ISSN 0005-1179, Automation and Remote Control, 2015, Vol. 76, No. 11, pp. 2000–2010. © Pleiades Publishing, Ltd., 2015.
Original Russian Text © I.V. Bychkov, G.A. Oparin, A.G. Feoktistov, V.G. Bogdanova, A.A. Pashinin, 2015, published in Avtomatika i Telemekhanika, 2015, No. 11, pp. 118–131.

## INTELLECTUAL CONTROL SYSTEMS

# Service-Oriented Multiagent Control of Distributed Computations

**I. V. Bychkov, G. A. Oparin, A. G. Feoktistov, V. G. Bogdanova, and A. A. Pashinin**

*Matrosov Institute for System Dynamics and Control Theory, Siberian Branch, Russian Academy of Sciences, Irkutsk, Russia*
*e-mail: idstu@icc.ru, oparin@icc.ru, agf@icc.ru, bvg@icc.ru, apcrol@gmail.com*

Received November 19, 2014

**Abstract**—Consideration was given to the multiagent methods and toolkits for efficient control of the job flows generated by the service-oriented applications. These designs were integrated within the framework of a unique technology supporting automation of solution of large scientific problems in the up-to-date cluster Grid whose computing nodes (clusters) can be of an involved hybrid structure. The novelty and practical significance of the methods and tools described in the paper lie in essential extension of the functionality of the computation control system of the cluster Grid, as compared with the existing ones, distribution and sharing of the Grid resources at various levels of job execution, and possibility of integrating intelligent computation control tools in the problem-oriented applications.

**DOI:** 10.1134/S0005117915110090

## 1. INTRODUCTION

Analysis of the tendencies in the construction technologies of the high-performance distributed systems both in this country and abroad suggests the need for integration of the methods and tools enhancing efficiency of loading the computation resources at solving large scientific problem, comprehensive allowance for the specificity of the knowledge domains in the course of solving these problems with the aid of user applications, and ensuring flexibility of applications use on the basis of the service-oriented approach.

For the time being, a wide spectrum of tools for construction of the service-oriented distributed computing environments has been developed including, for example, the general-purpose tools such as the Amazon Elastic Compute Cloud, Google AppEngine Microsoft Windows Azure, and Manjrasoft Aneka [1], specialized systems Globus Toolkit [2] and Unicore [3], as well as the programming language Opa [4]. Among the domestic tools for design of the service-oriented environments, the MathCloud system [5] and systems relying on the concept of Intelligent Problem Solving Environment (iPSE) [6] deserve mentioning. In the systems for construction of the service-oriented distributed computing environments, emphasis is usually made on simplification of the process of service design. In such systems, computations are controlled using the traditional metaplanners such as the GridWay [7] or Workload Management System [8], or with the use of specialized system software.

The traditional metaplanners do not sufficiently allow for the specific requirements of the users on the distributed resource. Therefore the users and administrators of the distributed computing environments have to tackle "manually" the sometimes contradictory problem of selection and allocation of resources by user's specification of each of its jobs and administrator's adjustment of the metaplanner configuration parameters with the aim of specifying the policies concerning the users' resources and jobs.

Complexity of the aforementioned problems gives rise to the need for automation and intellectualization of the processes of their solution. The multiagent systems (MAS) of computation control represent a widely used in practice approach to this problem [9]. Improvement of the quality of control decisions is often attained in MAS by using economical mechanisms for regulation of the demand for and the proposal of the resources of a distributed computing environment [1]. Two basic approaches to the multiagent control of computations can be specified [11]: interaction of MAS with local managers of the environment node resources with the aim of optimizing the use of resources and integrating the user application with the multiagent system of resource selection in order to enhance the efficiency of problem solution by the application.

In the former case, the use of MAS presupposes as a rule replacement of the metaplanners by special computation control agents. Thereby, each user becomes, independently of its will, a global user interacting with the resources of the distributed computing environment only through MAS. This constrains the potentialities of a wide circle of local users willing to solve their problems at the particular nodes of the environment without an "intermediary." In the latter case, in the presence of numerous applications of different users efficiency of the computation control systems can be appreciably reduced owing to the competition of the application agents for the common shared resources.

The present paper considers a MAS enabling integration of both aforementioned approaches to control of computations in the cluster Grid-system of the computing kind having nodes (clusters) of complex hybrid structure. The hybrid cluster comprises computing modules (hardware components) supporting various technologies of parallel programming and differing in their computing characteristics.

## 2. CONTROL OF COMPUTATIONS AT THE LEVEL OF GRID-SYSTEMS

Control of computations at the Grid-system level is realized by the MAS with a given organizational structure. The agents' actions are coordinated using the general group behavior rules. The agents operate in compliance with the given roles, and in the virtual community of the agents certain rules are defined for each role. MAS includes resource allocation agents and a controlling agent. The resource allocation agents can be united into virtual communities (VC). In various VCs arising in the MAS, the agents can coordinate their actions through cooperation or competition.

At the level of Grid-system, the purpose of MAS is to distribute the job flows arriving to the system so as to maintain the system operational performance within the limits set by the Grid-system administrator. The setting represents a specification of the process of problem solution carrying the information about the required computing resource, executed applied programs, input/output data, as well as other necessary data. All jobs are classified according to their computational characteristics [12]. Among the characteristics of the Grid-system performance, there is the node economy of the Grid-system and the system itself as a whole, the mean indices of the queuing time, the coefficients of successful completion, and the mean cost of executing jobs of various classes at the nodes of the Grid-system and in the system as a whole. The block diagram of the computation control system at the level of the Grid-system is depicted in Fig. 1.

This diagram shows the Grid-system as an object of control, its nodes being represented by heterogeneous computing clusters (CC), hybrid one including. The flow $w_1$ of the user jobs of the Grid-system and the flow $w_2$ of the local CC users are the external disturbances of the control object. The results of distributions $d_1$ and $d_2$ of the flows $w_1$ and $w_2$ over CC are, respectively, the control actions of MAS and local CC users on the control object. The vector $r_1$ of parameters of the administrative CC policies is the master control of the object of control. The resource allocation agents catch the jobs of flow $w_1$ for more detailed adjustment of the requirements on the computing system contained in the jobs. Therefore, the flow $w_1$ is modified into the flow $w_1'$. The

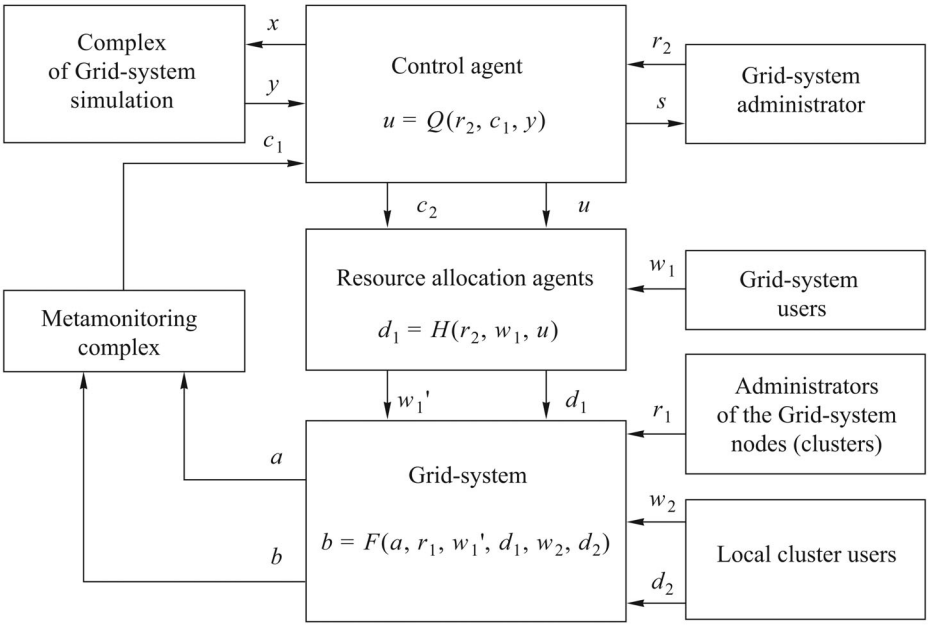


**Fig. 1.** Block diagram of the computation control system.

distribution $d_1$ of the flow $w_1'$ is done by the resource allocation agents on the basis of the economical mechanisms controlling demand and proposal of these resources [13]. The distribution $d_2$ of the flow $w_2$ is defined by the local CC users. The flows of jobs $w_1$ and $w_2$ are characterized by *dynamism* because the power and composition of the job flow vary in time, stochasticity because the job flow presupposes origination of random events, *heterogeneity* because the jobs correspond to different classes of problems and differ one from another in their specificity, *lack of feedback* because the number of jobs arriving within one time interval is independent of the number of jobs arriving within another time interval, *nonordinarity* because two or more jobs can arrive within the same time interval, and *stationarity* because the number of events arriving over a certain time interval depends on the length of this interval and is independent of its start instant.

The information about the computational characteristics of the Grid-system nodes is collected by the instrumentation of the metamonitoring complex [14] in the form of a file-oriented data structure $a$. The information about the current indices of the volumes of computational work at the executed and queued nodes of the Grid-system is also collected by the metamonitoring complex as a file-oriented data structure $b$. It is assumed that there exists some abstract relation $b = F(a, r_1, w_1', d_1, w_2, d_2)$ between the components of the structure $b$, on the one hand, and the computing characteristic of the nodes $a$, master control $r_1$ of the control object, job flows $w_1'$ and $w_2$, and the distributions $d_1$ and $d_2$, on the other hand. For different components of the structure $b$, this relation is representable by functional, statistical, ambiguous, or other map. In particular, for heterogeneous nodes the volume of the same computational work is determined with regard for the distinctions of their computational characteristics from the corresponding characteristics of the virtual reference node of the Grid-system.

The acquired information is transmitted to the controlling agent as the vector $c_1$ of aggregated indices of operation of the control object at its request. The requests of the controlling agent to the monitoring system are sent at a certain period of discreteness $T_1$ whose value is selected so as not to overload the computing environment of the Grid-system by information acquisition and at the same time to fix with the desired precision the instants when the indices of operation of the control

object approach their limit values. Part of information represented by the vector $c_1$ and topical for the resource allocation agents is transmitted immediately to these agents as the vector $c_2$.

The vector $r_2$ of the parameters of administrative policies of the Grid-system is the master action for the control agent. On the basis of information presented by the vectors $c_1$ and $r_2$, with the use of the simulation complex of the Grid-system the control agent at the preset time instants forecasts the dynamics of the performance indices of the control object for a certain time interval. The results of modeling are used to generate the vector $u$ of control actions on the operational algorithms of the agents of allocation of the VC resources by parametric adjustment of the algorithms. The following parameters of the operational algorithm of the agent of resource allocation are the elements of the vector $u$: value of penalty for "greedy" or "lazy" agents relative to the average volume of computational work at the node represented by the VC agents and the permissible deviation from this value. Simulation is initiated by the control agent with a certain period of discreteness $T_2 > T_1$. As soon as the vector $u$ of control actions is generated, it is transmitted to all VCs. The parametric adjustment of the operational algorithms of the resource allocation agents of a particular VC is carried out by using the established control actions with regard for the weight coefficients reflecting the computational characteristics of the nodes of this VC.

So, let $x$ and $y$ be the vectors of input and observed variables of the simulation model of the Grid-system. The observed variables are the performance indices of the Grid-system operation. The elements of the vectors $x_i$, $i = \overline{1, n_x}$, and $y_j$, $j = \overline{1, n_y}$, have their proper areas $X_i$ and $Y_j$ of permissible values. It is supposed that the effects of influence of the input parameters on the observed variables were examined in advance by the factor analysis at constructing and testing the Grid-system simulation model [15]. It is also assumed that the Grid-system administrator assigns to each $j$th element of the vector $y$ a criterion for computation of the estimate $\hat{y}_j$ of quality of importance of this element (tendency of its value to the minimum or maximum on the set of compared values) and its limit values $y_j^{\min}$ and $y_j^{\max} \in Y_j$. Some elements of the vector $x$ play the part of modified variables, make up the subset $X^*$, and are identified with the elements of the vector $u$: $u_q \equiv x_i$, $q = \overline{1, n_u}$, $i \in \overline{1, n_x}$, $1 \leqslant n_u < n_x$. As a rule, at solution of practical problems of control of computations in the Grid-system, in the course of simulation it is advisable to use $q \leqslant 8$. The number of values of the modified variable is determined from

$$\left( t_m \times \prod_{q=1}^{n_u} z_q \right) / n_c \leqslant T_2,$$

where $t_m$ is the mean time of running the simulation model defined by the metamonitoring complex on the basis of the computational history of model runs, $z_q > 0$ is the number of modified values of the $q$th variable, $n_c$ is the number of kernels of the node accommodating the control agent, $n_u < n_c$. The basic values corresponding to the accepted by default values of the configuration parameters of the actual computation control system (metaplanner, for example) of the Grid-system are used as the initial values of the modified variables. The rest of the values of the modified variables are selected from the corresponding areas of the permissible values with regard for the effects of the modified variable on the observed variables. The value of nonmodified input variables that are elements of the vector $x$ are given on the basis of the corresponding numerical information represented by the vectors $r_2$ and $c_1$.

Modeling simulates operation of the Grid-system by carrying out parallel multivariant calculations with the model running for each combination of the values of modified variables, and generating the set $V$ of variants of values of the observed variables. The value of $y_{jk} \in Y_j$ is an element of the $k$th variant $v_k \in V$ for the variable $y_j$, $j = \overline{1, n_y}$, $k = \overline{1, n_v}$. Choice of the subset $V^* \subseteq V$ of variants of values of the observed variables from the set $V$ with the aim of further determination of the values of elements of the vector $u$ is a multicriteria process. The control agent selects variants

for the subset $V^*$ on the basis of the lexicographic method if the administrator of the Grid-system can order in significance the observed variables or, otherwise, on the basis of the majority method.

The lexicographic method for choice of the variants of values of the observed variables uses the following rule for multicriteria choice described in [16]:

$$V^* = \Big\{ v_k \in V : (\forall v_l \in V \; \exists p \in \overline{1, n_y - 1} :$$
$$(\hat{y}_{1k} = \hat{y}_{1l}) \wedge \ldots \wedge (\hat{y}_{pk} = \hat{y}_{pl}) \wedge (\hat{y}_{(p+1)k} > \hat{y}_{(p+1)l})) \Big\},$$

where

$$y_j^{\min} \leqslant y_{jk} \leqslant y_j^{\max}, \quad j = \overline{1, n_y}, \quad k \in \overline{1, n_v}, \quad l \in \overline{1, n_v}, \quad k \neq l.$$

The majority method for choice of the variants of values of the observed variables uses the following rule for multicriteria choice [16]:

$$V^* = \left\{ v_k \in V : \left( \neg \exists v_l \in V : \sum_{j=1}^{n_y} \operatorname{sgn}(\hat{y}_{jl} - \hat{y}_{jk}) > 0 \right) \right\},$$

where $\operatorname{sgn}(0) = 0$, $y_j^{\min} \leqslant y_{jk} \leqslant y_j^{\max}$, $k \in \overline{1, n_v}$, $l \in \overline{1, n_v}$, $k \neq l$.

Use of the aforementioned methods of multicriteria choice is due to the fact that they are less complicated and, as compared with other existing methods for solution of such problem, from the computation standpoint are readily realizable; and the control agent needs minimal additional information from the Grid-system administrator.

Let a single $k$th variant of $v_k$ values of the observed variables be obtained as the result of generating the set $V^*$ to which the $k$th variant of values of the modified variables of the vector $x$ corresponds uniquely. By selecting among them the values $x_{ik}$ such that $x_i \in X^*$, we get the values of the elements of the vector $u$. If the resulting subset $V^*$ contains more than one variant of values of the observed variables, then the final choice of the single variant $v_l$ is done randomly. For $V^* = \emptyset$, the control agent generates the signal $s$ requiring from the administrator of the Grid-system new master controls to correct the current administrative policies of the Grid-system.

Therefore, the resource allocation agents perform control actions $d_1 = H(c_2, w_1, u)$ on the Grid-system where the control action $u = Q(r_2, c_1, y)$ is intended to enhance the quality of decisions made by the resource allocation agents through influencing the degree of agent intentions to execute jobs of different classes. The relations $H$ and $Q$ have the same nature as the considered relation $F$. It deserves noting that the control object continued operation at failure of any MAS agent, including the control one; at that only reduction in performance is possible.

At control of job flows by the resource allocation agents at the level of the Grid-system, the time to execute individual applications may increase because these agents fail to allow for some important features of the problem solution and user preferences for resource. In the following section we represent new additional agent tools for planning computations and allocation of resources at the level of applications which enable one to solve this problem to some extent.

## 3. CONTROL OF COMPUTATIONS AT THE LEVEL OF APPLICATIONS

Additional tools to control computations at the application level represent a virtual application community (VAC) designed to organize in the Grid-system parallel execution of the local user application. The main purpose of VAC is to select the least loaded CCs, activate a parallel application, monitor and transmit to the user the results of computations. VAC comprises the user agent, classification and planning agents, manager agent, and varying dynamically set of the local

agents. The three first agents receive and classify [12] the user request, plan computations, and generate a new—relative to the above scheme of the computation control system—job flow $w_3$ generated relying on the user application. The flow $w_3$ is transmitted to the manager agent carrying out its distribution $d_3$ to the local agents. The manager agent receives from the control agent the information about the computational characteristics of nodes and current indicators of the volumes of CC computational work in the form of the vector $c_2$ required for distribution. The manager agent is also responsible for restarting automatically the problem with new parameters (for a certain class of problems) and monitoring of solution of the user problem. The local agents are responsible for sending jobs to the local CC control system, analyzing the current CC state, and transmitting the results of job execution to the manager agent.

To distribute tasks to the local agents, used is the tender model where the computational work are used as lots, and the representative of the computing resources pretending to do computational work, as participants. The Vickrey auction considered in detail in [13] and used to realize the tender model enables on to reach a coordinated stable state of the auction participants by the end of bidding, as well as to provide a better balanced distribution of the jobs of the local users of clusters. Occurrence in the Grid-system of free resources in the course of solving the user problems entails redistribution at the system nodes of the MAS-controlled jobs.

The user application is formed as a Grid-service. The method of generation of the application Grid-services used below is based on a combination of the technologies of Web Services Resource Framework (WSRF) [17] and patterns [18] of interaction with the local CC resource managers. The instrumental environment High-performance Computing Service-oriented Multiagent System (HpcSoMaS) Framework developed by the present authors on the basis of the listed above technologies is used to develop VAC. The instrumental environment includes the library of classes and agent generation utilities, the pattern library to create services on the basis of the Representational State Transfer (REST) architecture [19] and the Simple Object Access Protocol (SOAP) [2]; ready-made services developed on the basis of the above libraries, specifications of service settings in the XML, patterns for the components of exchange of the XML messages and interaction with the local resource managers, and an instrument for generation of the description of the Grid-service in Web Services Description Language (WSDL) [21]. To start an application in the Grid-system, a job is generated for one of the local CC resource managers used in the system. Both the traditional metaplanners and the local CC control systems such as the Portable Batch System (PBS) [22] or Condor [23] can be used as such manager. Therefore, it is required to solve two basic problems within the framework of the proposed method: (1) establish a description of the service for application in WSDL and (2) convert the user request to the service into a computational job for the Grid-system.

On the whole, the service enables the user to formulate problems, configure adjustment and input of the source data for the application, obtain the results of computations, view current CC load and acquire information messages both by the mail and WEB interface facilities. Along with the listed above functions of system nature, in the designed VAC one can take into consideration the possibilities specified by the knowledge domain of the user application. Examples of such VACs are given below.

## 4. RESULTS OF STUDIES

A service to solve the problem of constructing the stability region in the space of two selected parameters $K$ and $T$ of the controller of a closed-loop control system obeying the differential equation $\frac{dx}{dt} = Ax$, where the elements of the matrix $A$ depend on the parameters $K$ and $T$, was realized as the first example of using VAC. This problem comes to solving a set of independent subproblems, that is, multivariant computations, for determination of stability of the matrix $A$

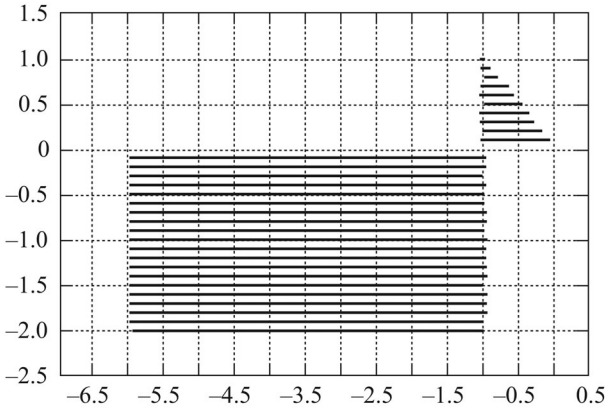


**Fig. 2.** Example of graphic image by the service of the stability region.

with variations of values of the parameters $K$ and $T$ within the given ranges $K_{\min} \leqslant K \leqslant K_{\max}$ and $T_{\min} \leqslant T \leqslant T_{\max}$ with the respective steps $\Delta K$ and $\Delta T$. By varying the parameters $K$ and $T$, constructed is a numerical grid underlying the set of subproblems. Execution of the job for solution of each subproblem is the necessary condition for solving the original problem. The eigenvalues of an arbitrary dense matrix are calculated using the algorithms represented in [24]. The result of the service represents tabular data and graphic image of the stability domain. For example, the stability region of the system

$$\frac{dx}{dt} = Kx + Ty, \quad \frac{dy}{dt} = x - z, \quad \frac{dz}{dt} = -x + y,$$
$$-2 \leqslant T \leqslant 1, \quad -6 \leqslant K \leqslant 0$$

is depicted in Fig. 2. For a more detailed description of the user work with this service the readers are referred to [25].

A service to solve the SAT problem (Boolean satisfiability problem) by multivariant computations on the basis of decomposing the original problem was realized as the second example. Decomposition of the SAT problem into an arbitrary number of independent subproblems whose joint solution provides an answer to the original problem is carried out using the method of splitting [26]. Each independent subproblem can be solved at its node of the Grid-system. In this case, the multivariant computations can be run in the mode of dynamic choice of resources. In the computational experiments, used were both the existing open-licence SAT solvers and those developed by the present authors [27, 28].

Using the WEB interface (user agent), the user formulates the problem: transmits the file with the original Boolean function in the conjunctive normal form (CNF) and then initiates problem solution. On the basis of the original data the user agent generates a set of subproblems and transmits them to the computation planning agent generating a job for each subproblem. Then, the information about the job pool is sent to the manager agent distributing jobs between the local agents. The local agent starts one of the SAT solvers to execute the job. The monitoring agents follow the state of each job. If additional free computing resources occur in the cluster Grid-system and the manager agent's job pool is empty, then the process of computations is suspended, one of the active subproblems is removed from the processor, decomposed further, additional jobs are generated, and computations are resumed. If one of the jobs has found solution, then, depending on the state of execution, the rest of the jobs either are removed from solution or dequeued. If after execution of all jobs no solution is found, then the user is informed about completion of problem solution with the answer "unsat."

**Table 1.** Comparison of the time of solving the SAT problem at distribution of the application jobs flow by the user ($d_2$) and VAC manager agent ($d_3$)

| | | Mean time of solving the sat-problem, s | | Minimal time of solving the sat-problem, s | |
|---|---|---|---|---|---|
| CNF | Number of variables/disjuncts | $d_2$ | $d_3$ | $d_2$ | $d_3$ |
| knight8 | 4096/491 024 | 183.0 | 132.0 | 61.0 | 0.6 |
| knight9 | 6561/1 007 603 | 499.0 | 359.0 | 282.0 | 199.0 |
| knight10 | 10 000/1 913 276 | 3599.0 | 2464.0 | 651.0 | 276.0 |

**Table 2.** Time of solution of the SAT problem obtained using hpcsat

| CNF | Variables/disjuncts | Time on 256 flows, s | Time on 1024 flows, s |
|---|---|---|---|
| rbsat-v1150c84314gyes7.cnf | 1150/84 314 | 1316 | 514 |
| toughsat_factoring_inf.cnf | 2878/15 516 | 2147 | 553 |
| gss-25-s100.cnf | 31 931/96 111 | 1985 | 126 |
| b04_s_2_unknown_pre.cnf | 123 133/801 488 | 2988 | 1640 |

For efficient use of the resources of Grid-system in the course of solving the SAT problem, developed were the hybrid methods of solution using concurrently more than one programming technologies such as the MPI technologies for organization of data exchange between the CC nodes, OpenMP technologies (of Pthread libraries) for organization of several processor flows within the framework of one CC node, and CUDA technologies for organization of computations on the graphic accelerators. Methods of dynamic redistribution of the job flows were also developed for the central and graphic processors.

Table 1 shows the results of solving the well-known Euler problem of the move of chess knight with indication of the average and minimal times for a series of 100 runs. Table 2 presents the results of solving a series of SAT problems with the use of the service of hybrid solver hpcsat developed at IDSTU SO RAN. The results are given for the problems which were not solved in 5000 s by serial and parallel SAT solvers that took part in the SAT Competition13. The results of the computing experiments suggest the following. In the first example, the application service is intended for realization of the multivariant computations with the use of the static selection of resources. All jobs generated by the computation planning agent must be completed prior to the start of computations. Obviously, for the problems of this kind the advantages of VAC over the local CC resource managers are insignificant. In the second example, the application service is intended to realize the multivariant calculations in the mode of dynamic selection of resources. In the course of problem solution the executed jobs may be suspended and then removed and the jobs queued to a node of the Grid-system may be moved to other nodes of the same environment and also new jobs may be generated. To solve the subproblems, used are both the serial and parallel solvers. In this case, application of VAC enables one to reduce essentially the time of problem solution as compared with the local CC resource managers owing to selection of the optimal solvers and possibility jobs migration between the Grid-system nodes.

In both examples, the services representing VAC are realized with the use of the framework considered in Section 3. In the first example, the specificity of the problem at hand requires that the functions of graphic visualization of the result be included in the service. In the second example, consideration was given to a special case of solving systems of Boolean equations, the SAT problem for which the previously constructed Boolean model was used. In the general case, however, the specificity of the knowledge domain described in detail in [27] was taken into consideration, first, by complementing the VAC functions listed in Section 3 by the functions for construction, analysis, and

rearrangement in various formats of the Boolean model, second, by indicating the mode of carrying out the computing experiment (problem solution, testing of the solver or model) depending on which the service requires that the user fills in various sets of parameters, and, third, by formulating a problem (search of the first solution, $k$ solutions or all solutions of the problem) corresponding to the knowledge domain of the problem under consideration. In the two first cases, upon reaching the result the incomplete jobs can be eliminated, which is done in the second example to reduce the time of problem solution and use rationally the computing resources. Additionally, the agents can select SAT solvers depending on the user preferences, analysis of the Boolean function structure, and availability of the computing resources.

The computing experiments were carried out in a heterogeneous cluster Grid-system of IDSTU SO RAN comprising the uniform CC "Blackford" with 20 computing nodes having each two four-core processors Intel Xeon 5345 EM64T ("Clovertown") 2.33 GHz, heterogeneous CC "Academician V.M. Matrosov" having 110 two-core nodes with 16-core processors AMD Opteron 6276 2.3 GHz ("Interlagos") based on the "Bulldozer" x86_64-microarchitecture and a node with graphic processors NVidia C2070 ("Fermi"), heterogeneous CC with GPU NVidia "Tesla" comprising four four-core processors Intel Xeon X5570 ("Nehalem") and 8 GPU NVidia "Tesla" C1060 with the total number of cores equal to 1920.

## 5. CONCLUSIONS

The present paper considered the service-oriented methods and tools to control the problem-oriented distributed computations in the cluster Grid-system integrated with the traditional meta-planners and local managers of the resources of the Grid-system nodes, including the original methods for conversion into the computing jobs of the user requests to the service-oriented environment, classification of jobs and decomposition of the environment resources according to the jobs classes, diverse multiagent tools for control of computations, new high-level framework for generation of the specifications and construction of the interfaces of the service-oriented applications.

The listed above methods and tools have some distinctions. First, they enable development and execution of the application services in different modes: control of computations both at the level of individual applications and at the level of job flows, allocation of resources required for execution of the service operations by special agents or traditional local resource managers, and use of static or dynamic planning of computations. Second, generation of the agents' VAC operating at the level of applications enables one to represent as the Grid-service some system functions of the applications, in particular, planning of computations, resource allocation and monitoring, determination of the job state, allowance to the specificity of the knowledge domain of problem solution in the course computation control.

## ACKNOWLEDGMENTS

This work was supported by the Russian Foundation for Basic Research, project no. 15-29-07955-ofi_m.

## REFERENCES

1. Rajkumar Buyya, R., Vecchiola, C., and Selvi, S.T., *Mastering Cloud Computing*, Burlington: Morgan Kaufmann, 2013.

2. Foster, I., Globus Toolkit Version 4: Software for Service-Oriented Systems, in *IFIP Int. Conf. Network and Parallel Computing*, Springer, 2006, pp. 2–13.

3. Streit, A., Bala, P., Beck-Ratzka, A., et al., UNICORE 6—Recent and Future Advancements, in *Forschungszentrum Jülich Zentralbibliothek*, 2010, http://hdl.handle.net/2128/3695.

4. Binsztok, H., Koprowski, A., and Swarczewskaja, I., *Opa: Up and Running*, Sebastopol: O'Reilly, 2013.

5. Astaf'ev, A.S., Afanas'ev, A.P., Lazarev, I.V., et al., Scientific Service-oriented Environment Based on the Web Technologies and Distributed Computation. Scientific Service in Internet: Scaling, Parallellelism, Efficiency, in *Proc. All-Russian Superkomp. Conf.*, Moscow: Mosk. Gos. Univ., 2009, pp. 463–467.

6. Bukhanovskii, A.V., Koval'chuk, S.V., and Mar'in, S.V., Intelligent High-performance Program Complexes for Modeling of Complicated Systems: Concept, Architechture, and Examples of Realizations, *Izv. Vyssh. Uchebn. Zaved., Priborostr.*, 2009, vol. 52, no. 10, pp. 5–24.

7. Herrera, J., Huedo, E., Montero, R., et al., Porting of Scientific Applications to Grid Computing on GridWay, *Sci. Program.*, 2005, vol. 13, no. 4, pp. 317–331.

8. Kovalenko, V.N., Complex Software of the Computing-type Grid, *Preprint of Keldysh IPM RAN*, Moscow: 2007.

9. Durfee, E.H., Distributed Problem Solving and Planning, in *Multiagent Systems: A Modern Approach to Distributed Artificial Intelligence*, Weiss, G., Ed., Massachusetts: MIT Press, 1999, pp. 121–164.

10. *Market-Oriented Grid and Utility Computing*, Buyya, R. and Bubendorfer, K., Eds., New Jersey: Wiley, 2010.

11. Toporkov, V.V., *Modeli raspredelennykh vychislenii* (Models of Distributed Computations), Moscow: Fizmatlit, 2004.

12. Bychkov, I.V., Oparin, G.A., Feoktistov, A.G., et al., The Distribution of Jobs in the Integrated Cluster System on the Basis of Their Classification, *Vychisl. Tekhnol.*, 2013, vol. 18, no. 2, pp. 25–32.

13. Bogdanova, V.G., Bychkov, I.V., Korsukov, A.S., Oparin, G.A., and Feoktistov, A.G., Multiagent Approach to Controlling Distributed Computing in a Cluster Grid System, *J. Comput. Syst. Sci. Int.*, 2014, vol. 53, no. 5, pp. 713–722, DOI: 10.1134/S1064230714040030.

14. Oparin, G.A., Novopashin, A.P., Sidorov, I.A., et al., Metamonitoring System for High-Performance Computing Environment, *Program. Prod. Sist.*, 2014, no. 2, pp. 45–48.

15. Boev, V.D., *Modelirovanie sistem. Instrumental'nye sredstva GPSS World* (System Modeling. Tools of GPSS World), St. Petersburg: BKHV-Peterburg, 2004.

16. Sholomov, L.A., *Logicheskie metody issledovaniya diskretnykh modelei vybora* (Logic Methods to Study Discrete Choice Models), Moscow: Nauka, 1989.

17. Czajkowski, K., Ferguson, D.F., Foster, I., et al., *The WS-Resource Framework. Version 1.0*, http://www.globus.org/wsrf/specs/ws-wsrf.pdf.

18. Bychkov, I.V., Oparin, G.A., Feoktistov, A.G., et al., The Service-Oriented Approach to Distributed Computing on the Basis of the Toolkit DISCENT, *Inform. Tekhnol. Komput. Sist.*, 2014, no. 2, pp. 7–15.

19. Kundu, P., Das, D., and Ratha, B., WSDL Specification of Services for Service Oriented Architecture (SOA) Based Implementation of a CRM Process, *Int. J. Sci. Engin. Res.*, 2012, vol. 3, no. 10, pp. 1–24.

20. Fielding, R.T., Architectural Styles and the Design of Network-based Software Architectures, *Dissertation*, Irvine: Univ. of California, 2000.

21. Walsh, A., *UDDI, SOAP and WSDL: The Web Services Specification Reference Book*, Upper Saddle River: Prentice Hall, 2002.

22. Henderson, R., Job Scheduling under the Portable Batch System, in *Job Scheduling Strategies for Parallel Processing*, New York: Springer, 1995, pp, 279–294.

23. Litzkow, M., Livny, M., and Mutka, M., Condor—A Hunter of Idle Workstations, in *8th Int. Conf. Distributed Computing Systems (ICDCS)*, Los Alamitos: IEEE CS Press, 1988, pp. 104–111.

24. Wilkinson, J.X. and Reinsch, C., *Handbook for Automatic Computation*, vol. II: *Linear Algebra*, New York: Shpringer, 1971.

25. Bychkov, I.V., Oparin, G.A., Feoktistov, A.G., et al., Service-oriented Approach to Multiagent Control of Distributed Computing, in *Proc. XII All-Russian Conf. Control Probl.*, Moscow: IPU RAN, 2014, pp. 8942–8953.

26. Oparin, G.A., Feoktistov, A.G., Bogdanova, V.G., et al., The Solution of Boolean Equations of High Dimensionality in the Distributed Computing Environment, in *Distributed Computing and Grid-Technologies in Science and Education. Abstr. Int. Conf. Dubna, Russia, June 29–July 2, 2004*, JINR, 2004, D11-2004-82, p. 65.

27. Oparin, G.A. and Bogdanova, V.G., REBUS—Intellectual Solver for Combinatorial Problems in Boolean Constraints, *Vestn. NGU, Inform. Tekhnol.*, 2008, vol. 6, no. 1, pp. 60–68.

28. Oparin, G.A. and Bogdanova, V.G., Parallel Computing Toolkit for Solving Boolean Equations on Multicore Processors, *Program. Prod. Sist.*, 2012, no. 1, pp. 10–14.

*This paper was recommended for publication by S.N. Vasil'ev, a member of the Editorial Board*