Parallel Smoother Based on Block Red-Black Ordering for Multigrid Poisson Solver

Masatoshi Kawai[1], Takeshi Iwashita[2,3], Hiroshi Nakashima[2], and Osni Marques[4]

[1] Graduate School of Informatics, Kyoto University, Japan
[2] Academic Center for Computing and Media Studies, Kyoto University, Japan
[3] JST, CREST, Japan
[4] Lawrence Berkeley National Laboratory, USA

Abstract. This paper describes parallelization techniques for a multigrid solver for finite difference analysis of three-dimensional Poisson equations. We first apply our block red-black ordering for parallelization of a Gauss-Seidel (GS) smoother, whose sequentiality is often problematic in parallelization of multigrid methods. Furthermore, we introduce a new multiplicative Schwarz smoother, in which multiple GS iterations are performed in each of red-black ordered blocks. Numerical tests are conducted on a cluster of multi-processor nodes comprising four quad-core AMD Opteron processors to examine the effectiveness of these parallel smoothers. The multi-process test using 216 processes in flat-MPI model shows that the block red-black GS smoother and its multiplicative Schwarz variant achieve 1.3 and 1.8 times better performance than the conventional red-black GS smoother, respectively.

1 Introduction

Solving Poisson equation problems often plays an important role in computational science simulations. To accelerate these simulations, the development of a fast Poisson solver is demanded. This paper focuses on the finite difference method for three-dimensional Poisson equation problems. In the finite difference analysis, it is important to efficiently solve the derived linear system of equations. In this paper, the multigrid method is used as the solver of the linear system. This method is suitable for large-scale problems, because it achieves a convergence rate independent from the number of degree of freedoms [1]. In this paper, we discuss the parallelization of the multigrid solver.

It is well known that the Gauss-Seidel (GS) smoother shows good convergence for the linear system arising in the discretized Poisson equation, and is superior to other smoothers such as (weighted) Jacobi. However, the GS smoother cannot be parallelized straightforwardly due to its data-dependency. Accordingly, several GS-based smoothers have been proposed for parallelization. The hybrid smoother combining weighted Jacobi and GS smoothers is a well-known easily parallelizable smoother based on the additive Schwarz method. Another popular method is to impose red-black ordering on GS smoothing to have a convergence

M. Daydé, O. Marques, and K. Nakajima (Eds.): VECPAR 2012, LNCS 7851, pp. 292–299, 2013.

rate superior to the hybrid smoother as well as a large degree of parallelism [2]. However, when the red-black GS smoother is implemented with stride memory accesses to the data array elements, the computational time for a smoothing step becomes longer than the sequential GS smoother due to poorer cache utilization.

To remedy this problem, we first introduced the block red-black ordering to parallelize the GS smoother [3]. Next, for further improvement of the solver performance, we have proposed a parallel multiplicative Schwarz smoother based on the block red-black ordering [4]. The smoothing step of the proposed smoother is faster than that of the red-black GS smoother, while it attains a good convergence rate comparable to the sequential GS smoother. In this paper, we mainly verify the effectiveness of the block red-black GS smoother and its variant in numerical tests of multi-process implementations, the solver performance on a multi-threaded environment was reported in [4].

2 Multigrid Solver for Three-Dimensional Poisson Equation

This paper deals with a multigrid solver for the finite difference analysis of a three-dimensional Poisson equation given by

$$-\nabla^2\phi = \rho \text{ on } \Omega, \tag{1}$$

where ρ is the given source, ϕ is the unknown spatial function, and Ω is the analyzed domain. Applying a 7-point finite difference scheme to (1), we obtain a linear system of equations to solve:

$$\boldsymbol{A}^{(0)}\boldsymbol{u}^0 = \boldsymbol{f}^0. \tag{2}$$

In this paper, we solve (2) by means of geometrical multigrid method, in which multiple coarse grids are generated from the original grid. Using these grids, the solution process of the multigrid method is given by Alg. 1. In the algorithm, $\boldsymbol{A}^{(i)}$, $\boldsymbol{I}_i^{i+1}$, $\boldsymbol{I}_{i+1}^i$ and L denote the coefficient matrix on i-th level grid, the restriction operator from i-th level to i+1-th level, the prolongation operator from i+1-th level to i-th level, and the number of grids, respectively. The vectors $\boldsymbol{u}^i$ and $\tilde{\boldsymbol{u}}^i$ are for i-th level unknowns and their approximation, respectively. In the analysis, the i-th level grid is twice as fine as the i+1-th level grid in each direction. We use the full-weighting restriction operator and tri-linear prolongation operator shown in, for example, [2] on the grids.

3 Parallelization of Geometric Multigrid Poisson Solver

3.1 Parallelization of Multigrid Method

The major components of a multigrid solver are a smoother and operators for restriction and prolongation. Since the restriction and prolongation operations are naturally parallelized by an usual domain decomposition, these operations

$$
\begin{aligned}
&\textit{Smoothing on } \boldsymbol{A}^{(0)}\boldsymbol{u}^0 = \boldsymbol{f}^0 \\
&\boldsymbol{f}^1 = \boldsymbol{I}_0^1 \left(\boldsymbol{f}^0 - \boldsymbol{A}^{(0)}\tilde{\boldsymbol{u}}^0 \right) \\
&\qquad \textit{Smoothing on } \boldsymbol{A}^{(1)}\boldsymbol{u}^1 = \boldsymbol{f}^1 \\
&\qquad\qquad \vdots \\
&\qquad\qquad \textit{Solve } \boldsymbol{A}^{(L-1)}\boldsymbol{u}^{L-1} = \boldsymbol{f}^{L-1} \\
&\qquad\qquad \vdots \\
&\qquad \textit{Smoothing on } \boldsymbol{A}^{(1)}\boldsymbol{u}^1 = \boldsymbol{f}^1 \\
&\tilde{\boldsymbol{u}}^0 \leftarrow \tilde{\boldsymbol{u}}^0 + \boldsymbol{I}_1^0\tilde{\boldsymbol{u}}^1 \\
&\textit{Smoothing on } \boldsymbol{A}^{(0)}\boldsymbol{u}^0 = \boldsymbol{f}^0
\end{aligned}
$$

Alg. 1. Procedure of L-level V-cycle multigrid method

on a grid point are mutually independent from others. On the other hand, the parallelization of the smoother is often problematic. For example, the weighted Jacobi smoother can be easily parallelized, but its convergence rate is inferior to the GS smoother. In the following subsection, we describe conventional GS-based parallel smoothers which have been used in many practical applications.

3.2 Conventional Parallel Smoothers

Hybrid Smoother. The hybrid smoother consists of the weighted Jacobi and the GS smoothers. It uses the weighted Jacobi method for boundaries of each domain-decomposed region allocated to a process/thread so that the GS smoother works on the interior region independently from those of other processes or threads. The smoother can be regarded as an additive Schwarz smoother, in which the GS method is used for the subdomain solver. The hybrid smoother is included in the popular multigrid solver library, BoomerAMG [5]. Although the hybrid smoother has the advantage of implementation easiness, it often entails a degradation in convergence. The convergence rate of the hybrid smoother depends on the number of processes/threads, and it is often worsen when that number is increased.

Red-black GS Smoother. In the 7-point finite difference scheme, each grid point has data dependence only on adjacent 6 points. Consequently, when we paint the grid points alternately by red and black, the grid points having an identical color can be updated independently of each other. That is, after the GS smoothing step is performed for red grid points in parallel, the smoothing step for black grid points is also processed in parallel. This parallel smoothing technique is called red-black GS (RB-GS) smoother. The convergence rate of the RB-GS smoother can be different from that of the sequential GS smoother in general.

However, it is known that the RB-GS smoother has good convergence when compared to the sequential one in homogeneous Poisson equation problems. Accordingly, the RB-GS smoother is the most widely-used parallel smoother for the problems.

In finite difference analyses, the RB-GS smoother is usually implemented with stride memory accesses of unknown and right-hand vector elements, because they are mapped from three-dimensional grid points with lexicographical ordering and thus red and black elements are arrayed alternately. This ordering is expedient for easy and efficient implementation of the whole of a simulation problem, and so it is for those of the restriction and prolongation. However, the efficiency of the smoother itself is degraded from the GS smoother because the stride accesses have poorer cache-line utilization causing lower cache-hit ratio and thus performance.

4 Block Red-Black Gauss-Seidel Smoother and Its Variants

In this paper, we aim to present a parallel smoother free from stride accesses for high-performance while keeping a convergence rate comparable to the conventional RB-GS smoother. For this, we first introduce block red-black ordering, which we originally proposed was for parallel ILU preconditioning [3], to parallelize a GS smoother. Next, we present a new multiplicative Schwarz smoother, being an enhanced version of the block red-black GS smoother.

4.1 Block Red-Black Ordering

Block red-black ordering is one of the parallel ordering techniques. In the ordering, the entire grid is first divided into multiple blocks. Next, red-black ordering is applied to the blocks as shown in Fig. 1, where a block of a color never has direct data-dependency on other blocks of the same color. This feature allows us to parallelize the smoothing step of the block red-black GS (BRB-GS) smoother, so that the GS smoothing is applied to all red blocks in parallel and then to all black ones also in parallel.

It is important that arbitrary ordering can be used in a block and thus the lexicographical one is used in the analysis. Consequently, this ordering makes the block-level GS smoother implemented without stride memory accesses. That is, by choosing the block size sufficiently large especially for the axis conforming to the memory address ordering, the accesses of unknown and right-hand vector elements in the GS smoothing in the block are made almost sequential. This means that a cache-line having a series of vector elements is almost fully utilized by a series of smoothing operations resulting in higher cache-hit ratio and thus more performance than the conventional (i.e., element-wise) RB-GS smoother.

As for the convergence rate, it is strongly expected that the BRB-GS's rate is sufficiently high and comparable to the RB-GS's and the sequential GS's. This expectation is based on the fact that the BRB-GS with the block size of one for

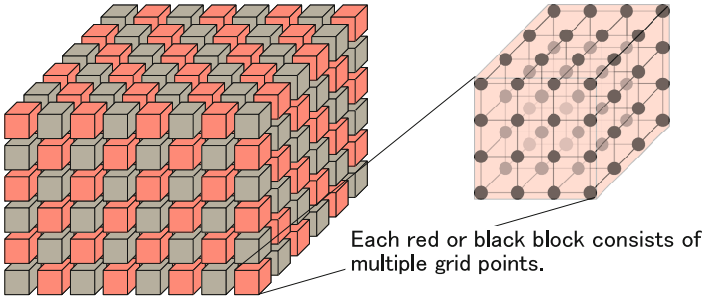

Fig. 1. Concept of block red-black ordering

each axis is just identical to the RB-GS while extremely large block size virtually gives us the sequential GS smoothing. Though the convergence with block sizes between these two extremes above needs to be investigated with real problems, our numerical tests discussed afterward support our expectation.

Furthermore, the cache-blocking technique for smoothing and other components, namely the restriction and the prolongation operations, shown in a context of the sequential multigrid solver in [6], can be easily applied to the BRB-GS. To use the technique in the BRB-GS, we only need to set the block size to match the cache size, and to execute the restriction or the prolongation operation just after/before the smoothing step in each block. Since this technique doubles the utilization of a cache line, it should significantly improve cache-hit ratio and thus performance.

4.2 Modified Block Red-Black Gauss-Seidel Smoother

In this subsection, we introduce a modified version of the BRB-GS smoother to increase the total solver performance. In this version, we simply increase the number of GS iterations in each red/black block from 1 to $\alpha > 1$. The smoother, denoted by mBRB-GS(α) hereafter, is regarded as a multiplicative Schwarz smoother rather than a parallel GS smoother based on parallel ordering. In the following, we discuss the advantage of this multiple iterations of block smoothing.

In general, increasing the number of iterations, namely β, in a smoothing step for the *whole* grid space, leads to improved convergence. However, since grids are usually much larger than the cache size, the computational cost for one smoothing step is also increased in proportion to β. As the smoother is dominant in the multigrid solver in term of computational cost, the total computational time is also proportional to β. Consequently, increasing β rarely reduces the total computational time unless the convergence is improved by a factor of β or more.

On the other hand, increasing the number of GS steps α for a *block* in the BRB-GS is expected to have different behavior. Let t_s be the computational time required for the first smoothing step in a block. When we set the block size less than the cache size, we can expect that the computational time for

the succeeding smoothing step $\tilde{t}_s$ is much less than t_s because of on-cache computation. Consequently, the computational time t_m for one smoothing step of mBRB-GS(α) is given by

$$t_m \approx t_s + (\alpha - 1)\tilde{t}_s < \alpha t_s. \tag{3}$$

From (3), even when the improvement in the convergence does not reach a factor of α, the total computational time can be reduced by increasing the number of GS steps in a block.

5 Numerical Results

5.1 Test Model and Used Parallel Computer

Numerical tests were conducted on the T2K Open Supercomputer at Kyoto University to examine the developed multigrid Poisson solver. The parallel supercomputer consists of SMP nodes, each consisting of four AMD quad-core Opteron 8356 (2.3 GHz) and 32 GB (DDR2-667) shared memory. The internal network between computational nodes, which is based on the DDR-InfiniBand technology, provides full bisection bandwidth and 8 GB/s for each node. The code was written in Fortran90 and MPI. In the present study, we only use the flat-MPI parallel programming model. It is noted that the multi-threaded implementation reported in [4] is based on OpenMP.

In the test model, the analyzed domain Ω is given by $[-0.5, 0.5]^3$ together with Dirichlet boundary condition of $\phi = 0$, and the source term is defined as

$$\rho(r) = \begin{cases} 1 & \text{if } r \leq 0.015 \\ 0 & \text{otherwise} \end{cases} \tag{4}$$

where r is the distance from the origin. To evaluate weak scalability, we fix the finest grid size per process at 128^3.

5.2 Performance Evaluation of the Multigrid Solver

Table 1 lists the computational time and the number of cycles of the multigrid solver with 1, 8, 64 and 216 cores (processes). Because the convergence of the hybrid smoother is 1.7 times slower than the sequential GS smoother, its parallel speedup is limited. On the other hand, the RB-GS and the BRB-GS smoothers attain a convergence rate comparable to that of the sequential GS smoother. However, the computational time for one multigrid cycle of the RB-GS is longer than that of the sequential GS because of the stride memory access. Consequently, only 63.7-fold (weak scaling) speedup is obtained by 216 processes compared with the sequential GS smoother. It is noted that the weak scaling speedup ratio is given by $(T_s \times P)/T$, where P is the number of processes (cores), and T_s and T are the elapsed time in sequential and parallel computations, respectively. Table 1 also indicates that the BRB-GS has advantage in

Table 1. Comparison of the computational time (s) and the number of cycles(in parenthesis) of parallel smoothers

	Number of processes			
	1	8	64	216
Seq.GS	3.55(10)	-		
Hybrid	-	9.99(17)	14.08(17)	15.08(17)
RB-GS	4.84 (9)	8.83(11)	11.39(10)	12.22(10)
BRB-GS	4.33(11)	6.22(11)	8.40(11)	9.43(11)

Table 2. Computational time (s) and the number of cycles(in parenthesis) of the multigrid solver using mBRB-GS on 216 processes

		p_o					
		1	2	3	4	5	6
p_r	1	10.38/11	8.23/9	8.33/9	7.58/8	7.84/8	8.12/8
	2	8.10/ 9	7.42/8	6.69/7	7.21/7	7.11/7	7.58/7
	3	7.54/ 8	6.75/7	6.78/7	7.05/7	7.30/7	7.59/7
	4	7.52/ 8	6.86/7	6.96/7	7.13/7	7.53/7	7.68/7
	5	7.85/ 8	7.16/7	7.45/7	7.80/7	7.92/7	8.24/7
	6	8.31/ 8	7.36/7	7.45/7	7.80/7	7.92/7	8.24/7

the computational time per cycle than the RB-GS, because of the more efficient cache utilization. Therefore the BRB-GS attains better solver performance than the conventional RB-GS smoother. The weak scaling speedup ratio of the BRB-GS reaches 81.3 by 216 processes.

Next, the performance of the mBRB-GS is examined. We carried out a preliminary test on single node using 16 treads for checking the computational time for the first and the second smoothing steps for the block, t_s and $\tilde{t}_s$. On the grid of 512^3, t_s and $\tilde{t}_s$ were measured 0.67 s and 0.11 s, respectively. The preliminary test confirms the inequality(3) because $\tilde{t}_s$ is approximately one sixth of t_s.

Table 2 shows the computational time and the number of iterations of multi-process parallel processing with 216 processes when the mBRB-GS(p_r) and the mBRB-GS(p_o) are used for the pre- and post-smoothing steps, respectively. Table 2 confirms that increasing the smoothing steps in the block leads to the improvement in the solver performance. The best result of the mBRB-GS was obtained when (p_r, p_o)=(2, 3) for 216 processes. Figure 2 shows the weak scaling speedup ratio of the solver with various parallel smoothers. In the test, (p_r, p_o) of the mBRB-GS was set to (2, 3). The numerical result shows the advantage of the BRB-GS and the mBRB-GS over conventional parallel smoothers. In the multi-process parallel processing, increasing the number of GS steps in the block of the mBRB-GS improves the convergence without increasing the number of MPI communications. Consequently, the mBRB-GS achieves 1.80 times better performance than the RB-GS.

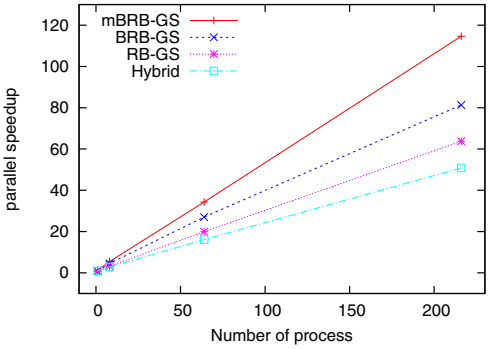

Fig. 2. Weak scaling speedup of parallel multigrid solver with various smoothers compared to a sequential solver with GS smoother

6 Conclusion

In this paper, we investigated the parallelization of a multigrid solver for three-dimensional Poisson equation problems, focusing on the parallel processing of the Gauss-Seidel (GS) smoother. First, we introduced the block red-black ordering technique to parallelize the GS smoother. In this method, the analyzed grid is divided into multiple blocks, to which the red-black ordering is applied. Numerical tests on 216 processes showed that the block red-black GS smoother can be 1.3 times faster than the conventional red-black GS smoother due to more efficient cache utilization. Next, we presented the modified version of the block red-black ordering GS smoother (mBRB-GS). In this version, we iterate GS smoothing in each block twice or more to have a like multiplicative Schwarz smoother. The smoother improves the convergence without largely increasing the computational time of one smoothing step. Consequently, the mBRB-GS can be 1.8 times faster than the RB-GS on 216 processes.

References

1. Trottenberg, U., Oosterlee, C., Achuller, A.: Multigrid. Elsevier Academic Press (2001)
2. Thoman, P.: Multigrid Methods on GPUs. VDM (2008)
3. Iwashita, T., Shimasaki, M.: Block red-black ordering: a new ordering strategy for parallelization of ICCG method. Int. J. Parallel Prog. 31, 55–75 (2003)
4. Kawai, M., Iwashita, T., Nakashima, H.: Parallel Multigrid Poisson Solver Based on Block Red-Black Ordering. In: Proc. Symposium on High Performance Computing and Computational Science, pp. 107–116 (2012) (in Japanese)
5. Henson, V., Yang, U.: BoomerAMG: A parallel algebraic multigrid solver and preconditioner. Applied Numerical Mathematics 41, 155–177 (2002)
6. Kowarschik, M., Rüde, U., Weiß, C., Karl, W.: Cache-aware multigrid methods for solving poisson's equation in two dimensions 64, 381–399 (1999)