A parallel multigrid solver on a structured triangulation of a hexagonal domain

Kab Seok Kang[1]

1 Introduction

Fast elliptic solvers are a key ingredient of massively parallel Particle-in-Cell (PIC) and Vlasov simulation codes for fusion plasmas. This applies for both, the gyrokinetic and fully kinetic models. The currently available most efficient solver for large elliptic problems is the multigrid method, especially the geometric multigrid method which requires detailed information of the geometry for its discretization.

In this paper, we consider a structured triangulation of a hexagonal domain for an elliptic partial differential equation and its parallel solver. The matrix-vector multiplication is the key component of iterative methods such as CGM, GMRES, and the multigrid method. Many researchers have developed parallel solvers for partial differential equations on unstructured triangular meshes. In this paper, we consider a new approach to handle a structured grid of a regular hexagonal domain with regular triangle elements. We classify nodes as either real or ghost ones and find that the required steps of data communication to assign the values on the ghost nodes is five. We show that the matrix-vector multiplication of this approach has an almost perfect scaling property.

The multigrid method is a well-known, fast and efficient algorithm to solve many classes of problems [1, 4, 5]. In general, the ratio of the communication costs to computation costs increases when the grid level is decreased, i.e., the communication costs are high on the coarser levels in comparison to the computation costs. Since, the multiplicative multigrid algorithm is applied on each level, the bottleneck of the parallel multigrid lies on the coarser levels, including the exact solver at the coarsest level. The additive multigrid method could combine all the data communication for the different levels in one single step. However, this version can be used only for preconditioner and need almost the double amount of iterations generally. The multiplicative version can be used as a solver and as a preconditioner, so we consider the multiplicative version only.

The feasible coarsest level of operation of the parallel multigrid method depends on the number of cores. The number of degrees of freedom (DoF) of the coarsest level problem will be increased as the number of cores is increased. To improve the performance of the parallel multigrid method, we consider reducing the number of executing cores to one (the simplest case) after gathering data from all cores on a certain level. This algorithm avoids the coarsest level limitation and numerical experiments on large numbers of cores show very good performance improvement.

Max-Planck-Institut für Plasmaphysik, EURATOM Associate, Boltzmannstraße 2, D-85748 Garching, Germany kskang@ipp.mpg.de

A different way to improve the performance of the parallel multigrid method is to use a scalable solver on the coarsest level. A good candidate for the coarsest level solver is the two-level domain decomposition method because these methods are intrinsically parallel and their required number of iterations does not depend on the number of sub-domains (cores). We consider BDDC [2] and FETI-DP [3] because these are well-known two-level non-overlapping domain decomposition methods and show very good performance for many problems.

In this paper we investigate the scaling properties of the multigrid method with gathering data, BDDC, and FETI-DP on a massively parallel computer.

2 Model problem and its parallelization

We consider the Poisson type second order elliptic partial differential equations on a regular hexagonal domain Ω with Dirichlet boundary conditions

$$\begin{aligned} c(x,y)u - \nabla \cdot a(x,y)\nabla u &= f, \quad \text{in } \Omega, \\ u &= 0, \quad \text{on } \partial\Omega, \end{aligned} \tag{1}$$

where $f \in L^2(\Omega)$, $c(x,y)$ is a non-negative function and $a(x,y)$ is a uniformly positive and bounded function. It is well known that the Eq. (1) has a unique solution.

The second-order elliptic problem (1) is equivalent to: find $u \in H_0^1(\Omega)$ such that

$$a_E(u,v) = \int_\Omega c(x,y)uv\,\mathrm{dx} + \int_\Omega a(x,y)\nabla u \cdot \nabla v\,\mathrm{dx} = \int_\Omega fv\,\mathrm{dx} \tag{2}$$

for any test function $v \in H_0^1(\Omega)$ where $H_0^1(\Omega)$ is the space of the first differentiable functions in Ω with zero values on the boundary $\partial\Omega$.

We consider a piecewise linear finite element space defined on a triangulation with regular triangles. This triangulation generate a structured grid and can be applied to a D-shape Tokamak interior region with conformal mapping. Let h_1 and $\mathscr{T}_{h_1} \equiv \mathscr{T}_1$ be given, where $\mathscr{T}_1$ is a partition of Ω into triangles and h_1 is the maximum diameter of the elements of $\mathscr{T}_1$. For each integer $1 < k \leq J$, let $h_k = 2^{-(k-1)}h_1$ and the sequence of triangulations $\mathscr{T}_{h_k} \equiv \mathscr{T}_k$ be constructed by the nested-mesh subdivision method, i.e., let $\mathscr{T}_k$ be constructed by connecting the midpoints of the edges of the triangles in $\mathscr{T}_{k-1}$, and let $\mathscr{T}_{h_J} \equiv \mathscr{T}_J$ be the finest grid.

Let us define the piecewise linear finite element spaces

$$V_k = \{v \in C^0(\Omega) : v|_K \text{ is linear for all } K \in \mathscr{T}_k\}.$$

Then, the finite element discretization problem can be written as follows: find $u_J \in V_J$ such that

$$a_E(u_J, v) = \int_\Omega fv\,\mathrm{dx} \tag{3}$$

for any test function $v \in V_J$, i.e., solve the linear system $A_J u_J = f_J$.

Let us now consider the parallelization of the above problem. We use real and ghost nodes on each core. The values on the real nodes are handled and updated locally. The ghost nodes are the part of the distributed sub-domains located on other cores whose values are needed for the local calculations. Hence, the values of the ghost nodes are first updated by the cores to which they belong to as real nodes and

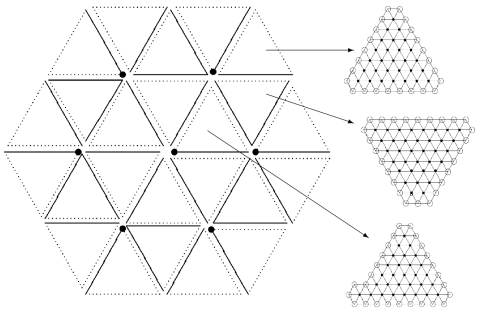

Fig. 1 The subdomains on 24 cores and real (–, •) and ghost (···, ◯) nodes on subdomains according to the types

then transferred to the cores that need them. To reduce data communication during matrix element computation, the computation of matrix elements on some cells can be executed on several cores which have a node of the cell as a real node.

We consider the way to divide the hexagonal domain into sub-domains with the same number of cores. Except for the single core case, we divide the hexagonal domain in regular triangular sub-domains and each core handles one sub-domain. Hence, feasible numbers of cores are limited to the numbers 6×4^n for $n = 0, 1, 2, \ldots$. For each core we have to define what are real and ghost nodes on the common boundary regions of the sub-domains. We determine the nodes on the common boundary of the sub-domains as the real nodes of the sub-domain which are located in the counterclockwise direction or in the outer direction from the center of the domain as shown in Fig. 1. For our problem with a Dirichlet boundary condition on the outer boundary, we can handle the boundary nodes as ghost ones. The values of these boundary nodes are determined by the boundary condition and thus do not have to be transferred bewteen cores.

We number the sub-domains beginning at the center and going outwards following the counterclockwise direction. Each sub-domain can be further divided into triangles; this process is called triangulation. In this process each line segment of the sub-domain is divided into 2^n parts. It can be shown that, independently of the total number of sub-domains and triangulation chosen, there are just three domain types. These give detailed information on the real and ghost nodes being connected to other sub-domains and cells which are needed to compute the matrix elements for the real nodes. To see how good the load balancing is, we measure the ratio of the largest number of real nodes to the smallest number of real nodes which is $\{2^n(2^n+3)\}/\{2^n(2^n+1)\}$ which tends to '1' as n is increased.

To get the values on the ghost nodes from the other cores for all sub-domains, we implement certain communication steps. The communication steps are the dominating part of the parallelization process and thus a key issue for the performance of the parallel code. The easiest way to implement the data communication would be that every ghost node value is received from the core which handles it as a real node value. However, such implementation would need several steps and the required number would then vary among the different cores. So this approach could be used for unstructured grids, but it would be too slow in our case. However, we solved the

problem by using a sophisticated data communication routine which needs a fixed number of steps for each core (that is, five).

Our dedicated data communication steps are as follows:

S1: Radial direction
S2: Counterclockwise rotational direction
S3: Clockwise rotational direction
S4: Radial direction (same as in S1)
S5: Mixed communications

3 Multigrid and domain decomposition methods

The motivation for the multigrid method is the fact that basic iterative methods, such as Jacobi and Gauss-Seidel methods, reduce well the high-frequency error but have difficulties to reduce the low-frequency error, which can be well approximated after projection on the coarser level problem. The multigrid method consists of two main steps, one is the smoothing operator and the other is the intergrid transfer operator. The former has to be easy to be implemented and be able to reduce effectively the high frequency error.

The other important operator is the intergrid transfer operator, which consists of the prolongation and the restriction operator. The intergrid transfer operators on triangular meshes have been studied in depth by many researchers, and their usage is mature.

The main issue with the parallelization of the multigrid method is execution time on the coarser level iterations. In general, the ratio of communication to computation on a coarse level grid is larger than on a fine level grid. Because the multigrid method works on both the coarse and fine grid levels, to get good scaling performance, we might need to avoid operating on the coarser level if possible. Usually, the W-cycle and the variable V-cycle multigrid methods require more work on the coarse level problems, so we consider for parallelization only the V-cycle multigrid method.

In addition to the execution time on the coarser level, we have to consider the solving time on the coarsest level. As a coarsest level solver, we can use either a Krylov subspace method or a direct method. The solving time of both methods increases with the problem size. So in considering the solution time of the coarsest level we need to find the optimal coarsening level, as well as the ratio of the communication to computation on each level.

From a certain level on, we can use a small number of cores to perform computations for the coarser levels. Among all the possible algorithms, let us consider the one which executes only on one core after having gathered all data.

Such a multigrid algorithm variation can solve the coarsest level problem on one core only, independent of the total number of cores. Instead of having only one core solving the coarser level problems and other cores idling, we choose to replicate the same computation on the coarser levels on each core; then we use these results for computations on the finer level. In the variant which we use, we use `MPI_Allreduce` which may yield a better performance than using combinations

of `MPI_Reduce` and `MPI_Bcast`, depending on the `MPI` implementation on the given machine.

Let us now consider another well known parallel solver, namely the domain decomposition method (DDM). The non-overlapping DDM is a natural method for problems which have discontinuous coefficients or many parts and are akin to being implemented on distributed memory computers. The non-overlapping DDM can be characterized by how it handles the values on the inner-boundary (that is, the common boundary of the two sub-domains). The condition number of the two-level non-overlapping DDM does not depend on the number of sub-domains. The BDDC and FETI-DP methods are well developed two-level DDM and have good performance when using a large number of sub-domains.

The BDDC algorithm [2] has been developed as an algorithm for substructuring, based on the constrained energy minimization concept. We follow the algorithm of [2] with a constraint matrix C_u which enforces equality of substructure DoF averaged across edges and at individual DoF on substructure boundaries (corner).

The FETI-DP method [3] imposes the continuity on the corner nodes which includes more than two sub-domains and the continuity on the edge nodes by using the Lagrange multipliers λ. By block Gauss elimination, we obtain the reduced system $F\lambda = d$ and solve it with PCGM with the Dirichlet preconditioner, as in [3].

4 Numerical experiments

As a model problem, we choose the simplest one with $c(x,y) = 0$ and $a(x,y) = 1.0$ in Eq. (1), i.e., the Poisson problem. To test the performance of our implementation, we use the finite element discretization formula which is the same for the finite volume discretization for this test problem. As a termination criterion for the solvers, we define a reduction of the initial residual error on the finest level by a factor of 10^{-8}.

The performance results reported in this paper were obtained on the HELIOS machine. The HELIOS machine is located in the International Fusion Energy Research Centre (IFERC) at Aomori, Japan. IFERC was built in the framework for the EU(F4E)-Japan broader approach collaboration. The machine is made by 4410 Bullx B510 Blades nodes of two 8-core Intel Sandy-Bridge EP 2.7 GHz processors with 64 GB memory and connected by Infiniband QDR. So it has a total of 70 560 cores total and 1.23 Petaflops Linpack performance.

We consider the multigrid method as a preconditioner of the preconditioned CGM with a localized Gauss-Seidel smoother which use old values on the ghost nodes. For the multigrid method, we use PCGM with the symmetric Gauss-Seidel method as a solver on the coarsest level and run two pre- and post-smoothing iterations for all cases.

We tested different data gathering levels on fixed numbers of cores. Without gathering data, the feasible coarsest level of the multigrid algorithm is the level that has at least more than one DoF per core. This level is the coarsest gathering level and depends on the number of cores. For instance, the coarsest gathering level of 384 cores is 5, of 1563 cores is 6, and 6144 cores is 7. Experimentally, setting the gathering level to the coarsest one shown always best performance. After gathering the data,

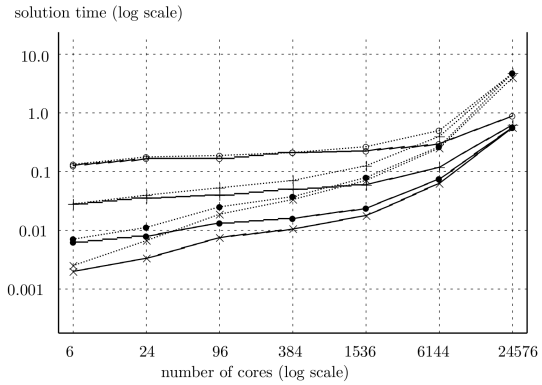

Fig. 2 The solution times in seconds of the multigrid method as a preconditioner for the PCGM with the Gauss-Seidel smoother with (–) and without (...) gathering data as a function of the number of cores for domains with 2.2K DoF (×), 8.5K DoF(•), 33.4K DoF(+), and 132K DoF(○) per core

all the computations are performed on one core. In this coarsest gathering level, the coarsest level does not impact performance as along as it is taken below level 6

In this paper, we use the simplest case only from the gathering level, all the data of the coarse problem are gathered on one core. In the case of large coarse problem, i.e., level greater than 6, a performance improvement could be expected by distributing it on many cores instead of one. But it has not been tested.

Let us now consider the performance impact when gathering the data on each core. To show that, we choose the coarsest level of the parallel algorithm as the coarsest gathering level. In the case of not gathering data, we have to use the coarsest gathering level as the coarsest level on which we solve the problem by using PCGM exactly. We tested four different cases, 2.2K, 8.5K, 33K, and 132K DoF per core and depicted the results in Fig. 2 which show that the gathering of the data is needed for large number of cores. The solution time of the solver in this case has a significant improvement for large numbers of cores and small number of DoF per core.

For a multigrid algorithm it is nearly impossible to fix the number of operations per core while increasing the total problem size, so we consider a semi-weak scaling by fixing the number of DoF of the finest level on each core. We tested six different number of DoF of the finest level on each core; from 2.2K DoF to 2.1M DoF and depicted the results in Table 1 together with the execution time (in bracket) of the matrix-vector multiplication which is the basic operation for iterative solvers and include the data communication step to update the values on the ghost nodes. The data shows that the matrix-vector multiplication has a perfect weak scaling property and the multigrid method as a preconditioner has really good semi-weak scaling properties when the number of DoF per core is large (compare 527K DoF and 2.1M DoF per core cases). Typically, the behaviour of multigrid algorithm implementations in weak scaling experiments is that they perform better as the number of DoF per core is increased.

The required number of iterations of the FETI-DP and BDDC methods does not depend on the number of sub-domains, but rather on the ratio of the mesh size of the triangulation (fine level, h) to the size of the sub-domains (coarse level, H). This

# cores	2.2K	8.5K	33.4K	132K	527K	2.1M
24	0.0034(0.000013)	0.0081(0.000055)	0.0356(0.00045)	0.1671(0.0031)	0.7046(0.0129)	2.824(0.052)
96	0.0075(0.000013)	0.0131(0.000056)	0.0406(0.00045)	0.1717(0.0031)	0.7114(0.0129)	2.825(0.051)
384	0.0104(0.000013)	0.0157(0.000056)	0.0502(0.00048)	0.2057(0.0031)	0.8397(0.0129)	3.327(0.052)
1536	0.0175(0.000013)	0.0244(0.000056)	0.0605(0.00051)	0.2209(0.0031)	0.8661(0.0129)	3.366(0.052)
6144	0.0633(0.000013)	0.0756(0.000056)	0.1192(0.00052)	0.3015(0.0031)	0.9476(0.0131)	3.471(0.052)
24576	0.5671(0.000014)	0.5630(0.000060)	0.6302(0.00054)	0.9105(0.0033)	1.6122(0.0141)	6.954(0.056)

Table 1 The solution times in seconds of the multigrid method as a preconditioner for the PCGM with the Gauss-Seidel smoother and the execution times of the matrix-vector multiplication (in bracket) according to the number of cores for domains with the several numbers of DoF per core

is shown in Table 2 where we list the required number of iterations of the FETI-DP and DBBC methods.

h/H	1/8		1/16		1/32		1/64		1/128	
# cores	FETIDP	BDDC	FETIDP	BDDC	FETIDP	BDDC	FETIDP	BDDC	FETIDP	BDDC
24	12	7	14	8	16	9	18	10	20	12
96	15	8	17	9	20	11	23	13	26	14
384	16	8	19	10	22	11	24	13	28	14
1536	16	8	20	10	23	11	26	13	29	14
6144	16	8	19	10	23	11	26	13	30	14
24576	16	8	19	9	23	11	26	13	29	14

Table 2 The required number of iterations of FETI-DP and BDDC according to the number of sub-domains and the ratio of the mesh size of the fine level (h) to the coarse level (H)

To implement the FETI-DP and BDDC methods, we have to solve local problems with Dirichlet and/or Neumann boundary conditions on each sub-domain and one globally defined coarse level problem. Furthermore, we need to communicate data with neighboring sub-domains and data on the coarse level. Solving the local problems and communicating the data with neighboring sub-domains are performed in parallel. So, these local steps do not alter the performance by changing the number of cores. Otherwise, the dimension of the global coarse level problem would grow as the number of cores increases. The dimension of the coarse level problem used for BDDC method is the same as of the coarsest gathering level used for the multigrid method. And the dimension of the coarse level problem used for FETI-DP method is one level below it.

We use the same gathering algorithm as in the multigrid method to solve the global coarser level problem. In both FETI-DP and BDDC, every sub-domain has some contributions to the matrices and vectors on the coarse level and uses the solution of the coarse level problem. So, we gather these contributions on each core using the `MPI_Allreduce` and use the solution after solving the coarse problem without any data communication.

To solve the local and global problems, we used two direct methods, the LAPACK (Intel MKL) library with dense matrix format and the IBM WSMP library with sparse matrix format, and the multigrid method as an iterative method. For large number of cores (more than 1536 cores), the global problems could be solved by either the iterative method or parallelized direct methods only on a small number of cores, due to the memory limitation. The solution time with parallel solver for

the global problems could be reduced as same as progressively reduced cores on the multigrid method.

For comparison to our previous results, we chose the solver which performes best. We tested five different cases with fixed number of DoF per core, from 55 DoF to 2200 DoF, and depicted the results in Table 3 together with the multigrid method. Results in Table 3 show that the FETI-DP is faster than the latter even though the DBBC requires a smaller number of iterations. These results also show that the weak scaling property is improved as the number of DoF per core is increased.

DoF/core	55			170			590			2200		
# cores	MG	FETI-DP	BDDC	MG	FETI-DP	BDDC	MG	FETI-DP	BDDC	MG	FETI-DP	BDDC
24	0.0009	0.0013	0.0014	0.0015	0.0020	0.0027	0.0019	0.0115	0.0216	0.0034	0.1007	0.2328
96	0.0022	0.0024	0.0028	0.0038	0.0034	0.0046	0.0054	0.0165	0.0298	0.0075	0.1287	0.3309
384	0.0043	0.0041	0.0067	0.0065	0.0057	0.0181	0.0080	0.0228	0.0439	0.0104	0.1414	0.3513
1536	0.0126	0.0131	0.0171	0.0152	0.0240	0.0367	0.0146	0.0512	0.0666	0.0175	0.1953	0.4056
6144	0.0582	0.0792	0.2954	0.0550	0.0988	0.3809	0.0584	0.1509	0.4849	0.0632	0.3864	1.2242
24576	0.5550	0.4961	1.8470	0.5762	0.5359	2.3163	0.5505	0.6883	2.3867	0.5671	1.0609	3.7620

Table 3 The solution times in seconds of the FETI-DP, the BDDC, and the multigrid method (MG) as a function of the number of cores for domains with the number of DoF per core

The solution times of the FETI-DP and the multigrid method for the smallest number of DoF per core cases (55 DoF per core) are almost the same. The multigrid method with gathering data is faster than the FETI-DP method. The difference of the solution time between the two methods increases as the number of DoF per core is increased, except for the largest number of cores (24576 cores).

5 Conclusions

We investigated the performance of the multigrid method with gathering data, BDDC, and FETI-DP on a regular hexagonal domain with regular triangulations and concluded that the first is the fastest solver for such a problem.

Acknowledgements This work was carried out using the HELIOS supercomputer system at Computational Simulation Centre of International Fusion Energy Research Centre (IFERC-CSC), Aomori, Japan, under the Broader Approach collaboration between Euratom and Japan, implemented by Fusion for Energy and JAEA. I would like to thank R. Hatzky and other HLST team members, B. Scott, and D. Tskhakaya for helpful discussions.

References

1. Bramble, J.: Multigrid Methods. Pitman, London (1993)
2. Dohrmann, C.R.: A preconditioner for substructuring based on constrained energy minimizations. SIAM J. Sci. Comput. **25**, 246–258 (2003)
3. Farhat C. Lesoinne M., e.a.: FETI-DP: A dual-primal unified FETI method – part I: A faster alternative to the two-level FETI method. Internat. J. Numer. Methods Engrg. **42**, 1523–1544 (2001)
4. Hackbush, W.: Multigrid Methods and Applications. Spinger-Verlag, Berlin (1985)
5. Hülsemann F. Kowarschik M., e.a.: Parallel geometric multigrid. Numerical Solution of Partial Differential Equations on Parallel Computers II, Lecture Notes in Computational Science and Engineering **51**, 165–208 (2006)