Application of Supercomputer Technologies in Agent - Based Models

Valery L. Makarov[1], Albert R. Bakhtizin[2], Nafisa V. Bakhtizina[3]

Central Economics and Mathematics Institute of Russian Academy of Sciences
Moscow, Russian Federation

[1]makarov@cemi.rssi.ru, [2]albert.bakhtizin@gmail.com, [3]bnvlad@yandex.ru

Abstract. This work contains a brief excursus on the application of supercomputer technologies in social sciences, first of all, – in part of the technical implementation of large-scale agent-based models. So, in this article we will consider the experience of scientists and practical experts in the launch of agent-based models on supercomputers, as well as on the example of an agent model developed by us of the social system in Russia we will analyze the stages and methods of effective projection of a computable core of a multi-agent system on the architecture of the acting supercomputer.

Keywords: agent-based models, parallel calculations, supercomputer technologies

1 Introduction

Computer modeling is the broadest, most interesting and intensely developing area of research and is in demand today in many spheres of human activity. The agent-based approach to modeling is universal and convenient for practical researchers and experts because of its benefits. These models allow modeling a system very close to reality. The emergence of agent-based models may be regarded as a result of the evolution of modeling methodology: a change from mono models (one model – one algorithm) to multi-models (one model – a set of independent algorithms). At the same time sets high requirements for computing resources. It's obvious, that for direct modeling of sufficiently long-term social processes in the country and the planet as a whole, significant computing capacities are necessary.

Supercomputers allow to enormously increase the number of agents and other quantitative characteristics (network nodes, the size of territory) of models, originally developed for use on ordinary desktop computers. For this reason, supercomputer modeling is a logical and desirable step for those simplified models, which have already passed practical approbation on conventional computers. Unfortunately, the specific architecture of modern computers does not guarantee that the software of a computer model will immediately work on a supercomputer. At least the paralleling

of the computable core, and often its deep optimization is required, because otherwise the use of expensive supercomputer calculation will most likely will not pay off.

2 Experience of Some Scientists and Practical Experts

2.1 Well-known Examples of Launching Agent-Based Models on Supercomputers

In September of 2006 a project on the development of a large-scale ABM (agent-based model) of the European economy — EURACE, i.e. Europe ACE (Agent-based Computational Economics) was launched, with a very large number of autonomous agents, interacting within the socio - economic system (Deissenberg, van der Hoog, Herbert, 2008). Economists and programmers from eight research centers in Italy, France, Germany, Great Britain and Turkey are involved in the project, as well as an advisor from Columbia University, USA, -Nobel laureate Joseph Stiglitz.

According to the developers, virtually all existing ABMs either cover only a single industry or a relatively restricted geographical area and accordingly, small populations of agents, while the EURACE presents the entire European Union, so the scope and complexity of this model is unique, and its numerical computation requires the use of supercomputers as well as special software.

The information about 268 regions in 27 countries was used to fill the model with necessary data, including some geo-information maps.

In the model there are three types of agents: households (up to 10^7), enterprises (up to 10^5) and banks (up to 10^2). They all have a geographical reference, and are also linked to each other through social networks, business relationships, etc.

EURACE was implemented using a flexible scalable environment for simulating agent-based models - FLAME (Flexible Large-scale Agent Modeling Environment), developed by Simon Coakley and Mike Holcombe[1] initially to simulate the growth of cells under different conditions (Coakley, Smallwood, Holcombe, 2006). With the help of the developed model several experiments were conducted in order to study the labor market. Without going into detail about the obtained numerical results, we note that, according to the authors, the main conclusion of the research is that the macroeconomic measures of two regions with similar conditions (resources, economic development, etc.) during a long period (10 years and more) may vary significantly, due to an initial heterogeneity of the agents[2].

In ABM EpiSims, developed by researchers from the Virginia Institute of bioinformatics (Virginia Bioinformatics Institute), the movement of agents is studied as well as their contacts within an environment as close as possible to reality and containing roads, buildings and other infrastructure objects (Roberts, Simoni, Eubank, 2007). To develop this model a large array of data was necessary, including information about the health of individual people, their age, income, ethnicity, etc.

[1] For more thorough information see www.flame.ac.uk

[2] More thorough information can be found on the website of the project: www.eurace.org.

The original goal of the research was to construct an ABM of high dimension (according to the number of agents) to be launched on the supercomputer, which could be used to study the spreading of diseases in society (Roberts, 2012). However, afterwards, in the course of work, another task was also being resolved, regarding the creation of specialized ABM++ software, which allows to carry out the development of ABM in the C++ language, and also contains functions, facilitating the allocation of the program code in use among the cluster nodes of the supercomputer. Apart from that, ABM++ provides for the possibility of dynamic redistribution of currents of calculations, as well as the synchronization of on-going events.

ABM++, the first version of which appeared in 2009, is the result of the modernization of the instrument, developed in 1990-2005 in the Los Alamos National Laboratory during the process of constructing large-scale ABMs (EpiSims, TRANSIMS, MobiCom).

Specialists of another research team from the same Bioinformatics Institute of Virginia created an instrument for the study of the particularities of the spreading of infectious diseases within various groups of society — EpiFast, among the assets of which is the scalability and high speed of performance. For example, the simulation of social activity of the population of Greater Los Angeles Area (agglomerations with a population of over 17 million people) with 900 million connections between people on a cluster with 96 dual- core processors POWER5 took less than five minutes. Such fairly high productivity is provided by the original mechanism of paralleling presented by the authors (Bisset, Chen, Feng et al., 2009).

Classic standard models of spread of epidemics were mostly based on the use of differential equations, however, this tool complicates the consideration of connections between separate agents and their numerous individual particularities. ABM allows to overcome such shortcomings. In 1996 Joshua Epstein and Robert Axtell published a description of one of the first ABMs, in which they reviewed the process of the spread of epidemics. (Epstein, Axtell, 1996). Agent models, which differ from each other in their reaction to the disease, which depends on the state of their immune system, are spread out over a particular territory. At that, in this model, agents, the number of which constitutes a mere few thousand, demonstrate fairly primitive behavior.

Later on, under the supervision of Joshua Epstein and Jon Parker at the Center on Social and Economic Dynamics at Brookings, one of the largest ABMs was constructed, which included data about the entire population of the US, that is around 300 million agents (Parker, 2007). This model has several advantages. First of all, it allows to predict the consequences of the spread of diseases of various types. Second of all, it focuses on the support of two environments for calculations: one environment consists of clusters with an installed 64-bit version of Linux, and the other of servers with quad-core processors and an installed Windows system (in this regard, Java was chosen as the language of the programming, although the developers did not indicate which particular version of Java they used). Third of all, the model is capable of supporting from a few hundred million to six billion agents.

The model in question (US National Model) includes 300 million agents, which move around the map of the country in accordance with the mobility plan of 4000×4000 dimensions, specified with the help of a gravity model. A simulation ex-

periment was conducted on the US National Model, imitating the 300-day long process of spreading a disease, which is characterized by a 96-hour incubation period and a 48-hour infection period. In the course of the study, among other things, it was determined that the spreading of the disease was declining, after 65% of the infected got better and obtained immunity. This model has repeatedly been used by the specialists of the Johns Hopkins University, as well as by the US Department of National Security, for research, dedicated to the strategy of rapid response to various types of epidemics (Epstein, 2009).

In 2009 a second version of the US National Model was created, which included 6.5 billion agents, whose actions were specified taking into consideration the statistical data available. This version of the model was used to imitate the spreading of the A(H1N1/09) virus all over the planet (Parker, Epstein, 2011).

Previously, this kind of model was developed by the Los Alamos National Laboratory (USA), and the results of the work with this model were published on April 10, 2006 (Ambrosiano, 2006). One of the most powerful computers which existed at the time known by the name of "Pink", which consisted of two 1024 processors with a 2,4 GHz frequency and a 2 GB memory each was used for the technical realization of the model. This large-scale model, composed of 281 million agents was used to study scenarios of the spreading of various viruses, including the H5N1, taking into consideration several possible operational interventions such as vaccinations, closing of schools and introducing of quarantines in some territories.

2.2 RepastHPC – Environment of Designing Agent-Based Models for High-Performance Computing

Special attention should be paid to the software, developed for the projecting of the ABM with the subsequent launching on supercomputers, - Repast for High Performance Computing (RepastHPC). This product was implemented using the C++ language and the MPI – a program interface for the exchange of messages among processors, executing the task in parallel mode, as well as the library Boost, which expands the C++ (Collier, North, 2011).

A dynamic discrete-event scheduler which carries out program instructions with conservative algorithms of synchronization, which provide for the delay of processors so as to follow a certain order of their completion, was implemented within the framework of RepastHPC.

In RepastHPC agents are divided between processors and every processor is connected to an agent, which is *local* in relation to this processor. In its turn the agent is local to the processor completing the program code, which describes the behavior of this agent. In addition to that, copies of other – *non-local* – agents may be present in any processor, which allows agents of the entire model to interact with these copies. For example, suppose the user, in his model, which presupposes parallel calculations, a will use two processors – P1 and P2, each of which creates a certain number of agents and has its own scheduler of completion of program instructions. Agents, the behavior of which is calculated on the basis of processor P1, are local in relation to this processor, and only within the framework of this processor can the program code

change their condition (the same applies for processor P2). Suppose processor P1 requests a copy of agent A2 from processor P2. Agent A2 is not local to the processor P1, and therefore the program code executed within the P1 processor cannot change the condition of agent A2. At that, agents implemented within the framework of the P1 processor, when necessary, may request information about the condition of agent A2, but the copy of A2 will remain unaltered. The alteration of the original A2 is possible only within the P2 processor, however, in this case RepastHPC synchronizes the changes of the condition of the agent amongst all processors3.

3 Adaptation of Agent-Based Models for The Supercomputer: Our Approach

In March of 2011 the model was launched on the supercomputer "Lomonosov", which simulated the socio-economic system of Russia for the next 50 years. This ABM is based on the interaction of 100 million agents, which hypothetically represent the socio-economic environment of Russia. The behavior of each agent is set by a number of algorithms, which describe its actions and interaction with other agents in the real world.

Five people participated in the project: two specialists of the Central Economic and Mathematical Institute of the Russian Academy of Sciences (V.L. Makarov, A.R. Bakhtizin) and three researches of the Moscow State University (V.A. Vasenin, V.A. Roganov, I.A. Trifonov). The data for the modeling was provided by the Federal Service of State Statistics and by the Russian Monitoring of the Economic Conditions and Health of the Population. A model for a standard computer was developed in 2009, and in 2011 it was converted into a supercomputer version. Below, we will examine the main stages and methods of effective projection of a computable kernel of a multi-agent system on the architecture of a modern supercomputer, which we have developed during the process of resolving the issue in question.

3.1 Parallel programming

It is important to understand that the scaling of programs for supercomputers is a fundamental problem. Although the regular and supercomputer programs carry out the same functionalities, the target functions of their development are usually different.

During the initial development of the complex application software the first and foremost goal is to try to minimize the costs of programming, personnel training, enhance the compatibility between platforms etc., and leave optimization "for later".

[3] More thorough information on the S/W can be found in the User Manual (Collier, 2012); a new version of the 1.0.1. package (dated March 5, 2012) can be downloaded from the following website: http://repast.sourceforge.net/repast_hpc.html

This is quite reasonable, since at the early stages, the priority of development is exclusively the functionality.

However, when the developed software is already being implemented, it is often discovered that there is not enough productivity for real massive data. And since modern supercomputers- are not simply computers, which work thousands of times faster than personal computers, in order to launch the program on a supercomputer it is necessary to introduce significant alterations first. Doing this effectively without special knowledge and skills is by far not always successfully achieved.

When the work is done properly and correctly significant increase in efficiency is usually achieved on the following three levels:

1. multisequencing of calculation;
2. specialization of calculative libraries by task;
3. low-level optimization.

3.2 Specialization and Low-Level Optimization

Before seriously talking about the use of supercomputers, the program must be optimized to the maximum and adapted to the target hardware platform. If this is not done, the parallel version will merely be a good test for the supercomputer, but the calculation itself will be highly inefficient.

To use a supercomputer without optimization and adaptation of the program to the target hardware platform is the same as sending a junior regiment on a combat mission: first it is necessary to teach the recruits how to properly carry out their tasks (specialization, optimization of software), as well as how to efficiently handle weapons (low-level optimization of software), and only then it can be considered an effective use of resources.

In the universal systems of modeling of the AnyLogic[4] type, the procedures presented are universal. And a universal code can often be optimized for a particular family of tasks.

3.3 Selection of a Modeling Support System.

Certainly, ABM can be programmed without a special environment, in any object - based language. In addition, the main shortcoming of the existing products for creating ABM except for RepastHPC, is the inability to develop projects that would run on a computing cluster (i.e. there is no mechanism for paralleling the process of executing the program code).

However, a more reasonable approach would be to use one of the proven systems for ABM - because of the unified implementation of standard ways of interacting agents. To save space, here, we will only consider the ADEVS system[5].

---

[4] AnyLogic – is an instrument for imitational modeling, which supports all approaches for creating imitational models: process-oriented (discrete event), system dynamic and agent-based, and any combination of the above. See http://www.anylogic.ru.

ADEVS is a set of low-level libraries for discrete modeling done in C++ language. Some of the advantages worth mentioning are the following:

- ease of implementation;
- high efficiency of models;
- support of basic numerical methods used for modeling ;
- built in paralleling of simulation with the help of OpenMP;
- the possibility of using standard means of paralleling;
- fairly rapid development of libraries in the current time;
- Cross-platform software;
- low-level software (current functionality does not impose any restrictions on the model);
- independence of the compiled code on unusual libraries;
- open source code.

However, significant shortcomings of this product is a complete lack of means of presentation and quite complicated modeling, when compared to, for example, AnyLogic. Therefore, this product cannot be used to build models on the consumer level, however, is an effective platform for implementing parallel simulations.

The main elements of the program when using the ADEVS library to build an ABM are the following:

- adevs simulator:: Simulator< X >;
- primitive of adevs agents:: Atomic< X >;
- model of adevs (container of agents)::: Digraph< VALUE, PORT >.

It was decided to develop the supercomputer version of the program described below, based on the ADEVS system, in view of its advantages listed above. Within the framework of this project, an MPI-version of the ADEVS simulator was created, as well as a visualization system of the calculation process based on the Qt library- a cross-platform set of tools for creating software written on the C++ programming language.

Next, we turn to a brief description of the developed model and the procedures for its subsequent run on a supercomputer.

3.4 The initial agent -based model

The first stage of development of the ABM described below, is to construct a tool that effectively solves the problem of research on conventional computers, as well as adjusting the parameters of the model. After its successful testing with a small number of agents (about 20 thousand - is the number of agents, with which conventional computers are able to perform calculations at a satisfactory rate and with good productivity, given the complexity of agents) it was decided to convert the model so

[5] The ADEVS system and its description can be downloaded from here:: http://www.ornl.gov/~1qn/adevs/

that it could be used on a supercomputer— this was the second stage of development. During the first stage the AnyLogic product was used, the technical capabilities of which, allowed to debug the model at a satisfactory speed and to configure its settings. Thus, the model for an ordinary computer was built in 2009, and in 2011 it was converted into a supercomputer version.

Fig. 1 shows the operating window of the developed ABM (dots - agents). During the operation of the system current information can be obtained on the socio-economic situation in all regions of Russia, including the use of cartographic data, changing in real time depending on the values of the endogenous parameters.

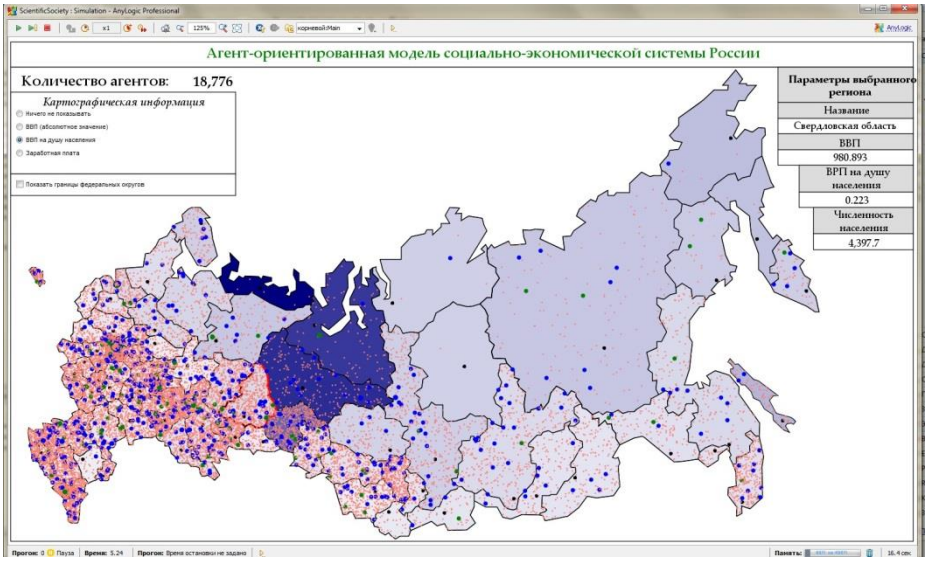

Fig. 1. Operating window of the developed ABM

The specification of the agents of the model was carried out taking into consideration the following parameters: age; life expectancy; specialization of parents; place of work; region of residence; income and others.

The specification of regions was carried out, taking into consideration the following parameters: geographic borders; population; number of workers (by type); GRP; GRP per capita; volume of investments; volume of investments per capita; average salary; average life expectancy; index of population growth, etc.

The major actions of agents are:

- aging (change from the category «children» to category «working age adults», and then - to «pensioners»);
- creating family;
- giving birth to a child;
- acquiring profession;
- religious activity;
- change of job;

• migration to other regions (or countries).

These actions are accompanied by the change in the number fo agents within corresponding groups with respect to (a) ethnicity, (b) religion, (c) nationality etc.

Statistics manuals of Rosstat, as well as sociological databases of RLMS (The Russia Longitudinal Monitoring Survey) were used to fill the model with the necessary data.

3.5 Conversion of the Model into a Supercomputer Program.

Earlier we had already discussed the problems of using ABM development tools for the realization of projects, carried out on the computing clusters of the supercomputer. Due to the difficulties in separating the computing part from the presentational part, as well as to the realization of the code using a high level of the JAVA language the productivity of the implementation of the code is significantly lower for AnyLogic than for ADEVS. Apart from that, it is extremely difficult to reprocess the generated code into a concurrently executed program.

Below is the algorithm of the conversion of the AnyLogic model into a supercomputer program.

3.6 Translation of the Model.

The models in the AnyLogic project are kept in the format of an XML-file, containing the tree diagram of the parameters necessary for the generation of the code: classes of agents, parameters, elements of the presentation, descriptions of the UML-diagrams of the behavior of agents.

During the work of the converter this tree diagram is translated into code C++ of the program, calculating this model. The entry of the tree is executed “depth wise”. At that, the following key stages are marked, and their combination with the translation process is carried out.

1. *Generating the main parameters.* The search for the root of the tree and the reading of the parameters of daughter nodes, such as the name of the model, address of assembly, type of model, type of presentation.

2. *Generating classes.* Generating classes (more detailed):

- configuration of the list of classes;
- reading of the main class parameters;
- reading of the variables;
- reading of the parameters;
- reading of the functions;
- generating a list of functions;
- reading the functions code;
- conversion of the functions code Java -> C++;
- reading of the figures and elements of control that are being used;
- generating the code of initialization of figures and elements of control;
- generating constructor, destructor and visualizer codes;

- generating the class structure;
- generating header code and source-files.

3. *Generating the simulator.* Search for the peak, containing the information about the process of simulation (controlling elements, significance of important constants, elements of presentation etc.).

4. *Generating shared files of the project* (main.cpp, mainwindow.h, mainwindow.cpp и etc.)

3.7 Import of Incoming Data

Data from the geoinformation component of the initial model (map of Russia), containing all of the necessary information is imported into the model as input data.

3.8 Generating Classes and the Transformation of the Functions Code

When generating the functions from the tree the following information is read: name of function, return type, parameters and its body.

Based on the list of classes constructed earlier, changes are introduced into the arguments of the functions, replacing the "heavy classes", i.e. all generated classes, classes of figures and other classes, which do not form part of the standard set, with corresponding pointers. The purpose of this is to save memory space and avoid mistakes when working with it. After that, the titles of the functions are generated, which are later inserted into the title and source-files. In the course of such reading the body of the function, by the means of the relevant function is transformed from a Java-based code into an analogical C++ code (this is possible due to the fairly narrow class of used functions; as for more complicated functions manual modification of the translated code is required), after which it is added to the list of bodies for this class.

In the course of the translation the need for the transformation of the initial functions code from the Java language to the C++ often arises. It can be presented in the form of sequential replacements of constructions, for example:

- *Transformation of cycles*: Java-format.
- *Transformation of pointers*. Java, unlike C++, does not contain such an obvious distinction between the object and the object pointer, hence the structure of the work with them does not differ. That is why a list of classes is introduced, in which it is important to use operations with object pointers, and not with the object itself, and all of the variables of such classes are monitored with the subsequent replacement of addresses to the objects with corresponding addresses to the object pointers within the framework of the given function.
- *The opening of "black boxes"*. In Java, and in the AnyLogic library in particular, there is a certain number of functions and classes, which do not have analogues in the C++ itself, nor in the ADEVS library. Due to this fact additional libraries shapes.h, mdb-work.h had been created, which compensate for the missing functions.

- *During the generating stage of the main parameters of the lists of classes* the name of the main class and the names of the modulated agent-classes are obtained. The procedure of adding an agent into the visibility range of the simulator is introduced into the code of the main class.

3.9 Generating Outside Objects

In the process of generating outside objects a separate function «Main::initShapes()» is created, which contains all of the "graphic information", i.e. the initialization of all figures, the classes of which had also been implemented in the shapes.h., is carried out within the framework of the function. The relevant example is presented in the following code fragment.

3.10 Generating Classes and the Code of the Title and Source Files

Based on all the data that has been read and generated the title and source files of the corresponding class are created.

3.11 Generating Simulation

For the generation of simulation it turned out to be enough to have the main.cpp, mainwindow.cpp, mainwindow.h files, written beforehand, in which the templates define the type of the main class and the added title files. When compiling the initial code the templates are replaced with the names of the classes received earlier (at the generating stage). This is enough for the double-flow simulation, which can later be replaced with a corresponding module for a multiprocessor simulation.

3.12 Additional Attributes

At the stage of analyzing the tree (see above) a tree, similar in structure, is formed for the generation of a C++ code, with the help of which the necessary attributes of compilation can be set (visualization of certain parts, visual validation of code recognition, additional flags of assembly etc.) , during the stage of preparation for translation.

After that, at receiving the command for transformation, the final compilation takes place, taking into consideration all of these attributes.

3.13 Assembly of the Ready Project

For the assembly of the translated project the QtCreator is used — cross-platform shareware integrated environment for work with the Qt framework.

3.14 Agent Code

With the help of the translator described above an initial code (except for the behavior pattern of agent) has been generated from the data of the files of the AnyLogic project (model.alp and others).

The behavior pattern of the agent must be generated from the diagram of conditions, however, currently the automation of this process has not yet been implemented. Therefore, a certain volume of the code had to be added to the generated code.

After the introduction of the necessary changes, a cross-platform application, repeating the main functionality of the given model, was achieved as a result of the compilation.

3.15 Statistics and Visualization of Time Layers

Given the non-interactive mode of the launching of the program on big supercomputers the collection of data and visualization were separated (this has to do with the load imbalance on clusters at various times of the day; as for exclusive access, it is simply impossible). After the recalculation of the model the information obtained can once again be visualized, for example, in the following manner (Fig. 2).

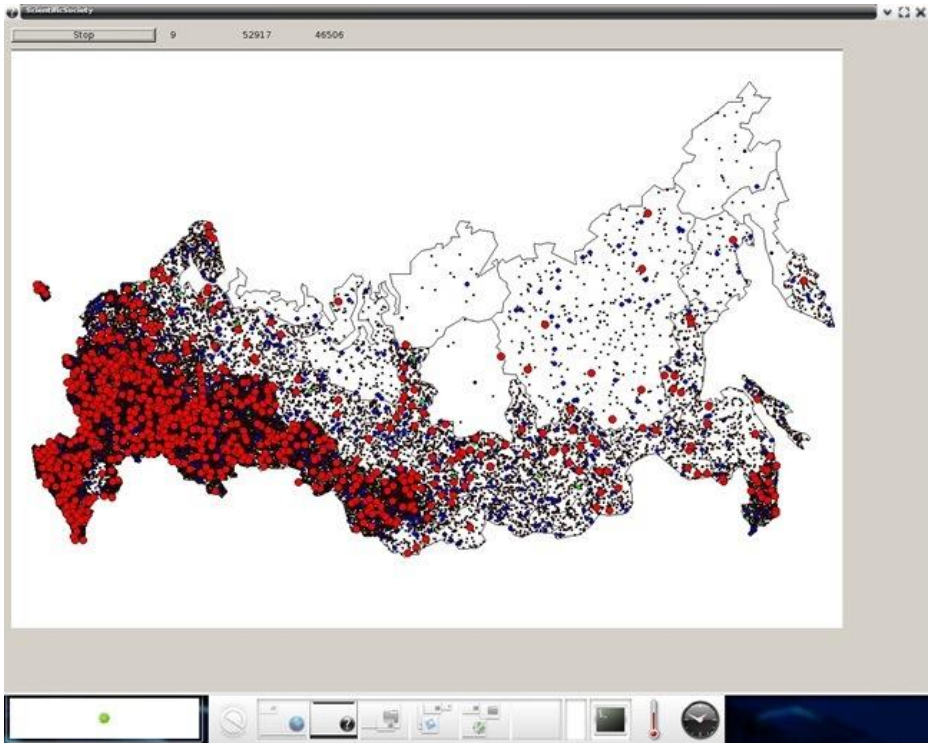

Fig. 2. Result of the work of the supercomputer program in graphic format

3.16 Supercomputers Available for Calculations

At the moment of making the calculations three supercomputers were available to us (Table 1), which were in the top five of the supercomputer rating of the Top-50 supercomputers in the CIS countries.

Table 1. Supercomputers available for research group

Position in Top -50	*Supercomputers*	*Nodes*	*CPU*	*Kernels*	*RAM/ node*	*TFlops*
1	«Lomonosov» (Moscow State University)	5 130	10 260	44 000	12GB	414
4	MVS-100K (Inter-agency Supercom-puting Center of the RAS)	1 256	2 332	10 344	8GB	123
5	«Chebyshev» (Moscow State University)	633	1 250	5 000	8GB	60

For our calculations we used two supercomputers – the «Lomonosov» and the MVS-100K.

4 Results

By using supercomputing technologies and optimizing the program code we were able to achieve very high productivity.

The optimization of the code and the use of C++ instead of Java allowed for the increase in speed of execution of the program. The model was tested in the following initial conditions: 1) number of agents — 20 thousand; 2) forecasting period — 50 years. The results of the calculations showed that the count time of the model using ADEVS amounted to 48 seconds using one processor, whereas the count time of the model using AnyLogic and a single processor amounted to 2 minutes and 32 seconds, which means that the development framework was chosen correctly.

As has already been noted before, an ordinary personal computer with high productivity is able to carry out calculations with satisfactory speed with a total number of 20 thousand agents (the behavior of each is defined by around 20 functions). At that the average count time of one unit of model time (one year) amounts to around one minute. When dealing with a larger number of agents, 100 thousand for example, the computer simply “freezes”.

Using the 1000 processors of the supercomputer and executing the optimized code, allowed to increase the number of agents to 100 million, and the number of model years to 50. At that, this enormous volume of research was carried out in a period of time, which approximately equaled 1 minute and 30 seconds (this indicator may slightly vary depending on the type of processors used).

Then we continued to increase the number of processors (under the same model parameters) in order to establish a dependency of the time for computation from the resources involved. Corresponding function is depicted on Fig.3.

Results of the modeling showed, that if the current tendencies ensue the population of the Siberian and the Fareast Federal Districts will significantly decrease, while in the Southern Federal District, on the contrary, there will be a significant increase in population. In addition to that, the modeling carried out gives reason to predict a gradual decrease of the GDP, as well as several other macroeconomic indicators.

The results of the experiments carried out using the developed ABM revealed that the scaling of the model in itself has certain meaning. For example, when launching the same version of the model for 50 model years using the same parameters (except for the number of agents: in the first case there were 100 million agents, in the second – 100 thousand agents) the results received, that is the scaled number of agents, had a difference of about 4,5%.

It can be assumed that in complex dynamic systems the same parameters (birth rate, life expectancy etc.) may produce different results depending on the size of the community.

Note a considerable increase of model productivity from the number of processors. At the same time we should mention that the model excludes almost any type of inter-agent interactions (which is conducted at macro-level). Therefore, we agenda for further research is to develop the complexity of the links between agents and to search for a most efficient way for a parallel program code.

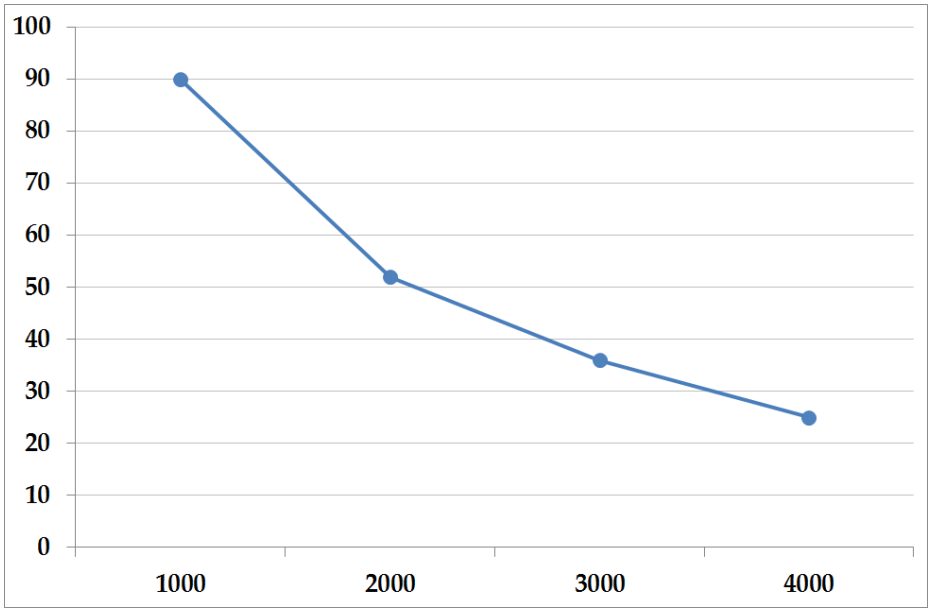

Fig. 3. Time for calculating the model as a function of the number of processors (X axes– number of processors, Y axes – time in seconds)

5 Acknowledgement

This work is supported by Russian Humanitarian Scientific Fund (grants № 14-02-00431).

References

1. Makarov V.L., Bakhtizin A.R., Vasenin V.A., Roganov, V.A., Trifonov I.A. (2011). Tools of supercomputer systems used to work with agent- based models // Software engineering. № 3.
2. Ambrosiano N. (2006). Avian Flu Modeled on Supercomputer // *Los Alamos National Laboratory NewsLetter*. Vol. 7. No.8.
3. Bisset K., Chen J., Feng X., Kumar VSA, Marathe M. (2009). EpiFast: A fast algorithm for large scale realistic epidemic simulations on distributed memory systems. Yorktown Heights, New York; 2009:430–439. Proceedings of 23rd ACM International Conference on Supercomputing (ICS'09).
4. Coakley S., Smallwood R., Holcombe M. (2006). From molecules to insect communities-How formal agent based computational modelling is uncovering new biological facts // Scientiae Mathematicae Japonicae 64 (2), 185-198.
5. Collier, N. and North, M. (2011). Repast HPC: A Platform for Large-Scale Agent-Based Modeling, in Large-Scale Computing (eds W. Dubitzky, K. Kurowski and B. Schott), John Wiley & Sons, Inc., Hoboken, NJ, USA.
6. Collier N. (2012). Repast HPC Manual. [Electronic resource] February 23. Access mode: http://repast.sourceforge.net, free. Screen title. Language.English. (date of request: may 2013).
7. Deissenberg C., Hoog S. van der, Herbert D. (2008). EURACE: A Massively Parallel Agent-Based Model of the European Economy // *Document de Travail No. 2008*. Vol.39. 24 June.
8. Epstein J.M., Axtell R.L. (1996). Growing Artificial Societies: Social Science from the Bottom Up. Ch. V. Cambridge, Massachusetts: MIT Press.
9. Epstein J.M. (2009). Modeling to Contain Pandemics // *Nature, volume 460*, p 687, 6 August.
10. Keith R.B., Jiangzhuo C., Xizhou F., Anil Kumar V.S., Madhav V.M. (2009). EpiFast: A Fast Algorithm for Large Scale Realistic Epidemic Simulations on Distributed Memory Systems ICS'09. June 8–12. N.Y.: Yorktown Heights.
11. Parker J. (2007). A Flexible, Large-Scale, Distributed Agent Based Epidemic Model. Center on Social and Economic Dynamics. Working Paper No. 52.
12. Parker J. Epstein J.M. (2011). A Distributed Platform for Global-Scale Agent-Based Models of Disease Transmission // ACM Trans Model Comput Simul. Dec 2011; 22(1): 2.
13. Roberts D.J., Simoni D.A., Eubank S. (2007). A National Scale Microsimulation of Disease Outbreaks. RTI International. Research Triangle Park. Blacksburg: Virginia Bioinformatics Institute.
14. Roberts D.J. ABM++ framework. https://www.epimodels.org/midas/Rpubabmplusplus.do (Accessed 2 August 2012).