Parallelization Performance of the Two-Phase Thermal-Hydraulics code, CUPID

Jae Ryong Lee[a*], Han Young Yoon[a], H.G. Choi[b]
[a]*Korea Atomic Energy Research Institute,1045 Daeduk-daero, Daejeon, Korea, 305-353*
[b]*Seoul National University, San 56-1, Shillim-Dong, Kwanak-Gu,Seoul 151-742, South Korea*
[*]*Corresponding author: jrlee@kaeri.re.kr*

1. Introduction

KAERI has been developing a component-scale thermal hydraulics code, CUPID [1]. The aim of the code is for multi-dimensional, multi-physics and multi-scale thermal hydraulics analysis. In our previous papers, the CUPID code has proved to be able to reproduce multi-dimensional thermal hydraulic analysis by validated with various conceptual problems and experimental data [1].

Recently, issues for a long transient calculation are raised which require a speedup of the calculation time. In order to make a best use of the multi-core hardware system, the CUPID code has being parallelized. By performing the verification for the fundamental problem set, the parallelization of the CUPID has been successfully confirmed. In addition, the parallelized CUPID code shows a good scalability. This code will be applied to calculate practical long transient phenomena.

2. Numerical Methodology

2.1 Governing equation

The CUPID code [1] adopts the two-fluid model for two-phase flows. In the two-fluid model, the mass, energy, and momentum equations for liquid and vapor phases are established separately, and then, they are linked by the interfacial mass, momentum, and energy transfer models. For a mathematical closure, the constitutive relations for the interfacial momentum transfer, the interfacial heat transfer and the wall heat partitioning are necessary.

2.2 Parallelization of the CUPID code

For the parallelization in the CFD codes, Two different methods are generally applied: OpenMP[2] and MPI[3]. OpenMP has a multi-thread concept which does not partition a computational domain but divides the do-loop into the number of processors and activates the thread to run the designated loop. It can be easily applied to the existing code without major modification but should be operated in the SMP (Shared Memory multi-Processor) computer. Because of the frequent activation and deactivation of the thread, the efficiency of the parallelization relatively low. On the other hand, the parallelization based on the domain decomposition can utilize the wide range of hardware from multi-core single PC to general supercomputer. The efficiency of parallelization is much better than the OpenMP. The CUPID code is parallelized based on this way with MPI (Message Passing Interface) [2]. MPI can be easily installed in any hardware and also shows good performance for the parallelization by using internal MPI libraries.

For the best scalability of the parallelized code, the computational domain should be decomposed considering the load balance. One of the simplest ways is that the users decompose the calculation domain during pre-processing, in which they should consider the number of meshes assigned on each processor to be even. The other way is to use the freeware software, METIS [3] which can automatically partition an arbitrary computation domain regardless of the shape of the control volumes. The CUPID code adopted both methods. After partitioning the computational domain, a local renumbering of the subdomain and an interface array should be defined.

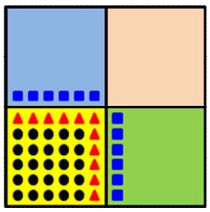


Fig 1 Interface array for subdomain

Fig 1 shows the example of the 4 partitioned subdomain. The variable at black dot (●) cell is named as interior variable and independent from the cells which are assigned in other processors. However, the array at the interface (red triangle ▲) is interior interface variable which is coupled with the arrays of other processors. The variable at the blue square (■) is exterior interface variable which is a buffer array for obtaining the variable from the other processors. During the calculation, the information at the interface cells is exchanged whenever the neighboring cells is involved.

2.3 CSR (Compressed Sparse Row) Format

Since the CUPID code is based on the unstructured grid system, the matrix might have irregular non-zero pattern. In order to handle this matrix, the CUPID code adopts iterative solver rather than a direct solver. Moreover, a preconditioning method is also used to enhance the speed of convergence of the solution.

Of various methods to make the matrix from the discretized governing equation which has irregular non-zero pattern, CSR scheme is adopted. CSR save only non-zero value and its position (row and column). The sparse matrix can be represented using four (4) arrays as follows;

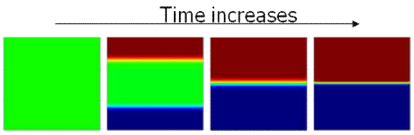
- au(:): array to save the non-zero value as row-by-row,
- ja(:): column of the non-zero value of au(:),
- ia(:): order of au(:) and ja(:) existence of the first non-zero value from each row,
- ju(:): order of au(:) and ja(:) existence of the diagonal value from each row.

These arrays are used for the ILU preconditioning [4]. Then, the parallelized iterative solver is used. Since the CUPID code produces an asymmetric matrix for two-phase simulation, the bi-conjugate gradient solver should be applied.

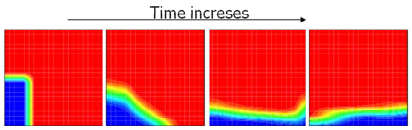
3. Result and Discussion

3.1 Verification of the solution

In order to confirm the accuracy of the solution of the parallelized CUPID code, the verification procedure is performed with several calculation set. Of the calculations, a few representatives are introduced in this paper as shown in Fig 3. Fig 3(a) is phase separation due to the gravity and Fig 3(b) is dam break simulation, which are fundamental examples to verify the two-phase capability.

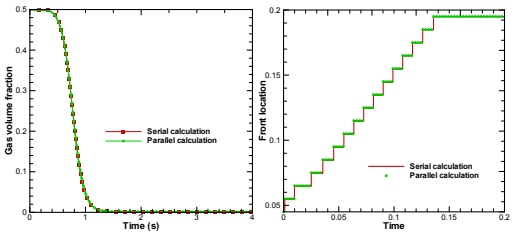


(a) Phase separation



(b) Dam break

Fig 2 Verification examples



(a) (b)

Fig 3 Verification result between serial and parallel calculation; (a) void fraction at designated position for phase separation; (b) front location of the liquid for dam break

Fig 4 shows the verification results between serial and parallel calculation. Both two calculations are typical two-phase conceptual problem to verify the discretized governing equation. Fig (a) is a time history of the void fraction at a designated position and Fig 5(b) is a front location of the liquid. It is shown that the solution from both serial and parallel computation is exactly same.

3.2 Scalability of the Parallelized CUPID code

In order to evaluate the performance of the parallelized CUPID code, the scalability is measured as shown in Fig 5. In general, the scalability depends on the number of computational grid points. Here, two grid systems are examined. For the relatively small grid system of 40,000, the parallelized CUPID code is scalable by using up to 4 processors and under-scalable when applying 8 processors. However, the grid system with 144,000 meshes, it shows super-scalable up to using 8 processors. Therefore, the number of meshes and the selection of how many processors to be used should be carefully considered.

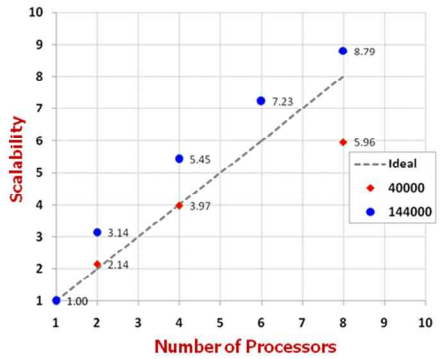


Fig 4 Scalability of the parallelized CUPID code

4. Conclusions

The CUPID code was parallelized with domain decomposition method. The MPI library was adopted to communicate the information at the interface cells. The CSR matrix formulation was applied to shorten the computational time. The parallelized CUPID code was verified against fundamental two-phase problem. Finally, in order to examine the scalability, a relatively large number of grid systems were applied so that the parallelized CUPID code showed super-scalability.

Acknowledgement

This work was supported by Nuclear Research & Development Program of the NRF (National Research Foundation of Korea) grant funded by the MEST (Ministry of Education, Science and Technology) of the Korean government(Grant code: 2012M2A8A4025647).

REFERENCES

[1] H.Y. Yoon, et al., “Multi-Scale Thermal-Hydraulic Analysis of PWRs using the CUPID Code”, Nuclear Energy and Technology, Vol.44, pp.831-846, 2012.
[2] http://www.mpi-forum.org
[3] Karypis, G. et al., "A software Package for Partitioning Unstructured Graphs, Partitioning Meshes, and Computing Fill-Reducing Orderings of Sparse Matrices V.4.0," U. of Minnesota, 1998
[4] Y.S. Nam, H.G. Choi and J.Y.Yoo, “AILU preconditioning for the finite element formulation of the incompressible Navier–Stokes equations”, Computer methods in applied mechanics and engineering, Vol.191, pp.4323-4339, 2002.