A heterogeneous parallel model of unstructured mesh finite element method based on CPU+GPU

Yu Lei*, Guoliang Peng, Yinjun Gao, Feng Han and Dong Wang

Northwest Institute of Nuclear Technology, Xi'an, China

* Corresponding Author Email: leiyu@nint.ac.cn

Abstract. Most of the existing numerical simulation programs using the unstructured mesh finite element are based on the traditional multicore processor architecture. With the increase of the number of computing meshes, the computing time is increasing, which leads to the common multicore CPU cluster can't meet the high computing demand of complex applications. In order to adapt to the trend of the heterogeneous development of high-performance computers, a heterogeneous parallel model of unstructured mesh finite element method is proposed in this paper. It can transplant the unstructured mesh finite element program framework to heterogeneous platform better and faster. The model realizes the efficient utilization of the multicore CPU by hierarchical parallelization, and realizes the efficient utilization of GPU by heterogeneous parallel rewriting for time-consuming computing hotspot. Finally, the model is applied to the parallel transplantation of CPU + GPU heterogeneous platform for the thermal radiation effect program. The results show that the model can reduce the programming difficulty and has good portability and extensibility.

Keywords: heterogeneous parallel model, unstructured mesh finite element method, CPU+GPU.

1. Introduction

The unstructured mesh finite element method is a basic algorithm for numerical simulations of thermal radiation effects and mechanics. It can deal with complex geometric shapes and nonlinear material behaviors, and has high precision and high efficiency characteristics. When dealing with large-scale complex structure problems, the traditional finite element method has many problems, such as large computation amount, long calculation time and high memory consumption. The exascale era of the high performance computing [1] has brought a new computing platform for the parallelization of the unstructured mesh finite element method. The heterogeneous parallelism of unstructured mesh finite element method [2] has become one of the hotspots of current research.

Because of the random structure of the unstructured mesh, the topological relation and storage format of the mesh are complex, and the numerical simulation quantity is large. The consumption of memory and CPU time of the computer is much larger than that of the structured mesh, and the mesh partition and parallel computation are more difficult. The parallel optimization of the unstructured mesh finite element method based on heterogeneous platform will result in the problems of discrete access and data transmission, which will reduce the performance of the program. In this paper, a heterogeneous parallel model of the unstructured mesh finite element method is proposed, which can effectively utilize the parallel characteristics of the algorithm and the hardware characteristic of the heterogeneous system architecture. Finally, the model is used to guide the heterogeneous parallel transplantation of the thermal radiation effect calculation program, and the numerical simulation experiment of typical application verifies the rationality and feasibility of the model.

2. The Heterogeneous Parallel Model

At present, the heterogeneous system mainly has the following architectures: CPU + GPU, CPU + MIC, Sunway heterogeneous multicore processor and Tianhe-3 (FT-2000+ and MT-2000+) [3] heterogeneous system, among which the typical feature of the multicore processor is to abandon the control unit in the general CPU chip. This allows more transistors to be used for computing, so they have higher parallel processing capabilities. Different heterogeneous processors have different

hardware architectures. In order to get the best parallel acceleration effect on a heterogeneous system, we must make adaptive parallel optimization according to the application characteristics.

When using traditional finite element method to solve the problem, it is necessary to form a matrix of total coefficient and it needs enough storage space. With the increasing of numerical computation scale of the thermal radiation effect, computational problems become more complicated. How to parallelize every step of the computing process for finite element method becomes the focus of current research. At present, the region decomposition method and the parallel method of solving equations are the two common parallel finite element methods.

In order to better use the computing resources of heterogeneous systems and give the full play to the parallel characteristics of the finite element method, a heterogeneous parallel model [4, 5] of unstructured mesh finite element method is proposed, as shown in Fig.1. According to the characteristics of physical parallelism, the model divides the mesh area and uses MPI process [6] to realize the parallelism among nodes; for multiple mesh calculations in each computing area, we can use multiple threads of the CPU cores in the node; computational-intensive hot spots in the program can be transferred from CPU core to accelerator or slave core, and parallel optimization can be carried out to improve parallel efficiency.

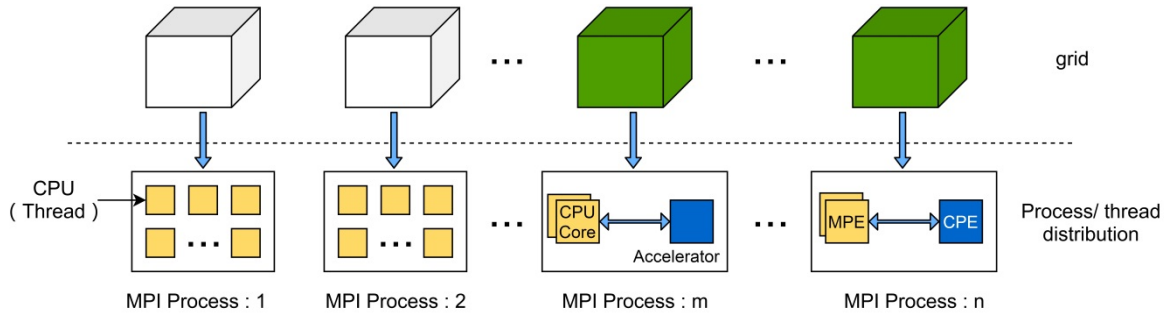


Figure 1. The heterogeneous parallel model of the unstructured mesh finite element methods.

In the heterogeneous parallel transplanting of concrete unstructured mesh finite element applications, the computing tasks are distributed to multiple computing nodes by coarse granularity domain decomposition, and the data exchange between nodes is used to realize efficient parallelism. The parallel code has the highest portability at this time, and it can be used quickly in different heterogeneous systems. In order to better use the heterogeneous resources of the system, heterogeneous parallel optimization must be carried out. The most efficient parallel approach is to parallelize the entire computing process of the node. This will result in low portability of the program, because different heterogeneity requires specialized rewriting. Therefore, we retain the thread-level parallel rewriting of nodes, and then perform heterogeneous parallelization rewriting of the time-consuming hot spots. For finite element methods, the solution of equations is the most time-consuming process, so we choose this part to carry out heterogeneous parallel optimization.

The difficulties of the unstructured mesh finite element program in heterogeneous parallel transplantation are task allocation, load balancing, discrete memory access of unstructured mesh and data communication. Based on the characteristics of the unstructured mesh finite element program, we propose some relevant solutions: (1) unstructured mesh applications generally have enough concurrent meshes, which can be divided into coarse-grained tasks by region decomposition, and the sub-regions can be assigned to the nodes by taking into account the partitioned boundaries that need to be communicated between nodes. (2) Each process or thread is allocated to use one or more concurrent computing areas. If a single node is allocated to use enough computing meshes, the meshes can continue to be decomposed and allocated to different CPU cores for calculation, which is helpful to the dynamic migration and merging of CPU multi-core tasks and dynamic load balancing within nodes. (3) The problem of discrete access of the unstructured mesh is caused by random access of the data, and the efficiency of access can be improved by increasing the locality of the data by data

rearrangement. (4) The frequency of the communication can be reduced by data compression, contiguous data merging, data reuse, and the cost of communication can be hidden by data transmission and computation simultaneously.

In the practical problem, most of the physical application problems are existed in the complex structures with irregular geometric shape. The strong adaptability of unstructured mesh in complex irregular region makes it better to apply. Unstructured meshes need more memory space to store location information, and the disorder of data storage also brings the problem of discrete access to the memory for unstructured meshes. With the increase of the mesh size, the problem of discrete access is more and more prominent, and the negative effect on computing efficiency is increased multiple. In the large-scale computation based on heterogeneous systems, this problem even becomes the main performance bottleneck of the technology development.

3. Heterogeneous parallel of the thermal radiation effect algorithm

In order to accomplish the heterogeneous parallelization of applications in heterogeneous systems more quickly, we adopt the traditional parallel strategy. The strategy is that we carry on the transformation of the heterogeneous parallelization for the most time-consuming part, and we carry on the traditional CPU parallelization in other computing processes. Therefore, we use two-stage parallel scheme in the heterogeneous parallel transplantation of the thermal radiation effect program. Firstly, the mesh region is divided according to the unstructured mesh file. Then the mesh region is allocated to many processes for calculation, and the different processes communicate with each other through the message passing interface (MPI) [6]. The internal computing hot spots of the process are accelerated using the heterogeneous GPU.

In the process of solving the finite element method, the solution of sparse linear equations [7] plays an important role in the whole calculation process. This part of the calculation is suitable for heterogeneous transplantation because it is relatively independent and has the characteristics of intensive computation and time-consuming. The input parameters of sparse linear equations are the coefficient matrix and the right-hand vector, and the output is the solution vector. The solution procedure of sparse linear equations mainly includes storage and calculation, in which the storage part refers to how to compress storage, and the calculation part refers to vector copy, vector addition (subtraction), vector inner product, and sparse matrix vector multiplication (SPMV). With the increase of the matrix size, the direct solution method of linear equations is difficult to be applied, and the iterative solution of linear equations becomes a better choice [8].

The method of bi-conjugate gradient stabilized (BICGSTAB) is an iterative method for solving asymmetric linear equations, and the PBICGSTAB is the preconditioned bi-conjugate gradient stabilized. It can be found that the PBICGSTAB method consists of four basic computation operations: vector addition/ subtraction, vector inner product, matrix vector multiplication and inverse matrix vector multiplication. We use the CUDA language to write a linear equation solver based on PBICGSTAB [9, 10]. In order to improve the CPU-GPU transmission efficiency, we convert the coefficient matrix of linear equations into CSR compression storage format at the CPU, which will reduce the amount of the transferred data between the CPU and the GPU. In the optimization of the GPU program, we find that the protocol and synchronization are the bottleneck problems that affect the whole GPU program. We have written high-quality protocol functions and adjusted the order of the computation, so that these problems can be effectively solved.

When the parallel program is implemented on the GPU, the compressed storage of sparse coefficient matrix for the PBICGSTAB algorithm is completed firstly in the CPU. After the matrix data and the right-hand vector are passed into the memory space on the GPU by the CPU core, the main calculation part of the solution of the equations [11] is completed by the GPU. Finally, the result is passed back to the CPU by the GPU, as shown in Fig.2. The CPU core and the GPU communicate data only twice, which minimizes the data transmission time between devices. The disadvantage is that the size of the problem is limited by the memory capacity of the GPU. In order to make interface

calls in the solution part of the equations, we compile the GPU program into a static library. The specific compilation instructions are as follows:

nvcc -lib pbicgstab.cpp -o libbicgstab.a

gcc -o main main.c -I/usr/local/cuda/include -lbicgstab -L. -lcudart -lcusparse -lcublas -L/usr/local/cuda/lib64

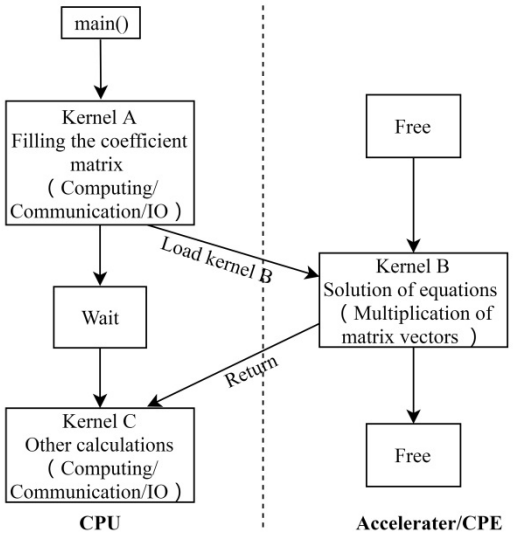
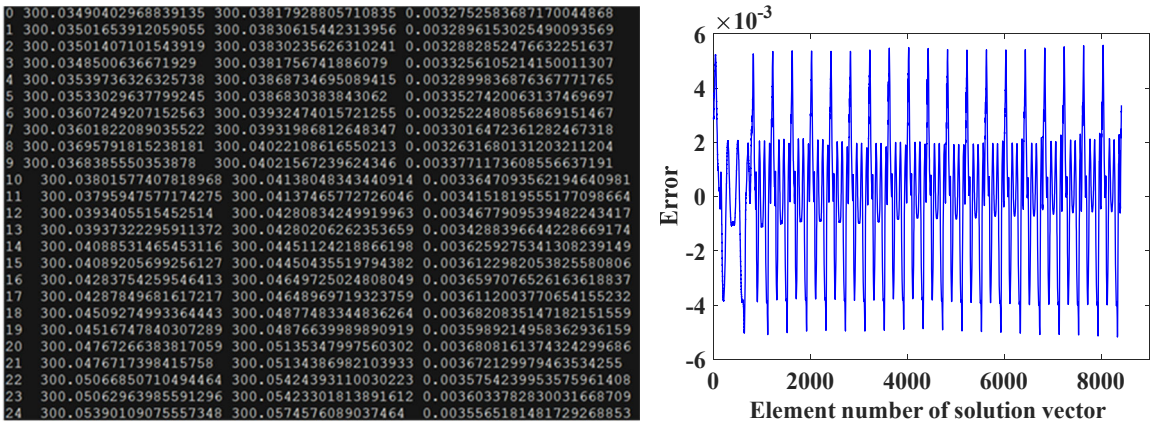


Figure 2. Heterogeneous parallel implementation process.

4. Results

The test environment we used includes ten NVIDIA A100 GPUs and two Intel Xeon Gold 6348 CPUs. In order to illustrate the correctness of the self- compiled GPU linear equation solver, we use the KSP solver of the PETSc [12] and the PBICGSTAB algorithm to solve the same linear equation system. Fig.3 (a) shows a part of the solution vector, in which the number is the first column, the KSP result is the second column, the GPU PBICGSTAB algorithm is the third column, and the difference between the two methods is the fourth column. Fig.3 (b) shows the difference curves of the results for the two methods, in which the number of the solution vector elements is a horizontal coordinate (the total number of elements is 8421) and the error is a vertical coordinate. Since the difference between the two is 10-3 magnitude, we think that the self-compiled GPU linear equation solver has the ability to calculate correctly.

In order to evaluate the performance of the program more accurately, we only show the computation time of the key code segments. The key code segment of TRE (the thermal radiation effect program) mainly includes the solution of equations, and the key code segment of TREGPU (the thermal radiation effect program of GPU) includes some CPU calculations and all GPU calculations. Table 1 shows the calculation time of the two-dimensional thermal radiation effect problem. It can be seen that the performance of the TREGPU is not as good as the TRE when the number of meshes is small; The performance of the TREGPU with 8 MPI processes and 8 GPUs is better than that of the TRE gradually when the number of meshes reaches 800,000, and the acceleration ratio of the TREGPU with 10 MPI processes and 10 GPUs is 1.3 compared with that of the TRE. Fig.4 shows the calculation results of the temperature field for the two-dimensional thermal radiation effect problem using the TREGPU.



(a) (b)

Figure 3. Comparison between the ksp solver and the GPU PBICGSTAB solver.

Table 1. Computational Time Comparison of the Key Code Segments for the Two-dimensional Thermal Radiation Effect Problem.

Number of Meshes	TRE		TREGPU	
	Number of MPI	Time(ms)	Number of MPI + GPU	Time(ms)
8421	1	6	1 + 1	114
	2	4	2 + 2	52
804201	8	1575	8 + 8	1230
	10	1337	10 + 10	1039

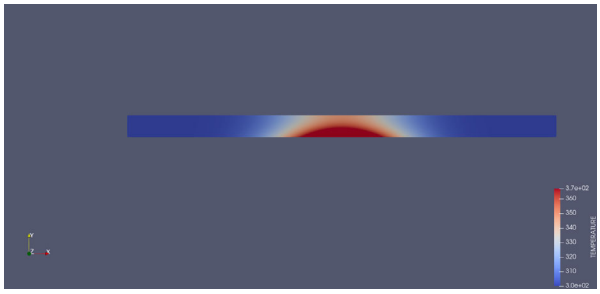


Figure 4. The Calculation of the Temperature Field for the Two-Dimensional Thermal Radiation Effect problem using TREGPU.

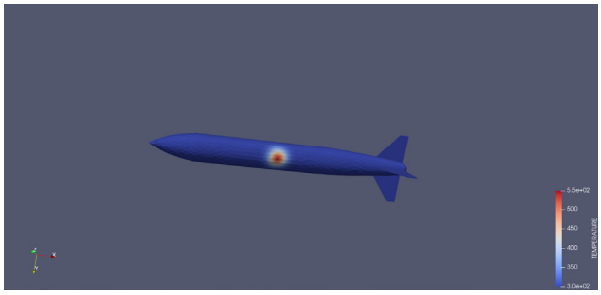


Figure 5. The Calculation of the Temperature Field for the Three-Dimensional Thermal Radiation Effect Problem using TREGPU.

Table 2. Computational Time Comparison of the Key Code Segments for the Three-dimensional Thermal Radiation Effect Problem.

Number of Meshes	TRE		TREGPU	
	Number of MPI	Time(ms)	Number of MPI + GPU	Time(ms)
6727	1	5	1 + 1	18
	2	4	2 + 2	13
507207	6	157	6 + 6	93
	8	97	8 + 8	42

We also tested the three-dimensional numerical examples, and the results are in good agreement with those of the two-dimensional numerical examples. Table 2 shows the calculation time of the three-dimensional thermal radiation effect problem. As the number of meshes increases, the performance of the TREGPU is gradually better than that of the TRE. The performance of the TREGPU with 6 MPI processes and 6 GPUs is better than that of the TRE when the mesh size reaches

500,000, and the acceleration ratio of the TREGPU with 8 MPI processes and 8 GPUs is 2.3 compared with that of the TRE. Fig.5 shows the calculation results of the temperature field for the three-dimensional thermal radiation effect problem using the TREGPU.

5. Conclusion

In this paper, a heterogeneous parallel model of the unstructured mesh finite element method is proposed to solve the problem that it is difficult to transfer the unstructured mesh finite element method to heterogeneous processor. The model has been successfully applied to the thermal radiation effect program, which shows the rationality and feasibility of the model. In addition, the model has many parallel ways, which can be extended to other multi-core systems according to the characteristics of the application.

References

[1] Qian De Pei, Wang Rui. Key issues in exascale computing. Scientia Sinica(Informationis), 2020, 50 (9): 1303-1326 (in Chinese).

[2] CAI Y, LI G Y, LIU W Y. Parallelized implementation of an explicit finite element method in many integrated core (MIC) architecture[J]. Advances in Engineering Software,2018 (116): 50-59.

[3] You, X, Yang, etc. Performance evaluation and analysis of linear algebra Kernels in the prototype Tianhe-3 cluster (Conference Paper) [J].Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics),2019,11416:86-105.

[4] Linghong Wu. Research on the Development and Application of Parallel Programming Technology in Heterogeneous Systems. Journal of Physics: Conference Series,2022: 1742-6588.

[5] Boro Sofranac, Ambros Gleixner, Sebastian Pokutta. Accelerating domain propagation: An efficient GPU-parallel algorithm over sparse matrices. Parallel Computing. 2022: 0167-8191.

[6] Message Passing Interface Forum, MPI: A Message-Passing Interface Standard, Version 3.1, High Performance Computing Center Stuttgart (HLRS), 2015, http: //mpi-forum.org/docs/mpi-3.1/mpi31-report.pdf, (Accessed 13 July 2020).

[7] NI Hong;LIU Xin. Many-core optimization for sparse triangular solver under unstructured grids [J] . Computer Science, 2019, 46 (S1): 518-522.(in Chinese)

[8] MERRILL D, GARLAND M. Merge-based parallel sparsematrix-vector multiplication [C] //Proceedings of International Conference for High Performance Computing, Networking, Storage and Analysis. Washington D. C. USA: IEEE Press, 2016: 1-12.

[9] “Nvidia CuBLAS,” https://developer.nvidia.com/cublas.

[10] “Nvidia CuDNN,” https://developer.nvidia.com/cudnn.

[11] Zhao Yu, Ma Xiaojun, Zhang Chengbin, Chen Jiujiu, Zhang Yuanhui. A GPU-accelerated particle-detection algorithm for real-time volumetric particle-tracking velocimetry under non-uniform illumination[J]. Measurement Science and Technology,2021,32(10).

[12] Satish Balay, Shrirang Abhyankar, Mark F. Adams, Jed Brown, Peter Brune, Kris Buschelman et al. PETSc users manual. Technical Report ANL-95/11 - Revision 3.9, Argonne National Laboratory, 2018.More references