

# Meta-monitoring system for ensuring a fault tolerance of the intelligent high-performance computing environment

I A Sidorov<sup>1</sup>, T V Sidorova<sup>2</sup> and Ya V Kurzibova<sup>3</sup>

<sup>1</sup>Matrosov Institute for System Dynamics and Control Theory of SB RAS, Lermontov St. 134, Irkutsk, Russia, 664033

<sup>2</sup>Limnological Institute of SB RAS, Ulan-Batorskaya St. 3, Irkutsk, Russia, 664033

<sup>3</sup>Irkutsk State University, Karl Marks St. 1, Irkutsk, Russia, 664003

ivan.sidorov@icc.ru

**Abstract.** The high-performance computing systems include a large number of hardware and software components that can cause failures. Nowadays, the well-known approaches to monitoring and ensuring the fault tolerance of the high-performance computing systems do not allow to fully implement its integrated solution. The aim of this paper is to develop methods and tools for identifying abnormal situations during large-scale computational experiments in high-performance computing environments, localizing these malfunctions, automatically troubleshooting if this is possible, and automatically reconfiguring the computing environment otherwise. The proposed approach is based on the idea of integrating monitoring systems, used in different nodes of the environment, into a unified meta-monitoring system. The use of the proposed approach minimizes the time to perform diagnostics and troubleshooting through the use of parallel operations. It also improves the resiliency of the computing environment processes by preventive measures to diagnose and troubleshoot of failures. These advantages lead to increasing the reliability and efficiency of the environment functioning. The novelty of the proposed approach is underlined by the following elements: mechanisms of the decentralized collection, storage, and processing of monitoring data; a new technique of decision-making in reconfiguring the environment; the supporting the provision of fault tolerance and reliability not only for software and hardware, but also for environment management systems.

## 1. Introduction

The development of a comprehensive monitoring system that would ensure the collection of data from a large number of heterogeneous components included in modern intelligent high-performance computational environment (IHPCE) is a difficult task because of the lack of appropriate standardized formats and protocols for obtaining the necessary information. There is a large number of software solutions that allow us to separately monitor the necessary components of IHPCE. In this regard, the most expedient and promising direction of research in creating integrated monitoring systems for IHPCE is the integration of existing local monitoring systems within a unified meta-monitoring system [1]. At the same time, the local monitoring system acts as a supplier of data. The data collecting and unification, expert analyzing the obtained information, and defining the necessary control actions are assigned to the meta-monitoring system.

The monitoring of the IHPCE components can be conventionally divided into the following categories:

- Monitoring and analysis of the software execution efficiency in IHPCE (control of the current state of computational processes and their individual copies, evaluation of the efficiency of the allocated resource use, etc.),
- Monitoring, testing, and diagnostics of hardware components of nodes (disks, processors, RAM, network interfaces, etc.),
- Monitoring of the IHPCE engineering infrastructure (uninterruptible power supply systems, climatic equipment, fire-fighting systems, etc.),
- Monitoring of the IHPCE computing infrastructure (monitoring of the current load of computing nodes, control of communication, control of transport and service networks, data storage systems, etc.),
- Monitoring of the IHPCE firmware (monitoring of the functioning of system services, task queues, agents, various subsystems, etc.).

The paper suggests an approach to complex monitoring of the IHPCE with multiagent control of computations [2, 3]. It is based on collecting and analyzing data received from a set of local monitoring systems that control the operation of hardware and software components of the environment. In addition, control effects on the IHPCE functioning are developed within the proposed approach.

## 2. Related work

*Tools for monitoring and analyzing the effectiveness of a program implementation in distributed computing environments.* A large number of systems have been accumulated in this category. They include program profilers, tools for monitoring the utilization of computational resources by means of copies of programs executed in distributed computing environments nodes, and tools for monitoring the utilization of network components. The description of these systems is represented in Table 1. Their comparative analysis is given in details in [4, 5].

**Table 1.** Systems for the monitoring and analysis of the program performance

System	Description	Reference
NWPerf	A system for analyzing the performance of a parallel program with the ability to provide data on its individual blocks. It implemented in the Python programming language.	<a href="https://github.com/EMSL-MSL/NWPerf">https://github.com/EMSL-MSL/NWPerf</a>
Arm MAP	A parallel, multi-threaded, and sequential profiler that provides comprehensive analysis on a specific set of metrics. It allows to analyze C, C ++, and Fortran programs.	<a href="https://www.arm.com/products/development-tools/server-and-hpc/forgemap">https://www.arm.com/products/development-tools/server-and-hpc/forgemap</a>
LAPTA	Tools for multidimensional analysis of dynamic characteristics of programs focused on supercomputers. It provides various types of graphical reports.	<a href="http://hpc.msu.ru/node/84">http://hpc.msu.ru/node/84</a>
mpiP	Lightweight profiler of MPI-programs. It enables to analyze programs in C, C ++, and Fortran.	<a href="http://mpip.sourceforge.net/">http://mpip.sourceforge.net/</a>
Integrated Performance Monitoring (IPM)	Advanced profiler of parallel programs with the ability to analyze data transfer processes, memory access, communication network, and disks. It supports various implementations of the MPI library.	<a href="http://ipm-hpc.sourceforge.net/">http://ipm-hpc.sourceforge.net/</a>
Intel® VTune™ Amplifier	Commercial software for analyzing the program performance. Its supports the analysis of the performance and scalability of programs, communication network bandwidth, and data caching.	<a href="https://software.intel.com/en-us/intel-vtune-amplifier-xe">https://software.intel.com/en-us/intel-vtune-amplifier-xe</a>

Tuning and Analysis Utilities	Tools for analyzing and visualizing the execution of parallel programs. It allows to analyze programs in C, C ++, Fortran, UPC, Java, and Python.	<a href="http://tau.uoregon.edu">http://tau.uoregon.edu</a>
HPCToolkit	Tools for the automatic detection of inefficient blocks of a parallel program with the reference to its source code. It focuses on the use in computing environments, including tens and hundreds of thousands of nodes.	<a href="http://hpctoolkit.org">http://hpctoolkit.org</a>
Paraver	A program performance analyzer based on event tracing and allowing detailed analysis of changes and distribution of a specific set of metrics. It supports the prediction of program behavior in different scenarios.	<a href="http://www.bsc.es/paraver">http://www.bsc.es/paraver</a>
Scalasca	Tools for optimizing parallel programs by measuring and analyzing their behavior during the execution. The main emphasis in identifying inefficient blocks is given for the synchronization of parallel programs.	<a href="http://www.scalasca.org">http://www.scalasca.org</a>

From the author's point of view, NWPerf and Paraver open-source packages are the most functional and perspective solutions for analyzing the efficiency of the parallel program execution in distributed computing environments.

*Monitoring, testing, and diagnostics of hardware components of computational nodes.* Unfortunately, only a small number of systems intended for detecting defects in the hardware components of distributed computing environments nodes are known. The description of these systems is represented in Table 2.

**Table 2.** Monitoring, testing, and diagnostic systems for hardware components of nodes

System	Description	Reference
Disparity	A software package that launches an MPI program on target nodes in order to detect possible malfunctions. It supports multiple modes of testing nodes (fast, advanced, etc.).	[6]
Coordinated Infrastructure for Fault Tolerant Systems	The system implements consistent processes for exchanging information about faults between nodes in order to develop a holistic picture of their state as a whole.	<a href="https://wiki.mcs.anl.gov/ciffts/index.php/CIFTS">https://wiki.mcs.anl.gov/ciffts/index.php/CIFTS</a>

The most interesting of them is the Disparty software tool, which allows to detect malfunctions of the components of the computing node during the downtime between the runs of instances of computational processes.

*Systems for monitoring the engineering infrastructure of distributed computing environments.* The systems represented in Table 3 are used to monitor the engineering infrastructure of supercomputer and data processing centers. However, almost all of them are proprietary and tied to the specialized equipment. Thus, they usually do not have the sufficient flexibility for monitoring the IHPCE infrastructure.

**Table 3.** Systems for monitoring the engineering infrastructure of distributed computing environments

System	Description	Reference
ClustrX	A resource management system that supports automatic shutdowns of equipment in the event of a failure of hardware and software components. The description of the monitored components is performed	<a href="http://www.t-platforms.com/products/software/clustrx/productfamily/clustrxwate">http://www.t-platforms.com/products/software/clustrx/productfamily/clustrxwate</a>

	in the Erlang scripting language.	h.html
EMC ViRP SRM	Software for monitoring a corporate storage of information resources and automating the generation of reports about their status. It designed to monitor specialized equipment only.	<a href="http://russia.emc.com/data-center-management/vipr-srm.htm">http://russia.emc.com/data-center-management/vipr-srm.htm</a>
Bright Cluster Manager	A toolkit of automating the creation and control of compute clusters in data centers or cloud platforms. It provides a variety of reports.	<a href="http://www.brightcomputing.com/products">http://www.brightcomputing.com/products</a>
Moab Cloud HPC Suite	Resource management system for supercomputers. It supports automation of planning, control, monitoring, and reporting.	<a href="http://www.adaptivecomputing.com/moab-hpc-basic-edition/">http://www.adaptivecomputing.com/moab-hpc-basic-edition/</a>
IBM cluster system management	Software management complex of large-scale computing clusters. It used predominantly on computing clusters manufactured by IBM.	<a href="https://www-01.ibm.com/common/ssi/cgi-bin/ssialias">https://www-01.ibm.com/common/ssi/cgi-bin/ssialias</a>

At present, non-commercial software products which could provide universal description of heterogeneous engineering equipment of a supercomputer center, creation of new objects, and setting the rules of their monitoring are not known to the author. Monitoring systems Nagios [7] and Zabbix [8] provide a set of tools for monitoring the engineering infrastructure of distributed computing environments, which in each case should be significantly improved.

*Monitoring the computation infrastructure of distributed computing environments.* Today, there are a large number of complex solutions in this category. The most popular complex systems are represented in Table 4.

**Table 4.** Systems for monitoring the computation infrastructure

System	Description	Reference
Ganglia	Scalable distributed monitoring system of computing cluster resources and cloud platforms with a hierarchical structure. It is the most common system used in computer centers.	<a href="http://ganglia.sourceforge.net">http://ganglia.sourceforge.net</a>
Nagios	A monitoring system for computing systems and networks that supports a wide range of functional capabilities for notifying an operator of possible malfunctions. It is often used to monitor telecommunication networks.	<a href="https://www.nagios.org">https://www.nagios.org</a>
Zabbix	A system for monitoring and tracking the state of the software and hardware of telecommunication networks, including network servers and services. It supports various databases for the data storage.	<a href="https://www.zabbix.org">https://www.zabbix.org</a>
ZenOSS	Monitoring software package that supports the automatic detection and configuration of monitoring parameters of various systems. It focused on cloud applications.	<a href="https://www.zenoss.com/">https://www.zenoss.com/</a>
Ovis2	A comprehensive monitoring system that provides high scalability and integration with other monitoring tools.	<a href="http://ovis.ca.sandia.gov/">http://ovis.ca.sandia.gov/</a>

The most popular system in this category is Ganglia. However, its standard set of functions does not meet the growing needs for monitoring the computation infrastructure of distributed computing environments. Often, the limited set of functions leads to the need for additional monitoring systems, such as Zabbix or Nagios. The most promising system in this category, from the author's point of

view, is Ovis2, which provides high scalability and wide possibilities for connecting various data sources.

*Monitoring middleware of distributed computing environments.* This category includes Nagios and Zabbix monitoring systems described above, as well as more specialized tools represented in Table 5.

**Table 5.** Systems for monitoring middleware of distributed computing environments

System	Description	Reference
Xymon	Software complex for monitoring the process of functioning of the system services of computing systems. The basic principle is to check the availability of network ports.	<a href="http://xymon.sourceforge.net/">http://xymon.sourceforge.net/</a>
Failure Testing Service	Toolkit for testing applications in the cloud environment. It allows testing in the framework of continuous integration.	[9]
CloudRift	System for testing microservice applications for cloud platforms. It enables to identify failures in individual segments of cloud programs.	[10]

In addition to the aforementioned systems, environment administrators usually develop specialized utilities to track the correct functioning of individual subsystems included into middleware. Such utilities are often implemented in the form of scripts running on schedule using the CRON service.

The results of a comparative analysis of the functionality of the developed meta-monitoring system with the capabilities of the key local monitoring systems described above is represented in Table 6. These results show the obvious advantages of the meta-monitoring system.

**Table 6.** Comparison of the developed meta-monitoring system with local monitoring systems

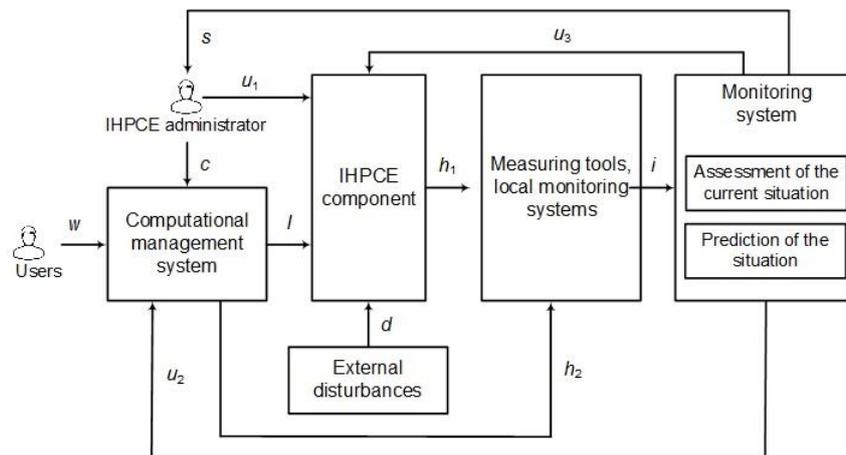
Feature	Lapta	Disparity	ClustrX	Zabbix	Ganglia	Nagios	Ovis2	Xymon	Meta-monitoring system
Analysis of the effectiveness of program implementation	+	-	-	-	-	-	-	-	+
Monitoring and diagnostics of computing nodes	-	+	-	+	+	+	+	-	+
Engineering infrastructure monitoring	-	-	+	+	-	+	-	-	+
Computing infrastructure monitoring	+	-	-	+	+	+	+	-	+
Testing of services and control subsystems	-	-	-	-	-	-	+	+	+

### 3. Scheme of the environment component control

The general scheme of the IHPCE component control using the meta-monitoring system is shown in Fig. 1. In this scheme, the IHPCE component acts as a control object. The administrator configures the

operation of the job management system, which handles the flow  $w$  of user tasks, using the vector  $c$  configuration parameters. He also creates affects  $u_1$  on the control parameters of the IHPCE component. The task management system determines the computational load  $l$  of the component in accordance with the flow  $w$ . The external disturbances  $d$  of the environment arise because of the actions of local users of the environment or events that occur during the operation of the engineering infrastructure.

The monitoring system collects the information  $i$  about the IHPCE component and computation management system with the help of measuring tools and local monitoring systems. This information is formed on the basis of the characteristics  $h_1$  of component status and the information  $h_2$  about functioning the computation management system. Based on the collected information, the meta-monitoring system assesses the current computational situation, predicts its development, and forms the control effects  $u_2$  and  $u_3$  on the IHPCE component and the computation management system in order to prevent or partially eliminate failures of hardware and software. In the event of a critical situation when such actions cannot be performed automatically, the meta-monitoring system sends the corresponding notification  $s$  to the environment administrator.



**Figure 1.** Scheme of the IHPCE component control.

#### 4. Meta-monitoring system architecture

The meta-monitoring system architecture is based on the principles of organization of multigenic systems [11] and includes the following main components:

- Interface for the user access to components of the meta-monitoring system, which allows to work with them in batch or interactive modes,
- Access level subsystem, which performs the differentiation of the access to the requested data,
- Agents that operate on the IHPCE nodes and carry out the data collection and processing. In addition, they interact with other agents.

A software agent installed in the IHPCE nodes is a program executed in the background mode. The agent collects data from the local monitoring systems, unifies the received data, and saves it in the local DBMS. It includes a subsystem for the failure diagnostics and environment reconfiguration. In addition, the agent has control subsystem that performs the execution functions of control actions and interaction with the agents of upper levels.

The possibility of data analysis and making necessary decisions on the side of the computing node is a key difference between the presented approach and existing solutions. In the well-known monitoring systems, the client installed in the nodes performs the functions of data collection and their periodic transmission to the control node. The centralized processing and analysis of the collected data are performed on the control node. This creates an additional extra load on the network protocol stack, which also requires CPU time, and has problems with scalability.

Agents of the developed meta-monitoring system consume about 37% less processor time in comparison with the Ganglia agents at the same frequency of interrogation of sensors. They transfer data to the central node of IHPCE or neighbouring agents only if necessary or on request. The processor time spent for the data analysis on the node is less the time spent on formation of network packets and control of their integrity. Thus, this reduces the load on the network stack and the central node of the monitoring system. In addition, it is possible to reduce the negative impact of the monitoring agent on the computational tasks performed in the nodes.

The measurement of node state metrics (processor, memory, etc.) is implemented by the functions of the SIGAR library [12]. This library is cross-platform. It allows unified access to the necessary information.

Integration of the meta-monitoring system with local monitoring systems is carried out in the specialized language that is a subset of the ECMA Script language [13]. This specialized language supports the call of external commands, network interaction, processing of output stream, regular expressions, and a number of other mechanisms for rapid implementation of non-standard sensors.

The subsystem of the data collection and processing is based on the principles of Round-robin Database. A volume of such databases does not change with time. Their fixed size is achieved due to the predefined number of records used cyclically to store data.

Nowadays, there are many implementations of cyclic databases (MRTG, RRDtools, etc.). At the same time, the performed tests have revealed a number of drawbacks in such systems related primarily to unacceptable performance in reading/writing data. We tried to create a cyclic database prototype of on the basis of the lightweight embedded relational database SQLite. However, the conducted experiments have shown its lower performance in comparison with RRDtools.

In this regard, we have made a decision to create own implementation of the cyclic database, which uses the specialized XML based format for storing structured information. We developed the mechanisms of data reading and writing, aggregation of data for a certain time interval, displacement of outdated data, data sampling in accordance to determined criteria, and means of data caching in memory. The developed database has demonstrated its efficiency in comparison with MRTG and RRDtools.

## 5. Practical application

The developed methods and tools for meta-monitoring IHPCE have been successfully tested in the Irkutsk Supercomputer Center of SB RAS [14]. IHPCE included three pools of nodes:

- 20 computational nodes with Intel Xeon E5-2695 v4 "Broadwell" processors with a total number of 720 cores,
- 10 computing nodes with AMD Opteron 6276 "Bulldozer"/"Interlagos" processors with the total number of cores 320,
- 20 computing nodes with Intel Xeon 5345 EM64T 2.33 GHz "Clovertown" processors with the total number of cores 160.

During the study of IHPCE by the meta-monitoring system, a list of hardware and software resources whose components were in a state close to critical or functioning with errors was revealed. The list of nodes, diagnostic messages of the meta-monitoring system, and node state description corresponding to the detected faults are given in Table 7.

**Table 7.** Meta-monitoring results

Pool number	Node number	Diagnostic message	Node status description
1	4	«warning node-4.matrosov.icc.ru loadavg5 43»	The average node load for the last 5 minutes exceeded 43 points.
1	13	«critical node-13. matrosov.icc.ru cpu-sys-p 77»	At the node, the loading of processor cores by the tasks of the operating system prevails.

2	112	«critical sm112.matrosov.icc.ru filesystem /home wtime 583816»	Writing to the /home directory is too slow.
1	14	«error node-14.matrosov.icc.ru down»	Node not available.
2	102	«critical sm102.tesla.icc.ru memory-used-ten 97»	On the node RAM is used by 97%.
3	7	«error node node-7.blackford.icc.ru filesystem /store du-free 0»	The node has run out of free disk space in the /store directory.

The analysis of data on the state of the IHPCE hardware and software resources collected by the meta-monitoring system revealed the inefficient operation of user applications, optimized the load of computing resources, and improved the reliability of the IHPCE operation. For example, when solving an important practical task of annotating the *Synedra acus* genome with the help of the MAKER software package [15], the prevalence of read-write operations in the network directory over computational operations performed on processor cores was revealed. In accordance with the detected inefficient use of resources, the package parameters indicating the location of directories for writing the results of calculations were automatically corrected. Local directories of nodes (for example, /tmp) were assigned as such directories, which allowed to significantly increase the efficiency of using processor cores in this package by more than 30%.

Another illustrative example of the successful applying of the developed meta-monitoring system is a significant improvement in power saving for one of the IHPCE pools, the nodes of which are outdated, but continue to be operated by users. These users solve their problems with the help of applications specialized in software and hardware features of the nodes in this pool.

The PBS Torque [16] system is used to control the completion of tasks in this pool. In order to automate the power consumption control in pool nodes the following meta-monitoring operations and rules have been developed:

- Operations to collect data from the sensors of the PBS Torque system about the used resources and tasks set in the queue, to enable and disable pool nodes, to change the pool configuration parameters,
- Set of output rules for the expert subsystem that define the conditions for applying these operations.

When a task is added to the PBS Torque queue on the pool's management node, the number of nodes in the pool required to solve it is automatically enabled using the Intelligent Platform Management Interface (IPMI) protocol. Then they are quickly tested and computational processes in these nodes are launched. After the task solution is completed, new tasks are waited for a specified period of time (usually 1-2 hours). In the case of their absence, the nodes are automatically switched off using the same IPMI protocol. As a result of automation in this pool with the help of meta-monitoring system, their daily power consumption was reduced by 34%.

The meta-monitoring system is great importance for evaluating the efficiency of the processes of functioning of the multi-agent system of distributed computing management [1, 2]. Permanent monitoring of the work of this multi-agent system has shown its higher fault tolerance to failures of software and hardware resources of IHPCE in comparison with other similar systems [17].

## 6. Conclusions

The paper addresses the relevant problem of monitoring the high-performance computing systems and ensuring their fault tolerance. We proposed a new approach to monitoring IHPCE (the environment with multi-agent management of distributed computing) and developed the specialized meta-monitoring system. The developed meta-monitoring system provides control, diagnostics, localization, and troubleshooting of the IPCE components. In addition, automatic reconfiguration of IHPCE in a finite number of steps enables minimizing the time of diagnosis and troubleshooting through the

parallel execution of their operations. The fault tolerance increase of nodes by means the preventive diagnosis and troubleshooting improves the reliability and efficiency of IHPCE.

The novelty of the presented approach includes the following elements:

- Special mechanism of decentralized collection, storage, and processing of monitoring data,
- Decentralized decision-making for the environment reconfiguration,
- Ensuring the fault tolerance and reliability for both the hardware and software of the environment, and the environment management system itself.

**Acknowledgment.** The study is supported by the Russian Foundation of Basic Research, project no. 19-07-00097 (reg. no. AAAA-A19-119062590002-7). This work was also supported in part by Basic Research Program of SB RAS, project no. IV.38.1.1 (reg. no. AAAA-A17-117032210078-4).

## References

- [1] Bychkov I V, Oparin G A, Novopashin A P 2015 Agent-Based Approach to Monitoring and Control of Distributed Computing Environment *Lecture Notes in Computer Science* 253-257
- [2] Bychkov I, Feoktistov A, Sidorov I, Kostromin R 2017 Job Flow Management for Virtualized Resources of Heterogeneous Distributed Computing Environment *Procedia Engineering* **201** 534-542
- [3] Feoktistov A, Sidorov I, Tchernykh A, Edelev A, Zorkalzev V, Gorsky S, Kostromin R, Bychkov I, Avetisyan A 2018 Multi-Agent Approach for Dynamic Elasticity of Virtual Machines Provisioning in Heterogeneous Distributed Computing Environment *Proc. of the Int. Conf. on High Performance Computing and Simulation* (IEEE) pp 909-916
- [4] Benedict S 2013 Performance issues and performance analysis tools for HPC cloud applications: a survey *Computing* 89-108
- [5] Mohr B 2014 Scalable parallel performance measurement and analysis tools – state-of-the-art and future challenges *Supercomputing frontiers and innovations* **1(2)** 108-123
- [6] Desai N, Bradshaw R, Lusk E 2008 Disparity: Scalable Anomaly Detection for Clusters *Proc. of the 37th International Conference on Parallel Processing* pp 116-120
- [7] Josephsen D 2007 *Building a Monitoring Infrastructure with Nagios* p 255
- [8] Zabbix. Available at: <https://www.zabbix.org> (accessed: 19.06.19)
- [9] Haryadi S G 2011 FATE and DESTINI: a framework for cloud recovery testing *Proc. of the 8th USENIX conference on Networked systems design and implementation* pp 238-252
- [10] Savchenko D, Radchenko G, Taipale O 2015 Microservices validation: Mjolnir platform case study *Proceedings of the 38th International Convention MIPRO* (IEEE) pp 248-253
- [11] Wooldridge M, Jennings N 1995 Intelligent Agents: Theory and Practice. *The Knowledge Engineering Review* **10(2)** 115-152
- [12] System Information Gatherer and Reporter API. Available at: <https://github.com/AlexYaruki/sigar> (accessed: 19.06.2019).
- [13] Standard ECMA-262: ECMAScript Language Specification. Available at: <http://es5.javascript.ru/> (accessed: 19.06.2019)
- [14] Irkutsk Supercomputer Center of SB RAS. Available at: <http://hpc.icc.ru> (accessed: 19.06.2019).
- [15] MAKER – genome annotation pipeline. Available at: <http://gmod.org/wiki/MAKER> (accessed: 19.06.2019).
- [16] PBS Torque. Available at: <https://github.com/adaptivecomputing/torque> (accessed: 19.06.2019).
- [17] Bychkov I, Feoktistov A, Kostromin R, Sidorov I, Edelev A, Gorsky S 2018 Machine Learning in a Multi-Agent System for Distributed Computing Management *Data Science. Information Technology and Nanotechnology 2018* (CEUR-WS Proceedings) **2212** 89-97