A comparative study of some distributed linear solvers on systems arising from fluid dynamics simulations

Désiré NUENTSA WAKAM [a,1] Jocelyne ERHEL [a] Édouard CANOT [b]
Guy-Antoine ATENEKENG KAHOU [c]

[a] *INRIA-Rennes, Campus de Beaulieu, 35042 Rennes Cedex*
[b] *CNRS-IRISA, Campus de Beaulieu, 35042 Rennes Cedex*
[c] *INRIA-Saclay, Parc Orsay Université, 91893 Orsay Cedex*

Abstract. This paper presents a comparative study of some distributed solvers on a set of linear systems arising from Navier-Stokes equations and provided by an industrial software. Solvers under consideration implement direct, iterative or domain decomposition methods and most of them are freely available packages. Numerical tests with various parameters are made easier by developing a unified toolbox that links with interface functions provided by these libraries. The intensive numerical tests performed on various sets of processors reveal the good performance results achieved by the recently proposed parallel preconditioner for Krylov methods based on an explicit formulation of multiplicative Schwarz [1].

Keywords. Fluid dynamics simulation, large linear systems, distributed solvers, parallel preconditioning, Multiplicative Schwarz, Additive Schwarz.

1. Problem Definition

In this paper, we are interested in finding a good solver for a class of large linear systems

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

where $A \in \mathbb{R}^{n\times n}$ is a real and unsymmetric sparse matrix, x, $b \in \mathbb{R}^n$ are respectively solution and right hand side vectors. The matrix A corresponds to the global jacobian matrix resulting from the partial first-order derivatives of the Reynolds-averaged Navier-Stokes equations. The derivatives are done with respect to the conservative fluid variables. There are various linear solvers libraries freely available and a task of finding a good one among them (for our set of linear systems) is not easy by itself. Although a theoretical analysis of the problem can suggest a class of solver, it is necessary to consider numerical comparisons on the problem being solved. These comparisons include, but are not limited to, memory usage, reliability, parallel efficiency, CPU time and accuracy in the final solution. So in this work, we present a comparative study of some

[1]Corresponding Author: Désiré NUENTSA WAKAM, IRISA, Campus de Beaulieu, 35042 Rennes cedex, e-mail address: desire.nuentsa_wakam@irisa.fr

distributed linear solvers on the above-mentioned set of linear systems. We do not have the pretension to consider all existing distributed solvers in this short study neither all aspects in the solvers as in [2,3]. At least, we expect this numerical study to suggest which method is appropriate for this problem. This study is also motivated by the performance achieved on these systems using the parallel preconditioned GMRES with the explicit formulation of multiplicative Schwarz [4]. As this kind of study needs many tests with various parameters, we have found useful to design a unified interface that helps us to link uniformly to the interfaces provided by the solvers. However, we should stress on the fact that our main goal in this work is not to offer a generic framework such as Trilinos [5] or Numerical Platon [6] but to test and compare each method suitable for our set of linear problems; so the toolbox is designed primarily to switch between all the solvers under study in some easy and uniform way. The paper is organized as follows. In the next section, the distributed solvers we used in this study are listed. The third part gives an overview of the toolbox. The section 4 is the major part of this work: it is devoted to the experimental comparisons. Concluding remarks are given at the end.

2. Distributed Linear Solvers

Traditionally speaking, the solvers suitable for the system (1) are based either on sparse direct or iterative methods. But with the actual state-of-art, the separation between these two classes is tight. Presently, techniques from the first class are used as preconditioners into the second class. Even in the second class, there are a variety of techniques based on Krylov subspace methods or multilevel methods (Multigrid, Domain decomposition). We first consider the solution with two distributed direct solvers, namely SuperLU_DIST [9] and MUMPS [10]. They are representative of two widely-used techniques in this class. Almost all aspects in both packages have been thoroughly compared [2] using a collection of matrices of reasonable size. Our guess is that the need of memory will become a bottleneck with our present collection of matrices. In fact, this memory usage can be reduced significantly when direct methods are used in incomplete form as preconditioner for iterative methods. So, in this work, we consider EUCLID [11], the recommended ILU preconditioner in HYPRE [12] library. Secondly, we focus on domain decomposition methods. Acting as preconditioners for the Krylov subspace methods (essentially GMRES method), they make use of previous methods to solve (more or less) the local problems induced by the decomposition.

When using Domain Decomposition methods to solve PDE equations, a classical scheme is to consider the splitting from the computational domain. Here, we consider rather a load balancing partitioning based on the adjacency graph of the matrix. Schur complement approaches use a partitioning without overlap while Schwarz methods are applied to partitions that are allowed to overlap. First, we consider the pARMS [8] package based on the first group. In the second group, we use the additive Schwarz preconditioner in the PETSc package.

The convergence of the Schwarz methods is better with a successive correction of the residual vector over the subdomains. This is the case in the Multiplicative Schwarz. However, it leads to an inefficient preconditioner in parallel environment due to the high dependencies of data between the subdomains. In a recent work [1], the authors proposed an explicit formulation of this preconditioner in order to dissociate the computation of the residual vector from the preconditioner application. This explicit form is used in conjunction with the parallel version of GMRES proposed in [13]. Hence, the preconditioned Newton-basis is first constructed in a pipeline over all processors [4]; then, a parallel version of QR factorization [14] is called to get an orthogonal basis. In this study, we use the result of that work which is expressed in the PETSc format and available in a library named as GPREMS (Gmres PREconditioned by Multiplicative Schwarz)[2].

3. Environment of Tests

Our main goal here is to build a ready-to-use interface toolbox such that we can uniformly test any method presented above. The PETSc installer tool is used to build compatible libraries of some of the solvers under study. Figure 1 gives a simplified overview of this architecture. The *routines for sparse matrix format* are provided to read data of systems from files (matrices and right hand side). These data can be in compressed Harwell Boeing format, in Coordinate Matrix Market format, or in compressed block sparse row. The *parameter routines* define *classes* that are used to select options for solvers as well as other parameters either from XML files or PETSc-style database options. At the top level, the test routines define interface functions to all solvers under consideration. So, we need only to choose a solver, edit or generate parameter file and give it to the test routine along with matrix and right hand side file. At the end of execution, the main statistics are returned in html (XML) or text file via the *statistics routines*. As our toolkit has a capability to switch between solvers transparently, it can be used to select automatically a particular solver given some properties of the linear system being solved such as the size of the matrix or its structural symmetry. However, as we shall see shortly with the results, this decision making is not easy.

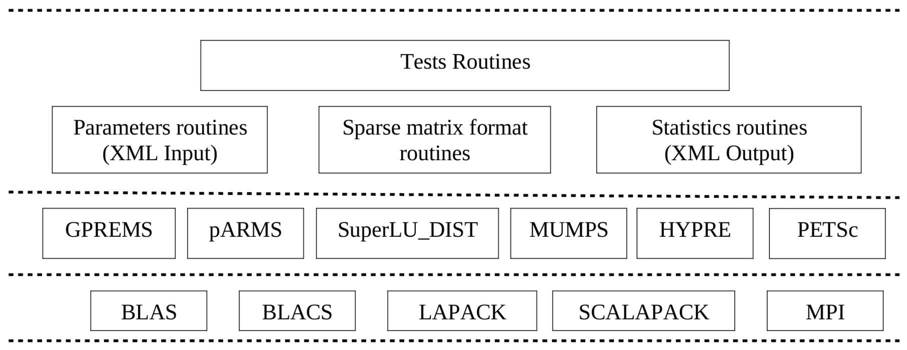
Tests Routines
Parameters routines (XML Input)
Sparse matrix format routines
Statistics routines (XML Output)
GPREMS
pARMS
SuperLU_DIST
MUMPS
HYPRE
PETSc
BLAS
BLACS
LAPACK
SCALAPACK
MPI

Figure 1. Architecture of our toolbox

[2]This library will be soon available for public use

4. Experimental Comparisons

Tests are carried out using the Grid'5000 experimental testbed, on *paradent* cluster in the Rennes site. Each compute node is a dual-cpu and each cpu is a quadricore Carri System CS-5393B (Intel Xeon L5420 at 2.5GHz) with a 32 GB shared memory. In the following, only one cpu is working in each node as no shared-memory programming paradigm was used. All nodes are connected through a Gigabyte Ethernet switch.

4.1. Test Matrices

All the matrices presented here are freely available upon request at [15]. In table 1, we list the characteristics of some of them. Integers n and nnz are respectively the size and

Table 1. Matrices of test

Idx	Matrix	n	nnz	origin
1	CASE_05	161,070	5,066,996	2D linear cascade turbine
2	CASE_07	233,786	11,762,405	2D linear cascade compressor
3	CASE_10	261,465	26,872,530	3D hydraulic gate case
4	CASE_17	381,689	37,464,962	3D jet engine compressor

the number of the nonzeros of the matrix.

4.2. Numerical Behavior, Parallel Efficiency and Fill-in with Direct Solvers

We consider the minimum degree (MD) and the nested dissection (ND) ordering. As the two direct packages (MUMPS and SuperLU_DIST) accept any pivotal sequence, any ordering method can be used. So we have used METIS as nested dissection ordering in both solvers. With approximate minimum degree (AMD) in SuperLU_DIST, we have observed that the fill-in produced was less than that in multiple minimum degree (MMD); however the factorization time is larger. So we have preferred to use the default ordering provided, *i.e.* MMD in SuperLU and AMD in Mumps.

First, the accuracy in the computed solution is considered. In table 2, we give the relative residual norm in the solution, *i.e.* $||b - Ax||/||b||$. Tests are done on 4 processors but the results are roughly the same on 8 or 16 processors. We have observed that in some cases, depending on the use of HSL-MC64 routine to permute large elements on diagonal, the computed solution could be wrong. With CASE_05 for instance, when MC64 is used after a nested dissection ordering, both methods do not achieve a good accuracy. With CASE_07, the situation is more complicated. Either with minimum degree or nested dissection ordering, the solution produced with SuperLU_DIST is not accurate. On the other side, without MC64 permutation, all systems are solved somehow accurately with both methods.

After the accuracy, we look at the increasing of memory needed during the factorization. So in table 3, the ratio of fill-in in factored matrices is given with respect to the nonzeros in the initial matrix i.e $fill = nnz(L + U - I)/nnz(A)$. The fill-in is larger when the permutation is performed to obtain large diagonal elements, particularly with the nested dissection ordering. As a result, it takes much more time to factorize the matrix, particularly for the largest case. In table 4, this preordering effect is shown for the

Table 2. Numerical behavior : Relative residual norm (4 processors)

Matrix	Ordering	SuperLU_DIST		MUMPS	
		No MC64	MC64	No MC64	MC64
CASE_05	MD	3.6e-14	3.5e-14	1e-13	9.4e-14
	ND	3.6e-14	1.3e-02	8e-14	29.9e-01
CASE_07	MD	1.8e-16	9.9e-01	4.7e-16	1.2e-13
	ND	3.6e-14	7.05e-01	3.6e-16	5.1e-10
CASE_10	MD	8.1e-13	8.8e-13	1.2e-12	1.5e-12
	ND	6.3e-16	8.3e-13	1.2e-12	1.4e-12
CASE_17	MD	7.3e-14	9e-12	6.3e-13	1.3e-10
	ND	7.5e-14	4.4e-13	8.2e-13	2.3e-10

overall CPU time on 16 processors. Observe that it takes twice CPU time with MC64 preordering in both methods. Surprisingly with METIS, the time in SuperLU_DIST is ten times larger when this preordering step is performed despite the fact that the fill-in is not so large as shown in table 3. For the MUMPS solver, these results confirm the advices that the maximum transversal should not be applied on matrices with nearly symmetry structure[2]

Table 3. Ratio of fill-in $(fill = nnz(L+U-I)/nnz(A))$: 4 processors

Matrix	Ordering	SuperLU_DIST		Mumps	
		No MC64	MC64	No MC64	MC64
CASE_05	MD	13	17	12	20
	ND	10	64	10	15
CASE_07	MD	30	32	30	34
	ND	22	126	21	27
CASE_10	MD	21.8	23.8	20.9	25.1
	ND	17.6	95.4	17.4	21.8
CASE_17	MD	115	138	100	119
	ND	61	74	58	77

Table 4. CASE_17: Effect of preprocessing on the CPU time (16 processors)

	SuperLU_DIST		MUMPS	
Ordering	No MC64	MC64	No MC64	MC64
MMD/AMD	3050	4045	4228	8229
METIS	1098	17342	1960	3605

The last aspects of interest are the overall time and the parallel efficiency. In table 5, we consider these aspects on the matrix CASE_17. T is the time in seconds while *Acc* and *Eff* are respectively the acceleration and the efficiency with respect to the time on 4 nodes. In this part, all tests are done without the permutation by MC64 as it leads in some cases to huge fill-in and consequently, large CPU factorization time. Note that MUMPS is slightly better than SuperLU_DIST on 4 processors. However, SuperLU_DIST performs better when we increase the number of processors. Moreover, it scales better than MUMPS. This result may come from the relatively slow network interconnecting the nodes [16]. Also in SuperLU_DIST, the amount of communication is reduced during the numerical factorization by using the static pivoting.

Table 5. Parallel efficiency with CASE_17

Ordering	Solver	P=4	P=8			P=16		
		T	T	Acc	Eff	T	Acc	Eff
METIS	SuperLU_DIST	3923	2073	1.89	0.94	1098	3.57	0.89
	MUMPS	3598	2969	1.21	0.6	1960	1.83	0.45

4.3. Parallel Behavior of Preconditioners

In the following, we strike to see the convergence of GMRES with the preconditioners mentioned in section 2; namely the parallel ILU preconditioner (EUCLID) in HYPRE, the restricted additive schwarz(ASM) available in PETSc, the Explicit Form of Multiplicative Schwarz(EFMS) used in GPREMS and the left Schur complement (SC) associated with the flexible GMRES in pARMS. To solve the local systems, we have used MUMPS in the case of ASM and EFMS while ILUK is used as approximate solver in the case of SC. The maximum number of iterations allowed is 4000 and the relative tolerance for the convergence is 10^{-9}. The size of the Krylov basis is 64 for the CASE_05 and CASE_07 cases and 128 for the largest ones.

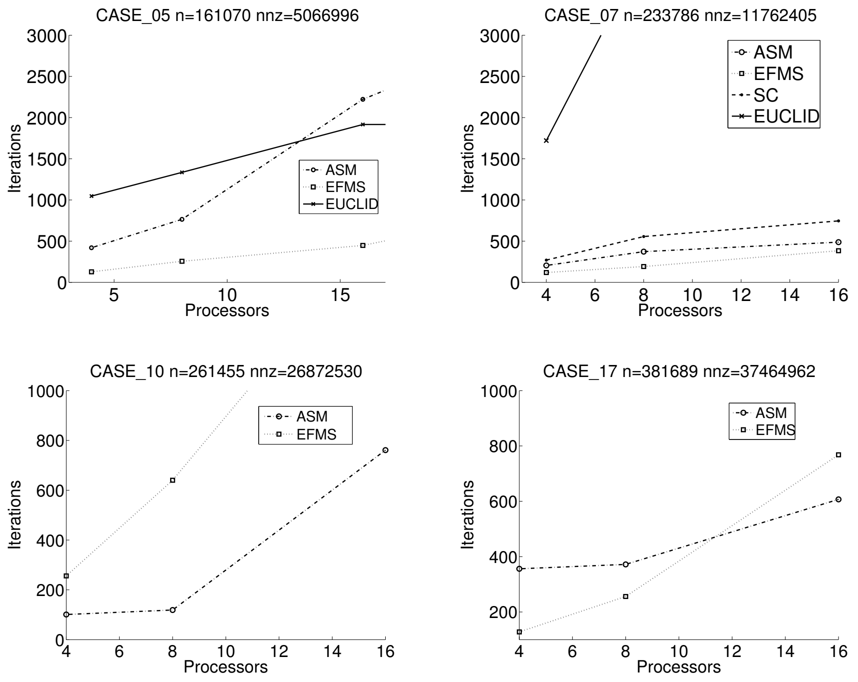

Figure 2. Number of iterations of GMRES

In figure 2 the number of iterations is given as a function of the computing nodes. On small systems (CASE_05, CASE_07) with all preconditioners, this number of iterations grows very fast with the number of processors. In many cases, the maximum number of iterations is reached before convergence. See for instance the CASE_07 with EUCLID on 8 nodes or more. For the largest cases, CASE_10 and CASE_17, GMRES with SC or EUCLID does not converge, whatever the number of nodes used. Thus, only ASM and EFMS are taken into account. On a small number of processors, with all cases but the CASE_10, EFMS gives less number of iterations than ASM. With the CASE_10, ASM performs better than EFMS. However, for more than 8 processors, the number of iterations increases very fast.

Figure 3 gives the time needed to converge to the right solution with respect to the number of processors. For the smallest case and the largest case, we compare direct solvers to preconditioned GMRES. METIS ordering is used in the two direct solvers without MC64 ordering. For the CASE_05, SuperLU_DIST and MUMPS are clearly faster than preconditioned GMRES. Also, the CPU time with ASM and EFMS tends to increase in CASE_05 and CASE_10. For the largest case, GMRES with ASM or EFMS performs better than direct solvers. However, ASM is better than EFMS with more than 4 processors. The main reason is that in EFMS, the residual vector is corrected in a pipeline through the subdomains whereas this correction is done almost simultaneously in ASM. On the other side, GPREMS do perform well regarding the number of iterations as shown in fig. 2.

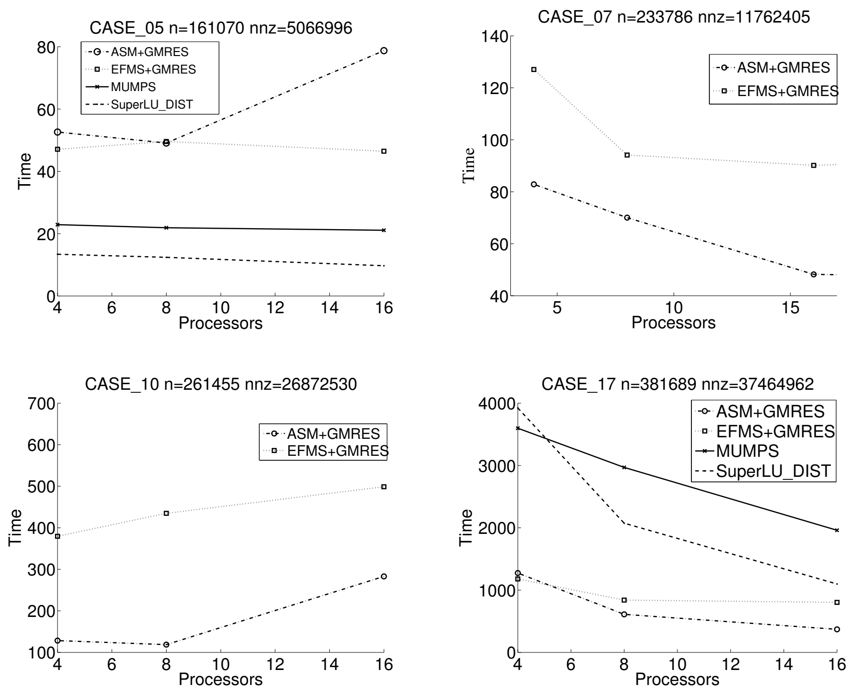

Figure 3. CPU time

5. Concluding Remarks

In this paper, we are interested in the numerical solution of some sparse linear systems issued from actual industrial CFD cases. The distributed solvers we have used are based either on direct, iterative or hybrid techniques. Usually direct solvers are robust, but we have observed here that they could fail to solve some of these systems with some nonobvious parameters. However, on small cases, they are markedly more efficient than other methods used in this study. Also, we have tested the ILU-EUCLID and the left Schur Complement preconditioner in pARMS library but on our set of linear systems, Schwarz-based preconditioners should be preferred. So, in this last category, the restricted additive Schwarz preconditioner performs well when it is associated with a direct solver on subdomains. However, we still need to take a very large Krylov basis which could be a bottleneck in the case of larger systems. Finally, one motivation in this work was to show the significant performance achieved by the parallel GMRES when it is preconditioned by one iteration of the multiplicative schwarz method. The results prove that this preconditioner is competitive among other domain decomposition methods. However, it still suffers from poor scalability. So we are investigating ways to improve this aspect by using some multilevel techniques.

Acknowledgment This work was supported by ANR-RNTL under the LIBRAERO contract. Experiments were carried out using the Grid'5000 experimental testbed (https://www.grid5000.fr).

References

[1] G.-A. Atenekeng Kahou, E. Kamgnia, B. Philippe. An explicit formulation of the multiplicative Schwarz preconditioner. *Applied Numerical Mathematics*, 57:1197-1213, 2007

[2] P. Amestoy, I. S. Duff, J-Y, L'Exclellent, X. S. Li, Analysis and Comparison of Two General Sparse Solvers for Distributed Memory Computers, *ACM Trans. Mathematical Software*, 27(4):388-421, 2001

[3] A. Gupta, Recent Advances in Direct Methods for Solving Unsymmetric Sparse Systems of Linear Equations. *ACM Trans. Mathematical Software*, 28(3):301-324, 2002

[4] G.-A. Atenekeng-Kahou. Parallélisation de GMRES préconditionné par une itération de Schwarz multiplicatif, *PhD thesis, University of Rennes 1 and University of Yaounde I*, December 2008.

[5] M.A. Heroux, R.A Bartlett, V.E. Howle, R.J. Hoekstra, J.J. Hu, T.G. Kolda, R.B. Lehoucq, K.R. Long, R.P. Pawlowski, E.T. Phipps, A.G. Salinger, H.K. Thornquist, R.S. Tuminaro, J.M. Willenbring, A. Williams, K.S. Stanley, An overview of the Trilinos project, *ACM Trans. Mathematical Software*, 31(3), 397-423, 2005

[6] B. Secher, M. Belliard, C. Calvin Numerical Platon: A unified linear equation solver interface by CEA for solving open foe scientific applications. *Nucl. Eng. Des. (2008)*, doi:10.1016/j.nucengdes.2008.06.025

[7] S. Balay, K. Buschelman, W.D. Gropp, D. Kaushik, M.G. Knepley, L.C. McInnes, B.F. Smith, H. Zhang, PETSc Web page. *http://www.mcs.anl.gov/petsc*, 2008

[8] Z. Li, Y. Saad, M. Sosonkina. $pARMS$: A parallel version of the algebraic recursive multilevel solver. *Numer. Linear Algebra Appl.*, 10:485-509, 2003

[9] X. S. Li and J. W. Demmel. SuperLU_DIST : A Scalable Distributed-Memory Sparse Direct Solver for Unsymmetric Linear Systems *ACM Trans. Mathematical Software*, 29(2):110-140, 2003

[10] P. Amestoy, I. Duff, J.-Y. L'Excellent, J. Koster, A fully asynchronous multifrontal solver using distributed dynamic scheduling, *SIAM J. on Matrix Analysis and Applications,* 23(1):15-41, 2001

[11] D. Hysom, A. Pothen. A scalable parallel algorithm for incomplete factor preconditioning. *SIAM J. on Scientific Computing*, 27:1689-1708, 2006

[12] R. Falgout, U. Yang, HYPRE: a Library of High Performance Preconditioners. *C.J.K.Tan, J.J. Dongarra, A.G. Hoekstra (Eds.), Lectures Notes in Computer Science,* 2331:632-641, Springer-Verlag, 2002

[13] J. Erhel, A parallel GMRES version for general sparse matrices. *Electronic Transaction on Numerical Analysis*, 3:160-176, 1995.

[14] R.B. Sidje, Alternatives for parallel subspace basis computation. *Numerical Linear Algebra with Applications*, 4:305-331, 1997.

[15] FLUOREM, The Fluorem Matrix Collection, *http://www.fluorem.com* LIB0721 2.0 / FP-SA, 2009

[16] É. Canot, C. de Dieuleveult, J. Erhel, A parallel software for a saltwater intrusion problem. *G. Joubert, W. Nagel, F. Peters, O. Plata, P. Tirado, E. Zapata (Eds.), Parallel Computing: Current and Future Issues of High-End Computing,* 33:399-406, NIC, 2006.