CATEGORY: ALGORITHMS - AL04
POSTER P6120
CONTACT NAME Erh-Chung Chen: s103062526@m103.nthu.edu.tw

ABCD Algorithm for Tridiagonal Solver

Erh-Chung Chen, Yu-Chuan Chen, Che-Rung Lee
National Tsing Hua University, Taiwan

Introduction

Tridiagonal linear system has a special structured matrix, as shown in Figure 1, which often appears in physics and computer graphics. Its special structure is suitable for ABCD (Augmented Block Cimmino Direct) algorithm to solve in parallel.

$$A = \begin{pmatrix} b_1 & c_1 & & & & \\ a_2 & b_2 & c_2 & & 0 & \\ & a_3 & b_3 & c_3 & & \\ & & \ddots & \ddots & \ddots & \\ & 0 & & \ddots & \ddots & c_{n-1} \\ & & & & a_n & b_n \end{pmatrix}.$$

Figure 1, format of tridiagonal matrix

Methodology

Algorithm - ABCD

ABCD algorithm, which is derived from augmented block cimmino method[1]. The matrix is partitioned with augmentations, where $C_{i,j} = A_{ij}A_{ji}^T$. Then, we can solve each block-rows in parallel. The format is shown in Figure 2.

However, the solution is changed due to the augmentations. To make the result unchanged, extra constraints are added to the system(equation 1).

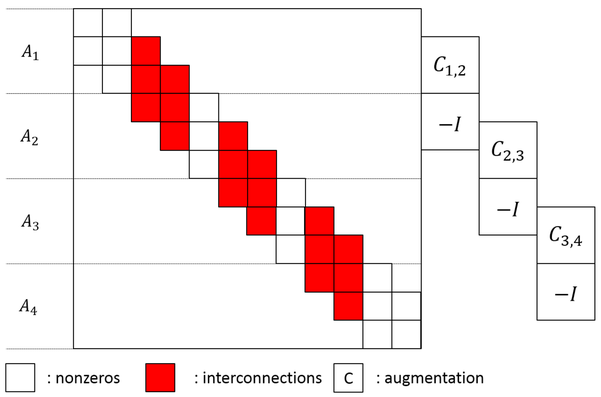

Figure 2, format of tridiagonal matrix and division technique

$$\begin{bmatrix} A & C \\ 0 & I \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ 0 \end{bmatrix}$$

$$Ax + Cy = b, y = 0$$

Then $Ax = b$.

Equation 1, we add extra constraints of $[0 \quad I]$ to make the result correct.

Now, the result is correct, but the extra constraints are not orthogonal to other block-rows. To solve the problem, we apply the projection method, where $P = \sum_{i=1}^{p} \bar{A}_i^T(\bar{A}_i\bar{A}_i^T)^{-1}\bar{A}_i$, to obtain the orthogonal complement of the subspace spanned by the block rows of $[A \quad C]$.

Accordingly, the equation 1 becomes equation 2.

$$\begin{bmatrix} A & C \\ B & S \end{bmatrix}\begin{bmatrix} x \\ y \end{bmatrix} = \begin{bmatrix} b \\ f \end{bmatrix}$$

Equation 2, $[B \quad S]$ is orthogonal to other block-rows. Now, we can solve the system in parallel.

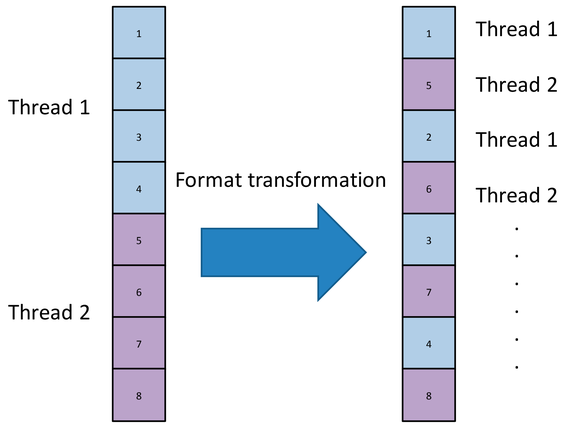

Figure 3, example of format transformation. Suppose there are 2 threads and 8 elements.

GPU implementation

We implemented the following techniques to achieve better performance on GPU:

1. data layout transformation

Data layout transformation helps us achieve the requirement of coalescing on GPU, as shown as in figure 3.

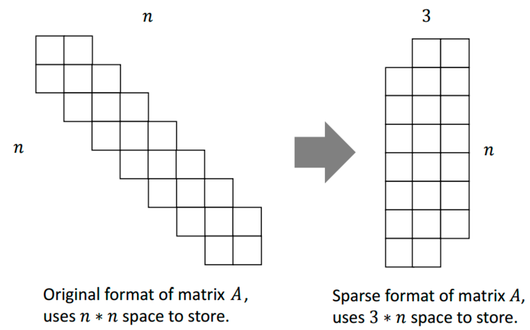

*Figure4 , sparse storage format of A. It is reduced to 3 * n for three diagonal storage.*

2. sparse storage

Because he tridiagonal matrix has nonzeros only at tridiagonal, a special format is preferred as shown as in figure 4.

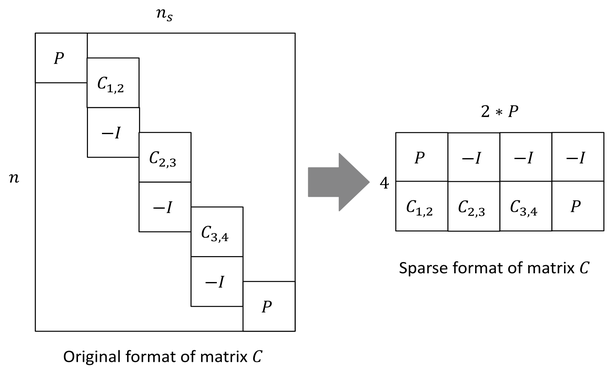

Figure 5, example of sparse storage and boundary padding. Because there are few elements in the augmentations, we change the storage way from dense to sparse. Then, we add padding P to unify works for different threads.

3. Boundary padding to reduce the branches

Because there are differences between different block-rows, we proposes the boundary padding to reduce the branches divergence, as shown in figure 5.

Evaluation

We implement the algorithm on both GPU and CPU to test the applicability. As shown in Figure 6, the performance of GPU version is about 15 times faster than that of the CPU version. The calculation time is similar when the matrix size is too small.

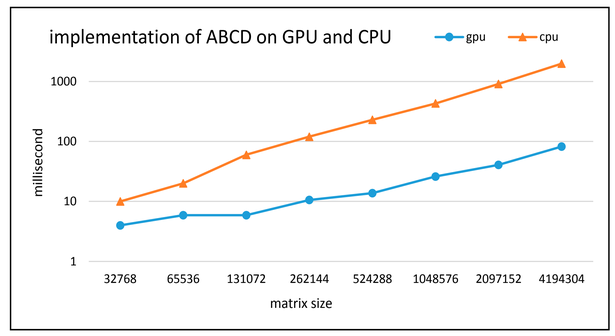

Figure6 , comparison between ABCD implementation on GPU and CPU.

Conclusion

With this work , we provide a fast tridiagonal matrix solver based on GPU and show the applicability. Using ABCD algorithm and SPIKE algorithm, we can solve a tridiagonal matrix in parallel with less calculation time than CPU.

Reference

[1] I. S. Duff, R. Guivarch, D. Ruiz, and M. Zenadi. "The augmented block cimmino distributed method. Technical Report TR/PA/13/11, CERFACS," Toulouse, France, 2013.
[2] E. Polizzi and A. H. Sameh, "A parallel hybrid banded system solver: The SPIKE algorithm," Parallel Computing, vol. 32, no. 2, pp. 177–194, 2006.