Development of GPU Based Parallel Computing Module for Solving Pressure Equation in the CUPID Component Thermo-Fluid Analysis Code

Jin Pyo Lee and Han Gyu Joo
Seoul National University, 599 Gwanak-ro, Gwanak-gu, Seoul, 151-744
jinpyo@snu.ac.kr

1. Introduction

In the thermo-fluid analysis code named CUPID, the linear system of pressure equations must be solved in each iteration step. The time for repeatedly solving the linear system can be quite significant because large sparse matrices of Rank more than 50,000 are involved and the diagonal dominance of the system is hardly hold. Therefore parallelization of the linear system solver is essential to reduce the computing time.

Meanwhile, Graphics Processing Units (GPU) have been developed as highly parallel, multi-core processors for the global demand of high quality 3D graphics. If a suitable interface is provided, parallelization using GPU can be available to engineering computing. NVIDIA provides a Software Development Kit(SDK) named CUDA(Compute Unified Device Architecture)[2, 3] to code developers so that they can manage GPUs for parallelization using the C language. In this research, we implement parallel routines for the linear system solver using CUDA, and examine the performance of the parallelization. In the next section, we will describe the method of CUDA parallelization for the CUPID code, and then the performance of the CUDA parallelization will be discussed.

2. Method

The CUPID code is written in FORTRAN and the linear solver is the BiConjugate Gradient STABilized (BiCGSTAB) method, which is one of the Krylov subspace methods. The BICGSTAB algorithm requires 4 vector inner products, 2 matrix-vector products and 2 solutions of the preconditioner equation at each iteration step[3]. Additionally in order to find the norm of the residual vector, this algorithm needs one more scalar calculation. The point diagonal preconditioner is used in CUPID. The point diagonal preconditioner is to constitute the matrix by taking the diagonal of entries of the linear system. Although this preconditioner may not be as efficient as the other preconditioners such as the Incomplete LU(ILU) conditioner, but it is very simple and easy to implement particularly in parallel computation.

The pressure matrix of the CUPID code is sparse and as no more than 10 entries are given for a single row. Therefore, the matrix information is stored using two 2-dimensional arrays and two 1-dimensional arrays in order to save the computing time and memory. The 2-dimensional arrays contain the value and the location of offdiagonal components, and the 1-dimensional arrays store the diagonal values and the number of neighbors for each row, respectively. However, the 2-dimensional arrays are complicated to handle in the GPU parallelization, so the array structures are converted to 3 equivalent 1-dimensional arrays. These arrays represent the location of value of the offdiagonal elements, and the values of diagonal entries.

The parallelization algorithm of the CUPID code using CUDA contains the following.

[1] Initialization of parameters
[2] GPU memory allocation
[3] Copy information of matrix, preconditioner to GPU
[4] Implement loop of BiCGSTAB
    [4-1] Compute scalar product in parallel
    [4-2] Solve the precondition equation in parallel
    [4-3] Compute matrix-vector product in parallel
[5] Finish loop if the convergence condition is satisfied

3. Performance Examination

The CPU model employed to compare the performance is Intel Core2 Quad Q9400 2.66GHz. In contrast, the GPU model employed in this research is GeForce 9600 GT (containing 8 Multi Processor = 32 Scalar Processor)

3.1. Implementation condition

The GPU's performance is superior to CPU in single precision floating point operation. However, GPU shows poor performance in the double precision calculation compared to the CPU because GPU's core, scalar processor is very lightweight, which implies that the GPU parallelization in double precision operation might be useless. In order to examine the parallelization performance fairly, all the double precision variables in the CUPID linear system solver is converted to single precision one.

The CUPID code solves large linear systems and the matrix dimension can be larger than 10,000. Therefore, the number of block in the CUDA parallelization should be carefully chosen. If the number of blocks is insufficient, the CUDA kernel might be broken down due to the overflow of GPU's cache memories. Also, in order to enhance the performance, the number of thread should be chosen as the dimension greater than the number of blocks. In this case, each thread made such that it computes one row of matrix.

3.2. Results

The test was performed in sequence given in the following tables. The size of matrix used in this test is 11220 by 11220 and 125000 by 125000. And then these matrices are involved in 2 phase rectangular parallelepiped problem[1, 4].

※ In case of Table 1~3, iteration contains only one kernel and number of iterations fixed to 2000.

Table 1. Performance comparison for Parallelized Matrix-Vector Product Kernel

# of Block	32	64	128	256	512
GPU [sec] Dim 11220	0.687	0.683	0.681	0.668	0.671
CPU [sec] Dim 11220	1.29				
GPU[sec] Dim 125000	-			6.00	6.08
CPU[sec] Dim 125000	11.4				

※ It is impossible to implement in this algorithm under 25 blocks, 210 blocks due to overflow of shared memory in case of dimension 11220 and dimension 1250000 respectively.

Table 2. Performance comparison for Parallelized Solution of Preconditioner Equation

# of Block	32	64	128	256	512
GPU [sec] Dim 11220	0.043	0.054	0.038	0.045	0.039
CPU [sec] Dim 11220	0.177				
GPU [sec] Dim 125000	-			1.86	1.93
CPU [sec] Dim 125000	2.24				

※ This kernel consist of the same algorithm with above.

Table 3. Performance comparison for Parallelized Scalar Product

# of Block	4	8	16	32	64
GPU [sec] Dim 11220	0.103	0.059	0.053	0.049	0.050
CPU [sec] Dim 11220	0.116				
GPU [sec] Dim 125000	1.97	1.65	1.03	0.965	0.953
CPU [sec] Dim 125000	1.39				

※ This kernel consist of the tree-like summation algorithm different from above. In the case, the shared memory

Table 4. CUPID code computing time with CPU and GPU

matrix dimension	# of iterations	CPU Intel Core2 Quad [sec]	GeForce 9600 GT [sec]	speed up
(small) 11220 by 11220	25	0.15	0.04	3.75
	50	0.27	0.07	3.85
	75	0.35	0.11	3.18
	100	0.43	0.15	2.87
(large) 125000 by 125000	25	1.07	0.41	2.61
	50	1.95	0.81	2.41
	75	2.92	1.22	2.39
	100	3.88	1.64	2.37

※ CPU: Intel Core2 Quad Q9400 2.66GHz

※ Number of blocks fixed to 128

※CUPID code used in this test contains above three kernels

4. Discussion

Table 1~3 shows that the respective kernel has its own specific number of blocks to give the best performance. Considering these results, 128 blocks were used for the CUPID run shown in Table 4. The results of Table 4 show that the performance of GPU is better than CPU by up to about 4. However, the improvement of performance decreases with the number of iterations and the size of matrix. This tendency might result from transmission data of FORTRAN to C and C to CUDA. Therefore, we will achieve better higher performance if the CUPID code is converted into C language program. Also, inappropriate data structure needs to be restructured.

5. Conclusion

In this research, the linear system module of the CUPID code was successfully parallelized by the CUDA package. Especially the increase of performance is up to about 4. Also, through this research it was found that iterative method is well parallelized using CUDA. Although the performance of GPU is not linearly dependent on the number of GPU cores, It is possible to obtain relatively high performance with much less cost and smaller space by using GPU. However, other preconditioner is not yet parallelized because its algorithm is so complicated. This point can be overcome by sophisticated algorithm. Also, GPU parallelization shows poor performance in double precision. This defect can be solved by improvement of GPU.

REFERENCES

[1] J. J. Jeong *et al*, A semi-implicit numerical scheme for transient two-phase flows on unstructured grids, Nucl. Eng. Deg. **238**, 3403 (2008).

[2] "NVIDIA CUDA™ Programming Guide", ver. 3.0 , pp.1-3, 2009

[3] Jens Breitbart, Research Group Programming Languages/ Methodologies Dept. of Computer science and Electrical Engineering, Universitat Kassel, pp.21-25, August 7, 2008

[4] Joo Han Gyu, Jung Yeon Sang, Lee Min Jae, CUPID 압력방정식 선형계 신속해석모듈 개발, Seoul National University, 2009