GPU-based Parallelization of the CUPID Code: The Strategy and Preliminary Results

Ik Kyu Park *, So Hyun Park, Il Jin Kim
Korea Atomic Energy Research Institute, 1045 Daedeok-daro Yuseong-gu Daejeon Korea
**Corresponding author: gosu@kaeri.re.kr*

1. Introduction

KAERI has developed the thermal-hydraulic analysis code CUPID, a computational multi-physics fluid dynamics (CMFD) code, which enables to handle both CFD-scale and component-scale analysis, single-phase and two-phase simulations, and multi-physics coupling applications [1,2]. For practical uses, both computational performance and numerical accuracy are crucial to CFD code. The MPI based parallelization with domain decomposition has been a significant breakthrough in enabling the CUPID code to handle large scale simulations [1].

In recent, the CUPID code faces to coupling with a high reliable Monte Carlo neutron physics code, PRAGMA[3]. Since the PRAGMA code is utilized on multi-threads of GPU cards, the CUPID code needs to be parallelized in a similar manner. Therefore, the CUPID code has begun to be parallelized using GPU. This paper discussed the part of GPU-based parallelization of CUPID and compared computational performance with single CPU core performance.

2. Methods and Results

2.1 Implementation of GPU-based parallelization on the CUPID code

The CUPID code primarily consists of a matrix solver; 1) Construct the solver matrix, 2) construct the source vector, 3) Calculate the model and correlations. The physical models are mainly contribute to the source vector, but the interfacial drag and heat transfer models associate to the solver matrix.

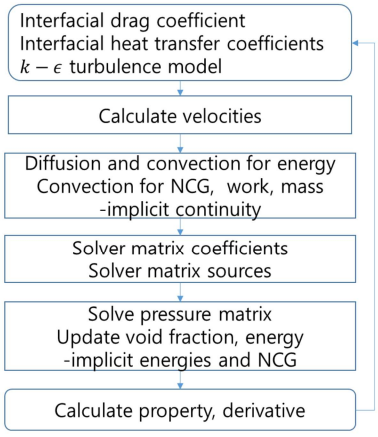

Fig. 1. The CUPID simulation flow for the single phase turbulent flow

The Figure 1 shows the CUPID simulation flow for the single-phase turbulent flow. As shown in Fig.1, 1) Matrix solver, 2) Matrix construction, and 3) Source vector construction are essential part of the CUPID code. And the numerous physical models and correlations are added according to the application.

Therefore, we profiled the performance of CUPID code using representative application and identify the minimum required subroutines in order of importance and occupancy. Then we set the strategy for implementing GPU-based parallelization on the subroutines which occupy the most computational resources: 1) Solve pressure matrix, 2) Solver matrix coefficient, 3) Calculate property and derivative, 4) Diffusion and convection for implicit method, 5) the rest of the code. In this paper, the representative application was single liquid phase turbulent flow in the rectangular channel.

2.2 GPU-based parallelization of the matrix solver

The original CUPID code adopts the BiCGSTAB with incomplete LU (iLU) preconditioner to solve the pressure matrix which denotes 'Solve pressure matrix' step in Fig. 1. This subroutine is the most crucial part of the simulation flow, so that we start GPU porting from the BiCGSTAB solver.

The BiCGSTAB solver include 1) vector reduction, 2) scalar calculation, 3) vector addition, 4) Matrix-vector multiplication (Matvec), and 5) Lower-Upper(LU) operation. Figure 2 shows the example of the coefficient matrix which should be solved by BiCGSTAB algorithm.

In this study, we apply the fundamental directive kernel '!$cuf kernel do <<<*,*>>>' to scalar calculation and the vector addition subroutines. And, Matvec calculation and LU operation are parallelized manually without directive kernel. For the LU operation, the coloring method was implemented in order to separate the matrix coefficient from neighboring cells. Figure 2 shows the example of cell coloring and the resultant coefficient matrix.

14	7	16	8
4	13	6	15
10	3	12	5
1	9	2	11

Fig. 2. The colorized cells and the solver matrix coefficients

Table 1. Comparison of the computational performance for BiCGSTAB solver

Test case	Mesh type	Number of cells	Coloring level	iLU Preconditioner(s)		MatVec Time(s)		Solver Time(s) 2LU+2MatVec	
				CPU	GPU	CPU	GPU	CPU	GPU
VD16 HMTA	3D-S	25,200	2	10	2	3	0.33	13	7
VD16 RBLA	3D-S	63,000	2	95	31	38	4	154	98
VD6 F.D.	3D-S	17,403	2	98	31	32	5	142	105
VD5 ROCOM	3D-U	37,044	4	242	73	136	6	443	179
VD14_H2P1	3D-U	113,865	4	425	45	282	3	869	132
VD23_WH14x28	3D-S	13,050	2	90	35	25	5	124	121
VFS13 N.C.	2D-S	25,600	2	178	54	46	9	252	193
VFS8_Turb3D	2D-S	48,400	2	214	24	58	4	305	85
iSMR protoype mesh	3D-U	15,398,292	6	45,565	1,076	37,689	2	109,609	1,983

Table 1 presents the computational performance comparison with single CPU core. The results indicate that the GPU-based parallelization of the BiCGSTAB solver shows reasonable performance.

2.3 GPU-based parallelization of matrix and source vector construction

Matrix and source vector construction indicates calculation of solver matrix elements and the vector elements. And these elements are formulated by the diffusion and convection equations of the mass and energy. The same implement method is applied to this subroutines: The directive kernel '!$cuf kernel do <<<*,*>>>' is applied in straightforward. The key aspect of the subroutine is utilizing cell-based variables within the face-based streaming processor and vice versa- leveraging face-based variables within the cell-based streaming processor.

2.4 GPU-based parallelization of model and correlation calculation

This phase calculates 1) the interfacial drag and heat transfer coefficients of two-phase, 2) steamtable, and 3) turbulence model. The explicit velocity calculation also can be included. This simple subroutines are parallelized using the directive kernel '!$cuf kernel do <<<*,*>>>'. The parallelization is straightforward because the most formulations have no dependency on the neighbor cell variables.

3. Conclusions

,

In this paper, we have developed the GPU-based parallelized CUPID code. First, we set the development strategy using the performance profiling of CUPID. The most time-consuming subroutines that calculated BiCGSTAB solver was parallelized in the inital step. The computational performance was compared with the single CPU core, and it shows the reasonable performance. Afterward, parallelization was carried out step by step: 1) the matrix and vector construction, 2) model and correlation calculation. As a result, the performance of certain test cases shows that GPU-based parallelization of the CUPID code is promising for enhancing computational speed.

ACKNOWLEDGMENTS

This work was supported by the Korea Atomic Energy Research Institute (KAERI) [524520-25].

REFERENCES

[1] Y.H. Choi, and Y.Y. Han, Numerical Analysis of the ROCOM Boron Dilution Benchmark Experiment Using the CUPID Code, Nuclear Engineering and Design 341, 2019.

[2] I.K. Park, J.R. Lee, Y.H. Choi, and D.H. Kang, "A multi-scale and multi-physics approach to main steam line break accidents using coupled MASTER/CUPID/MARS code," Annals of Nuclear Energy 135 (2020) 106972.

[3] N. Choi and H. G. Joo, "Domain decomposition for GPU-Based continuous energy Monte Carlo power reactor calculation," Nuclear Engineering and Technology, vol. 52, pp. 2667-2677, 2020.