Advances in Engineering Research, volume 181
International Conference on Electronical, Mechanical and Materials Engineering (ICE2ME 2019)

Multiagent Resource Dispatching in a Heterogeneous Cloud Environment

I.A. Kalyaev[1], A.I. Kalyaev[2] and I.S. Korovin[2,*]
[1]Southern Scientific Center of the Russian Academy of Sciences, Taganrog, Russia
[2]Southern Federal University, Russia
[*]Corresponding author

***Abstract*—The cloud computing paradigm has appeared: large-scale distributed calculations based on a pool of abstract, virtualized, dynamically redistributable computational resources, granted to external users via Internet on their demands. A cloud computing environments (CCE) get wider application for solution of large-scale scientific and technical problems from various subject domains, such as high-energy physics, sciences about the Earth; chemistry and biology; cosmology and astrophysics; ecological and man-made safety; industry, pharmacology and pharmacy; material science; oil and gas production; medicine, etc.. The specific feature of such large-scale scientific problems is their complex inner structure, which, as a rule, consists of several data coupled subtasks. Besides, it happens rather often that efficiency of such subtasks considerably depends on the type of computational resource, which is used for their implementation. In our work "A modified method of multi-agent resource dispatching in a heterogeneous cloud environment" we present the formal problem statement of multi-agent resource dispatching in a heterogeneous cloud environment, and also we proposed the multi-agent CCE dispatcher functioning principles. This article presents the algorithms of the task agent and the recourse agent of multi-agent resource dispatching in a heterogeneous cloud environment.**

Keywords—component; formatting; style; styling; insert

I. The Algorithm of the Task Agent

When Customer has placed the task graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$ of the task Z_l on the BB, the graph gets the agent AZ_l. The main function of the agent AZ_l is to minimize the solution time of the task Z_l. Since the solution time of the task Z_l, first of all, depends on the execution time of the most complex thread (the critical path) of the graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$, then, in the first place, the agent is to distribute its subtasks to the CCE resources. For this, the agent selects the most complex thread in the task graph and specifies the required (possible) execution start time $t_s^1 = t_{curr}$. After that, it places the thread $\mathbf{H}_1$ on the BB for execution.

When the agents of all resources R_j $(i = 1,2,...,N)$ have analysed their execution capabilities of the thread $\mathbf{H}_1$, they represent their "propositions" to the agent AZ_l of the task Z_l. The agent AZ_l chooses the "proposition" from the agent AR_p, which is capable to provide execution of such subthread, for $\mathbf{H}_{1p}^1 =< q_1^1, q_2^1, ..., q_b^1 > \subseteq \mathbf{H}_1$ which the value

$$E_p = \frac{\sum_{i=1}^{b} v_i^1}{t_{fp}^1 - t_s^1} \quad (1)$$

is the highest. Here v_i^1 $(i = 1,2,...,b)$ is the computational complexity of the subtask O_i, assigned to the vertices $q_i^1 \in \mathbf{H}_{1p}^1$; t_{fp}^1 is the execution completion time of the subthread $\mathbf{H}_{1p}^1$, calculated by formula (2).

The agent AR_p joins the society $\mathbf{R}_l$, which solves the task Z_l. The number of the agent AR_p, responsible for execution of the subthread vertices, and their execution time, calculated by formula (3), are assigned to all vertices of the subthread $\mathbf{H}_{1p}^1$ of the graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$.

Then the agent AZ_l removes the subthread $\mathbf{H}_1^1$ from the thread $\mathbf{H}_1$. As a result, a new thread $\mathbf{H}_1^2 \subseteq \mathbf{H}_1$ is formed. The agent AZ_l assigns to this new thread the possible execution start time t_s^2, specified by the execution time of the subthread $\mathbf{H}_{1p}^1$, i.e. $t_s^2 = t_{fp}^1$. After that, the agent AZ_l places the thread $\mathbf{H}_1^2$ on the BB for execution.

Further on, the agent AZ_l once again collects the propositions, concerning participation in execution of the thread $\mathbf{H}_1^2$, from the agents AR_j $(j = 1,2,...,N)$. It chooses the "proposition" from the agent AR_c, which is capable to provide execution of such subthread $\mathbf{H}_{1c}^2 =< q_{b+1}^1, q_{b+2}^1, ..., q_m^1 >$ $(m \le k)$, for which the value E_c (see formula (5)) is the highest. As a result, the agent AR_c joins the society $\mathbf{R}_l$. The number of the agent AR_c, responsible for execution of the subthread $\mathbf{H}_{1c}^2$, and the execution time t_1^j calculated by formula (6), are assigned to all vertices of the subthread $\mathbf{H}_{1c}^2$ of the graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$.

The same process continues until all vertices (subtasks) of the thread $\mathbf{H}_1$ are distributed, i.e. a new subthread $\mathbf{H}_1^m = \varnothing$.

The task agent AZ_l removes the thread $\mathbf{H}_1$ from the task graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$, and, as a result, a new graph $\mathbf{G}_l^1(\mathbf{Q}_l^1, \mathbf{X}_l^1) = \mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)/\mathbf{H}_1$ is formed. In this graph the task agent AZ_l selects the most complex thread $\mathbf{H}_2$, which is placed on the BB for execution. When all vertices (subtasks) of the thread $\mathbf{H}_2 =< q_1^2, q_2^2, ..., q_r^2 >$ are taken by the agents AR_j $(j = 1,2,...,N)$, in the graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$ each vertex of the thread $\mathbf{H}_2$ gets the number of the corresponding agent AR_p assigned to this vertex, and the execution time t_i^2 $(i = 1,2,...,r)$. Then the task agent AZ_l is to check initial data consistency for the subtask O_b, assigned to the vertex q_b^1 of the thread $\mathbf{H}_1$, incident with the target vertex of the thread $\mathbf{H}_2$. For this, the task agent AZ_l compares the execution completion time t_f^2 of the thread $\mathbf{H}_2$, calculated by formula (7), with the execution time t_{b-1}^1 of the previous vertex q_{b-1}^1 of the thread $\mathbf{H}_1$. If $t_r^2 > t_{b-1}^1$, then it means that the results of execution of the thread $\mathbf{H}_2$ and required for execution of the subtask O_b will be obtained later than the results of execution of the operation O_{b-1} assigned to the vertex q_{b-1} of the thread $\mathbf{H}_1$. In this case the task agent AZ_l is to correct the execution time for all next vertices $q_b^1, q_{b+1}^1, ..., q_k^1$ of the thread $\mathbf{H}_1$ increasing them by the value $\Delta t = t_f^2 - t_{b-1}^1$.

Then the thread $\mathbf{H}_2$ is removed from the task graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$, and a new graph $\mathbf{G}_l^2(\mathbf{Q}_l^2, \mathbf{X}_l^2) = \mathbf{G}_l^1(\mathbf{Q}_l^1, \mathbf{X}_l^1)/\mathbf{H}_2$ is formed. In this graph the most complex thread $\mathbf{H}_3$ is selected, and the agent AZ_l places it on the BB for execution (see Figure I). The process continues until all vertices of the task graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$ are assigned to the agents AR_j $(j = 1,2,...,N)$ and their scheduled execution time is specified. After that the agent AZ_l of the task Z_l informs Customer about the scheduled solution time t_l^k of its task Z_l, calculated by formula which the value

$$W_{3,i_3,k_2+1} = \sum_{j=1}^{k_2} W_{3,i_3,j} - 0.5, \quad \mathrm{i}_3 = 1 \ldots \mathrm{k}_3 . \tag{2}$$

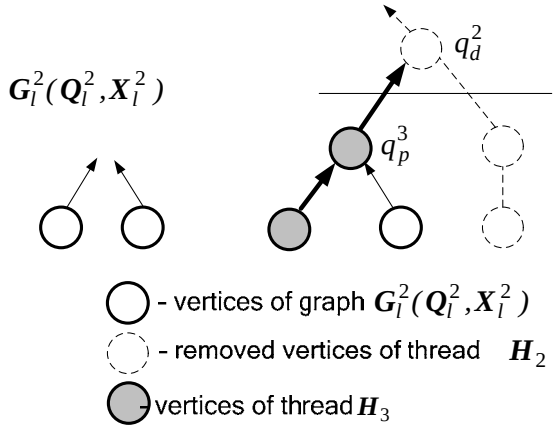

FIGURE I. THE GRAPH $\mathbf{G}_l^2(\mathbf{Q}_l^2, \mathbf{X}_l^2)$ MODIFIED BY THE AGENT AZ_l

If Customer agrees with the suggested solution time of its task Z_l, then the agent AZ_l informs all agents of the society $\mathbf{R}_l$ that it is necessary to execute their operations.

A formal algorithm of the agent $i < b$ of the task Z_l, developed according to the given description, is presented below.

II. THE ALGORITHM 1

1) $j = 1$; $\mathbf{G}_l^j(\mathbf{Q}_l^j, \mathbf{X}_l^j) = \mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$; $\mathbf{R}_l = \varnothing$.

2) The most complex thread $\mathbf{H}_j =< q_1^j, q_2^j, ..., q_k^j >$ *is selected in the graph* $\mathbf{G}_l^j(\mathbf{Q}_l^j, \mathbf{X}_l^j)$*:* For this thread the value $V_j = \sum_{i=1}^{k} v_i^j$ is the highest. Here v_i^j is the computational complexity of the operation O_i^j assigned to the vertex q_i^j $(i = 1,2,...,k)$.

3) $m = 1$; $\mathbf{H}_j^m = \mathbf{H}_j$; $t_s^m = t_{\text{curr}}$.

4) The agent AZ_l *places the thread on the BB for execution.*

5) $r = 1$; $p = 0$; $E_p = \infty$.

6) The agent receives a "proposition" to execute the subthread $\mathbf{H}_{jr}^m =< q_1^j, q_2^j, ..., q_b^j > \subseteq \mathbf{H}_j^m$ $(b \le k)$, *and the execution start time* t_{sr}^m *and execution completion time* t_{fr}^m *from the agent* AR_r.

7) If $E_r = \sum_{i=1}^{b} v_i^j / t_{fr}^m - t_s^m \ge E_p$, *where* v_i^j *is the computational complexity of the vertex* q_i^j $(i = 1,2,...,b)$, *then go to 9, else*

8) $E_p = E_r$; $p = r$

9) $r = r + 1$; *if* $r \le N$, *then go to 6, else.*

10) The agent AR_p *joins the society* $\mathbf{R}_l$ *which executes the task* Z_l, *and is responsible for execution of the thread* $\mathbf{H}_{jp}^m$*:* In the graph $\mathbf{G}_l^j(\mathbf{Q}_l^j, \mathbf{X}_l^j)$ the number of the agent AR_p and the execution time are assigned to the vertices of the thread $\mathbf{H}_{jp}^m$. The execution time is calculated as follows

$$t_f^j = t_{sp}^m + \sum_{i=1}^{f} \frac{v_i}{S_p(O_i)} \ (f = 1,2,...,b) \tag{3}$$

11) $\mathbf{H}_j^{m+1} = \mathbf{H}_j^m / \mathbf{H}_{jp}^m$; *if* $\mathbf{H}_j^{m+1} = \varnothing$, *then go to 13, else*

12) $m = m+1$; $t_s^m = t_f^b + (d_{b,b+1} / Y_p)$ *; go to 4.*

13) If $j = 1$ *or* $t_k^j \le t_{f-1}^{j-1}$, *where* t_k^j *is the completion time of the thread* $\mathbf{H}_j$; t_{b-1}^{j-1} *is the execution time of the vertex* q_{b-1}^{j-1} *of the thread* $\mathbf{H}_{j-1} =< q_1^{j-1}, q_2^{j-1}, ..., q_r^{j-1} >$, *the vertex* q_b^{j-1} *which is incident with the target vertex* q_k^j *of the thread* $\mathbf{H}_j$, *go to 15, else*

14) In the graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$ *a new prearranged execution time* $t_i^{j-1} = t_i^{j-1} + \Delta T \quad (i = b, b+1, ..., r)$ *is assigned to the vertices* $q_b^{j-1}, q_{b+1}^{j-1}, ..., q_r^{j-1}$, *of the thread* $q\mathbf{H}_{j-1}$ *:* Here $\Delta T = t_k^j - t_{b-1}^{j-1}$. The agent AR_c, which is responsible for execution of the vertices $q_b^{j-1}, ..., q_r^{j-1}$ of the thread $\mathbf{H}_{j-1}$, is usually informed about correction of the execution time.

15) $\mathbf{G}_l^{j+1}(\mathbf{Q}_l^{j+1}, \mathbf{X}_l^{j+1}) = \mathbf{G}_l^j(\mathbf{Q}_l^j, \mathbf{X}_l^j) / \mathbf{H}_j$, *if* $\mathbf{G}_l^{j+1}(\mathbf{Q}_l^{j+1}, \mathbf{X}_l^{j+1}) = \varnothing$, *then go to 17, else*

16) $j = j+1$, *go to 2.*

17) Customer is informed about the scheduled solution time $t_l^k = t_k + (d_{k,k+1} / Y_p)$ *of its task* Z_l, *where* t_k *is the execution time of the target vertex* q_k *of the graph* $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$, $d_{k,k+1}$ *is the size of the result data assigned to the edge outgoing from the target vertex* q_k *;* Y_p *is the bandwidth capacity of the channel of the resource* R_p, *which is responsible for execution of the vertex* q_k.

18) If Customer agrees with the scheduled solution time of its task Z_l, *then the agent* AZ_l *informs all agents of the society* $\mathbf{R}_l$ *that it is necessary to execute their operations.*

19) If the agent $AR_p \subseteq \mathbf{R}_l$ *reports that execution of its thread* $\mathbf{H}_{jp}^m =< q_1^j, q_2^j, ..., q_b^j >$ *is completed, then the agent* AZ_l *compares the scheduled completion time* t_f^j *of the thread with the current time* t_{curr} *:* Here the scheduled completion time is calculated as $t_f^j = t_b^j + (d_{b,b+1}^j / Y_p)$, where t_b^j is the execution time, assigned to the target vertex $q_b^1 \in \mathbf{H}_{ip}^m$; $d_{b,b+1}^j$ is the size of data, assigned to the outgoing edge of the thread $\mathbf{H}_{ip}^m$. If $t_{curr} > t_f^j$, then Customer is informed that solution of its task Z_l is delayed for $\Delta t = t_{curr} - t_f^j$, and the scheduled execution time of all next vertices of the graph $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$ is increased by Δt.

20) If the target vertex q_b^j *of the thread* $\mathbf{H}_{jp}^m$ *is the target vertex* q_k *of the graph* $\mathbf{G}_l(\mathbf{Q}_l, \mathbf{X}_l)$, *then Customer is informed that its task* Z_l *is executed. The task* Z_l *is removed from the BB.*

III. Conclusions The Resource Agent Algorithm

The agent AR_p is to search job for its resource R_p. For this the agent AR_p asks regularly the BB, and if it finds a certain thread $\mathbf{H}_j^m$, placed for execution by the agent AZ_l of the task Z_l, it tries to join the society $\mathbf{R}_l$ which is responsible for its execution. So the agent AR_p selects in the thread $\mathbf{H}_j^m =< q_1^j, q_2^j, ..., q_k^j >$ a subthread $\mathbf{H}_{jp}^m =< q_1^j, q_2^j, ..., q_b^j >$ $(b \le k)$. The subtasks, assigned to the vertices of the subthread $\mathbf{H}_{jp}^m$, belong to a set $\mathbf{O}_p$, i.e. a set of subtasks, executed by the resource R_p.

Besides, the agent AR_p specifies the time t_{sp}^m, when it is able to start execution of the subthread $\mathbf{H}_{jp}^m$ (i.e. when it has executed all operations, assigned to it earlier), and also the execution completion time t_{fp}^m of the subthread $\mathbf{H}_{jp}^m$, calculated by formula (2).

The selected subthread $\mathbf{H}_{jp}^m$ is sent to the agent AZ_l of the task Z_l as a "proposition" from the agent AR_p for joining the society $\mathbf{R}_l$, responsible for execution of the task Z_l. If the "proposition" from the agent AR_p is the best among all others, i.e. the value E_p, calculated by formula (5), is the highest subthread $\mathbf{H}_{jp}^m$, the agent AR_p gets confirmation, that the subthread $\mathbf{H}_{jp}^m$ is assigned to it.

The agent AR_p starts execution of the thread $\mathbf{H}_{jp}^m$ precisely according to the scheduled execution start time t_{sp}^m (see Figure II). Before execution of any subtask O_i^j, assigned to the vertex q_i^j $(i = 1, 2, ..., b)$ of the thread $\mathbf{H}_{jp}^m$, the agent AR_p checks availability of all initial data required for execution. If some initial data is not available, then the agent AR_p goes to a standby mode. When all initial data, required for execution of the subtask O_i^j, is available the agent AR_p starts execution of the subtask O_i^j with the help of its own resource R_p. When all subtasks assigned to the vertices of the subthread $\mathbf{H}_{jp}^m$ are executed, the agent AR_p informs the agent of the task Z_l about execution of the thread $\mathbf{H}_{jp}^m$.

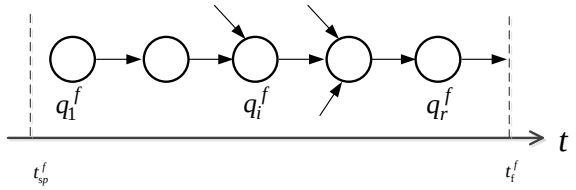

FIGURE II. THE EXECUTION OF THE SUBTHREAD $\mathbf{H}_{jp}^m$ BY THE AGENT AR_p

The following formal algorithm corresponds to the process, described above.

IV. The Algorithm 2

1) The agent AR_p *asks the BB and tries to find job for its resource* R_p.

2) If the agent AR_p *has found a thread* $\mathbf{H}_j^m =< q_1^j, q_2^j, ..., q_k^j >$ *of the task* Z_l, *placed on the BB by the agent* $\mathbf{H}_j^m$, *then it selects a subthread* $\mathbf{H}_{jp}^m =< q_1^j, q_2^j, ..., q_b^j >$ $(b \le k)$ *in the thread* $\mathbf{H}_j^m$, *whose vertices fulfill the condition* $O_i^j \subseteq \mathbf{O}_p$ $(j = 1, 2, ..., b)$, *where* O_i^j *are subtasks assigned to the vertex* q_i^j *of the thread* $\mathbf{H}_{jp}^m$.

3) The agent AR_p specifies the execution start time t_{sp}^m and the execution completion time t_{fp}^m for the thread $\mathbf{H}_{jp}^m$ as follows

$$t_{fp}^m = t_{sp}^m + \sum_{i=1}^{b} \frac{v_i^j}{S_p(O_i)} + \frac{d_{b,b+1}^j}{Y_p} \quad (4)$$

4) The "proposition" from the agent AR_p to participate in execution of the thread $\mathbf{H}_j^m$ of the task Z_l is sent to the agent AZ_l.

5) If the agent AZ_l confirms, that the agent AR_p has joined the society $\mathbf{R}_l$, then the agent AR_p starts execution of the thread $\mathbf{H}_{jp}^m = < q_1^j, q_2^j, ..., q_b^j >$ in the time t_{sp}^m.

6) $i = 1$

7) The agent AR_p checks availability of the initial data required for execution of the subtask O_i^j assigned to the vertex $q_i^j \in \mathbf{H}_{jp}^m$: If the initial data is not available, the agent AR_p goes to a standby mode.

8) When the required initial data is available, the agent AR_p executes the subtask O_i^j with the help of its resource R_p.

9) $i = i + 1$, if $i < b$, then go to 7, else

10) The agent AR_p informs the agent of the task Z_l that the subtasks of the thread $\mathbf{H}_{jp}^m$ are executed

11) Go to 1.

To estimate the efficiency of the suggested methods and algorithms of CCE multi-agent dispatcher during solution of a flow of large-scale scientific problems, it is necessary to analyze functioning of the CCE in various modes. Due to the fact, that the simulated computational environment is a complex system, it is necessary to organize a targeted experiment for its complete research.

Since the CCE consists of a great number of nodes, we suggest to use the average value of the coefficient of system nodes relative load as a criterion, which describes efficiency of the developed methods and algorithms in relation to loading of computational resources. The coefficient of system nodes relative load is the ratio of the processor time, spent on solution of useful tasks by all nodes, to the total processor time spent by all nodes during operating of the system. From now forward let us use identify this parameter as Y1.

For research we have chosen several parameters of the system to estimate their influence on Y:

- the number of nodes;
- the rate of new tasks;
- the performance of nodes;
- the bandwidth capacity of the data channel among the nodes;
- the computational complexity of tasks;
- sizes of data transferred among subtasks;
- the number of subtasks in the task.

Due to the extremely large number of experiments, required for analysis of influence of various values and parameter combinations on CCE efficiency, we have decided to divide all variable parameters of the model into three groups according to their degree of impact on the parameter Y1:

- primary parameters which specify the application area of the system (the number of nodes and rate of new tasks).
- secondary parameters which characterize separate nodes and tasks existing in the system (the performance of the nodes, the bandwidth capacity of data channels in the node, the computational complexity of tasks and size of data transferred among subtasks).
- tertiary (subjective) parameters which depend only on the user, and therefore hardly predictable (the number of subtasks in some task).

To make experiments for various sets of parameters, mentioned above, in close-to-real conditions we have implemented a distributed program model of a CCE with a multi-agent dispatcher, which contained the following components:

- a program agent and a program agent interface;
- a bulletin board;
- a visual user interface;
- a visual administrator interface.

The block diagram of the architecture of the complex is given in Figure 3.

The interface of the program agent is given in Figure 4.

The visual user interface is intended to:

- describe user tasks;
- place input data on the bulletin board;
- receive task execution results from the bulletin board.

The visual user interface is given in Figures 5 and 6.

The visual administrator interface is intended to:

- input and edit configuration data of hardware and software tools and save them in configuration files.
- stop, restart of certain user tasks, placed on the selected bulletin board.
- port user applications among computational nodes;
- demonstrate current placing of user applications in the network;
- demonstrate current status of computational resources and loading of network channels.

The visual administrator interface is given in Figure III-VI.

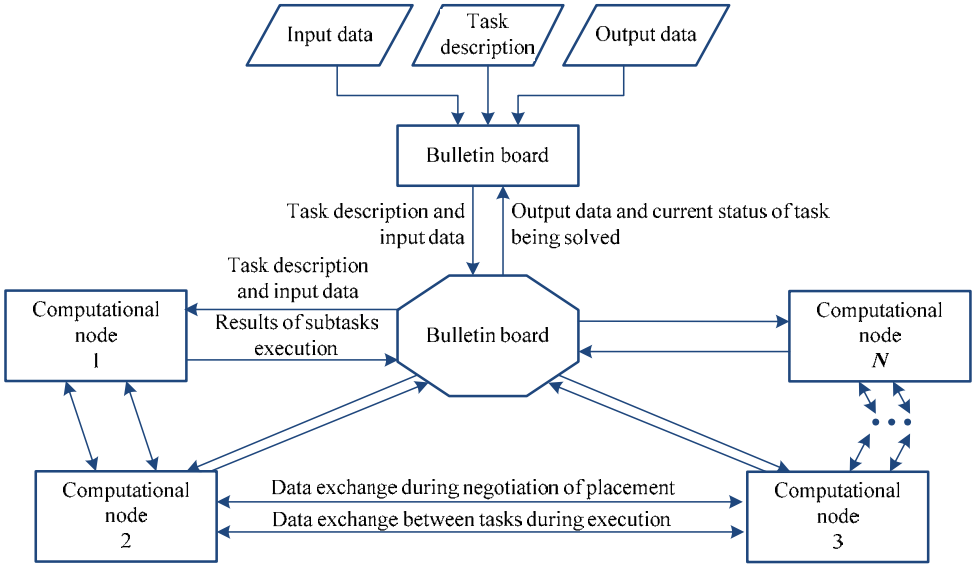

FIGURE III. THE ARCHITECTURE OF THE SOFTWARE

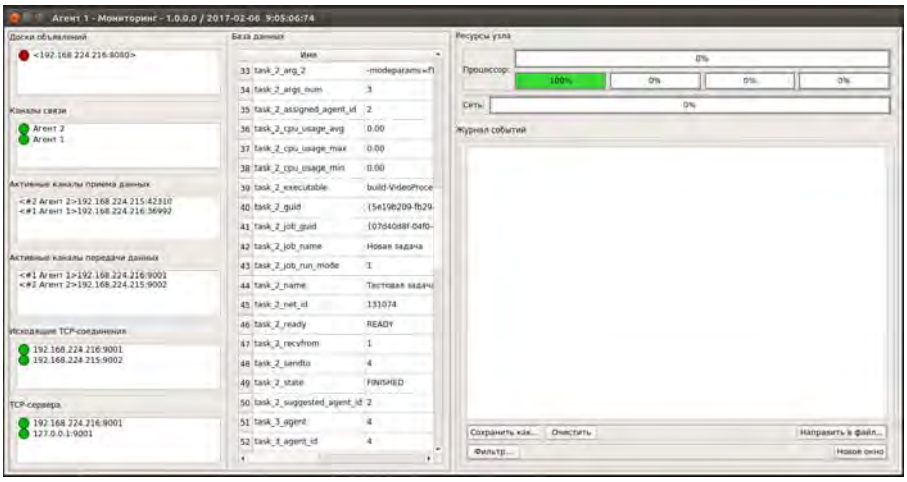

FIGURE IV. THE PROGRAM AGENT INTERFACE

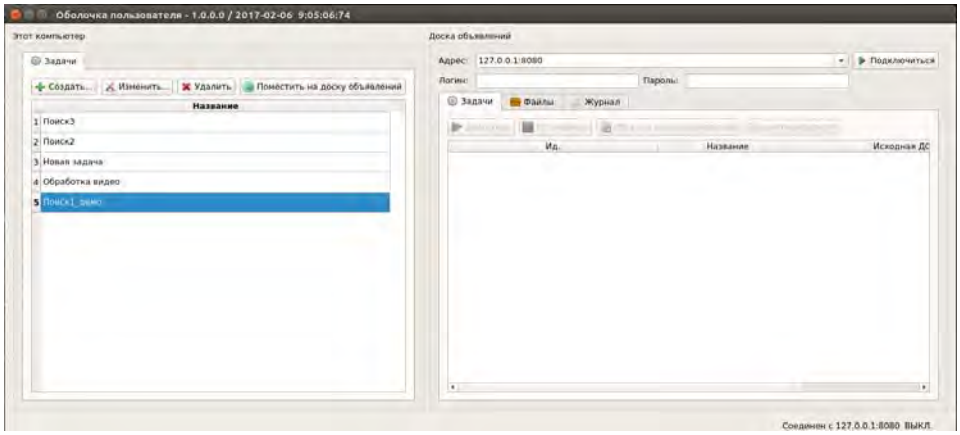

FIGURE V. VISUAL USER INTERFACE

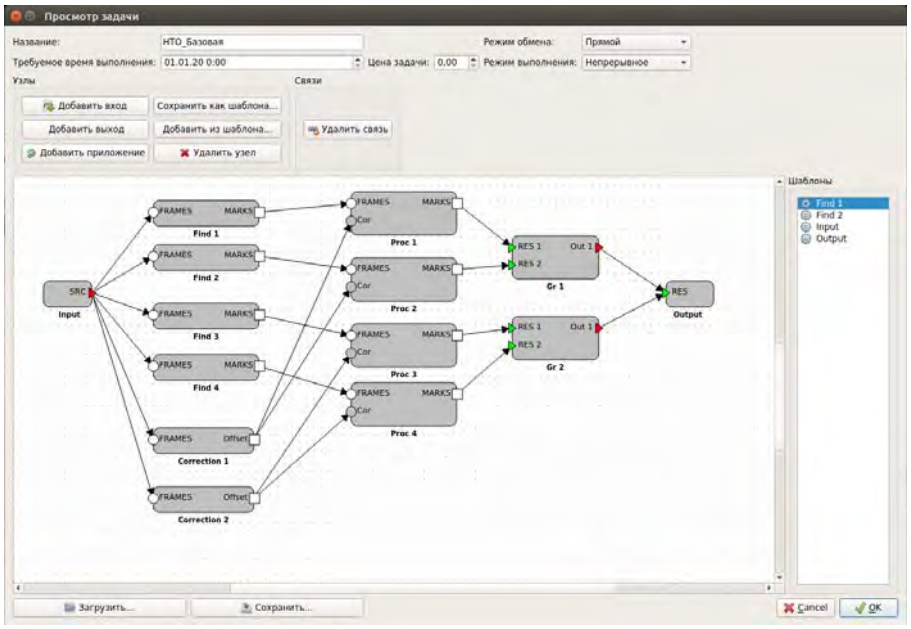

FIGURE VI. VISUAL TASK EDITOR INTERFACE

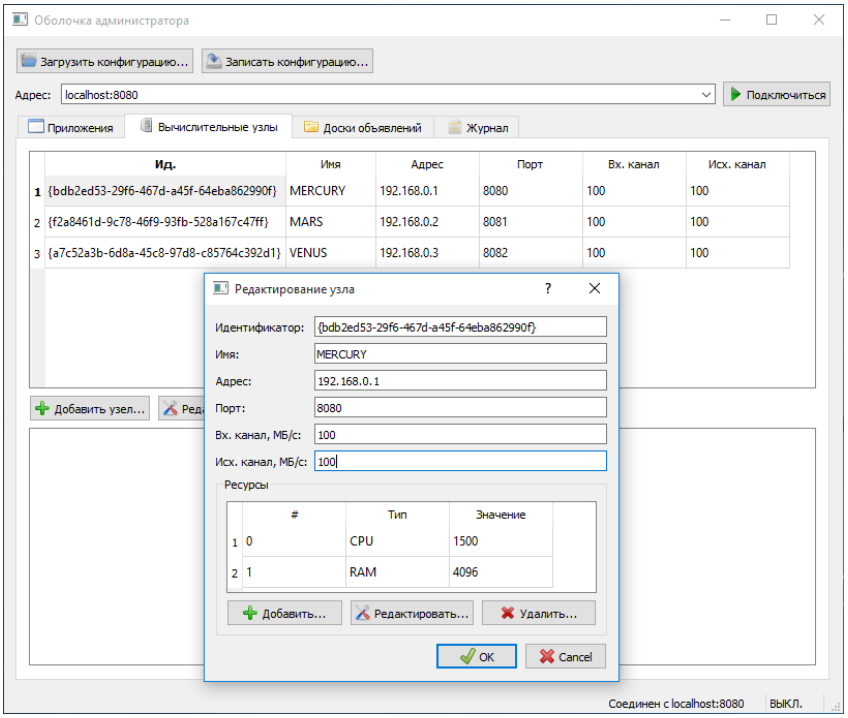

FIGURE VII. VISUAL ADMINISTRATOR INTERFACE

For the developed distributed CCE program model with multi-agent dispatcher we have designed a test stand, which contained 10 computational modules. Each module consisted of 1 to 16 processor cores, connected by a shared network with the bandwidth capacity up to 1000 Mbit per second. Taking into account capabilities of virtualization, the test stand has provided simulation of a CCE operating with various values of the initial parameters such as:

- the size of resources of the CCE;
- the performance of resources during solution of subtasks of various types;
- the bandwidth of the channels, which connect resources with cloud infrastructure;

On the base of parameter combinations, formed earlier, we have performed sets of experiments for various groups of system parameters.

The research results have proved that the suggested method is efficient, the values of the parameter Y1 in various sets of experiments (even for large number of system nodes) remain high enough. The final average value of efficiency of executors is 71.53 for all sets of experiments.

So, in spite of the fact, that CCE dispatching during task solution is realized at the level of executors, loading of computational resources remains high enough. That is why it is possible to conclude that the developed methods and algorithms are effective from the system executors' point of view [10-12].

Research results has proved that the main advantage of the suggested multi-agent approach to resource dispatching in CCE is adaptation of computational process to computational capabilities of CCE resources. Besides, in comparison to traditional, centralized organization of the dispatcher of the cloud environment, in this case requirements to servers (bulletin boards) become more simple. Owing to this, penalty of cloud environment organization can be considerably reduced. As a result, cost of cloud computations can also be reduced, and process of cloud environment scaling can be simplified.

So, it is possible to conclude, that the suggested approach is usable for design of enterprise CCEs for solution of a flow of large-scale tasks. Owing to such CCEs it will be possible to increase considerably efficiency of computational equipment usage, and to increase flexibility and fault tolerance of computational process.

ACKNOWLEDGMENT

Research is being conducted due to financial support of Russian Ministry of Science and Higher Education, contract № 14.575.21.0152.

REFERENCES

[1] B. Kepes, "Understanding the cloud computing stack: Saas, paas, iaas," Divers. Ltd., pp. 1–17, 2011.

[2] Y. Jadeja and K. Modi, "Cloud computing-concepts, architecture and challenges," in Computing, Electronics and Electrical Technologies (ICCEET), 2012 International Conference on, 2012, pp. 877–880.

[3] T. Dillon, C. Wu, and E. Chang, "Cloud computing: issues and challenges," in Advanced Information Networking and Applications (AINA), 2010 24th IEEE International Conference on, 2010, pp. 27–33.

[4] S. Bhardwaj, L. Jain, and S. Jain, "Cloud computing: A study of infrastructure as a service (IAAS)," Int. J. Eng. Inf. Technol., vol. 2, no. 1, pp. 60–63, 2010.

[5] F. Tao, L. Zhang, V. C. Venkatesh, Y. Luo, and Y. Cheng, "Cloud manufacturing: a computing and service-oriented manufacturing model," Proc. Inst. Mech. Eng. Part B J. Eng. Manuf., vol. 225, no. 10, pp. 1969–1976, 2011.

[6] M. Armbrust et al., "A view of cloud computing," Commun. ACM, vol. 53, no. 4, pp. 50–58, 2010.

[7] K. Chen, C. Hu, X. Zhang, K. Zheng, Y. Chen, and others, "Cloud Computing," IEEE Netw., p. 4, 2011.

[8] L. Griebel et al., "A scoping review of cloud computing in healthcare," BMC Medical Informatics and Decision Making. 2015.

[9] D. C. Marinescu, Cloud Computing: Theory and Practice. 2013.

[10] I. S. Korovin, M. V Khisamutdinov, and A. I. Kalyaev, "On DSS Implementation in the Dynamic Model of the Digital Oil field," in IOP Conference Series: Materials Science and Engineering, 2018, vol. 307, no. 1, p. 12052.

[11] I. S. Korovin, A. Kalyaev, M. Khisamutdinov, D. Ivanov, and G. Schaefer, "Application of neural networks for modelling centrifugal pumping units of booster pump stations for a two-phase gas-liquid mixture," in Informatics, Electronics and Vision & 2017 7th International Symposium in Computational Medical and Health Technology (ICIEV-ISCMHT), 2017 6th International Conference on, 2017, pp. 1–6.

[12] I. Korovin, M. Khisamutdinov, A. Kalyaev, D. Ivanov, and G. Schaefer, "Neural network model of pumping units in oil preparation and pumping complex," in Informatics, Electronics and Vision & 2017 7th International Symposium in Computational Medical and Health Technology (ICIEV-ISCMHT), 2017 6th International Conference on, 2017, pp. 1–4.