The software structure for agent-oriented simulation with distributed dispatching

E V Melnik¹, A Y Ostroukhov ², I S Pukha ², A A Rodina¹

¹ Federal Research Centre, Southern Scientific Centre of Russian Academy of Sciences, 41, Chekhov Street, 344006, Rostov-on-Don, Russian Federation
² Scientific research institute of multiprocessor computer systems of Southern Federal University, 2, Chekhov Street, 347928, Taganrog, Russian Federation

E-mail: evm17@mail.ru

Abstract. This paper deals with the problem of the implementation of the agent-oriented simulation on high-dimensional models in the social science domain. It was shown that high dimensionality of models and the complexity of their development using the existing tools were the main constraint factors. The analysis of existing agent-oriented simulation systems was held. Some special aspects of platforms, on which such systems were performed, were considered. The influence of clustering on the simulation rate was shown. The structure of the software for the agent-oriented simulation using the existing agent-oriented simulation systems on the basis of the distributed scheduler, which provides clustering optimization of a model problem, was proposed.

1. Introduction

At the present time there are a great number of investigations in the social science domain due to the growth of adversarial social phenomena [1, 2]. Nowadays the most advanced analysis tool, which is applied in the social science domain, is simulation on the basis of an agent-oriented approach [3]. In works [4-6] the results of the above mentioned method for analysis of large social groups were shown.

The successful application of the agent-oriented simulation in the social science domain takes place due to the fact that the method closely corresponds to the domain field.

It should be emphasized that there are some peculiarities for the agent-based simulation in the social science domain:

- large agent population (hundreds of thousands of agents),
- complex agent behavior,
- dividing the agent by groups/types,
- using the high-efficiency cluster solution.

2. The analysis of agent-oriented simulation software

There are some key problems of the agent-oriented simulation application for real tasks:

- inadequate agent models due to the lack of domain data;
- inadequate tools [8-10], which requair some special user’s knowledge, which does not link with a domain.

In [9] the latest agent-oriented simulation, systems (AOSS) were presented. The results of
comparative analysis of AOSSs in terms of “model development complexity - scalability” are presented in Table 1.

**Table 1.** The AOSS comparison in terms of “model development complexity - scalability”.

| Complexity          | Simply/ Easily | Moderately | Hard / Complicatedly |
|---------------------|----------------|------------|----------------------|
| Extreme scale       | Repast HPC, MATSIM, PDES-MAS, Swarm |
| Large scale         | FLAME, LSD (2D/3D), MASS, Pandora, UrbanSim, AnyLogic (2D/3D), AOR Simulation, CloudSim, CybelePro, D-OMAR, Echo, Ecolab, FLAME GPU (3D), GridABM, HLA_Agent, HLA_Repast, Repast-J, Repast Symphony (2D/3D), Brahms, BSim(2d/3D), |

We note that the majority of AOSS, which are oriented to performe scaled models, in order to achieve a required efficiency, keep an operation on clusters and have optimized implementations (for example, Flame-HPC, Repast-HPC).

Due to enhancement of the efficiency of graphics processors as compared to general-purpose processors [11] the problem of AOSS implementation on GPU has become currently important.

AOSSs based on FLAME-GPU, which are aimed at using the NVidia made graphics accelerator, was developed in [12]. The above mentioned AOSS performs modeling of scaled systems without an accessibility to clusters.

However, in the context of GPU modeling, it should be taken into account that, since a particular limit, GPU efficiency begins to reduce the population growth much faster than in cases when clusters are used. Though, FLAME-GPU can be considered as the necessary alternative to cluster solutions in case of the out-of-the-box solution, when the population size is not extrimal and GPU has certain efficiency.

To sum up, it can be said that FLAME, FLAME-HPC, Repast-HPC, FLAM-GPU are optimal solutions among the open AOSSs in the social science domain, which can be described in terms of large models and complex agent behavior.

3. The analysis of implementation platforms for AOSSs

On the basis of AOSS analysis, it can be concluded that AOSS efficiency depends on the agent model which is used. As a result, the choice between the cluster and GPU solutions is rather problematic, particularly, at the initial stages.

The AOSS comparison on the basis of the implementation platform is presented in Table 2.

**Table 2.** The AOSS comparison on the basis of implementation platform.

| Cluster implementation platform | GPU implementation platform |
|---------------------------------|------------------------------|
| Pros | Cons | Pros | Cons |
| no scalability limits | high cost | high efficiency | scalability limits |
| high efficiency | fast data transmission | low cost | agent model restriction due to GPU memory size |
| | the problem of model location in the cluster | simple deployment and usage | |
| | complex deployment and usage | visualizing results | |
We can observe in Table 2 that GPU-based solutions are much better in terms of the cost/performance ratio. At the same time, model size restrictions do not allow us to use GPU-based solutions for extremely large size models. To some extent it can be negated by means of several GPUs in a workstation computer. However, it requires buying specialised GPUs and a mother card. But this approach does not solve a scaling problem. It just increases a limit when AOSS simulation performance begins to reduce. It follows that a combined solution, which involves clusters of workstations and GPU, can be considered as an optimal. Herewith, the number of workstations can be increased according to scaling demands. In this case, the problem of communication between GPUs both intra-cluster and intra-workstations, is meant to be solved. To deal with this problem NVIDIA company developed NCCL - NVIDIA Collective Communications Library, which provides parallel computation primitives on multi-GPU and multi-node environment [13].

Cluster solutions use the powerful interfaces of date transmission both between cluster nodes and cluster interconnects. For example, NCCL provides date transmission between nodes by means of InfiniBand verbs, libfabric, RoCE and IP Socket.

Another way to solve a scaling problem involves middleware using. It provides model code cross-compilation. This approach allows us to use some AOSSs of various types with various implementation platforms. The similar approach using of OpenABL is presented in [14].

It must be mentioned that the foregoing approach allows us to solve the problem of high complexity model development. As it shown in [9], highly-efficient AOSSs are characterized by high complexity model development (Table 1). Herewith, for this purpose a general-purpose programming language is used. If model often changes during the modeling process, it is necessary to attract programmers to solve some current problems. As a result, the bottle neck of modeling is the rate of model change by the programmers for needs of domain experts instead of the execution speed of a model. There is some information about the cost of model development in eLOC for various AOSSs in [14]. We can see that FLAME GPU demonstrates the cost of development, which can be compared with ABL cost. It comes from the fact, that a pattern system is used for model development [11, 15].

Another way to simplify the process of model development is middleware usage. It represents a graphical interface for model development. In this case, original model code is generated automatically on the basis of input data. This approach can be easily implemented for AOSS, which uses the various configuration files for model description (for instance, FLAME, FLAME GPU). The step of code generation should perform on a separate module, as in [14], in order to increase its functionality (for example, various AOSS supporting) regardless of middleware.

Secondly, factors, which have an impact on model execution speed, should also be considered. A scheduler, which executes an agent replacement, should be put in the cluster system. The scheduler is responsible for monitoring of the state of the cluster and carries out agent replacement when computational node workload exceeds the threshold or heavy network load takes place. In order to provide performance and scalability of the system, the scheduler should be distributed. In [16,17] the description of distributed scheduler usage in the context of failover problem solving in distributed computing environment was presented.

4. The assessment of factors having an impact on the simulation rate

It must be mentioned that there are some factors having an impact on the simulation rate:
- cluster node performance,
- speed of data transmission interface between the cluster nodes,
- intensity of communications between the cluster nodes,
- complex agent behavior.
There are some methods to speed up the simulation rate:
- cluster node performance enhancement,
- high-speed network interface,
- effective agent distribution.
The first two methods can be used if there is an opportunity to choose a cluster or to change its configuration. In the context of a given cluster these methods cannot be used.

In this work we will dwell upon the last method. For this purpose, we should assess in what way the ineffective distribution of agent groups influences the simulation rate.

Let the intensity of communications be from $N = \sum_{i=1}^{M} N_i$ agents, which are split into $M$ groups:

$$\sum_{i=1}^{M} N_iS_i = \sum_{i=1}^{M} N_i(S_i^{in} + S_i^{out}) = \sum_{i=1}^{M} N_iS_i^{in} + \sum_{i=1}^{M} N_iS_i^{out}$$  \hspace{1cm} (1)

where $S_i^{in}$ - the intensity of communications among the group;

$S_i^{out}$ - the intensity of communications between agents from other groups.

A network society has some characteristics. First of all, it is a large agent population. Second, internal communications exceed the external ones $S_i^{in} >> S_i^{out}$. In the context of the ineffective group distribution it leads to considerable traffic increasing between the cluster nodes:

$$\sum_{i=1}^{M} N_iS_i^{in} >> \sum_{i=1}^{M} N_iS_i^{out}.$$  \hspace{1cm} (2)

Let us assess the volume of a single node transmission considering the following assumptions:
- an agent participates only in one group,
- each group is fully located on one node.

Let agents be involved in groups of equal size $N_i$, which are equally distributed on computational nodes ($k$ groups per node).

Let $l$ be all nodes, and then the total number of agents is described by the following equation:

$$N = k l N_i.$$  \hspace{1cm} (3)

We assume that agents communicate with each other with intensity $S_i^{in}$ within a cluster and $S_i^{out}$ - between clusters. Considering the fact that each cluster is located on one computational node, information exchange within a cluster does not lead to occurrence of traffic in the network. From among $S_i^{out}$ communications a part of them $(k-1)N_i$ is transmitted to agents of a cluster, which are located on the same node as an agent-transmitter is. It also does not lead to occurrence of traffic in the network. The number of communications which are transmitted by each agent of any node to agents of other nodes is equal to $S_i^{out}(N-kN_i)/N$. And for all agents it is equal to $kN_iS_i^{out}(N-kN_i)/N$. Herewith, the agents of a node will receive communications from agents of each other nodes:

$$kN_iS_i^{out}(N-kN_i)/(l-1)N.$$  \hspace{1cm} (4)

Consequently, a generic traffic of a node is calculated using the following formula:

$$T = 2kN_iS_i^{out}(N-kN_i)/(l-1)N = 2kN_iS_i^{out}(1-kN_i/klN_i) = 2kN_iS_i^{out}(1-1/l).$$  \hspace{1cm} (5)

From the given expression we can see that the volume of data transmission grows in a linear direction when:
- increasing the group size,
- increasing the number of clusters on the node,
- increasing the transmitted communications.

It must be mentioned that:
- increasing the number of nodes increases the traffic limitedly ($2kN_iS_i^{out}$ - maximum),
- the number of nodes cannot be greater than the number of clusters $l_{max} = N/N_i$.

In view of the foregoing the following conclusions can be made:
- increasing the number of agents on one of the nodes will lead to:
  1. increasing the computation time;
2. increasing the data transmission time.
   - synchronization of agent performance within a simulation cycle in case of different computation and data transmission time leads to reduction in the computation system effectiveness and the simulation time increases respectively.

The obtained expressions allow evaluating the possibility of simulation, depending on model parameters and communication interface.

In figure 1 a “node traffic – number of agents per node” diagram is presented. It allows evaluating the possibility of simulation, depending on model size and equipment configuration – number of nodes and communication interface parameters.

![Figure 1. “Node traffic - number of agents per node” diagram.](image1)

In figure 2 there are diagrams, which allow one to evaluate the possibility of simulation in terms of intensity of communications between agents.

![Figure 2. “Intensity of agent communications - number of agents per node” diagram.](image2)
It is worth noting that, in case when an agent can be involved in more than one group simultaneously, these groups should be located on one computational node. If it is impossible to implement such distribution, then it is necessary to reduce the number of agents included in groups, which are located on different computational nodes.

5. The software structure for agent-oriented simulation

To provide clustering controlling during the simulation, the software structure for agent-oriented simulation is presented in figure 3.

![General software structure for agent-oriented simulation](image)

Figure 3. General software structure for agent-oriented simulation.

The system involves a control node and execution nodes.

The control node is a coordinator of the whole system. AOSS is not performed on it. A scheduler on the control node executes only communication between schedulers from other nodes.

The following components are deployed on the control node:
- model development module, which provides domain-specific tools for model development and setting;
- visualization and analysis module, which provides simulation result representation and analysis tools;
- control module, which provides simulation control (launch, stop and so forth);
- model generator, which creates configuration files on the basis of data from model development module that are necessary for AOSS performance;
- scheduler, which provides monitoring of the state of the model implementation platform using data exchange with other schedulers and performs AOSS configuration data transmission to other nodes.

AOSS is launched on the execution nodes. The scheduler provides the AOSS configuration and control, and is performed on the same node.

The system operates in the following manner (we assume that a model has been developed and the initial distribution has been prepared):
1. Schedulers are launched on each node.
2. Schedulers exchange data about their own nodes condition, coordinating the approval of their conceptualization about the system.
3. When synchronizing is reached, a scheduler of control node transfers data, which needed for simulation (configuration files, initial distribution files, etc)
4. After having received needed data, schedulers perform a resynchronizing. When the resynchronizing is successfully completed, schedulers run simulation.
5. During the simulation, schedulers monitor the state of the environment: nodes operability, nodes processor utilization, data transmission channel loading, etc.
6. In case of abnormal situations (node failure, loading threshold exceeding of a computing resource) schedulers break simulation.
7. On the basis of the last completed iteration schedulers create a new initial distribution and perform a synchronizing of a new system state.
8. After system synchronizing the simulation is run.
9. If it is necessary, 7-8 steps are repeated until simulation is completed.

In the event that there is no need to use a cluster, the system can be deployed on a single node. In this case the scheduler also provides control functions.

It must be mentioned that when we use a single node configuration, the scheduler is not necessary. A software structure can be simplified. In this case AOSS access to configuration data should be provided and control functions should be put in the control module.

6. Conclusion
The problem of agent-oriented simulation in social science domain was examined in this work. It was shown that the main difficulties in this field are high dimensionality of models and high complexity of model development due to shortcomings of existing modeling tools.

The analysis of existing AOSSs was held and some peculiarities of platforms, on which AOSS is launched, were examined. Some approaches to model scaling and simplification of model development were considered.

The authors proposed to use a GPU-based cluster as a platform for agent-oriented simulation. The assessment of factors, which have an impact on the simulation rate, was held. The evaluation of simulation possibility, depending of model parameters and implementation platform, was obtained.

The software structure and software operation algorithm for agent-oriented simulation using existing AOSSs on the basis of the distributed scheduler was proposed in this work. This solution provides the optimization of simulation problem clustering.

7. Acknowledgments
The reported study was funded by the RFBR project, 18-29-22046mk.

References
[1] Minaev V A, Ovchinskij A S, Skryl’ S V, Trostynskij S N 2012 How to control the collective consciousness: modern models (Moscow: New Russian University)
[2] Latynov V V 2013 The psychology of communicative effect (Moscow: Institute of Psychology of Russian Academy of Sciences)
[3] Gilbert G N, Troitzsch K G 2005 Simulation for the social scientist (New York, NY: Open University Press)
[4] Deissenberg C, Hoog S van der, Dawid H 2008 EURACE 204 (2) 541–552
[5] Okrepilov V V 2015 Economy of Region 301–313
[6] Makarov V L, Bahtizin A R, Sushko E D, Vasenin V A, Borisov V A, Roganov V A 2016 Herald of the Russian Academy of Sciences 86(3) 252-262
[7] Makoveyev V. 2016 Economic and social changes: facts, trends, forecast 5 (47) 272-289
[8] Pavón J, Sansores C, Gómez-Sanz J 2006 Proc. of the 4th European Workshop on Multi-Agent Systems EUMAS’06 (Lisbon, Portugal) p 11
[9] Abar S, Theodoropoulos G K, Lemarinier P, O’Hare G M P 2017 Computer Science Review 24 13–33
[10] Railsback S F, Lytinen S L, Jackson S K 2006 Simulation 82 (9) 609–623
[11] Richmond P, Walker D, Coakley S, Romano D 2010 Briefings in Bioinformatics 11(3) 334–347
[12] Flexible Large Scale Agent Modelling Environment for the GPU (FLAMEGPU) Retrieved from: http://www.flamegpu.com
[13] NVIDIA Collective Communications Library (NCCL) Retrieved from: https://developer.nvidia.com/nccl.
[14] Cosenza B, Popov N, Juurlink B, Richmond P, Chimeh M K, Spagnuolo C, Cordasco G, Scarano V 2018 Euro-Par 2018: Parallel Processing ed. M Aldinucci, L Padovani, M Torquati vol 11014 (Cham: Springer International Publishing) pp 505–518
[15] Richmond P, Chimeh M K 2017 Proc. International Conference on High Performance Computing & Simulation (Genoa) (Italy: IEEE) p 11-17
[16] Melnik E, Klimenko A 2018 Agent-Based Approach to Distributed Information and Control System Reconfiguration Proc. of the Computational Methods in Systems and Software vol 661, ed R Silhavy R, P Silhavy, Z Prokopova (Springer: Cham) pp 196-205
[17] Melnik E V, Klimenko A B 2016 Informational and control system configuration generation problem with load-balancing optimization Proc. IEEE 10th Int. Conf. Appl. Inf. Commun. Technol. (NY: IEEE) p 1-5