Transactions of the Korean Nuclear Society Spring Meeting
Jeju, Korea, May 17-18, 2018

GPU-based Parallel Krylov Linear System Solver for CMFD calculation in nTRACER

Junsu Kang and Han Gyu Joo[*]
Seoul National University, 1 Gwanak-ro Gwanak-gu, Seoul, 08826, Korea
[*]*Corresponding author: joohan@snu.ac.kr*

1. Introduction

Heterogeneous computing systems using Graphic Processing Unit (GPU) are one of the most popular high performance computing platforms. Massively parallel architecture of GPUs allows more power-efficient and higher throughput than CPUs. Therefore, the inclusion of GPU in the neutron transport calculation is increasingly explored.

nTRACER is a direct whole core transport calculation code based on the planar Method Of Characteristics (MOC) solver and the axial Simplified P_3 (SP_3) solver in the framework of 3D Coarse Mesh Finite Difference (CMFD) method [1]. The feasibility of GPU acceleration of MOC calculation in nTRACER had been explored, and the result was successful [2].

GPU acceleration of CMFD calculation in nTRACER is required to keep up with enhanced performance of GPU-based MOC. For the solution of the large sparse linear system involved in the CMFD formulation, the preconditioned Krylov iterative solver, especially preconditioned BiConjugate Gradient Stabilized solver (pBiCGSTAB), has been used as the linear solver of nTRACER. Therefore GPU-based pBiCGSTAB linear system solver is constructed in this work.

The optimal strategies for implementation of GPU-based pBiCGSTAB for CMFD calculation will be discussed. Though nTRACER already has well optimized sequential CPU-based CMFD calculation scheme, conventional optimization strategies could cause overhead cost in parallel heterogeneous system. Therefore different strategies are implemented and evaluated through numerical experiments. The resulted GPU-based solver's performance will be compared with conventional nTRACER solver.

2. CMFD calculation in nTRACER

2.1. CMFD formulation

In CMFD calculation, the sub-pin level fine mesh in MOC calculation is homogenized into pin-cell based coarse mesh. Then coupling coefficients are generated to construct coarse mesh diffusion problem. An eigenvalue problem arises as a result of CMFD formulation and be solved by power iteration. The linear system solution is required for power iteration as:

$$\mathbf{M}\phi^{(n+1)} = \frac{1}{k_{eff}^{(n)}}\boldsymbol{\chi}\mathbf{F}\phi^{(n)}, \tag{1}$$

where $\mathbf{M}$ represents neutron migration by diffusion and scattering, $\mathbf{F}$ represents neutron generation by fission. Due to the large-scale, sparsity and asymmetry of $\mathbf{M}$, the linear system in Eq. (1) is solved by pBiCGSTAB in nTRACER.

2.2.Two-level CMFD

nTRACER employs two-level CMFD formulation. The first level is the multi-group CMFD (MG CMFD) based on MOC transport solution. The second level is the group-condensed CMFD (GC CMFD) which has same coarse mesh geometry as multi-group CMFD with condensed energy groups. The MG CMFD calculations are accelerated by the GC CMFD calculation.

Figure 1 shows the two-level CMFD algorithm. GC CMFD is repeated with frequency number 5 which is determined empirically. In this work, 'outer iteration' represents the power iteration for eigenvalue update in CMFD and 'inner iteration' represents pBiCGSTAB iteration for linear system solving.

```
for i = 1, 2, 3 ...
    1 outer iteration of multi-group CMFD
    if $\|\mathbf{r}_i\| / \|\mathbf{r}_0\| < \varepsilon_{MG}$, then quit
    if (i (mod 5) = 0) then
        for j = 1, 2, 3 ...
            1 outer iteration of group-condensed CMFD
            if $\|\mathbf{r}_j\| / \|\mathbf{r}_0\| < \varepsilon_{GC}$, then quit
```

Fig. 1. Iteration strategy of two-level CMFD calculation.

2.3.Gauss-Seidel group sweep scheme

The migration matrix $\mathbf{M}$ in Eq. (1) can be separated to $\mathbf{D}$ and $\mathbf{S}$, which represent migration by diffusion and scattering respectively, as follows:

$$\mathbf{M} = \mathbf{D} - \mathbf{S}. \tag{2}$$

Usually $\mathbf{S}$ contains most of nonzero entries of $\mathbf{M}$. Therefore $\mathbf{S}$ has been excluded in the inner iteration of nTRACER by putting $\mathbf{S}$ in the RHS of linear system and solving the linear system group by group in Gauss-Seidel manner as follows:

$$\mathbf{D}_g\phi_g^{(n+1)} = \lambda^{(n)}\boldsymbol{\chi}_g\psi^{(n)} + \sum_{g'=1}^{g-1}\mathbf{S}_{g'g}\phi_{g'}^{(n+1)} + \sum_{g'=g+1}^{G}\mathbf{S}_{g'g}\phi_{g'}^{(n)}. \tag{3}$$

If up-scattering doesn't exist in the problem, only one sweep over energy group is enough to get the exact solution. Usually the up-scattering reaction is infrequent. Therefore the solution of whole linear system is determined after only one group sweep over whole energy range and one additional group sweep over energy range where up-scattering exists.

The lack of robust consideration of the relation between multi-group spectra of this scheme can be resolved by group-condensed CMFD in the multi-grid manner. As a result the total floating point operations (flop) required for inner iteration is reduced with similar convergence rate when compared to explicit solution of the linear system in Eq. (1).

3. Implementation Strategy for GPU

3.1. spMv algorithms

BiCGSTAB algorithm is composed by Sparse matrix-vector multiplication (spMv), dot product and vector update. spMv is the most time-consuming operation in the BiCGSTAB algorithm. Since spMv has low arithmetic intensity, memory access pattern determines the performance of spMv. Therefore the performance of spMv algorithm differs by the matrix structure. To find optimal spMv algorithm for CMFD linear system, suitability of 2 compressed sparse row (CSR) sparse matrix format based algorithms and 1 sliced ELLPACK (SELL) based algorithm are compared in 4.2. Detailed implementation in reference papers [3, 4] are omitted in this paper and summarized description for each algorithm is provided below.

3.3.1. CSR-scalar

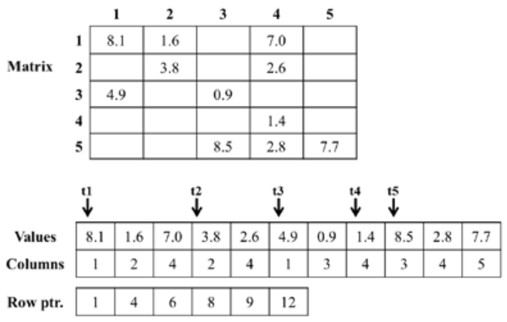

Fig. 2. Example of CSR sparse format and memory access pattern of CSR-scalar.

CSR is one of the most widely used sparse matrix format due to its wide applicability. The most straight forward CSR-based spMv algorithm is CSR-scalar. This algorithm assigns computation works of a row to each thread. As shown in Figure 2, threads access entries that are apart from each other. While this non-coalescent memory access works well in CPU, it wastes the transfer capacity in GPU.

3.3.2. CSR-2step

Several modified CSR-based algorithm has been proposed to obtain better performance in GPU. The algorithm proposed by Gao et al. [3] separates spMv into 2 step as partial product step and summation step. Fully coalescent memory access is achieved in partial production step. In this work, this algorithm will be referred as 'csr-2step'.

3.3.3. SELL-P spMv

Anzt et al. [4] proposed spMv algorithm based on SELL-P format which is extension of SELL format. This algorithm will be referred as 'sellp' in this work. SELL slices the matrix into blocks of rows and pads the rows with zeros to achieve same length among rows in the same block. Since the data is stored in column major order, natural coalescent memory access in spMv is achieved. But overhead cost can be caused in SELL format because padded zeros have to be processed too.

3.2. Minimization of overhead communication cost

Computation cost can be categorized into arithmetic cost and communication cost. Communication cost includes all types of data movements. In heterogeneous computing platform composed by CPUs and GPUs, overhead communication cost is caused by data transfer and synchronization between a CPU and a GPU. Usually these overhead communication costs are heavy.

Iterative solvers require enormous communications between processor units while the amount of arithmetic work per communication is small. When considering the extremely high flop/s of GPU and heavy overhead communication cost, runtime of GPU-based iterative solver is dominated by communication cost.

To minimize the overhead communication cost, all operations in the pBiCGSTAB algorithm are moved to the GPU in this work. Though some operations like reduction algorithm in the dot product or operations with scalar variables are slower on GPU, overall performance is improved. Only one data transfer is remained per an inner iteration, the residual error norm $\|r_0\|$, which is inevitable for convergence check.

The Gauss-Seidel group sweep scheme, which is explained in 2.3., provides great reduction of arithmetic cost. However communication cost increases since this scheme solves small linear systems for each group rather than solve a large linear system relating all groups.

The performances of Gauss-Seidel group sweep and explicit solution scheme are compared in 4.3.

3.3. Approximate inverse preconditioner

nTRACER has employed block Incomplete LU (BILU) preconditioner that is well customized to CMFD problem. However triangular solve required to applicate ILU-type preconditioner is naturally sequential. Though several methods exist to achieve parallelism in triangular solve, the scalabilities of these methods are limited that utilizing massive parallelism of GPU is usually impossible. So another type of preconditioner called sparse approximate inverse (SPAI) is used for GPU platform. SPAI approximates inverse of matrix $\mathbf{A}$ in the linear system $\mathbf{Ax} = \mathbf{b}$ by Frobenius norm minimization for prescribed sparsity pattern $\mathcal{S}$:

$$\min_{m \in \mathcal{S}} \left\| \mathbf{AK} - \mathbf{I} \right\| = \sum_{j=1}^{N} \min_{m \in \mathcal{S}} \left\| \mathbf{AK}_j - \mathbf{e}_j \right\|, \qquad (4)$$

where $\mathbf{K}_j$, $\mathbf{e}_j$ are j-th column of preconditioner and identity matrix respectively [5]. As SPAI approximates the inverse of coefficient matrix, application of it in BiCGSTAB algorithm can finished by a spMv which is naturally parallelizable.

3.4. Single precision arithmetic

Usual commercial GPUs are specialized for single precision arithmetic. But round-off error accumulated in iterative calculation can limit the accuracy of solution with single precision. The strategy called 'iterative refinement' is employed to use single precision arithmetic for linear system solution on GPU with same level of accuracy as double precision calculation [6].

4. Numerical Experiment Results

4.1. Experimental setup

Table 1. Hardware specifications.

	Model	SP Gflop/s	DP Gflop/s	Band width (GB/s)
CPU	Intel® i7-6700	108.8	54.4	34.1
GPU	Radeon R9 280x	4096	1024	288

Table 2. Properties of the test matrices.

CASE	#group	NNZ ($\times 10^3$)	#row($\times 10^3$)	NNZ/row
C5G7	7	590	73	8.1
VERA#4	47	3,761	153	24.6
VERA#5	47	26,928	1,098	24.5

The heterogeneous system which combines a CPU and a GPU is set to obtain experimental results. The theoretical peak processing power and memory bandwidth of the processors are shown in Table 1.

Benchmark problems with different properties are chosen as test problems to observe performance tendency of algorithms [7, 8]. Table 2 shows properties of test problems. For VERA #4-5 case, GC CMFD calculation is carried out with 8 condensed groups. The performance for VERA #5, which is realistic 2D core problem, will be the final performance criteria of new GPU-based iterative linear system solution. VERA #5 is practically the largest problem handled by single GPU in realistic application, because the 3D core problem will parallelized into planes in multi-GPU system. The outer iteration convergence tolerance is set as 0.1 for MG CMFD and 0.2 for GC CMFD. The inner iteration convergence tolerance is set as 0.01.

4.2. spMv performance comparison

Table 3. Storage overhead of SELL-P format.

	C5G7	VERA#4	VERA#5
SELL-P NNZ ($\times 10^3$)	748	4,024	28,906
overhead/NNZ (%)	27	7.0	7.3

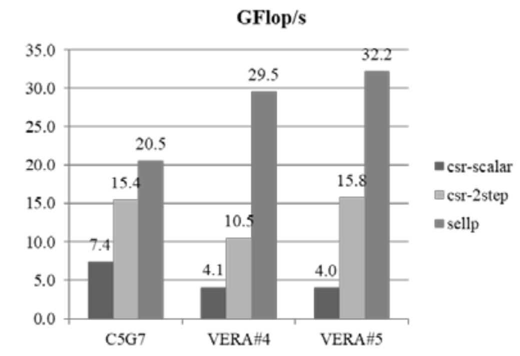

Fig. 3. Estimated flop/s of spMV algorithms.

Table 3 shows the storage overhead caused by the SELL-P format. The arithmetic cost of the spMv is proportional to the number of nonzero entries (NNZ). Since NNZ per row is similar in same group, small number of group causes discrepancy among row lengths in each block of SELL-P. As a result, sellp algorithm requires 27% more flop in C5G7 problem while it requires 7% more flop with VERA cases. Figure 3 shows performance comparison results of spMv algorithms. Estimated flop/s in figure 3 only counts effective flop and neglects additional arithmetic works in sellp algorithm. Despite storage overhead, sellp outperforms CSR-based algorithms. When see the difference between problems, the performance of sellp is improved with large NNZ per row while csr-scalar shows opposite tendency. But even in the C5G7 case, which has very sparse matrix due to small number of group, the performance of sellp is superior to CSR-based algorithms.

4.3. Calculation scheme comparison

The performance of Gauss-Seidel group sweep and explicit solution scheme is evaluated by the computation time required for linear system solving (Ax=b time) in one CMFD calculation. Table 4 shows the performance comparison results. In given computing system, explicit solution is superior to Gauss-Seidel group sweep scheme due to smaller overhead communication cost, although total flop count is larger with explicit solution. When see the tendency, Gauss-Seidel scheme has advantage in problem with large geometry and many number of group that has relatively large arithmetic work per communication. But even in VERA#5, which is practically the largest problem, explicit solution shows better performance. When considering explicit solution will be benefitted from high-end GPU with higher throughput, which reduces time for arithmetic work, explicit solution is more suitable for GPU-based solution.

Table 4. Calculation scheme comparison results.

CASE		*#outer iteration*	*Flop count* ($\times 10^9$)	*Ax=b time (s)*
C5G7	G-S.	8	0.32	0.16
	explicit	9	0.72	0.10
VERA #4	G-S.	7	0.65	0.83
	explicit	7	3.42	0.19
VERA #5	G-S.	10	7.07	2.0
	explicit	7	26.0	0.95

4.4. Performance of GPU-based solution

Table 5. VERA#5 calculation results.

	CPU	CPU+GPU
# of MOC (# of CMFD)	3 (4)	3 (4)
# MG outer iter.	32	31
# GC outer iter.	185	141
flop count ($\times 10^9$)	22	208
Preconditioner construction (s)	0.11	4.39
Ax=b time (s)	30.38	8.02

The performance of GPU-based CMFD linear system solution and conventional CPU-based sequential nTRACER CMFD linear system solution are compared. GPU-based solver uses explicit scheme with SPAI preconditioner while conventional solver uses Gauss-Seidel group sweep and BILU preconditioner. VERA #5 problem is solved for comparison. Table 5 shows the comparison results. In this work, every parts of CMFD calculation except linear system solving is done in sequential environment. Due to heavy construction cost of SPAI, time for preconditioner construction is increased in GPU-based solution. Since SPAI requires more inner iteration than BILU preconditioner and explicit solution requires more flop than Gauss-Seidel group sweep, flop count is much larger with GPU-based solution. Total flop required until convergence is about 9 times larger with GPU-based solution. In spite of flop amount, the Ax=b time of GPU-based solution is reduced to 26% level of conventional solution.

5. Conclusion

GPU-based pBiCGSTAB solver is constructed to accelerate CMFD calculation in nTRACER. The evaluation of spMv algorithms in GPU shows sellp algorithm which allows fully coalescent memory access outperforms other algorithms. Fully coalescent memory access maximizes data transfer per cycle on GPU. In addition, explicit solution of whole linear system shows better performance than Gauss-Seidel group sweep on GPU due to the overhead cost caused by communication between CPU and GPU. Both sellp and explicit solution increases the total arithmetic cost when compared to conventional algorithms. These results show that the performance of Krylov iterative solver in parallel heterogeneous system is determined by communication cost rather than arithmetic cost.

The resulted GPU-based solver reduces CMFD linear system solving time to roughly 26% in 2D core problem, though the total flop count is about 9 times larger than the conventional solution. However this research is limited on single GPU calculation on single plane, research on multi-GPU calculation on 3D problem is required as future research.

ACKNOWLEDGEMENTS

This research is supported by National Research Foundation of Korea (NRF) Grant No. 2016M3C4A7952631 (Realization of Massive Parallel High Fidelity Virtual Reactor)

REFERENCES

[1] Y. S. Jung et al., “Practical Numerical Reactor Employing Direct Whole Core Neutron Transport and Subchannel Thermal/Hydraulics Solvers,” Annals of Nuclear Energy, Vol. 62, pp. 357–374, 2013.
[2] N. J. Choi, J. S. Kang, H. G. Joo, “Massively Parallel Method of Characteristics Neutron Transport Calculation with Anisotropic Scattering Treatment on GPUs,” International Conference on High Performance Computing in Asia-Pacific Region, Tokyo, Japan, Jan 28–31, 2018.
[3] J. Gao, R. Liang, J. Wang, “Research on the conjugate gradient algorithm with a modified incomplete Cholesky preconditioner on GPU,” Journal of Parallel and Distributed Computing, Vol. 74(2), pp. 2088-2098, 2014.
[4] H. Anzt et al., “Acceleration of GPU-based Krylov solvers via data transfer reduction,” The International Journal of High Performance Computing Applications, Vol. 29(3), pp. 366-383, 2015
[5] M. J. Grote, T. Huckle, “Parallel Preconditioning with Sparse Approximate Inverse,” SIAM Journal on Scientific Computing, Vol. 18(3), pp. 838-853, 1997
[6] N. J. Choi, et al., “Performance Comparison of Linear System Solvers for the CMFD Acceleration on GPU Architectures,” Transactions of the Korean Nuclear Society Spring Meeting, Jeju, Korea, May 17-18, 2018.
[7] M.A. Smith, E.E. Lewis, B.C. Na, “Benchmark on Deterministic 2-D/3-D MOX fuel assembly transport calculations without spatial homogenization,” OECD/NEA report, NEA/NSC/DOC(2003)16, 2003.
[8] A. T. Godfrey, “VERA Core Physics Benchmark Progression Problem Specifications,” Rev. 4, CASL-U-2012-0131-004, 2014.