solving large linear systems. On the other hand, iterative methods provide an approximation to the solution after a finite number of steps. They are easier to parallelize than the direct methods and can take advantage of the computer power of distributed computing architectures, such as clusters and grids, to solve large sparse linear systems.

In this computational context, there are two types of challenges for iterative parallel algorithms depending on whether they are synchronous or asynchronous: The heterogeneity of the computing nodes and the inherent latency of the network [2]. The heterogeneity of the processors can limit the parallel synchronous iterative method to the speed of the slowest processor. The synchronous exchange of data between processors forces each processor to wait until the other processors have finished their work before processing the next iteration. The network latency increases the communication time between computing nodes and might have a great impact on the execution time of fine grained parallel algorithms that require many data exchanges. The asynchronous iterative methods and multisplitting methods are the most adapted to this environment because they reduce the need for data synchronization between the computing nodes.

In a previous work [8], the iterative multisplitting method and Krylov subspace minimization, both presented in Section III, were combined into one method. The conducted numerical experiments showed that with this new approach the convergence of the multisplitting method can be accelerated by using the Krylov subspace minimization method and can outperform classical iterative methods such as the GMRES [16] and the Block-Jacobi multisplitting method [2]. In this paper, this previous work is extended in order to furthermore improve the convergence speed of the method through two means: the first step consists into taking advantage of HYPRE boomerAMG preconditioner, as the role of a preconditioner is to condition a given problem to a more suitable form for a numerical solving method. The second step consists in improving the minimization by eliminating the need for synchronization. This last improvement not only reduces the number of outer and inner iterations required to converge, it also reduces the execution time of the inner iterations.. The experiments conducted over the Grid5000 platform while solving a 2D Laplace problem, showed that as expected the new method converges faster than the general multisplitting algorithm and the algorithm presented in [8].

The remainder of the paper is organised as follows: Section II discusses existing work on this topic and introduces the reader to the PETSc library. Section III is devoted to the introduction of the multisplitting algorithm and then to the hybrid algorithm based on the multisplitting method and the Krylov subspace. The context and results of the experiments, as well as their interpretation, are covered in Section IV. Section V recapitulates the main contributions of this paper and present some future work.

II. Related work

Multisplitting is a parallel iterative method, first introduced in [14] by O'Leary and White, in which they presented several basic convergence results. It is based on several splittings of the coefficient matrix A from the linear system of equation $Ax = b$, and can be seen as a generalization of the classical Block Jacobi parallel iterative algorithm [2]. Since its first introduction, the scientific literature has dealt with different variants of multisplitting, e.g., asynchronous versions [6], different convergence conditions [1], [3], and other two-stage algorithms [6], [9].

In [12], the authors proposed a practical serial implementation and numerical examples for the Krylov subspace method in the case where the preconditioning matrix results from multisplitting. They showed that they could implement the methods efficiently by creating subsets of processors tasked with generating the search direction and another subset of processors responsible for the minimization over the result subspace. The fact that the minimization task does not need to send frequent updates to the direction generating task minimizes synchronization. They also studied the convergence properties of various forms of the algorithm.

The authors of [7] worked on GREMLINS (GRid Efficient Method for LINear Systems), a parallel iterative solver adapted to the grid computing environment and known to penalize traditional large sparse linear solvers. The solver is in fact based on the multisplitting algorithm, where the inner solver is either the serial direct method LU or the serial iterative method GMRES. Moreover, the GREMLINS algorithm can be executed synchronously or asynchronously without any further modifications.

Extending their earlier work, the authors of [7] applied the Jocabi algorithm and other block-level multisplitting algorithms using asynchronous and sychronous communications. They then used a Krylov subspace method-based approach within each block and investigated the performance and scalability of these approaches on up to 32768 cores on a Cray XE6.

A. Introduction to PETSc

Since developing parallel applications can cause some difficulties for scientists and engineers, they can rely on high-level software libraries to ease some processes. Such libraries help developers by providing abstractions for mathematical operations, data representation, and managing parallel layouts of data, while internally using communication libraries such as MPI (Message Passing Interface) implementations and PVM (Parallel Virtual Machine) [4] PETSc (Portable Extensible Toolkit for Scientific Computation) is a collection of data structures and procedures developed by Argonne National Library for scalable (parallel) solution of scientific applications modeled by partial differential equations. It is based on the message-passing model, as presented in Figure 1, and combines the robustness and efficiency of MPI with a carefully designed and implemented parallel numerical library. MPI is recognized as the standard message exchange library for developing distributed numerical applications in C. PETSc provides a wide range of iterative methods to solve PDEs and related problems, Figure 1. PETSc can be configured through parameters passed on the command line which makes it very easy to test various solvers and preconditioners on the go. In addition, PETSc is available in C and FORTRAN programming languages and can be extended through well-known packages like Hypre preconditioner or external packages like BLAS/LAPACK. Finally, PETSc offers a good compromise between usability and efficiency, promoting code reuse and flexibility while separating parallelism concerns from algorithm selection [5].

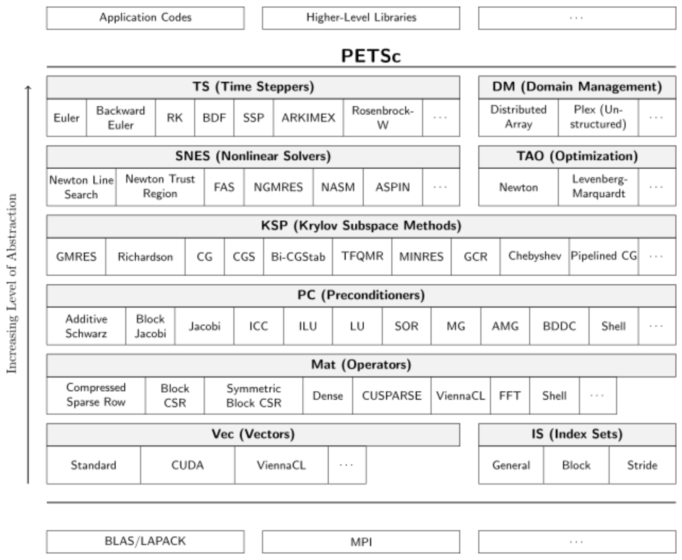

Fig. 1. Numerical libraries of PETSc [5]

III. Algorithm Krylov Multisplitting

The hybrid algorithm discussed in this paper is based on two well-known iterative methods: The multisplitting approach, which is the basis of the classical Jacobi method, and the error minimization approach, which inspired the classical GMRES method. This section first explains the multisplitting method in details, then it presents the multisplitting method combined with the minimization process. The last part of this section is devoted to the implementation of the algorithm with PETSc.

A. Multisplitting

Let us consider $A \in \mathbb{R}^{n\times n}$, a large sparse square and non-singular matrix, b the right-hand-side vector and x the vector solution of the following linear system:

$$Ax = b, \tag{1}$$

The idea behind the multisplitting method is that matrix A, can have several decompositions [14]. If A can be partitioned in L different ways:

$$A = M_l - N_l,\ l = 1, \ldots, L, \tag{2}$$

where M_l and N_l are $n \times n$ matrices, and M_l are non-singular. Then a multisplitting of A is defined by:

$$B = \sum_{l=1}^{L} D_l M_l^{-1} N_l, \tag{3}$$

where D_l are non-negative and diagonal weighted matrices that add up to the identity matrix, $\sum_{l=1}^{L} D_l = I$.
Equation 3 leads to the equation:

$$x^{k+1} = \sum_{l=1}^{L} D_l M_l^{-1} (N_l x^k + b),\ k = 1, 2, \ldots \tag{4}$$

where x^k is the solution of the equation at iteration k. The initial guess x^0 is set before the iteration process begins. Another equivalent form can be written:

$$x^{k+1} = Bx^k + Gb, \tag{5}$$

where

$$G = \sum_{i=1}^{p} D_i M_i^{-1}, \tag{6}$$

The convergence of such a multisplitting method has been studied and it was demonstrated that to converge the spectral radius of matrix B (as defined below) should be lower than 1:

$$\rho(\sum_{i=1}^{p} D_i M_i^{-1} N_i) < 1, \tag{7}$$

Part of the motivation for using the multisplitting method is that it decomposes the linear system into smaller sub-systems which can be easily solved in parallel using a distributed computing architecture. Communication is only required to aggregate the results using the diagonal weighting factors. Thus each subset of computing units is assigned a linear sub-system of the following form:

$$x_l^k = M_l^{-1} N_l x_l^{k-1} + M_l^{-1} b,\ l \in \{1, \ldots, L\} \tag{8}$$

where x_l^k is the solution of the l sub-system at iteration k. The solution of the linear system $Ax = b$, at iteration k, can then be written as follows:

$$x^k = \sum_{l=1}^{L} D_l x_l^k,\ k = 1, 2, \ldots \tag{9}$$

The multisplitting algorithm which performs well [13], [15] can be accelerated by employing strategies based on Krylov subspace methods [10]. The Krylov subspaces, which are at the heart of several well-known iterative methods, such as GMRES, have already been investigated and proven to be reliable. To speed up the convergence of the problem at hand, our approach uses both the multisplitting technique and the Krylov method.

B. Multisplitting with minimization of residual algorithm

The linear systems produced in Equation 8 can be solved either using a direct or an iterative method. When an iterative method is used, the resulting resolution method is called a *two-stage* iterative algorithm. In the rest of the paper, the *inner and outer iteration* corresponds to the multisplitting method and the Krylov minimization method, respectively.

In order to solve Equation 1, a splitting is first applied to the linear system. This splitting leads to the decomposition of A into L non-intersecting blocks or sub-matrices as shown in Figure 2. Each A_l is a rectangular sub-matrix of size $n_l \times n$, and X_l, B_l are sub-vectors of size n_l each, such that $\sum_{l=1}^{L} n_l = n$.

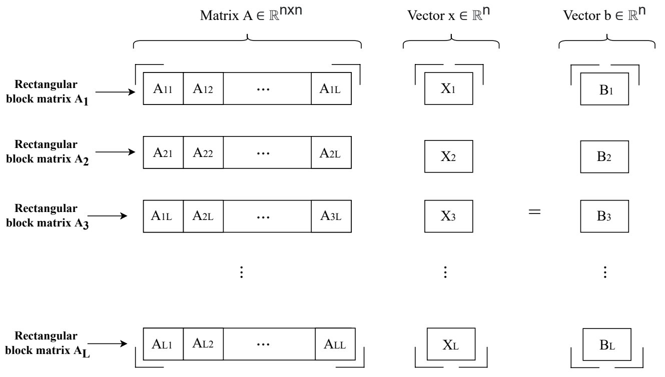

Fig. 2. The multisplitting method splits the matrix A, the vector solution $x \in \mathbb{R}^n$ and the vector of the right-hand side $b \in \mathbb{R}^n$ into L different blocks, each is processed by a group of processors.

The multisplitting form of the linear system is defined as follows:

$$\forall\, l \in \{1, ..., L\},\ A_{ll} X_l + \sum_{\substack{m=1 \\ m \neq l}}^{L} A_{lm} X_m = B_l, \tag{10}$$

where A_{lm} is a sub-block of size $n_l \times n_m$ of the rectangular matrix A_l, $X_m \neq X_l$ is a sub vector of size n_m of the solution vector x and $\sum_{m \neq l} n_m + n_l = n$, for all $m \in \{1, \ldots, L\}$.
Each sub-system, (eq. 11) can then be solved independently by the *inner iteration* solver:

$$\begin{cases} A_{ll} X_l = Y_l, \text{ such that} \\ Y_l = B_l - \sum_{\substack{m=1 \\ m \neq l}}^{L} A_{lm} X_m, \end{cases} \tag{11}$$

The outer iteration is dedicated to minimizing an error function over a Krylov subspace to speed up convergence of the inner iteration above. The minimization process needs a basis vector which is the Krylov subspace. This subspace consists of successive solutions generated from the solutions of the L splittings:

$$S = \{x^1, x^2, ..., x^s\},\ s \leq n, \tag{12}$$

where for $j \in \{1, ..., s\}$, and

$$x^j = [X_1^j, ..., X_L^j], \tag{13}$$

is a solution of the global linear system. Until the stop criteria is reached, the iteration process of the algorithm is resumed with a fresh starting estimate $\tilde{x} = S\alpha$. α is defined as follows:

$$\alpha = [\alpha_1, ..., \alpha_L], \tag{14}$$

and each $\alpha_l, l \in \{1, \ldots, L\}$ is defined as follows:

$$R_l \alpha_l = b_l, \tag{15}$$

where $R_l = A_l S$ is a dense rectangular matrix of size $n_l \times s$ and $s << n_l$ (s much lower than n_l). This leads to solve a system of normal equations

$$R_l^T R_l \alpha = R_l^T b, \tag{16}$$

which is associated with the least squares problem

$$minimize \, \|b_l - R_l \alpha_l\|_2 \tag{17}$$

where R_l^T denotes the transpose of matrix R_l. The main steps of the algorithm are summarized in Algorithm 1.

Note: In the previous version of this algorithm [8], the value of α in equation 15 was computed after merging the $R_l, l \in \{1, \ldots, L\}$ matrices from each block. The equation was as follows:

$$R\alpha = b, \tag{18}$$

where $R = \{R_1, \ldots, R_l\}$. This leads to solve Equation 18 in each block. This equation has more unknowns than Equation 15 and yields to more execution time during the steps of converting the system into a square symmetric system (Equation 16) and then solving the equation with a solver. The new approach is therefore intuitively less costly in computing time and reduces the load of work on each subset of processors. Moreover, there is less communication in this part. In fact, inter-blocks synchronization to exchange sub-vectors α_l of size $n_l \times 1$ is less costly in time than the synchronization of sub-matrices R_l of size $n_l \times s$.
The PETSc library provides a broad selection of data structures, solvers and preconditioners to implement fast and efficient parallel numerical codes. However, implementing a parallel method in PETSc is not a trivial task. The developer have to identify the most appropriate objects to use. For Algorithm 1, choosing the right data structure for storing the sparse matrix A was of great importance for implementing this algorithm in PETSc. Moreover, Allocating the memory for the matrix before trying to insert data, saved a lot of time during the experimentation. Many solvers were evaluated in order to find out which one was the most suitable for our method.

Algorithm 1 A two-stage linear solver

Input: A_l (sparse sub-matrix), b (right-hand side vector), L (number of blocks)
Output: x (solution vector)

```
1:  $k$ = current iteration number
2:  $\epsilon$ = Tolerance
3:  $X_l^j$ = Block level multisplitting solution vector
4:  $x_l^j = [X_1^j, ..., X_L^j]$
5:  $X_l$ = Block level solution vector after minimization
6:  $x = [X_1, ..., X_L]$
7:  $S$ = Minimization Krylov basis vector
8:  Load $A_l$, $b$
9:  Set $x$ to the an initial guess $x^0$
10: do
11:     for j = 1,...,s do
12:         Compute the right hand side $Y_l$ (Equation 11)
13:         Solve the block level linear system $A_{ll} X_l^j = Y_l$
            with GMRES parallel method
14:         Update the multisplitting global solution $x_l^j$ by
            assembling the block level solutions $X_l^j$ (Equation 13)
15:         Update Krylov basis S: Add $x_l^j$ to the $j^{th}$ column
            of S (Equation 12)
16:     end for
17:     Compute $R_l = A_l \times S$
18:     Solve the linear equation $R_l^T R_l \alpha = R_l^T b$ with the
        CGLS parallel solver
19:     Compute the block level solution $X_l = S \times \alpha$
20:     Update the global solution $x$ by assembling the local
        solutions $X_l$
21: while $\|b - Ax\|_2 > \epsilon$
```

IV. Numerical experiments

This section contains a detailed description of the experiments conducted over Grid50000 to evaluate the performance and the scalability of our approach. In the next subsections, the context of the experiments, the tuning of the parameters, the obtained results, and their interpretation are presented.

A. *Context of the experiments*

The chosen partial differential equation to be solved is the 2D Laplace equation. The finite difference method is used to discretize it and a five-point stencil is used. The right hand side of the linear equation, vector b, represents the boundary conditions of Laplace's equation on a rectangular domain D. Each block portion of the matrix A was multiplied by a vector whose elements was set to 1. The result which is a symmetric positive definite linear system, was assigned to each block portion of vector b.

$$\begin{cases} \nabla^2 u = \frac{\partial^2 u}{\partial x^2} + \frac{\partial^2 u}{\partial y^2} = 0, \; (x, y) \in D \\ \text{Boundary conditions: } u \text{ prescribed on } \partial D \end{cases} \tag{19}$$

Three implementations were used for the experiments:

- ***Multisplitting with global minimization***: This method corresponds to the algorithm described in [7]. The R value, appearing in equation 18, is synchronized between the blocks after the multisplitting iterations and before the minimization process.

- ***Multisplitting with local minimization***: This is an updated version of the previous algorithm where the required synchronization between the blocks has been reduced as explained in Section III-B.
- ***General multisplitting***: This corresponds to the multisplitting algorithm without minimization that was implemented using the PETSc library.

In the rest of this paper, each method will be referred to by the name it was given in the above description.

All experiments were conducted over the Grid5000 platform which is a large-scale and flexible testbed for experimental research in all areas of computer science with a focus on parallel and distributed computing, including cloud, HPC, Big Data, and AI. Grid5000 consists of 31 clusters, 828 nodes and 12,328 cores geographically distributed across 8 sites, 7 in France and 1 in Luxembourg, as illustrated in Figure 3. A 10 Gbps backbone network is dedicated to the connection of all sites.

The clusters *paravance* in Rennes and *grisou* in Nancy were selected to run the experiments because their computing nodes have the same specifications in terms of computing power and memory. They are all equipped with two Intel Xeon E5-2630 v3 processors with 8 cores per CPU and 128 GiB of memory. The bandwidth of the paravance's network is 2x10 Gbit/s and 4x10 Gbit/s for grisou.

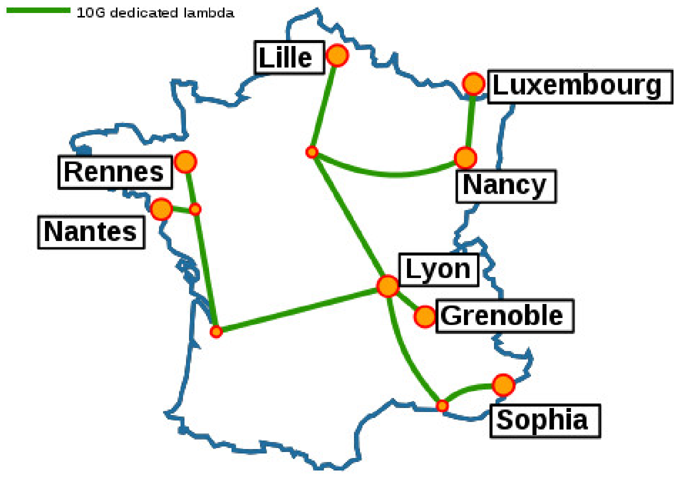

Fig. 3. Grid5000 clusters distribution on Jan.17

B. Parameters tuning

The number of blocks was set to 2 to enable rapid convergence [8]. So the splitting, which is basically a Block Jacobi splitting [2], consists in dividing matrix A into two rectangular matrices. For the inner iterations solver (i.e multisplitting), GMRES, with a restart parameter equal to 30, was used because the Gradient Conjugate solver does not provide the possibility to restart it after a given number of iterations. The maximum number of iterations was set to the default value 10000, and the relative tolerance fixed to 10^{-12}. HYPRE boomerAMG was used as a preconditioner for the inner iteration. Figure 5 shows the impact of using a HYPRE boomerAMG preconditioner on the execution time of the algorithm.

For the minimization step, the Conjugate Gradient method for Least-Squares problems (CGLS) [11] was used as solver . No preconditioner was set because CGLS does not support any preconditioner. The maximum number of iterations was set to the default value of 500, and the relative tolerance was fixed to 10^{-30}, after testing multiple values. For the tolerance of the outer solver, ϵ was set to the value of 10^{-3}. Different values for the size of the Krylov subspace (s) was tested in the minimization process. The results of these tests are presented in Figure 4 and they demonstrate that 4 is the most appropriate value for s.

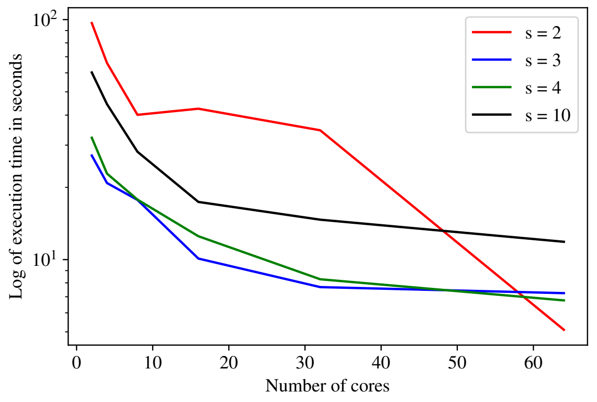

Fig. 4. Variation of execution time of the Multisplitting with local minimization implementation, with different values of minimization Krylov subspace size (s) and number of blocks equals 2 on the clusters paravance and grisou (Rennes site and Nancy). Laplace 2D equation with problem size 1448^2 .

C. Results and discussions

With more computing resources, parallel programs should be able to minimize the overall execution time. To assess the performance and the scalability of the proposed algorithm while varying the size of the linear system and/or the number of processes, two cases, the strong and weak scaling, are studied in this section. This evaluation also gives a good estimate of the required computing resources for different sizes of the problem.

In the strong scaling case, the number of processors is increased while the problem size remains constant. Figure 6 shows how the three methods, discussed in this paper, perform while solving a 2D Laplacian problem of size 1448^2 components. The two multisplitting with minimization methods perform well in terms of computing time when compared to the multisplitting method without minimization, comforting the intuition that the minimization process contributed in accelerating the convergence of the inner iterative process (i.e the multisplitting process). Furthermore, the local minimization multisplitting is the fastest version since the synchronization

Pb. size	Nb. cores	Multisplitting with global minimization		Multisplitting with local mimization		Ratio
		Time (s)	Residual	Time (s)	Residual	
1024²	4 (2 x 2)	331.06	9.573901e-04	14.20	4.635208e-04	23.31
1448²	8 (2 x 4)	555.86	9.843735e-04	16.25	4.716789e-05	34.21
2048²	16 (2 x 8)	898.20	9.884657e-04	27.76	4.726606e-04	32.36
2896²	32 (2 x 16)	1812.25	9.951199e-04	27.56	3.397707e-04	65.75
4096²	64 (2 x 32)	2864.78	8.731609e-04	35.00	2.913539e-04	81.85
5792²	128 (2 x 64)	4915.09	9.977175e-04	55.21	3.035485e-04	89.03

TABLE I
WEAK SCALING RESULTS ON TWO SITES (RENNES AND NANCY)

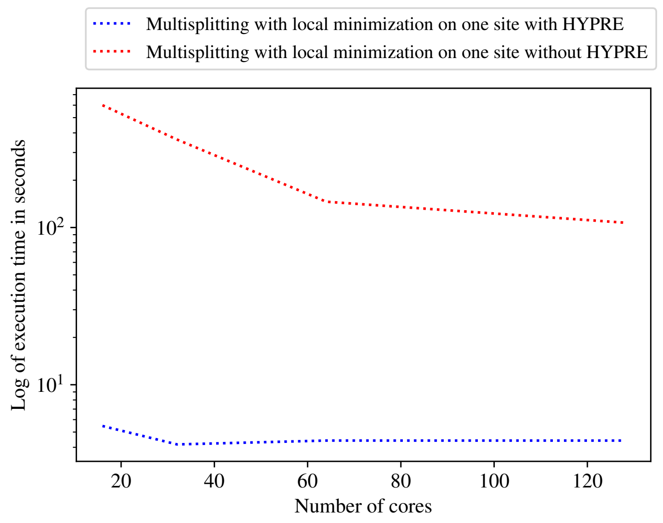

Fig. 5. Using the multisplitting method with local minimization and with or without the HYPRE boomerAMG preconditioner to solve 2D Laplace equation of size 1024^2 components.

steps between blocks are reduced and the sub-system to solve by the minimizer is smaller. The results of the experiments, conducted over a single cluster and two geographically distant clusters, showed that the different methods approximately scale in the same way on both architectures. This suggests that the network latency does not significantly penalize any of the three methods.

The second case is the weak scaling, for which both the problem size and the number of processors are increased at the same rate. Table I shows how the two multisplitting with minimization algorithms performed when the size of the problem and the number of cores was doubled at the same time. The number of components by core is approximately equal to $262,144$. The multisplitting with local minimization outperformed the multisplitting version with global minimization for all the tested problem sizes. The results presented in Table I showed a speed up ranging between 23x and 89x when the execution time of both implementations is compared.

V. CONCLUSION AND PERSPECTIVES

This paper presents an improved version of the Krylov multisplitting algorithm based on the residual minimization. The number of communications between the computing nodes was reduced when compared to the previous version which

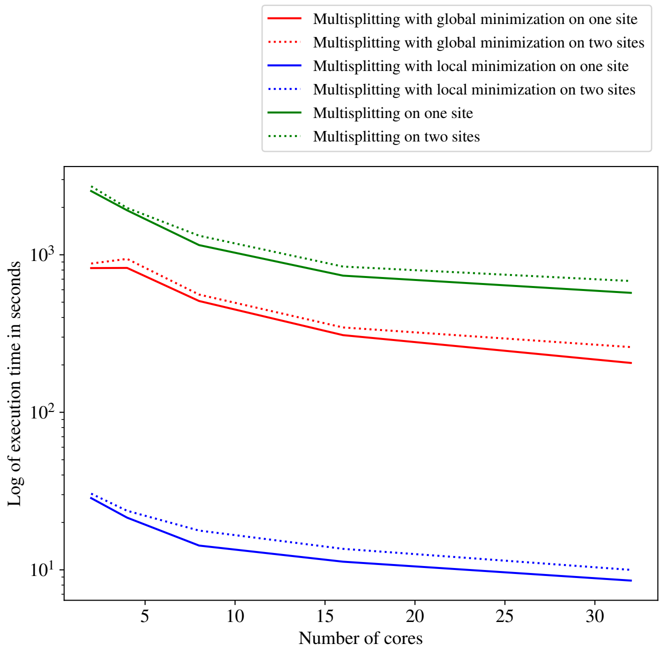

Fig. 6. Strong scaling with up to 32 cores and 2 blocks to solve a 2D Laplace equation of size 1448^2 components

also decreases its computation time. Moreover, the use of the local minimization process seems (based on our experiments) to increase the convergence speed. The number of iterations in the inner solver was significantly reduced. The algorithm was implemented with the well known PETSc library and it was evaluated in a grid environment. The experiments were conducted over two geographically distant clusters from Grid5000 and showed that the local minimization is up to 89X faster than the global minimization version when solving the 2D Laplace problem.

In a future work, it is our intent to further extend this work by investigating other linear equation systems and solving larger problems using different solvers available in the PETSc library. We also plan on mathematically proving the convergence of the method with local minimization. Finally, we think that the asynchronous version of the multisplitting method could considerably benefit from the local minimization. Therefore, the application of this improvement to the asynchronous method will be studied.

VI. Acknowledgement

This work was supported by the ANR ADOM (contract ANR-18-CE46) and it was partially supported by the EIPHI Graduate School (contract "ANR-17-EURE-0002"). We would also like to thank the Mésocentre de Calcul de Franche-Comté and Grid'5000.

References

[1] Jacques M. Bahi. Asynchronous Iterative Algorithms for Nonexpansive Linear Systems. *Journal of Parallel and Distributed Computing*, 60(1):92–112, 2000.

[2] Jacques Mohcine Bahi, Sylvain Contassot-Vivier, and Raphaël Couturier. *Parallel iterative algorithms: from sequential to grid computing*. CRC Press, 2007.

[3] Zhong-Zhi Bai, Violeta Migallón, José Penadés, and Daniel B. Szyld. Block and asynchronous two-stage methods for mildly nonlinear systems. *Numerische Mathematik*, 82(1):1–20, March 1999.

[4] Pavan Balaji, Darius Buntinas, Satish Balay, Barry Smith, Rajeev Thakur, and William Gropp. Nonuniformly communicating noncontiguous data: A case study with petsc and mpi. In *2007 IEEE International Parallel and Distributed Processing Symposium*, pages 1–10. IEEE, 2007.

[5] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Steven Benson, Jed Brown, Peter Brune, Kris Buschelman, Emil Constantinescu, Lisandro Dalcin, Alp Dener, Victor Eijkhout, William D. Gropp, Václav Hapla, Tobin Isaac, Pierre Jolivet, Dmitry Karpeev, Dinesh Kaushik, Matthew G. Knepley, Fande Kong, Scott Kruger, Dave A. May, Lois Curfman McInnes, Richard Tran Mills, Lawrence Mitchell, Todd Munson, Jose E. Roman, Karl Rupp, Patrick Sanan, Jason Sarich, Barry F. Smith, Stefano Zampini, Hong Zhang, Hong Zhang, and Junchao Zhang. PETSc/TAO Users Manual. Technical Report ANL-21/39 - Revision 3.16, Argonne National Laboratory, 2021.

[6] Rafael Bru, Violeta Migallón, Jos Penad, Es Daniel, and Daniel Szyld. Parallel, Synchronous And Asynchronous Two-Stage Multisplitting Methods. *Electronic Transactions on Numerical Analysis*, 3, August 1998.

[7] Raphaël Couturier, Christophe Denis, and Fabienne Jézéquel. GREMLINS: a large sparse linear solver for grid environment. *Parallel Computing*, 34(6-8):380–391, 2008. Publisher: Elsevier.

[8] Raphaël Couturier and Lilia Ziane Khodja. A scalable multisplitting algorithm to solve large sparse linear systems. *The Journal of Supercomputing*, 71:1–12, December 2014.

[9] Andreas Frommer and Daniel B. Szyld. H-Splittings and two-stage iterative methods. *Numerische Mathematik*, 63(1):345–356, December 1992.

[10] Gene H Golub and Charles F Van Loan. Matrix Computations. edition, 1996.

[11] Magnus R Hestenes and Eduard Stiefel. Methods of conjugate gradients for solving. *Journal of research of the National Bureau of Standards*, 49(6):409, 1952.

[12] Chiou-Ming Huang and Dianne P. O'Leary. A Krylov multisplitting algorithm for solving linear systems of equations. *Linear Algebra and its Applications*, 194:9–29, 1993.

[13] Michael Neumann and Robert J Plemmons. Convergence of parallel multisplitting iterative methods for M-matrices. *Linear algebra and its applications*, 88:559–573, 1987. Publisher: Elsevier.

[14] Dianne P. O'leary and R. E. White. Multi-splittings of matrices and parallel solution of linear systems. *SIAM J. Alg. Disc. Meth*, pages 630–640, 1986.

[15] Theodore S Papatheodorou and Yiannis G Saridakis. Parallel algorithms and architectures for multisplitting iterative methods. *Parallel computing*, 12(2):171–182, 1989. Publisher: Elsevier.

[16] Youcef Saad and Martin H Schultz. GMRES: A generalized minimal residual algorithm for solving nonsymmetric linear systems. *SIAM Journal on scientific and statistical computing*, 7(3):856–869, 1986. Publisher: SIAM.