PHASE: An Environment for Parallel High-performance Agent-based Simulating

Jiamin Guo¹, Bo Li¹, Yingliang Zhao¹, Yuan Yu¹ and Zuolin Chen²
¹ The School of Electronic and Information Engineering, Xi’an Jiaotong University, Xi’an, Shaanxi, 710049, P.R. China
² People's Liberation Army 63626 troops, China.
*bolec@xjtu.edu.cn

Abstract. Agent-based simulation models have become an increasingly popular research and management tool in many different fields. However, most of the platforms are foreign, and there is no relevant public parallel simulation platform available in China. So it is necessary to construct a public high-performance parallel computing platform for Agent-based Simulation. This paper uses X10 language to design a highly scalable Agent-based simulation platform by using an asynchronous partitioned global address space model, and describes the construction principle and implementation process of the simulation platform from four core modules, the data organization and partitioning mechanism, Agent communication mechanism, simulation promotion mechanism and data collection mechanism. And we verified the correctness and feasibility of the simulation platform by running the classic GameOfLife model. Our work fills the gap that there is no public high-performance parallel simulation platform in China and contributes to the future research work.

1. Introduction
Our goal in this work is to design an Agent-based modeling and simulation platform with high versatility and high parallelism. Our approach is to use the high parallel X10 programming language and the Asynchronous Partitioned Global Address Space (APGAS) model to complete the related design and implementation of the platform. We are not the first to use the X10 language as the language of the development platform (see Japan XASDI [1]), but we believe that there are many unique ideas in our approach, especially in different core parts using different and effective methods.

1.1. ABMS
ABMS (Agent Based Modeling System, ABMS) is a scientific research method, which can be used to study the autonomous agent with interactive behaviors to describe the system abstractly and to study the emergence phenomenon of the system through bottom-up modelling [2,3]. ABMS allows the modeler to specify individual agent behavior rules for each agent, describe the environment in which the individual is located, and then execute to determine possible system results according to rules [4].

1.2. X10
The X10 parallel programming language is a typical implementation of the APGAS model [5]. It can improve programming efficiency by using X10 to write programs, because it does not need to pay too much attention to the heterogeneous details of the underlying computing resources, and so it is also
easy to write and the X10 language can be used to write multi-thread distributed programs suitable for execution in heterogeneous cluster environments.

The core concepts in the X10 language are the Place and the Activity. The Place can be thought of as a virtual shared-memory multi-processor, a unit of computation. And the calculation of the X10 language is done by executing the Activity, which can be considered a lightweight thread.

1.3. APGAS Model
In the process of developing X10 language, IBM first proposed the Asynchronous Partitioned Global Address Space (APGAS) model [6]. The APGAS model extends the implementation framework of PGAS and provides more asynchronous support for the program, making the model not limited to programs that express SPMD types [7].

2. Related Work And Motivation

2.1. Related Work
As for the research of parallel multi-agent simulation and simulation platform, in China, most researchers carry out some simulation experiments through the simulation platform, one part is to improve the performance, and few people research the development of the platform. At present, the foreign popular multi-agent social simulation platforms mainly include: the Repast HPC platform developed jointly by The University of Chicago and ANL based on Repast [8], the FLAME (Flexible Large-scale Agent Modelling Environment) platform jointly developed by Rutherford Appleton Laboratory and the University of Sheffield [9], the SASSY (Scalable Agent-Based Simulation System) platform developed by Maria Hybinette et al [10], XASDI (X10-based Agent Simulation on Distributed Infrastructure) platform based on ZASE developed by IBM Research-Tokyo [1], D-Mason simulation platform based on Mason developed by George Mason University [11], while in China, YU-SUPE platform developed by National University of Defense Technology.

The Repast HPC platform is a parallel extension of the Repast platform, implemented in the C++ language. For the data storage and management of the Agent, the platform uses the two data structures Context and Projection. And the Context is used to store the Agent object, the Projection is the mapping between the Agent and the model environment. For communication, the platform proposes a package communication mechanism, which uses the MPI in the Boost library to call the explicit Send and Receive functions to complete the communication. For the advancement of simulation, it uses the discrete event scheduling strategy.

FLAME is implemented in C language. For data storage and management and communication, the platform uses a Message Board mechanism, that is, the data is designed as a set of message boards, and then the Agent can obtain information and interact by accessing to the message boards. For the advancement of the simulation, the platform uses the execution of the Agent's behavior and the transition of the state to advance.

SASSY can be seen as a middleware between Agent-based API and PDES emulation kernel. For data storage and management, the Agent and simulation environment are divided into several areas, which would be allocated to different processes. For communication, SASSY uses the Interest Management (IM) mechanism to interact. For the advancement of the simulation, the platform uses parallel discrete event technology, and uses event-driven mode combined with optimistic synchronization strategy for orderly simulation.

The XASDI platform is implemented in the X10 language. The platform has established a high-performance multi-agent simulation platform for traffic conditions in Tokyo, Japan. For data storage and management and communication, the X10 language's own mechanism and data structure are implemented, such as DistArray, Block, at(P) S, Team toolkit, etc. For the advancement of the simulation, global clock control is implemented through Place0 node, and the simulation is advanced in parallel with the X10 communication primitive.

The D-MASON platform is a distributed version of MASON. For data partitioning and storage,
before the simulation starts, the simulation environment is divided into several areas. For communication, D-MASON uses a Publish-Subscribe mechanism. Who subscribes to my message, who will I send a message to. For the advancement of the simulation, the platform considers ABM as a step-by-step calculation, which is to calculate Agent behaviors in successive simulation steps. While in each simulation step, it is divided into three phases: communication, synchronization and simulation.

2.2. Motivation
First of all, through the above analysis, the Repast HPC platform uses a tight synchronization time management mechanism, and this mechanism does not further optimize the synchronization mode, and the parallel simulation execution performance is not fully demonstrated. The topology of the Agents in the FLAME platform is not clear enough in the simulation modeling process, which makes the simulation efficiency not high enough [12]. SASSY is mainly suitable for local interaction intensive models, and so for the frequent interactions models between nodes it obviously will have communication bottlenecks. The XASDI platform is mainly used for traffic simulation, and its applicability is not extensive enough. The data division in D-MASON is static, and load balancing cannot be guaranteed during the simulation process. YU-SUPE is a platform for internal research projects in schools in China. It is not public, and the platform is not very versatile and easy to use.

Secondly, when many non-computer people are working on Agent modeling in his fields, if there is no theoretical basis, they have to implement the modeling process of the entire project from start to finish, which is time-consuming and labor-intensive. Although the model environment may not be the exactly same in different fields, it is similar or even repetitive in some basic simulation coding work. Therefore, if there is a general parallel agent simulation platform, the simulation modeling cycle in different fields can be greatly reduced, thereby the costs will be saved.

In order to solve the above problems comprehensively, this paper mainly combines the X10 high parallel programing language and the discrete event-based simulation advancement mechanism to develop the Parallel High-performance Agent-based Simulating Environment (PHASE) with APGAS model.

3. PHASE Platform Design And Implementation
The PHASE platform designed and developed in this paper consists of four core parts—data organization and partition mechanism, Agent communication mechanism, parallel simulation advance mechanism, and data collection mechanism.

3.1. Data Organization And Partition Mechanism
Agent data organization and partition generally have two ways: one is based on environmental, and the other is based on Agent. And Agent environments can generally be divided into four types: Grids, Networks, Irregular Polygons, and Geographic Information Systems (GIS). In this paper, the data organization model combines two division methods, which is to use environment-based partitioning and building through Grids at the beginning of simulation, and to use Agent-based partitioning in the simulation process. The reasons are as follows:

- Grids is a two-dimensional spatial topology with a simple structure and can be seemed as a checkerboard and be suitable for most models. Figure 1 shows two partitioning ways based on grid.

- We can obtain Agent and other operations according to unique categories or two-dimensional coordinates. It is not necessary to assign each agent a unique identification number ID like other simulation platforms (such as ReapstHPC). At the same time, various entities in the environment are also abstracted into different types of Agent, which is quite scalable by supporting the custom functions in the Agent and the environment.

- Store all agents using a distributed array (DistArray). Since each agent is distinguished from other agents by category and grid coordinates. In some models consisted of multi types of
Agent, each type of agent can be stored using the DistArray data structure in X10. When the program runs, all data in the DistArray is separated into several blocks by the largest average, and then these blocks are assigned to each Place in some way, which ensures that the static balance of the load is partitioned before the simulation.

- We use the Agent-based partitioning method in the simulation process. However, this method is not global, but a local load migration. For example, in the simulation process, if some nodes were overloaded, the system will move from the overload node to the underload node. The migration of the agent is performed, so that the load of each computing node is roughly in a dynamic balance state, and the parallel performance of the computer can be fully utilized.

![Figure 1. Grid-based band and block partitioning in the GameOfLife model](image)

In addition, we have also tried some other ways of partitioning. Such as, Cluster based partitioning, that is, we divided the distributed-intensive Agents into the same Place. See [13].

### 3.2. Agent Communication Mechanism

The Agent communication mechanism adopts a combination of global communication and local communication.

- Global communication. The basic operation `at ()` in the X10 allows direct access from one compute node to another compute node. For example, execute `at (place(2)) s` in Place1; this statement indicates that the Activity executed in the current Place1 will pause and execute the S statement in Place2. Therefore, it is easy to complete the operation of mutual access operation in multiple Places. However, the time overhead of such operations is too large because these calculations belong to communication operations across compute nodes.

- Local communication. If all communication modes use global communication, it will inevitably lead to frequent cross-Place communication, and the computational overhead will be extremely large, which will affect the iterative speed of the simulation system. In order to reduce the computational overhead, the locality principle is used for redundancy calculation [14], and the number of global communication is reduced by storing the nonlocal data into the local (current computing node) redundant area in advance, which also reduces the correlation between the computing nodes. So that the calculation no longer relies on frequent global communication to obtain nonlocal data, and then can be executed concurrently to the maximum extent.

As shown in Figure 2, the dark part is the redundant area, the light part is the local sub-model data block, and a number of redundant areas are added on the outside of the sub-model data block in each place for storing. When updating their state, the Agents can obtain the nonlocal data in local area.
Figure 2. Concept of the redundant area

As shown in Figure 2, the dark part is the redundant area, the light part is the local sub-model data block, and a number of redundant areas are added on the outside of the sub-model data block in each place for storing. When updating their state, the Agents can obtain the nonlocal data in local area.

Figure 3. Redundant area

In addition, in order to further improve communication efficiency, we have also made various optimizations for communication methods, such as optimization of redundant area layers, optimization of minimum traffic, and optimization of communication forms. As shown in Figure 3, the optimization of the number of redundant areas means the selection of the value of $R$. The minimum traffic
optimization means that when updating the status information of the agents in each Place, the Agents in the work area does not need to be acquired from Place0, that is, it does not need to acquire the all data, and only need to obtain the overlap area data from Place0, thus reducing Inter-node communication from \((B + 2R) \times (B + 2R)\) to \(((B + 2R) \times (B + 2R) - B \times B)\). As shown in Figure 4, the optimization of communication form means that each Place gets data from Place0, which will increase the communication load pressure of Place0. In order to reduce the pressure, data can be exchanged directly between the Places to update local information.

3.3. Parallel Simulation Advancement Mechanism

We design the discrete event advancement mechanism based on conservative strategies similar to the SASSY. As shown in Figure 5, the global controller in Place0 controls the global simulation time. Each Place also has its own local simulation time. The synchronization of the Place is adjusted by the global controller in Place0. Each Place has its own local event table and scheduler for storing and managing local events and determining the execution time and order of local events. The global controller in Place0 selects event with the global timestamp minimum from all Places. When the simulation starts, the scheduler in each Place continuously and cyclically selects event with the local timestamp minimum event from the local event table, then the global controller selects the event with the global timestamp minimum and then executes the event. And the simulation will end until executing the end event. In addition, we combine \texttt{finish, clock}, \texttt{when} and other primitives in the X10 language to achieve message synchronization among Places.

![Figure 5. Schematic diagram of Simulation advancement mechanism](image)

3.4. Data Collection Mechanism

Data collection in Parallel Agent simulation is timely and completely storage and management of the data generated during the parallel agent simulation process, so as to provide the in-depth relevant details for the simulation participants in the simulation or after the simulation, which is an extremely important step to understand the simulation process. This platform uses “Decentralized” collection at any location mechanism. It is to define a class (DataCollection), and then the user can directly call the corresponding function in its model code. As a result, the relevant data will be captured and stored in the data collecting file.
4. Experimental Environment And Results Analysis

4.1. Experimental Environment

The hardware and software environment are as shown in table 1 and table 2.

| Table 1. Hardware environment |
|-------------------------------|
| Hardware | hpms-1 | hpms-2 |
| Architecture | X86_64 | X86_64 |
| CPU(s) | 16 | 8 |
| On-line CPU(s) list | 0-15 | 0-7 |
| Number of threads per core | 2 | 1 |
| Number of cores per socket | 4 | 8 |
| Socket(s) | 2 | 1 |
| NUMA node | 2 | 2 |
| CPU MHz | 2200.000 | 800.000 |
| L1d cache | 16K | 64K |
| L1i cache | 64K | 64K |
| L2 cache | 2048K | 512K |
| L3 cache | 6144K | 5118K |
| NUMA node0 CPU(s) | 0-7 | 0-3 |
| NUMA node1 CPU(s) | 8-15 | 4-7 |
| Company ID | AuthenticAMD | AuthenticAMD |

| Table 2. Software environment |
|-------------------------------|
| Software | version |
| Linux system | Ubuntu-14.04.4 |
| MPI | Mpich-3.2 |
| SSH | 3.6 |
| JDK | Jdk-8u111-linux-x64 |
| ANT | Apache-ant-1.9.7 |
| X10dt | X10-2.6 |

4.2. GameOfLife Model Experiment And Result Analysis

In order to verify the correctness and feasibility of the PHASE simulation platform, we simulated the classic model of ABMS-GameOfLife model on the PHASE. The relative steps for adding the GameOfLife model to the parallel simulation platform are as follows:

- **Agent.** Add all the attributes of the entity in the Agent.x10 file. The existing two-dimensional coordinate position and status information are available. Others can be added after the comment “AddCode”;
- **Grid.** The grid parameters are set in the Grid.x10 file. In this example, the grid is initially set
8

to 360 in both landscape and portrait orientation.

- **Event.** The simulation event is designed in the EventSequence.x10 file. In this example, we set four events, which are game initialization (init()), Agent neighbor information acquisition (AgentNeighborState()), and game evolution according to rules (Interactive()), Agent information display (Display()). And at the same time they should be written to the corresponding step according to the scheduling order.

- **Model.** Event scheduling is designed in the Model.x10 file to run the specified event at the specified time. It should be noted that an end event is designed to stop the simulation.

- **DataCollection.** Data collection in the DataCollection.x10 file can occur in any given event or at any time. If the latter, you need to customize the data collection event.

- **Run.** Run main function and count the simulation time in the Run.x10 file. And configure the simulation properties in the Setting.x10 file. In this example, the initial number of agents, the iteration step, and the directory where the data collection files are stored.

In this simulation experiment, according to the model, we added some methods such as location acquisition, state acquisition, and neighbor acquisition. In addition, for each type of Agent that needs to communicate across the Place, a redundant area is designed in every Place by using the DistArray data structure, and the nonlocal data is required in advance into the local boundary buffer by the asynchronous operation (async()) before the calculation, thereby Communication could be hided. In addition, the model initial parameters set are as follows: The environment scale is a 360 × 360 grid environment, the number of compute node (the Place) is 2, the initial number of Agent is 20000, and the total steps is 200. The number of alive Agent changes as shown in Figure 5.

![Figure 5](image)

**Figure 5.** The change of the number of Agents in the simulation

As shown in the Figure 6, the number of alive Agent decreases rapidly at the simulating initial time. This is because the Agent is randomly distributed, and then according to the simulation rules, many Agents could die, and a few steps later, the changing speed of alive Agents number will gradually slow down, and finally maintain a certain number of ups and downs in a dynamic balance. The conclusion of this experiment is a good description of the simulation process of the GameOfLife model. Also the number of death and rebirth of the Agent in each step both can be viewed in the logging file of collecting data.

5. **Summary And Future Work**

This paper describes the design principle and implementation process of the simulation platform from four core modules: data organization and division mechanism, Agent communication mechanism, parallel simulation advance mechanism and data collection mechanism, and verify the correctness and feasibility of the basic functions of the platform by simulating the classic model of GameOfLife on the
platform. The related design and experiment details of the platform has been publicly released on the Github platform (Github: https://github.com/hpmsxjtu/PAGE) for studying and using by relevant scientific researchers. And at the same time, it fills the gap that there is no public high-performance parallel simulation platform in China. Moreover, the platform is more versatile, not just for traffic simulation.

The PHASE simulation platform makes full use of the advantages of APGAS and has an ideal adaptability to models based on grid space. However, it is a topic worthy of further studying that how to extend the platform from the grid structure to the other topology structures such as network and geographic information system. In addition, the PHASE platform still has several problems to be solved in practical applications, such as dynamic load balancing and further communication optimization. At present, we have integrated the simulation advance mechanism and data collection mechanism into a toolkit for the other simulators to use, and the related data organization and division mechanism and communication mechanism toolkit have just been completed initially. And the basic functions of the platform have been realized and there are many extra functions that need to be further refined, and it is expected to form a more completed high-performance Agent simulation platform in the future.

Reference

[1] Suzumura T, Kato S, Imamichi T, et al. X10-based massive parallel large-scale traffic flow simulation[C]// Acm Sigplan X10 Workshop. 2012.
[2] Klügl F, Bazzan AL. Agent-based modeling and simulation[J]. AI Magazine, 2012, 33 (3): 29-41.
[3] North M J , Macal C M . Managing Business Complexity: Discovering Strategic Solutions With Agent-Based Modeling and Simulation[J]. Oxford University Press, 2007, 35(6):1120-1122.
[4] Dubitzky W , Kurowski K , Schott B . Repast HPC: A Platform for Large-Scale Agent-Based Modeling[C]// Wiley-IEEE Press, 2012.
[5] Charles P , Grothoff C , Saraswat V A , et al. X10: an object-oriented approach to non-uniform cluster computing.[C]// Acm Sigplan Conf on Object-oriented Programming. ACM, 2005.
[6] Yelick K, Dan B, Chen W Y, et al. Productivity and performance using partitioned global address space languages[C]// Parallel Symbolic Computation, Pasco, International Workshop, July, University of Western Ontario, London, Ontario, Canada. 2007.
[7] Hao Zhijun. Research on Cluster Resource Management and Optimization Technology for Asynchronous Partitioned Address Space Programming Model [D]. 2014.
[8] Collier N, North M. Parallel Agent-based simulation with Repast for High Performance Computing[J]. Simulation, 2013, 89 (10): 1215-1235.
[9] Kiran M , Richmond P , Holcombe M , et al. FLAME: simulating large populations of Agents on parallel hardware architectures[C]// International Conf on Autonomous Agents & MultiAgent Systems: Volume. International Foundation for Autonomous Agents and MultiAgent Systems, 2010.
[10] Hybinette M , Kraemer E , Xiong Y , et al. SASSY: A Design for a Scalable Agent-Based Simulation System using a Distributed Discrete Event Infrastructure[C]// Proceedings of the Winter Simulation Conf WSC 2006, Monterey, California, USA, December 3-6, 2006. DLBP, 2006.
[11] Cordasco G , Chiara R D , Mancuso A , et al. A Framework for Distributing Agent-Based Simulations[C]// International Conf on Parallel Processing. Springer-Verlag, 2012.
[12] Dubitzky W, Kurowski K, Schott B. Large-Scale Computing Techniques for Complex System Simulations[M]: John Wiley & Sons, 2012.
[13] Wang Y, Lees M, Cai W, et al. Cluster based partitioning for agent-based crowd simulations[C]. Winter Simulation Conf, 2009: 1047-1058.
[14] Chen Li, Zhang Zhaqing, Feng Xiao-bing. Redundant computational segmentation of optimized communication in distributed storage systems[J]. Journal of Computer, 2003, 26(2).