INTERNATIONAL JOURNAL OF
NUMERICAL ANALYSIS AND MODELING, *SERIES B*
Volume 5, Number 1-2, Pages 13–20

DEVELOPMENT OF A RESTRICTED ADDITIVE SCHWARZ PRECONDITIONER FOR SPARSE LINEAR SYSTEMS ON NVIDIA GPU

HUI LIU, ZHANGXIN CHEN, SONG YU, BEN HSIEH AND LEI SHAO

Abstract. In this paper, we develop, study and implement a restricted additive Schwarz (RAS) preconditioner for speedup of the solution of sparse linear systems on NVIDIA Tesla GPU. A novel algorithm for constructing this preconditioner is proposed. This algorithm involves two phases. In the first phase, the construction of the RAS preconditioner is transformed to an incomplete-LU problem. In the second phase, a parallel triangular solver is developed and the incomplete-LU problem is solved by this solver. Numerical experiments show that the speedup of this preconditioner is sufficiently high.

Key words. Restricted additive Schwarz preconditioner, linear solver, ILU, parallel triangular solver, GPU

1. Introduction

A restricted additive Schwarz (RAS) preconditioner is a general parallel preconditioner for speedup of the solution of sparse linear systems, which was developed by Cai et al. [4]. This preconditioner is a cheaper variant of the classical additive Schwarz preconditioner, and it is faster in terms of iteration counts and CPU time [4, 5]. Nowadays, RAS is the default parallel preconditioner for the solution of nonsymmetric sparse linear systems in PETSc [1, 4] and has been used in PHG [20]. Our long-term goal is to develop and implement this type of preconditioners for numerical reservoir simulation [7, 8].

GPU, which was used only for graphics processing in its earlier development, is now much more powerful in float point calculation than conventional CPU. It has been used in many scientific applications, such as FFT [16], BLAS [2, 3, 16], Krylov subspace solvers [18, 13, 14, 15] and algebraic multigrid solvers [11]. Algorithms for basic matrix and vector operations are well understood now. However, due to the irregularity of sparse linear systems, the development of efficient parallel preconditioners on GPU is still challenging. In this paper, we introduce, study and implement a RAS preconditioner for speedup of the solution of sparse linear systems on NVIDIA Tesla GPU. For a given matrix A whose size is $n \times n$, a sub-problem is constructed and written as a smaller $i \times i$ matrix, $i \leq n$ [4]. Following this idea, combining all sub-problems together, the final problem becomes a diagonal block matrix problem, $\mathrm{diag}(A_1, A_2, \ldots, A_k)$, which can be solved by incomplete-LU factorization, where k is the number of sub-problems. We have recently developed a parallel triangular solver in [15], where a new matrix format, HEC (hybrid ELL and CSR), and a modified level schedule method on GPU have been introduced. This parallel triangular solver will be used in the current development and study of the RAS preconditioner. Numerical experiments performed show that the speedup of this preconditioner is sufficiently high.

Received by the editors January 5, 2014 and, in revised form, March 19, 2014.
1991 *Mathematics Subject Classification.* 65F08, 65F50, 65Y04, 65Y05, 65Y10.
This research was supported by NSERC/AIEE/Foundation CMG and AITF Chairs.

The layout is as follows: In §2, basic knowledge and our deduction of RAS are introduced. In §3, our parallel triangular solver is described. In §4, numerical experiments are employed to test our GPU version RAS preconditioner. In the end, some conclusions are presented.

2. Restricted Additive Schwarz Preconditioner

We consider a linear system:

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

where $A = \left(A_{ij}\right)$ is an $n \times n$ nonsingular sparse matrix. Denote by L the lower part of A. Then the non-zero pattern we use is $L + L^T$, where L^T is the transpose of L. Also, we define an undirected graph $G = \{W, E\}$, where the set of vertices $W = \{1, \ldots, n\}$ represents the n unknowns and the edge set $E = \{(i, j) : A_{ij} \neq 0,\ A_{ij} \in L + L^T\}$ represents the pairs of vertices [4].

The graph G is partitioned into k non-overlapping subsets by METIS [12], denoted by $W_1^0, W_2^0, \ldots, W_k^0$. We always assume that all subsets of W are sorted in ascending order according to the column indices. For any subset W_i^0, a 1-overlap subset W_i^1 can be obtained by including all the immediate neighboring vertices in W [4]. Repeating this process, a δ-overlap subset W_i^δ can be defined, and the resulting overlapping subsets are $W_1^\delta, W_2^\delta, \ldots, W_k^\delta$.

For any nonempty subset V of W with N ($N > 0$) vertices, we define a mapping $m : V \to W$ by

$$m(p(j)) = j, \tag{2}$$

where $p(j)$ is the position of vertex j in V. Then we introduce matrix B as follows:

$$B_{ij} = A_{m(p(i))m(p(j))}. \tag{3}$$

In this case, B is an $N \times N$ matrix.

Applying this definition, for any W_i^δ with N_i vertices, we introduce the mappings $m_1, m_2, \ldots, m_k$. Using equation (3), we obtain submatrices, $A_1, A_2, \ldots, A_k$. When a RAS preconditioner is applied to the solution of linear systems, these submatrices can be solved simultaneously. We now assemble these submatrices and solve an enlarged system:

$$M = \mathrm{diag}(A_1, A_2, \ldots, A_k), \tag{4}$$

where M is an $(N_1 + N_2 + \ldots + N_k) \times (N_1 + N_2 + \ldots + N_k)$ matrix. This matrix can be solved by ILU(k) or ILUT. The structures of L and U are of the following form:

$$L = \mathrm{diag}(L_1, L_2, \ldots, L_k), \quad U = \mathrm{diag}(U_1, U_2, \ldots, U_k). \tag{5}$$

The final problem is how to solve the lower and upper triangular problems. The whole assembling procedure is described in Algorithm 1.

Algorithm 1 Assembling a RAS preconditioner

1: Constructing the undirected graph G using pattern $L + L^T$;
2: Partitioning G using METIS;
3: Constructing subgraph W_i^δ and the corresponding mapping m_i;
4: Assembling submatrix A_i;
5: Factorizing A_i and obtaining the lower and upper triangular matrices L_i and U_i, respectively.