Design of General HLA Simulation Federate Based on GPU

Guo-hua Zhu[1], Min Cao[1], Hai-zhou Wang[1], and Li-feng Wang[2(✉)]

[1] School of Artificial Intelligence, Jianghan University, Wuhan 430056, China
zhuguohua1215@tom.com
[2] BaiSe University, BaiSe 533000, China
wanglifeng1101@tom.com

Abstract. In view of the operation efficiency of federate in HLA based simulation scheme, this paper analyzes the method of calling GPU computing resources in HLA federate, and designs a general simulation federate structure based on GPU. By comparing the results of the simulation scheme based on CPU and GPU, it is proved that under the premise of ensuring the stability of simulation federation operation, the simulation Federate based on GPU can greatly speed up the running speed of the Federation members and improve the simulation development and operation efficiency.

Keywords: Simulation · HLA · Federate · GPU · Social information · Wireless network

1 Introduction

HLA (high level architecture) builds simulation system according to the idea and method of object-oriented. It is the technology of dividing simulation members and building simulation federation on the basis of object-oriented analysis and design [1]. HLA mainly considers how to integrate federations on the basis of federates, that is, how to design the interaction among federates to achieve the purpose of simulation. It does not consider how to build federations from objects, but how to build federations under the assumption of existing members, which is an important reason why it is called "advanced architecture". The basic idea of HLA is to use object-oriented method to design, develop and implement the object model of simulation system, so as to achieve high-level interoperability and reuse of simulation federation [2]. In HLA, interoperability is defined as one member can provide services to other members and receive services from other members. HLA itself can not fully realize interoperability, but it defines the architecture and mechanism to realize interoperability among Federation members. In addition to facilitating the interoperability among members, HLA also provides a flexible simulation framework for Federation members. In HLA, the distributed simulation system used to achieve a specific simulation purpose is called Federation, which consists of several interactive simulation object models [3]. HLA defines the basic principles and methods

S. Liu and X. Ma (Eds.): ADHIP 2021, LNICST 417, pp. 124–132, 2022.
https://doi.org/10.1007/978-3-030-94554-1_11

of Federation and federate member construction, description and interaction. It should be noted that the Federation can join a larger Federation as a member.

With the development of computer technology, the practical problems solved by computer simulation technology are becoming more and more complex, and the running speed of simulation systems has been unable to meet the needs of practical problems. GPU (graphics processing unit) has a unique advantage in the field of computer simulation acceleration due to its natural super computing power [4]. Compared with GPU computing, CPU needs to support parallel and serial operations at the same time. It needs a strong versatility to deal with different data types. At the same time, it also needs to support complex universal logic judgment, which will introduce a large number of branch jump and interrupt processing. All these make the internal structure of CPU extremely complex, and the proportion of computing units is reduced. GPU is faced with highly unified, interdependent large-scale data and pure computing environment that does not need to be interrupted. Therefore, GPU chips are much simpler than CPU chips, and GPU designers use more crystal tubes as execution units, which also leads to the super computing power of GPU. For some lightweight computing intensive HLA Simulation members, GPU based application research has made some progress in recent years. At present, scholars in related fields have done some research on the design of general HLA Simulation federate, and achieved some research results. In Ref. [5], this paper introduces the development process of the lbts-sims Federation, and gives experience and lessons in the process of federation development. The data generated by legacy simulation is merged into logical groups to achieve low latency, run-time efficiency and data distribution optimization within Federation. Lbts-sims currently includes four federates: a scene control federate (a legacy ground truth server) and three new local HLA federates. The system can be used as a guiding ideology for HLA developers to integrate legacy simulation with HLA Federation. However, developers customize their own products according to business requirements. There are some problems, such as no unified standard, poor cross platform performance, weak descriptive ability, and inconsistent hardware support for various versions. Based on the above background, this paper designs a general HLA Simulation federate system based on GPU. By introducing the basic implementation of Federation in HLA architecture, a general federation architecture is constructed, and a General Federation programming model based on GPU is designed, so that users can improve the general method of HLA Federation calling GPU processor. This system can speed up the operation of simulation members and improve the efficiency of simulation development and operation.

2 Establishing a General Simulation Member Solution

HLA (high level of Architecture) is a complete set of distributed simulation technical specifications. Its core goal is to solve the simulation support environment of traditional distributed simulation technology. RTI provides general and independent simulation support services. On the issue of resource reusability, this system still can not meet the current needs. Therefore, it is very necessary to research and apply the universal member model framework based on HLA. It can integrate a large number of federate simulation members, design a common member model shared by all simulation members, reduce

the degree of resource redundancy waste, and improve the model compatibility attribute of simulation system, as well as the expansibility of simulation. At present, there are universal federates that implement C++ interface, but this paper implements universal federates of java interface.

Regular federation members are only suitable for specific simulation models. The advantage of the general member is that when the simulation program changes, the general member does not modify the simulation model description file (such as XML file) according to the new simulation program, so as to obtain a new simulation model, and solves the problem of programmers when the simulation program changes. The simulation members made major revisions and rewrites, and described the algorithm of the simulation entity through the DLL file, which was used to describe the dynamic behavior of a certain entity in the federation, such as the control algorithm of the PID controller, etc. [6]. The General Federation member is responsible for loading DLL algorithm and reading the member model file XML, and joining it as a member in the Federation, updating the attribute value, receiving the reflection attribute value, receiving the interactive command from the control end, so as to control the status of simulation members (start, pause, stop, etc.) through the control end.

Therefore, according to the processing flow of federate member interaction data, its composition is mainly divided into: member initialization, data receiving layer, data processing layer, data distribution layer, exit federate execution. The processing flow of Java based general federate member interaction data is as Fig. 1.

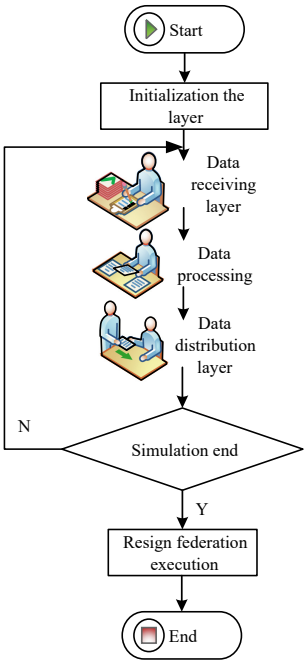

Fig. 1. Processing flow of interaction data of General Federation members based on Java

Initialization the layer, whose functional requirements: prepare the Federation members to participate in simulation promotion, namely, universal Federation member initialization. Its initialization content: Determine basic information such as federated member ID (Member IDentity, identity ID), federated member name (MemberName), determine and load corresponding algorithm components, such as DLL (Dynamic Link Library). The control end has completed the creation of the federation execution (createFederationExecution), waiting for the control command of the control end, the federation members join the corresponding federation execution (joinFederationExecution), declare the relationship between publish and subscribe, and establish the logical port relationship for information and data transmission and reception. Determine the time advancing strategy, whether the federation members are time restricted (Time Regulation), and whether it is time controlled (Time Constrained).

Data receiving layer (DRL), its functional requirements: Receive and store interactive data, provide data support for the data processing layer of federation members [7]. Its main content: according to the ordering relationship declaration in the initialization module, establish the corresponding data relationship storage table, temporarily store the temporary data and obsolete data, such as the latest data will be retained until the third time after receiving the data, and then be replaced.

Data processing layer (DPL), its functional requirements: In accordance with the time to promote service management under the premise of completing the corresponding interactive data processing, in order to get the processing results, and then by the data distribution layer processing. The main contents are as follows: Select the correct received data as the entry parameter of the federate; use the unique algorithm component of the federate to process the received data, get the processing results, and submit the processing results to the data distribution layer.

Data distribution layer (DDL), its functional requirements: according to the published relationship statement to accurately complete the data transmission [8]. Its main contents are: Obtaining the return data of the processing layer; confirming the time advance permission; if it is allowed to advance, it can continue to advance the simulation; confirming the data transmission channel; completing the data transmission.

The functional requirements of design federation execution (RFE): the federate ends the simulation and exits the Federation Execution [9]. Its main content: Receive the command of the control and exiting the federation execution, or the Federation members exit the federation execution by themselves, and the Federation members release their own simulation resources.

By building a general federate, the function of each interface can be realized, and the HLA interface framework and federate model are successfully separated, which makes the development of simulation federate easier and makes the model compatibility and reusability of simulation system greatly enhanced, and provides a foundation for the algorithm implementation of general calling GPU resources.

3 Design of Universal HLA Simulation Federate

The main design goal of universal federate is to enable HLA Simulation members to call GPU computing resources in a unified and simple way. The first is the production of DLL. HLA general simulation members are based on Java. From the point of view of programming language, CUDA and OpenCL both support C/C++ natively. It's troublesome to access other languages. To call GPU computing resources through Java code, you can access CUDA or OpenCL through JNI, or use various GPU Programming libraries based on Java version of JNI, such as jcuda [10].

Our idea is to use C/C++ programming to call the GPU resources, and then make the program into a DLL for Java program to call. By calling the DLL file, we can call the GPU computing resources.

Developing a special simulation member based on the general simulation member program. In the simulation member, we first need to declare the publishing and ordering relationship of object class and interaction class in each simulation member. Data forwarding is to forward the data content of the party who produces the data (Publishing member) to the party who is interested in these messages (subscribing member). The key to solving the problem of data distribution is to make the corresponding relationship between the publishing party and the subscribing party. Declaration management determines the direction of message forwarding, that is, to determine the message source (Publishing member), also to determine the ordering relationship between the message object (ordering member) and the publishing relationship of the simulation member itself, so that each member can accurately receive the information they need [11]. After the simulation operation, with the promotion of simulation, each simulation member constantly updates their object class attributes, and the simulation members constantly generate the data to be processed. During the simulation promotion, the simulation members pass the data to the DLL in the form of input parameters by calling the DLL file. After the data is processed by GPU, the data will be returned to the simulation member. The simulation member controls the whole simulation promotion according to the final received data.

Each simulation federate program is divided into two modules. One module is a general simulation member, which is used to arrange the HLA simulation logic processing part of the system model to run on the computer, in order to complete the HLA simulation federation operation [12]. The other is DLL file, which aims to complete the calculation function of simulation members by calling the GPU computing resources, and speed up the running speed of simulation members through GPU. Among them, the relationship between simulation member module and DLL module is as Fig. 2.

DLL module realizes the calling algorithm of GPU computing resources. The internal operations of DLL file to call GPU computing resources are as follows: firstly, the CPU allocates a block of memory in its internal cache, creates a data table on this memory, and saves the data sent by simulation members to be processed in this data table. As like as two peas, GPU also has a data table that is exactly the same in the internal cache. GPU sets the algorithm according to the algorithm set in advance in CPU, calculates the data table in the GPU internal cache, and finally sends the processed data to CPU, then returns it to the simulation member by CPU [13]. In terms of processing details, first create a view dataview on the internal cache of the CPU through

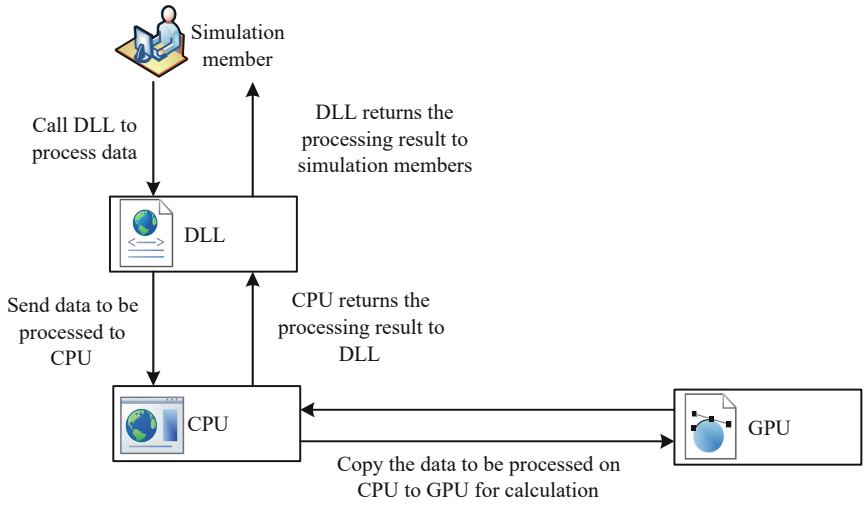

Fig. 2. Relationship between GPU based simulation member module and DLL module

the array_view<float,1>dataView(n,&arr[0]) method. Then run the code on the GPU through the function parallel_for_each(), and the GPU completes the complex calculation processing. Finally, the results obtained after running on the GPU are copied from the GPU to the CPU through the dataView.synchronize() function, so that the program can call the GPU computing resources.

4 Simulation Test

A simulation scheme is designed based on HLA simulation system and p-rti simulation server software. There are three simulation models: aircraft, tank and control center. Aircraft, tank and control center each form a simulation member, and record their coordinate information at each simulation time. The starting coordinates of aircraft and tank are (0, 0) and (100, 100), respectively, and the position information is sent to the control center. After receiving the position coordinates sent by other members, the control center calculates and processes the position coordinate data of aircraft and tank by calling DLL. If the distance between the aircraft and the tank is greater than the set distance, the control center sends a message to let the aircraft and the tank advance, and the aircraft and the tank update their position coordinates according to the specific motion formula. In this way, the simulation will continue until the plane and the tank reach a predetermined distance, and the control center will send the firing instruction to the tank to fire the plane, and then the simulation propulsion will end. The simulation member processing flow based on GPU is as Fig. 3.

The scheme will be implemented in two ways. The first way is that all simulation members' computation processing is undertaken by CPU. The second way is CPU + GPU, in which the HLA logic processing of simulation members is undertaken by CPU and the computation part is undertaken by GPU. In the process of simulation member

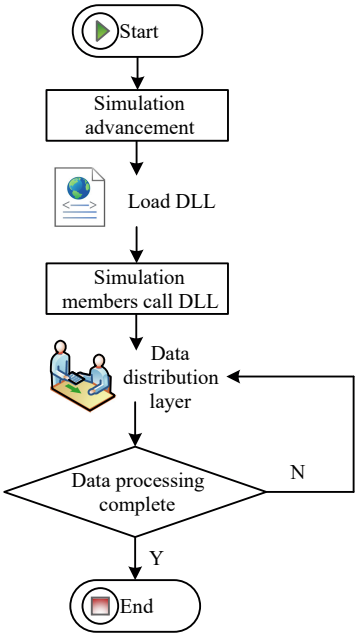

Fig. 3. Flow chart of simulation member processing based on GPU

promotion, the object class continuously sends data to the members who have ordered its object class objects, and receives the data sent by the object class they ordered. In this process, the simulation members need to load the DLL, and realize data processing by calling the DLL many times until the data processing is completed. The simulation steps are set at 1000 steps, 2000 steps and 3000 steps respectively. The simulation running time based on CPU and GPU is compared and analyzed, and the running results are as Fig. 4.

After the HLA Simulation members call the GPU computing resources successfully, they run the simulation cases several times, and all the simulation schemes can be promoted normally, which proves that the HLA Simulation based on GPU is feasible and reliable. Observe the average time of the simulation members when the number of propulsion steps is 1000, 2000 and 3000. It can be seen from the data in Fig. 4 that the GPU takes only about 1/3 of the CPU time when the simulation members have the same number of propulsion steps, which indicates that the HLA Simulation members run faster than the HLA Simulation members running on the CPU after calling the GPU computing resources, which proves that the GPU can optimize the running speed of HLA simulation.

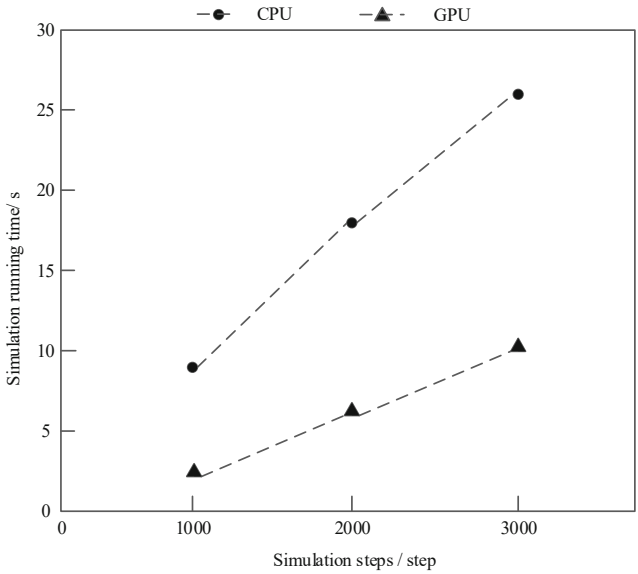

Fig. 4. Comparison analysis results of CPU and GPU based simulation running time

5 Conclusion

This paper aims to optimize the efficiency of HLA Simulation members by using the computing power of GPU. GPU is a special graphics core processor, because of its programming trouble and lack of data tools, it has not been given due attention in the development of HLA based simulation system for a long time. Making full use of GPU's massive data throughput and powerful floating-point computing ability will greatly improve the program performance. This paper designs a federation HLA simulation scheme based on GPU, and constructs a general simulation federation structure based on GPU by analyzing the computing resources of HLA Federation GPU. By allocating the computing part of HLA Simulation members to GPU and using its computing resources to realize floating-point computing processing of simulation data, the time consumption of simulation calculation can be greatly reduced, the running speed of joint members can be accelerated, and the efficiency of simulation development and operation can be improved. The disadvantage of this method is that the design and programming complexity of HLA simulation will be increased, and the development cost will be increased. Therefore, in the following research, we need to optimize the complexity of simulation design and programming, so as to reduce the development costs.

Fund Projects. Hubei Natural Science Foundation guidance project (2019cfc887).

References

1. Sna, B., Fm, A., Am, C., et al.: Reliable identification of the first transition velocity in various bubble columns based on accurate sophisticated methods. Chem. Eng. Res. Des. **165**, 409–425 (2021)

2. Humood, K., Mohammad, B., et al.: On-chip tunable Memristor-based flash-ADC converter for artificial intelligence applications. IET Circ. Devices Syst. **14**, 107–114 (2020)
3. Geneugelijk, K., Spierings, E.: PIRCHE-II: an algorithm to predict indirectly recognizable HLA epitopes in solid organ transplantation. Immunogenetics **72**(8), 119–129 (2019)
4. Zheng, Z., Jza, B., Dong, Y.A., et al.: CFD simulation of fluidized magnetic roasting coupled with random nucleation model. Chem. Eng. Sci. **229**, 116148 (2020)
5. Savaan, H., Duman, L., Din, M., et al.: Migrating a legacy simulation to HLA: lessons learned integrating with new native HLA simulations. In: IEEE Fall SIW 2003. IEEE (2019)
6. Pd, A., Gza, B., Yc, A., et al.: Reasonable permutation of M2e enhances the effect of universal influenza nanovaccine. Int. J. Biol. Macromol. **173**, 244–250 (2021)
7. Herman, A.P., Gan, J., Yu, A.: GPU-based DEM simulation for scale-up of bladed mixers. Powder Technol. **382**, 300–317 (2020)
8. Tsai, M., Tian, Z., Qin, N., et al.: A new open-source GPU-based microscopic Monte Carlo simulation tool for the calculations of DNA damages caused by ionizing radiation — part I: core algorithm and validation. Med. Phys. **47**(4), 1958–1970 (2020)
9. Warren, C., Giannopoulos, A., Gray, A., et al.: A CUDA-based GPU engine for gprMax: open source FDTD electromagnetic simulation software. Comput. Phys. Commun. **237**, 208–218 (2019)
10. Ma, B., Gaens, M., Caldeira, L., et al.: Scatter correction based on GPU-accelerated full Monte Carlo simulation for brain PET/MRI. IEEE Trans. Med. Imaging **39**(1), 140–151 (2019)
11. Liu, S., Liu, D., Srivastava, G., Połap, D., Woźniak, M.: Overview and methods of correlation filter algorithms in object tracking. Complex Intell. Syst. **7**(4), 1895–1917 (2020). https://doi.org/10.1007/s40747-020-00161-4
12. Liu, S., Lu, M., Li, H., et al.: Prediction of gene expression patterns with generalized linear regression model. Front. Genet. **10**, 120 (2019)
13. Liu, S., Li, Z., Zhang, Y., et al.: Introduction of key problems in long-distance learning and training. Mob. Netw. Appl. **24**(1), 1–4 (2019)