Organization and Protection on the Basis of a Multi-agent System of Distributed Computing in a Computer Network to Reduce the Time for Solving Large-Scale Tasks

Sergey Khovanskov(✉), Konstantin Rumyantsev, and Vera Khovanskova

Southern Federal University, Rostov-on-Don, Russia
{Sah59, rke2004}@mail.ru, v.s.khovanskova@gmail.com

Abstract. Special compilers are often used to solve multivariate tasks with time constraints. However, in this case, the cost of solving the problem is significantly increased and the time required to organize access to such a computing environment is required. At present, the use of distributed computing organized in a computer network is one of the most accessible and widespread technologies for reducing the time for solving large-scale tasks. Many different approaches for organization of distributed computing in a computer network are grid technology, metacomputing (BOINC, PVM and others). The purpose of most of the existing approaches for creating centralized systems of distributed computing is their main disadvantage.

We propose to organize solutions to such a problem as multivariate modeling by creating distributed computing in a computer network based on a decentralized multi-agent system. A typical computer network is selected as a computing environment. A self-organizing distributed computing system based on a decentralized multi-agent system is proposed as a computer system. A system is a set of agents performing the same algorithm. We propose an agent algorithm for a decentralized multi-agent system. Agents working on this algorithm create a self-organizing distributed computing system and protect the results of calculations from such a thunderstorm as "denial of service".

Keywords: Distributed computing · Information protection
Computational process

1 Introduction

Nowadays special computers are often used to solve multivariant tasks with time constraints. However, this significantly increases the cost of solving the problem and requires time to organize access to such a computing environment.

Distributed computing is one of the most accessible and common technologies for reducing the time for solving complex multivariant problems [1–6].

Different computing environments are used to implement such calculations. The multiprocessor computer, cluster computing system, multi-computer system, or an

P. K. Singh et al. (Eds.): FTNCT 2018, CCIS 958, pp. 292–303, 2019.
https://doi.org/10.1007/978-981-13-3804-5_22

ordinary computer network can be used as a computing environment. The most accessible computing environment on which distributed computing is possible is a computer network that has a sufficient or excessive number of data centers (local, wide area networks). Most available computing environment where it is possible to perform distributed computing is a computer network having a sufficient or excessive number of data centers (LAN/WAN).

Currently, there are many different approaches for organization of the distributed calculations in computer network technology grid, metacomputing (BOINC, PVM and others). Most existing approaches are designed to create centralized distributed computing systems. This is their main drawback. In the global network there is a real threat to the operability of the distributed computational processes due to the extreme instability of the computing environment and the actions of intruders. We propose to use self-organizing distributed computing system based on a decentralized multi-agent system for the solution of large-task and to reducing the threats to the existence of distributed computing and the security of the obtained results [7–10] (see Fig. 1).

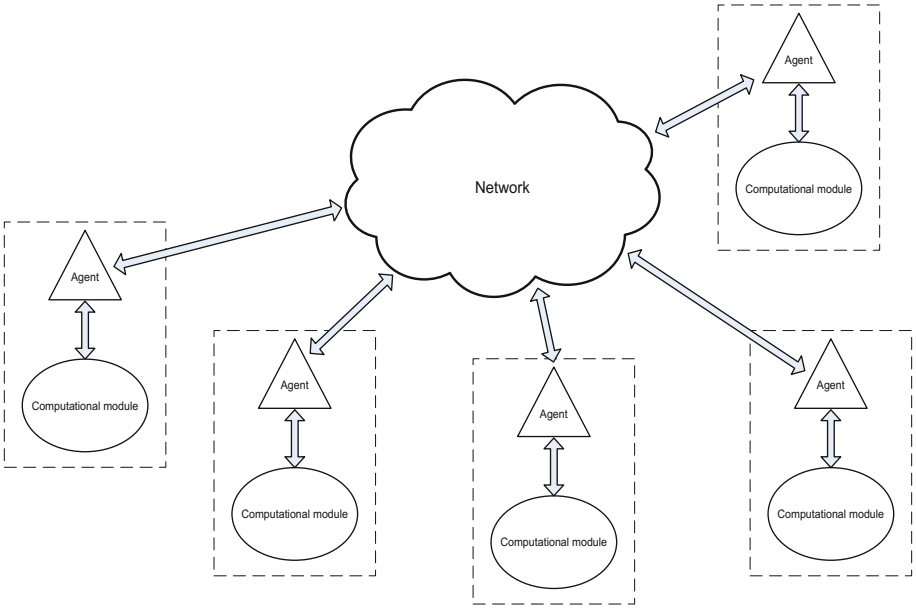

Fig. 1. Structure of the agent program of the multi-agent system

Multiagent system is a set of agents, each of which represents a software module and placed on a separate computer. The agent performs the office only of its computer and therefore its work is independent. It organizes the decision of tasks on your computer initiates the communication with the computers of other agents, performs processing of information provided by them and based on it make decisions.

Decentralized multi-agent system is a set of networked computers. Each computer is under the control of his agent. All agents execute the same algorithm. The result is a peer-to-peer computing system. Each agent works independently from the other agents, the exchange between agents is done using broadcast messages. It allows in the process of implementing distributed computing scaling a multi-agent system without affecting functionality of processes computing.

We the developed the multi-agent system algorithm for the realization and protection of distributed computations in computer networks. The algorithm allows organizing the distributed computing system based on the nodes of computer network [11–20].

2 Implementing a Multi-agent System

The system should be decentralized—each agent should have equal rights and be able to exchange messages with other agents.

Let's formulate requirements to the algorithm of the agent:

- the agent should monitor the computing processes running on managed computer;
- agents must share the computational load for the organization of distributed computing;
- agents must redistribute its processing load depending on the performance of their computers;
- each agent needs to store all the results of the execution of a large task;
- multi-agent system must ensure the protection of distributed computing against threats from intruders.

The algorithm was developed containing a set of rules that each agent must perform to organize distributed computing in a computer network and implement requirements.

A multi-agent system is a set of agents M in the form of the same program modules of agents $\{m_1, m_2, \ldots, m_n\} \in M$. The set M is superimposed on the set of network computers $\{p_1, p_2, \ldots, p_j\} \in P$ $(P > M)$ so that the agent m_i is located on the corresponding computer p_i of the network. Each agent module (agent) controls the resources of the p_i computer and monitors the load on the Wi. All multi-agent system M, managing computers $\{p_1, p_2, \ldots, p_j\} \in P$, organizes a system of distributed computing to solve the whole set of tasks $\{w_1, w_2, \ldots, w_n\} \in W$. The set M is a peer-to-peer set of agents working on the same program.

At the beginning of the organization of distributed computing, the agents $\{m_1, m_2, \ldots, m_n\} \in M$ are operating in the computer network $\{p_1, p_2, \ldots, p_n\} \in P$ on M. At the first stage, the $m_i \in M$ agent receives the basic information for the organization of distributed computations in the set M. It includes the computational load W of the M system and indicating which part of w_i of the total volume W the agent must perform. To monitor the execution of the computational load, each agent has two tables

for work. The first W_{nrez} table includes all outstanding tasks. The second W_{rez} table includes the completed tasks with the results of the execution $\{W_{nrez}, W_{rez}\} \in W$.

At the initial stage of organizing distributed computing in a computer network $w_i \subseteq W$.

After the agent receives the load $m_i \in M$ and general information about the system, he initiates on his computer p_1 a computational process to perform wi_1, performing the actions in accordance with the rule of computational load.

3 Algorithm of the Agent for Organizing Distributed Computing and Ensuring the Safety of the Results of Computing Processes

Each agent of a multi-agent system is located on its network computer. Each computer is an autonomous computing system, the work of which does not depend on other computers on the network.

The agent m_i monitors the state of the p_i computer and manages its operation in accordance with the "computational load" rule. If p_i does not perform calculations, then the agent m_i selects from the list of its computational load W_i the next order in the order and passes it for execution to the computing block of the computer p_i. The choice is made by sequentially viewing the list of computing load $Wi \in W_{nrez}$ by the agent.

The algorithm of actions of the agent m_i by the rule "computational load execution".

1°. The agent $m_i \in M$ checks whether the next task $w_j \in Wi$ is completed? If not then to point 7°.
2°. Agent m_i receives the result of the task $w_j \in Wi$.
3°. The m_i agent looks at the list of uncompleted Wi_{nrez} jobs.
4°. Load Wi completely fulfilled $Wi_{nrez} = 0$? If yes, go to item 7°.
5°. Selection by agent m_i of the next job $w_j + 1$ from the list of uncompleted computing load Wi_{nrez}.
6°. Transfer the selected job $wj + 1$ to the computer $p_i \in Pz$.
7°. Go to another rule.

Due to the "computational load" rule, each computation node p_i continuously performs computational load Wi. The process is performed completely under the control of the agent m_i. This allows you to optimally use the computing resources of each computer.

To implement the interaction between the agents $\{m_1, m_2, \ldots, m_n\} \in M$, during the execution of distributed calculations, agents exchange messages with results among themselves. Exchange between agents occurs against the background of computational load carrying out by computers controlled by the agents $\{m_1, m_2, \ldots, m_n\} \in M$. The agent m_i, having received the result from the other agent, writes $w_j \in W$ into its table of

the results of the general computing load. At the end of the work, each agent stores all the results of the solutions to tasks W.

The algorithm of actions of m_i agent according to the rule "transfer the obtained results to the other agents".

1°. Verification is not passed to their computational load $w_j \in Wi$? If Yes, go to item 2°, if not then to paragraph 5°.
2°. To form a package for transmitting information agents $\{m_1, m_2, \ldots, m_n\} \in M$.
3°. Free medium? If Yes, go to item 4°, if not to the point 5°.
4°. To send a package with information about the result of the calculation $w_j \in Wi$ from all agents $\{m_1, m_2, \ldots, m_n\} \in M$.
5°. Go to the selection rules of behavior.

The execution time of the entire computing load W by the multi-agent system M is equal to the time for executing the average load Wi by the agent $m_i \in M$.

Incomplete or slow execution of computational load by agents of the distributed computing system can be caused not only by the low speed of individual computers, but also by the consequence of the implementation of a denial-of-service threat. A denial-of-service attack can lead to a disruption in the performance of some compute nodes and, as a consequence, the termination of the operation of the distributed computing system itself. To ensure the protection of distributed computing from this threat, each agent performs actions according to the rule of monitoring the completeness of the execution of the total computing load W.

The agent m_i performs the tracking of the completeness of the execution of W with each obtaining of results from both agents $m_j \in M$ and from the computer p_i and recording of the results obtained in the list. W_{rez}. If the entire Wi load is performed in full, then the agent scans and selects from W_{nrez}. The job and and passes it on to execution p_i.

The algorithm of actions of the agent m_i according to the rule of monitoring the completeness of execution of the general computing load.

1°. The agent $m_i \in M$ checks whether all tasks included in its computational load are executed $Wi_{nrez} = 0$? If yes then go to item 5°, if not to item 2°.
2°. The agent m_i selects from the table of the general computing load W_{nrez} the job w_j by which the result is not obtained.
3°. Does the agent m_i check that he performed w_j before? If yes then go to item 5°, if not then go to item 4°.
4°. The agent m_i transmits the selected task to its computer $p_i \in Pz$.
5°. Go to the next rule.

A graphical representation of the algorithm in Fig. 2

Due to the implementation of the rule for monitoring the completeness of the overall computing load, there is a redistribution of the load between the agents of the multi-agent system. When you attack "denial of service" and the failure of one or more

of the agents, the load is redistributed among the remaining healthy compute nodes of the multi-agent system. This ensures high resiliency of the distributed computing system created on the basis of a multi-agent system in a computer network.

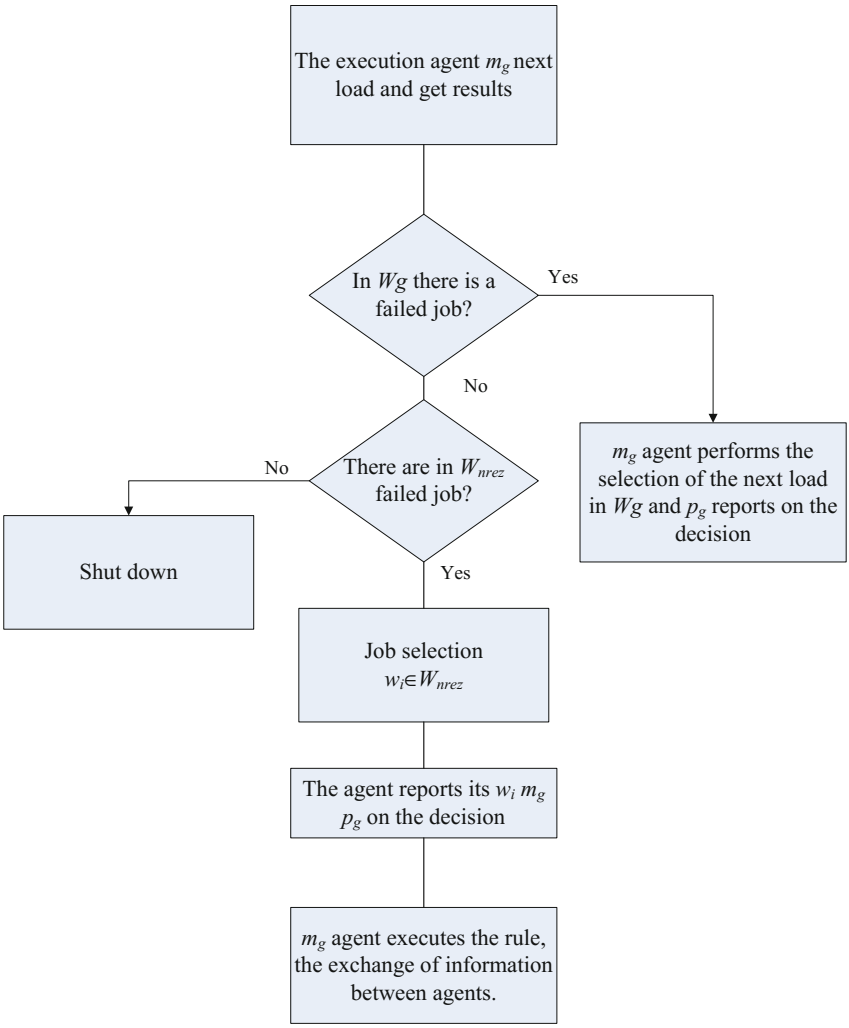

Fig. 2. Structure of the agent program of the multi-agent system

Based on the developed algorithm was written in Python and streamlined program of work agent multi-agent system. The program structure is shown in Fig. 3.

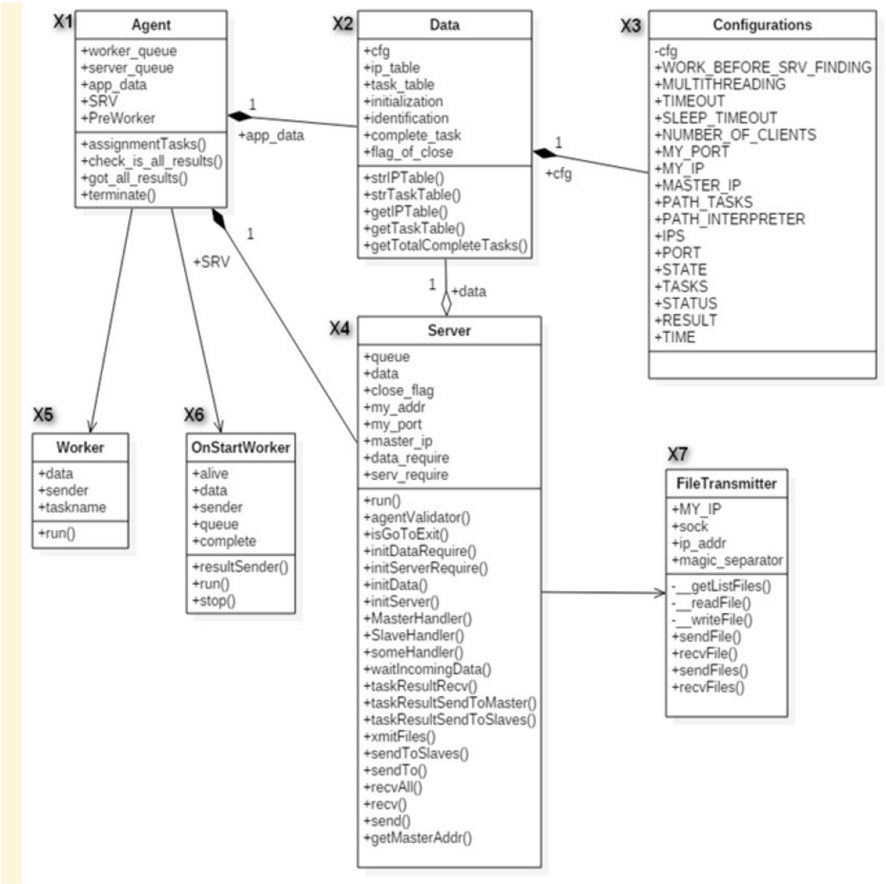

Fig. 3. Structure of the agent program of the multi-agent system

4 Estimating the Detection of a False Result in a Centralized Multi-agent System

Each agent of a multi-agent system fulfilling the developed algorithm of distributed computing allows expanding the system by including free computers in it. For this purpose, the agent module and its computational load are transferred to the free computer. Scaling a multi-agent system reduces the computational burden on each agent and reduces the execution time of a larger volume task. The agent module can be located on any network computer, including on the computers of the global Internet. This increases the degree of threat to the security of processes and the results of distributed computing. The management agents verify the correctness of the results obtained from the agents of a multi-child system for protecting distributed computing from the threat of receiving a false result.

Let's calculate on a concrete example the probability of finding a false result for a centralized multi-agent system. The calculation is feasible for a multi-agent system consisting of thousands of agents performing calculations and one managing agent. Assume that among the many agents that make up the multi-agent system, there are intruders, each of which transmits false results of calculations. The controlling agent from the results obtained from the agents selects some and checks their correctness by repeating the calculations. The results to be checked are randomly selected, since the managing agent, having limited computing resources, does not know exactly which agent is the attacker.

The probability of P_{olr} detecting one false result in a centralized multi-agent system, which is constantly formed by an attacker, is determined by the Bernoulli formula. The Bernoulli formula makes it possible to determine the probability of occurrence of a certain event under independent conditions. This suggests that the occurrence of an event in an experiment does not depend on the appearance or non-appearance of the same event in earlier or subsequent tests.

$$P_n(m) = \frac{n!}{m! * (n-m)!} * p^m * (1-p)^{n-m} \tag{1}$$

where m is the number of times the event occurred;

p - probability that the event will occur;

n is the number of repetitions of the experiment.

For our case, the number of repetitions of the experiment n is the average computational load kr for each agent. It depends on the total amount of computational load W and the number of agents N in the multi-agent system M. Load kr is calculated by formula

$$n = kr = \frac{W}{N} \tag{2}$$

where N is the number of agents in the multi-agent system M;

W is the total amount of computational load.

The probability of occurrence of an event in one experiment is determined by the number of agents of the multi-agent system:

$$p = \frac{1}{N}; \tag{3}$$

To calculate the probability of finding a false result, we substitute in our formula (1) our data from formulas (2) and (3).

$$P_{olr} = \frac{\left(\frac{W}{N}\right)!}{m! * \left(\frac{W}{N} - m\right)!} * \left(\frac{1}{N}\right)^m * \left(1 - \frac{1}{N}\right)^{\frac{W}{N}-m} \tag{4}$$

where m is the number of false results found.

Since an attacker can generate false results for his entire computing load, the probability of generating a false result is $ko = 1$. If the probability of forming a false result is $ko = 0.5$, this means that it produces false results for only half of its computational load.

In accordance with the structure of the multi-agent system, formula (5) is used to calculate the probability of detecting a false result, reflecting the dependence of the probability on the number of control agents and the probability of creating false results by the attacker:

$$p = \frac{b}{N} * ko; \tag{5}$$

ko - probability of false results creation by an intruder;

b is the number of control agents.

Substituting in formula (4) instead of probability p, calculated from formula (3), the value of p calculated by formula (5), we will form the probability of detection of at least one false result ($m = 1$) from the attacker by the controlling agent ($b = 1$). Probabilities of detecting at least one false result in a centralized multi-agent system that are generated by an attacker with probabilities $ko = 1$, $ko = 0.8$ and $ko = 0.6$ (see Fig. 4).

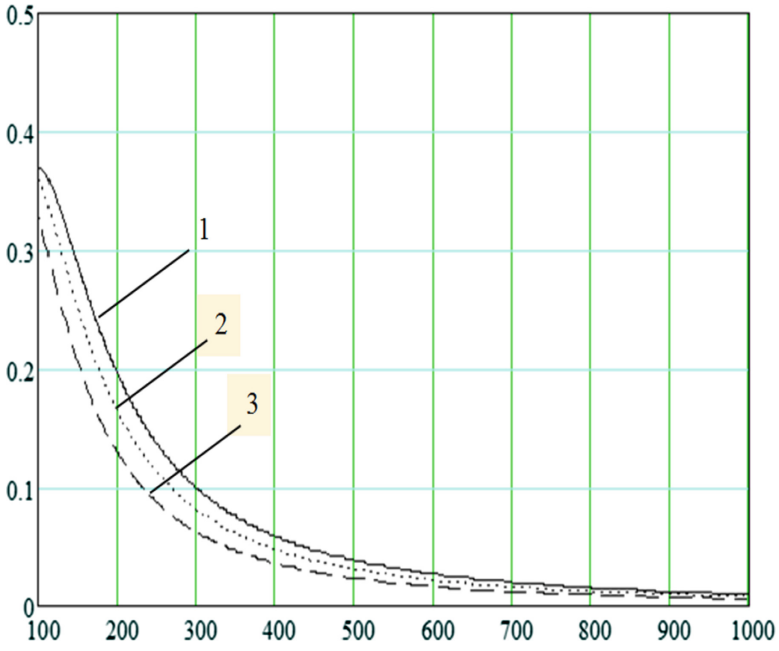

Fig. 4. The probability of detection of at least one false result when the probability of the formation of a "false result" (1) $ko = 1$ (2) $ko = 0.8$ and (3) $ko = 0.6$.

The calculation is performed for a centralized multi-agent system with one managing agent. The number of tasks for a large-volume problem $W = 10000$ for a different number of agents N of a multi-agent system from 100 to 1000. When analyzing the obtained graphs, it is seen that the probability of detecting at least one false result by the managing agent decreases with the increase in the number of system agents. With the number of agents $N = 1000$, the probability of detecting at least one false result is $P_{olr} = 0.005$. When $N \to \infty$ the probability $P_{olr} \to 0$. The reason for this is when scaling the multi-agent system the number of results you get a managing agent, increases. Managing agent is unable to verify all the results transferred to him by the agents of multi-agent system M.

Reducing the likelihood of false results is also affected by a decrease in the probability of false results from the attacker, since in this case the total number of false results in the multi-agent system decreases, which reduces the probability of detection of false results by the managing agent.

Similarly, the probability of detection by the managing agent of at least 2 false results for one intruder is calculated, similarly to 3 and 4 (*Polr2*, *Polr3*, *Polr4*). In the formula (4) we substitute $m = 2, 3, 4$.

The probabilities P_{olr2}, P_{olr3}, P_{olr4} of the detection agent of false results are calculated with one attacker. Probabilities of detection of 2, 3 and 4 false results in a centralized multi-agent system of those that are formed by an attacker with probabilities $ko = 1$. are shown in Fig. 5.

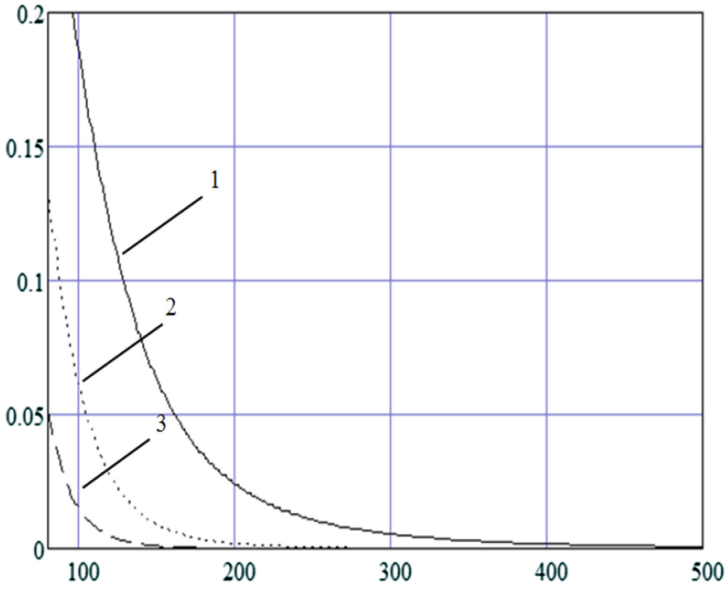

Fig. 5. Probability of detection with $ko = 1$ (1) 2x, (2) 3x and (3) 4 false results.

Comparing the graphs in Figs. 3 and 4, we can conclude that in the centralized multi-agent system the probability of finding false results decreases with the increase in the number of false results that the controlling agent should detect.

5 Conclusion

The algorithm was developed for the organization and protection of distributed computations in computer networks based on multi-agent system with the aim of reducing the solution time of large-scale problems. We developed the algorithm for decentralized multi-agent agent system, which allows securing distributed computing based on multi-agent systems in computer networks and reducing the solution time of large-scale problems. Decentralized computer system provides higher protection efficiency of the processes of distributed computing than centralized in an unstable computing environment of a computer network. Agents, working on information in the article, algorithm, perform their own distribution between a given computational load for the organization of distributed computing. In the process of implementing distributed computing, the agents communicate with multicast messages pass each other the results of the calculations and redistribute among themselves the given computational load depending on the performance of computers. It allows you to provide in addition to reducing the solution time of large-scale problems, the protection efficiency of computational processes and computing results from the substitution. It increases the degree of protection for distributed computing from the threat of "denial of service" and safety results of the decision from the threat of a "false" result compared to a centralized computing system. Implementation of the algorithm for decentralized multi-agent system program was written in Python. The agent program was installed on 3 computers. The results of multi-agent system in the network showed that an organized system of distributed computing works. The system performs integrity monitoring of the results of the solutions of a large problem. At the organization of the distributed computing system decreases the computational load on computers with low productivity through the redistribution of computational load among the agents.

References

1. Kureichik, V.V., Kureichik, V.M., Sorokoletov, P.V.: Analysis and a survey of evolutionary models. J. Comput. Syst. Sci. Int. **5**(46), 779–791 (2007)
2. Khovanskov, S.A., Norkin, O.R., Litvinenko, V.A.: Algorithm of optimization of computing loading of the organization of the distributed calculation. In: Proceedings of the Congress on Intelligent Systems and Information Technologies, IS-IT 2011, Scientific Publication in 4 Volumes, No. 4, pp. 142–145. Physmathlit, Moscow (2011)
3. Pljonkin, A., Rumyantsev, K.: Quantum-cryptographic network. In: 2016 IEEE East-West Design and Test Symposium (EWDTS). https://doi.org/10.1109/ewdts.2016.7807623. Electronic ISSN 2472-761X. ISBN 978-150900693-9
4. Khovanskova, V.S., Khovanskov, S.A., Litvinenko, V.A., Litvinenko, E.V.: Primenenie parametricheskoj adaptacii v algoritmah postroeniya ortogonal'nogo dereva SHtejnera. Informatika, vychislitel'naya tekhnika i inzhenernoe obrazovanie **4**(28), 9–16 (2016)

5. Khovanskov, S.A., Litvinenko, V.A., Litvinenko, E.V.: Gibridnyj metod upravleniya tochnost'yu resheniya ehkstremal'nyh zadach na grafah. Izvestiya YUFU. Tekhnicheskie nauki **7**(144), 112–116 (2013)
6. Kshemkalyani, A.D., Singhal, M.: Distributed Computing: Principles. Algorithms, and Systems. Cambridge University Press, Cambridge (2008)
7. Khovanskova, V.S., Khovanskov, S.A., Litvinenko, V.A.: Ocenka sokrashcheniya vremeni postroeniya svyazyvayushchih derev'ev cepej s pomoshch'yu raspredelyonnoj vychislitel'noj sistemy. Informatika, vychislitel'naya tekhnika i inzhenernoe obrazovanie **4**(28), 34–42 (2016)
8. Khovanskov, S.A., Norkin, O.R.: The approach to the implementation of the algorithm trace connections of electronic components in a distributed computer networks. Int. J. Innov. Inf. Manuf. Technol. **1**, 26–31 (2014)
9. Foster, I., Zhao, Y., Raicu, I., Lu, S.: Cloud Computing and Grid Computing 360-Degree Compared. http://arxiv.org/pdf/0901.0131.pdf. Accessed 14 Sept 2012
10. Kotenko, V.V., Rumyancev, K.E., Kotenko, S.V.: Identifikacionnyj analiz v informacionno-telekommutacionnyh sistemah. Monografiya. Rostov-na-Donu: Izd-vo YUFU (2014)
11. Madkour, A.M., Eassa, F.E., Ali, A.M., Qayyum, N.U.: Mobile-agent-based systems against malicious hosts. World Appl. Sci. J. **2**(29), 287–297 (2014)
12. Muñoz, A., Pablo, A., Maña, A: Multiagent systems protection. Adv. Softw. Eng. Article ID 281517, 9–12 (2011)
13. Guan, X., Yang, Y., You, J.: POM-a mobile model against malicious hosts. In: Proceedings of High Performance Computing in the Asia-Pacific Region, vol. 2, pp. 1165–1166 (2000)
14. Khovanskov, S.A., Khovanskova, V.S., Litvinenko, V.A., Norkin, O.R.: The algorithm for determining the direction of building relations in a distributed computing system. In: Proceedings of the Congress on Intelligent Systems and Information Technologies, IS&IT 2012, vol. 4, pp. 49–50. Physmathlit, Moscow (2012)
15. Khovanskova, V.S., Khovanskov, S.A., Litvinenko, V.A.: Algoritm organizacii bezopasnyh raspredelennyh vychislenij na osnove mnogoagentnoj sistemy.Izvestiya YUFU. Tekhnicheskie nauki **10**(183), 146–158 (2016)
16. Müller, J.P., Fischer, K.: Application impact of multi-agent systems and technologies: a survey. In: Shehory, O., Sturm, A. (eds.) Agent-Oriented Software Engineering, pp. 27–53. Springer, Heidelberg (2014). https://doi.org/10.1007/978-3-642-54432-3_3
17. Wooldridge, M.: An Introduction to Multiagent Systems, p. 484. Wiley, New Jersey (2012)
18. Khovanskov, S.A., Khovanskova, V.S.: Metody zashchity raspredelennykh vychisleniy. Modernizatsiya sovremennogo obshchestva: problemy. puti razvitiya i perspektivy: sbornik materialov VI Mezhdunarodnoy nauchno-prakticheskoy konferentsii, pp. 104–107. Logos, Stavropol (2015)
19. Khovanskov, S.A., Khovanskova, V.S.: Povysheniye stepeni zashchity raspredelennykh vychisleniy. Sovremennoye sostoyaniye estestvennykh i tekhnicheskikh nauk: Materialy XVIII Mezhdunarodnoy nauchno-prakticheskoy konferentsii (20.03.2015). M.: Izd-vo «Sputnik +», pp. 96–100 (2015)
20. Khovanskova, V., Khovanskov, S.: Mul'tiagentnye sistemy: koncepcii zashchity, Bezopasnost' mul'tiagentnyh sistem. – Technical and natural sciences: Theory and practice. In: Proceedings of Materials of International Scientific e-Symposium, Russia, Moscow, 27–28 March 2015, Kirov, pp. 167–175 (2015)