I.J. Intelligent Systems and Applications, 2017, 7, 41-51
Published Online July 2017 in MECS (http://www.mecs-press.org/)
DOI: 10.5815/ijisa.2017.07.05

Distributed Computer System Resources Control Mechanism Based on Network-Centric Approach

Zhengbing Hu
School of Educational Information Technology, Central China Normal University, No. 152 Louyu Road, 430079, Wuhan, China
E-mail: hzb@mail.ccnu.edu.cn

Vadym Mukhin, Yaroslav Kornaga and Yaroslav Lavrenko
National Technical University of Ukraine "Kiev Polytechnic Institute", Pr. Pobedy, 37, 03056, Kiev, Ukraine
E-mail: {v.mukhin, y.kornaga, y.lavrenko}@kpi.ua

Oksana Herasymenko
Taras Shevchenko National University of Kiev, Volodymyrska Street, 64/13, 01601, Kiev, Ukraine
E-mail: oksgerasymenko@gmail.com

Abstract—In this paper, we present the development of a decentralized mechanism for the resources control in a distributed computer system based on a network-centric approach. Intially, the network-centric approach was proposed for the military purposes, and now its principles are successfully introduced in the other applications of the complex systems control. Due to the features of control systems based on the network-centric approach, namely adding the horizontal links between components of the same level, adding the general knowledge control in the system, etc., there are new properties and characteristics. The concept of implementing of resource control module for a distributed computer system based on a network-centric approach is proposed in this study. We, basing on this concept, realized the resource control module and perform the analysis of its operation parameters in compare with resource control modules implemented on the hierarchical approach and on the decentralized approach with the creation of the communities of the computing resources. The experiments showed the advantages of the proposed mechanism for resources control in compare with the control mechanisms based on the hierarchical and decentralized approaches.

Index Terms—Distributed computer system, resource control system, decentralized resource control, tasks scheduling, network-centric approach.

I. Introduction

The wide use of distributed computer systems (DCS) has become a daily routine. The increase in the number of users of such systems leads to the fact that DCS are constantly increasing in scale. In addition, such systems are designed to serve simultaneously a large number of users, which requires efficient resource control techniques in such systems [1, 2].

As in recent years, there is a tendency to create a large scale DCS, which are intended to serve a big number of users, the resource control in such systems require the development of new methods and tools. For the scalable DCS is important to apply the decentralized approach to resource control [3, 4, 5].

Let DCS is a set of heterogeneous computing nodes (CN) connected by heterogeneous data channels, and these channels have multichannel capability. Channeling can be both physical and topological. Physical multi-channel is meant the realization of frequency or time division of a single physical data channel. Such channel can be fiber-optic lines, the physical nature of which involves the differentiation of modes of frequencies, including duplexing. The topological multi-channel is the implementation of multiple alternative routes between the DCS nodes, which are connected by the topologically redundant communication system. We may combine the physical and topological multi-channels for the high-speed, reliable data transfer. The scheduler of such a distributed system should consider the features of this system in order to increase of efficiency of its functioning.

So, the development of the scheduler for DCS described above should be based on the decentralized approach. Let us describe some details of the decentralized control approach for DCS resources control and of the schedulers that are implement it.

II. Analysis of Recent Researches and Publications

Currently there are the following main approaches in the resource control of distributed computer systems: centralized control of resources, decentralized control and hierarchical one. Each of these approaches has its advantages and disadvantages. More details of the features, advantages and disadvantages of these approaches can be found in [3, 4, 6].

One of the main components of resource control system (RCS) of DCS is a scheduler which coordinates the solution of the tasks that arrive for processing to the distributed system by placing them on available computing resources [7]. It is important to meet the requirements of the user and to avoid the idles for the computational resources [8]. In [7] there are identified the following key characteristics of the DCS scheduler: the run time of the task, DCS performance and the response time of the system.

In recent years, there was devoted much attention to the study of decentralized resource control techniques for the distributed computer systems to improve the performance of the DCS in the context of its scalability and reliability [9]. In [9] are described the comparative analysis of the existing decentralized distributed computer systems and the mechanisms that are used for resource control. This study contains a comparative analysis of mechanisms of resource control of DCS in different contexts.

Additionally, [10, 11, 12] presents more recent developments of DCS resources control systems, that are implemented on a decentralized basis.

The development in [11] uses the exchanges of network messages between DCS compute nodes for the organization of distributed computing. The messages of several types come, each of which provides a specific functions of the algorithm for the resources allocation. This planning framework supports several heuristic algorithms and it also provides the load balancing for the DCS nodes.

In [12] there is considered the DCS, which consists of heterogeneous computing elements (CE), which are interconnected by a computer network. In addition to CE, the system has passive nodes, the so-called "Bulletin boards", which are used to allocate users tasks and to collect some service information. On the each CE runs an agent, which provides their interaction between themselves and with the "Bulletin Board". The interaction algorithm is designed in a such way, that in order to solve the task, received by one of the CE from "Bulletin boards", the computational elements are joined in the community. The community is designed to provide the user requirements on the time allotted for the task solution. The CE performs the monitoring of their performance and assesses the time for which the task is performed. Also, CE may initiate the community reformatting in the case when the available DCS resources can not completed the task on time.

As it stated in [9], the decentralized technologies are the effective and reliable for the control of tasks flow and data storage in a DCS and is a promising field of research, in this way the subject matter is relevant today.

III. Problem Statement

The main goal of this work is to develop a RCS module for DCS with decentralized mechanism for resource control on the basis of a network-centric approach and to analyze its parameters along with comparing to the other implementations of the RCS.

IV. The Basic Idea

The effectiveness of a decentralized distributed system depends on the level of coordination and cooperation between its elements [9, p. 389], therefore, the improving of the coordination mechanism for joint action allows us to improve the parameters of the system. One of the effective approaches to resources control is the network-centric approach, which shows its effectiveness in various areas.

The principles of network-centric control was developed for the defense sector [13], and then started to be applied in other areas of control of complex systems [14, 15]. The basis of network-centric control is the empowerment of the peripheral components of the system in making them the independent decisions. It is important to note that the peripheral components, with the aim of interaction between themselves, form the so-called horizontal links for data exchange or to performs their functions. These links provide the new qualitatively characteristics of the control system. In addition, it should be emphasized another principle of functioning of control system based on a network-centric approach – the principle of self-organization, which means that the system components work as a community to achieve a common goal. The implementation of these principles for the organization of decentralized resources control will allow to develop an effective and flexible mechanism for DCS resource control.

Let us to develop the concept of resource control for the distributed computer system based on a network-centric approach. First, we divide the components by the level, then we determine the functionality of each component and develop the mechanism of their interaction.

Since the resource control mechanism is decentralized, so there is no any central component, which is implemented the overall control of the tasks scheduling in the system. We form the next levels of the DCS components:

- the first level – schedulers;
- the second level – brokers;
- the third level – computing resources.

In literature we can find various goals of the scheduler and broker in the DCs, and sometimes these terms are seen as synonyms. In this paper we consider these components as different tools. In order to achieve the decentralized control in the DCS there is in use the several schedulers, each of which interacts with a set of resource brokers. All schedulers are equivalent among themselves and are linked as "fully connected", the resources brokers also are equally between themselves. Each scheduler interacts with a set of brokers, but not simultaneously. In turn, the DCS resource broker is subordinate only to a single scheduler and has links of the "fully connected" form with all the brokers of the system.

Every broker has in its subordination the several computational resources, and the one resource is subject to a single broker. Brokers can transfer resources among themselves in the case of a request for it.

The functions of the scheduler are:

– get the tasks from the user (more precisely the parameters of the task, the input data remain on the user's computer)
– determine the number of CE needed to perform the tasks and their computational performance in order the task to be completed on time;
– the choice of the broker that will receive the task for execution;
– the load balancing between schedulers and brokers;
– get results from the broker and redirect them to the user.

The functions of broker:

– get the task (parameters of task) from the scheduler and place it in a queue;
– form of a set of resources to perform tasks due to the requirements of the scheduler;
– the location of the task on the DCS nodes;
– control of the task execution.
– control of the run time of tasks and the ensuring of their timely implementation;
– interaction with the other brokers to obtain the necessary resources in case of shortage, or providing its own resources by the request from other brokers;
– send results of task executing to the certain scheduler, which was originally sent of this task by the user.

The functions of computing resource:

– obtain the sub-tasks (or parts of tasks) with the input data for it from the user's computer;
– execute sub-tasks;
– storage of output data of tasks until they will be requested by other computing resources;
– signal to the broker about the sub-task completion.

A structure scheme of a DCS resource control system on the basis of the network-centric approach is shown in Fig.1.

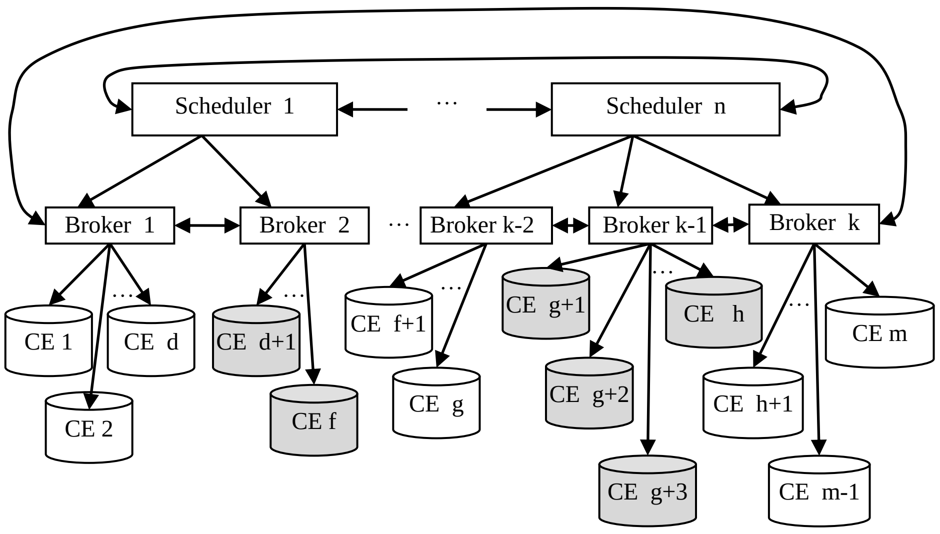

Fig.1. The structure scheme of DCS resource control on the basis of a network-centric approach

We will develop the variant of the implementation of the module for the DCS resource control and compare its parameters with the RCS modules, the implementation of which is based on the other principles.

V. The Development of the RCS Module for DCS

In order to study the characteristics of the different mechanisms of DCS resource control we develop the RCS modules that are implement the following approaches to resource control:

– a decentralized approach based on the communities of computing resources;
– hierarchical approach;
– network-centric approach.

A. Task Structure Description

First, the task is represented in MPL-form (multi-layer parallel form) as a set of subtasks (st), each placed at the certain their level. The input streams for subtasks of the first level are the input data provided by the user prior to the task execution. In turn subtask of the last level has one output stream that is the output, i.e. the result of the task.

The study was conducted taking into account the fact that the task runs asynchronously, i.e. the parallel subtasks can be run at different levels, if they received all the input data.

Let us present the main characteristics of the tasks (Table 1) for further description of the development of RCS modules in more detail.

Table 1. The main parameters of the task

Parameter	Description
levelAmount	Number of task levels
maxLV	The maximum number of subtasks at the level
subTaskAmount	Number of subtasks
MItotal	Computational complexity of the task in MI (Millions Instructions). Defined as the sum of the computational complexity of all subtasks
MIcompl	The total computational complexity of subtasks that are already finished at the time of determining this parameter
MIuncompl	The total computational complexity of subtasks that have not yet been finished at the time of determining this parameter. Calculated by the formula $MIuncompl = MItotal - MIcompl$
MIlastST	The computational complexity of the last subtask MI
Tmax	The maximum time for which the task should be performed
TinWork	Time spent on the task. It is calculated as the difference between the current time and the time when the task was started. The time of the task waiting in the system is not taken into account

The development of scheduler module is made taking into account the MPL-form for the task and, here is assuming that the tasks are executing asynchronously.

B. Development of RCS Module on the Basis of the Formation of Communities of Computing Resources

Based on [12] let us design a decentralized scheduler, which forms the communities to process the task. The main concept is based on the algorithm described in [12], which was partially modified by us with the goal to unify all investigated RCS for the comparison of the obtained results.

This RCS functioning as a set of communities that are created by the compute nodes to execute the task. On the each compute resource is run so called the resource agent, that provides the resource control and performs the interaction with agents of other resources. In addition, in a distributed computer system is the so-called "Bulletin board", which store the tasks, that are submitted by users, and the agents receive the tasks from these "boards".

Before the task will be run, we form the community, which starts from the connection of the at least one agent to the selected task. The task is selected by FIFO principle, the priority is not taken into account. The agent of resource with performance $resPerf$ is connected to the community to complete the task when the condition (1) is satisfied:

$$\left(\begin{array}{l}\dfrac{MIlastST}{resPerf} + k_{corspnd} * \\ * \dfrac{MItotal - MIlastST}{comSumPerf + resPerf * (\max LV - com\,Re\,sAmnt)}\end{array}\right) < T_{max}, \quad (1)$$

where $k_{corspnd}$ is the correspondence coefficient; $comSumPerf$ – the total productivity of all resources of DCS, which agents are already connected to the community; $com\,Re\,sAmnt$ – the number of agents that are connected to the community.

Otherwise, the agent selects the next task. The community is built until the number of agents becomes equal $\max LV$. Then agents start the distribution of subtasks among themselves. It is important to note that the resource does not leave the community in case when there no any subtask which is ready to perform; in this case it is "idle", being connected to the community. At the same time, if the community cannot be formed for a certain limited time, the agent is disconnected from the community and tries to connect to another community.

As noted above, the distribution of subtasks among resources begins immediately after the required number of agents are connected to the community. Firstly, we selects the any subtask, which is ready to perform, i.e. all the necessary input data for which are ready. It may the input data for the task, and the data obtained on one of the previous stages of the calculation as well. Then the sub-task sent to the free resource (within the community) with the maximum performance rate. If there no any ready to run subtask the resource wait until it appears.

After the some resource from the community have completed the sub-task, it evaluates its performance during its execution and then it determines whether the community is able to complete the task on time. In case when the community will not fit into the allotted time, the resource agent initiates the dissolution of the community and informs the other agents about this procedure. It is important to note that the agent who currently performs the subtask, will continue their work and will be disconnected from the community only after its finish. In case when all the agents are disabled, the process of forming community begins again and the disabled agents choose the another task to execute from the queue. It should be noted that the dissolution of the community is not happening in the case when the only the last level subtask left to perform, and it task is transferred into the most productive community resource. This is because the loss of time to reform the community can be more than time loss due to execution of the subtask on the another free resource, which will connect to a new community, formed for the same task.

C. Implementation RCS Module Based on a Hierarchical Approach

In order to compare the different approaches for the DCS resources control we realize the RCS module on the basis of the hierarchical approach. The main components of the module is the TheEntryPoint and the Scheduler. The scheduler has a certain set of resources in the submission.

In this RCS user's tasks are entering to the entry point, which are defines by the scheduler, which receives the task for execution. In order to select a certain scheduler, there is performed the selection from the all schedulers that can perform the task for a time less than T_{max}, and then among them is found a scheduler with the smallest workload. The workload is determined by dividing of the total computational complexity of all tasks in the scheduler's queue to the total productivity of all its DCS resources. In the case when the task cannot be completed on time by one of the scheduler, the entry point returns it to the user with the status "Rejected".

The scheduler receives the task and places it last in the queue. Further there is performed a selection of resources for each task in the queue. Unlike the previous approach, for the task is not available to get resources in quantity $\max LV$, but only the amount, which is necessary to perform subtasks of the first level. The performance of the resources, that are chosen for the task, is determined in the same way, as it was implemented in the previous RCS module (see (1)). In case when after performing of the some subtasks there the amount of task (i.e., those for which you have all the input data), more than the allocated resources for this task in the certain time, the scheduler will provide the necessary amount of resources.

The performance of the resources, on which the subtask will be relocated, must satisfy the condition (2):

$$perf > \left(\frac{MIlastST * res_{amnt} + k_{corspnd} * (MIuncompl - MIlastST)}{res_{amnt} * \left(T_{max} - T_{inWork}\right)} \right) \tag{2}$$

where $perf$ – the performance of the resource to which the subtask will be placed; res_{amnt} – number of resources which will be allocated to the task. In case if the resources with performance that satisfies the condition (2) are absent, the subtasks will be placed on any available resources.

Due to the fact, that the tasks are executed asynchronously, i.e. at any moment you may need to provide additional resources to the task, there is a need to reserve a certain set of system resources in order to be allocated to the location of the subtasks. If this is not done, there may happened situation when you begin to run a significant number of tasks and the all DCS resources will be distributed between them. In this case, the possibility to provide the additional resource to the task is absent, and this fact will lead to a significant increase of the execution time of tasks, because the parallel execution of the subtasks is not possible.

The DCS resources reserving procedure is not taken into account the resource productivity, but only their amount. For each task it is reserved 35% $\max LV$ of the tasks.

Thus, the task will run when two conditions are fulfilled:

1) there are sufficient resources with appropriate performance to allocated on them the subtasks of the first level of task (exactly the amount of subtasks are ready to perform, since the input data for them are the input data of the task itself);

2) the amount of unused resources of scheduler is not less than the total amount of reserve resources for all currently performed tasks, including the task for the resources are allocated.

Each task in the queue is checked for these two conditions and, in case if they are satisfied, then this task starts to run.

D. Development of RCS Module for DCS on the Basis of a Network-Centric Approach

A structure scheme of a DCS resource control system on the basis of the network-centric approach is shown on Fig. 1. According to this scheme, the main components of the developed module are: Scheduler, Broker and Resources.

The tasks are sent by the user to one of the Scheduler, which determines the amount and the parameters of the resources necessary to complete the task. The scheduler initiates the information exchange about the resources and task queues of other schedulers status (idle/busy). Then it determines the schedulers to submit the task for execution. It should be noted, that the only information about the task is passed here, and the task with the input data will be send to the resources, where it will be executed. In case when even the all existing resources of the DCS (not only free at the moment but all the resources of a distributed system) will be in use, the task cannot be completed on time, such a task is returned to the user with the status "Rejected".

As soon as the task was received from other scheduler, it is transmitted for execution to the broker with the smallest current load in case when there the sufficient amounts of the necessary resources in the system, or the task is placed in the task queue of the scheduler otherwise.

The principle of the resources selection by performance parameter is similar to that described in the preceding two paragraphs (see (1)). In addition, this RCS module provides the resource reservations on the same principle as the RCS module on the basis of the hierarchical approach.

In order to allocate the subtasks directly to CE it is sent to the broker together with the requirements on the performance of the resources defined by the scheduler. Broker generates a set of compute nodes from the available resources, and this set should meet the specified requirements. In the case when the available DCS resources is not enough, it requests the other brokers on the additional resources. After the set of resources are formed, task is starting.

Similarly to the two previous cases, here the task executes asynchronously, i.e. at any time, you may need to add the resources to the task. In this case, the broker determines the performance of DCS resources in order to ensure the timely completion of tasks (see (2)), then tries to find them in a system and allocates there the subtasks that are ready to execute. If the broker does not have the required resources, it send the request to other brokers to add the resources. In the case where none of the brokers is unable to provide such a resource, we may use any of the free resources. In the absence of free resources on the

DCS, the task is waiting for them, and it continues to run on the available resources.

VI. Experiments and Discussion

A. *Experiments Preparing*

To study the parameters of the suggested RCS modules for DCS they are implemented in Java using GridSim library [16]. The algorithm of these modules work is described in the previous paragraphs of the paper. The experiments are aimed to study the distribution of the computing resources between tasks, so therefore the resources of a different type are not under consideration (data storage, data channels of DCS and others). These researches taking into account other types of resources may be a future deal.

It is important to note that the components of RCS in all three implementations are interacting by exchanging the service messages, and although the impact of their transmission on the information exchange rate in the DCS communication channels is not directly determined, anyway it is important to analyze the number of messages per task, depending on the mechanism of DCS resources control. So, the goal of the experiments is to analyze the functioning of the RCS modules by the following parameters:

– the number of the messages exchanges between components of the RCS module for a single task;
– the percentage of tasks completed on time;
– the waiting time of tasks in the system;
– the utilization (employment) of resources;
– total execution time of tasks batch.

The results of these experiments significantly depend on the parameters of DCS compute nodes as well as on the parameters of the incoming set of tasks, so the experiments were planned to be identical for the all developed RCS modules.

Thus, for the experiments we take the sets of tasks in amount of 100, 250, 500 and 1000 tasks. The format of the tasks is described above. The parameters of the tasks were generated by pseudo-random number generator according to the data presented in table 2.

Table 2. The boundaries of the task parameters

Parameter name	Lower parameter limit	Upper parameter limit
Number of task levels	3	5
Number of subtasks at the level	2	5
Computational complexity of the subtask	50000 MI	200000 MI
The volume of the input data stream (for the subtasks of the first level)	1000 Mb	10000 Mb
The number of input data streams for the subtask (for subtasks of all levels except the first one)	1	5

The tasks are entering in the DCS in the same sequence and at the same time for different experiments. The researches used the distributed computer system of 50 nodes, the performance of which is in the range of 300 MIPS (Million Instructions per Second) up to 700 MIPS. Table 3 presents the division of the resources into categories according to their performance.

Table 3. The Categories of DCS resources and their amount

Category of resources	Performance boundaries	Number of resources
Category I	601-700 MIPS	15
Category II	501-600 MIPS	15
Category III	401-500 MIPS	12
Category IV	300-400 MIPS	8

For RCS module on the basis of the hierarchical approach, the resources were distributed among 5 schedulers in the equal amount, i.e., by 10 resources.

For RCS module on the basis of a network-centric approach, the resources were distributed among 5 schedulers and 10 brokers, and each scheduler has submitted 2 brokers. Thus, each broker directed 5 resources at the beginning of the experiment.

Since the absolute values of the time for the task execution are depended significantly on the such parameters as resource performance, computational complexity of the task, the volume of input and output data for the task, it is expediently to abstract from the absolute values. Therefore, we introduce the concept of a quantum of time and in this study we will take it equal to one hour. If the parameters of tasks or parameters of the computing resources of the RCS is changed, in this way the quantum value also will be different.

The RCS modules have certain parameters, which depend on RCS functions. The table 4 lists the parameters of the modules that were set for the experiments. The following designations are in the table:

– Module A – RCS module based on the communities of computing elements;
– Module B – RCS module based on a hierarchical approach;
– Module C – RCS module based on the network-centric approach.

Table 4. The parameters of RCS modules for the experiments

Parameter	Value	Note
T_{wait}	1/6 quantum	Module A
k_{reserv}	0.35	Module B, Module C
$k_{corspnd}$	1.5	Module B, Module C
Period of re-distribution of the tasks	2,5 quantum	Module C
Number of schedulers	5	Module B, Module C
Number of brokers per scheduler	2	Module C

For the RCS module, which is based on the hierarchical approach, the DCS resources were distributed among 5 schedulers equally, i.e. by 10 resources (Table 4). For the RCS module based on the network-centric approach, the resources were distributed among 5 schedulers and 10 brokers, and 2 brokers reporting to each scheduler (Table 4). Thus, each broker was managed 5 resources at the beginning of the experiment.

Each experiment was repeated 10 times, then the average value was calculated on the results obtained, which is presented in this paper as a result of the experiment.

B. Analysis of the Number of Service Messages Exchanges Between Module Components Per Task

The interaction of the module components is realized by the exchange of service messages. The messages, that are transmitted over the network, create the additional traffic, which can lead to a decrease in the bandwidth of the channels. Let's analyze the number of messages per one task for the various RCS modules. We counted the number of messages, transmitted to each other by the RCS module components, and then we calculate the number of messages per one task. In the case if the connections between the components of the RCS module are realized, for example, through shared memory, the information exchange is performed by writing/reading data from memory and, accordingly, was not taken into account into the messages counting process.

As it shows the diagram (Fig. 2), the number of messages does not depend on the number of tasks in the set. But the number of messages depends on the structure of the task and requires more detailed research, which will be performed further. Also, you can also see that the RCS module based on the hierarchical approach is characterized by a lack of messages, which is quite natural due to the principles of its implementation.

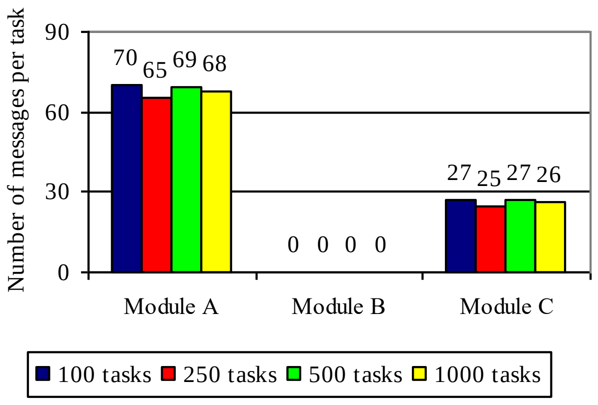

Fig.2. The number of messages per task when the different RCS are in use

It should be noted that for the RCS module based on the network-centric approach the number of messages can be larger in case when the more complex algorithms for the resource allocation are applied. In this case, the first possible resource was selected, the performance of which can ensure the timely execution of the task, so the number of messages is low.

C. Analysis of the Part of the Tasks that are Completed on Time

One of the most important parameters of the task during it processing the RCS is the maximum time period for which it should be finished. It should be noted that as the initial point is taken not the moment when the task enters the system, but the moment when the first subtask is placed on the computing resource.

So, in this case we analyze the correctness of the resources selection for the sub-tasks execution, but the intensity of the tasks flow into the system and the system's throughput are not taken into account.

Fig. 3 shows a diagram of the part of the tasks (in %) that was performed in time for each of the RCS modules and for certain set of tasks.

As can be seen from the Fig. 3, the part of tasks that was performed on time is slightly different for each RCS modules, as well as for the different sets of tasks, and in each case is above 90%. Thus, we may state, that for these parameters (Table 4), each of the RCS modules ensures the timely execution of more than 90% of the tasks from the set.

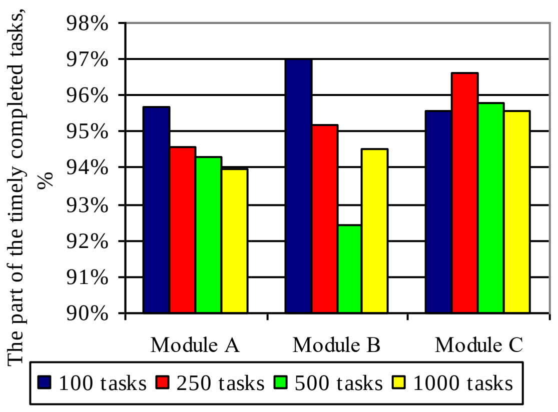

Fig.3. The part of the timely completed tasks for the various RCS modules

D. Analysis of the Waiting Time for the Task in the System and of the Total Execution Time of the Tasks Set

One of the important criteria for the RCS functioning evaluation is the waiting time for the task in the DCS. The waiting time for the task is determined from the moment when the task is entered into the system until the first subtask is placed on the computational resource. It should be noted that the waiting time, among the other things, depends significantly on the intensity of the tasks flow and on the performance of computing resources. Let's analyze these parameters under condition of invariability and RCS, so we define the differences between the average waiting time of the task in DCS.

The experiments were conducted for sets of 100, 250, 500 and 1000 tasks. For each task, the waiting time was fixed, we were repeated 10 experiments for each of the input data. Based on the collected data, we count the average waiting time of the task in the system.

Fig. 4 shows the dependence of the average waiting time for the task in the DCS for the different RCS modules for sets of 100, 250, 500, and 1000 tasks.

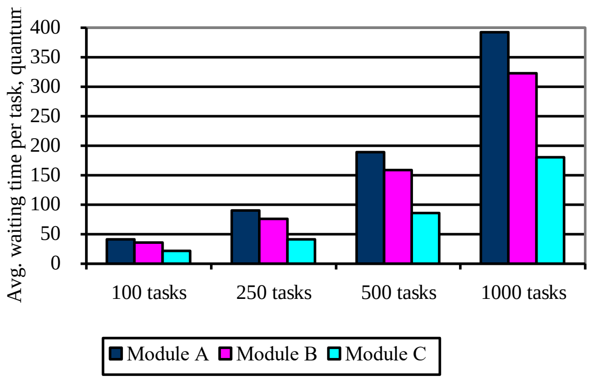

Fig.4. The dependence of the average waiting time for the tasks for the different RCS modules and for different number of tasks in the set

Since the waiting time depends on many parameters, as mentioned above, these experiments are aimed to analyze not the absolute values of the time, but rather comparing the waiting time for various RCS modules. As can be seen from the histogram, the waiting time for Module B is less than the waiting time for Module A by 13%-18%, although this gap is gradually increasing with the number of tasks growth in the set. The waiting time for Module C has been significantly reduced compared to the waiting time for Module A, the reduction is from 47% for 100 tasks to 54% for 1000 tasks.

Let's analyze the execution time for the set of tasks.

The execution time of the task set was determined from the moment when the first task was sent to the execution until the last task from the set was received to execute. As the experiments shows (Fig. 5), the execution time of the task set for Modules A and B is almost the same for different task packets, while the execution time of the task package for Module C was decreased in compare to Module A respectively by 16.9%, 18.37%; 20.3%, 20.7% for packets of 100, 250, 500, 1000 tasks.

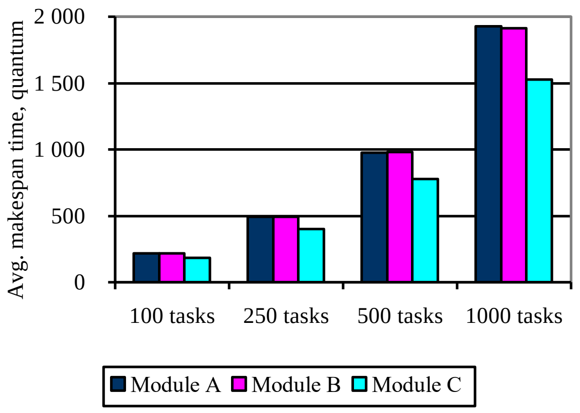

Fig.5. The execution time for the task set for the different RCS modules of and for the different number of tasks in the set

Thus, we compare the results in Figs. 4 and 5, so we can draw certain conclusions:

a) despite the fact that the waiting time for the task for Module B was reduced in comparison with Module A, the execution time of the task package for these two modules is almost the same. This indicates the fact, that the task execution time has slightly increased for Module B. Taking into account the specifics of the task execution algorithms for these Modules, which were described above, this is quite natural, since in the Module A, at the begin of the task running, the number of resources is equal to the maximum number of subtasks at the level, and these resources are inalienable until the task will be completed. So, the situations when it is necessary to place the next sub-task for execution, and the free resource is unavailable, do not arise. At the same time, for the Module B, such situations are possible, because only a small part of resources are reserved, in addition, any resources are reserved despite its performance. Therefore, for the Module B, there may be situations when a free resource with the required performance is unavailable, or when there are no any available resources at all. For example, during the experiments such situations arose: for a set of 100 tasks was fixed an average of 31 cases of lack of resources with the required productivity and 2 cases of lack of free resources. For a set of 1000 tasks, these indicators grew to 356 and 13 cases, respectively;

b) the waiting time of the task for Module C (in comparison with Module A) decreased almost twice, while the time for the task set execution was reduced by only 16% -20%. So, we can conclude that the time of the tasks execution has increased. The reason for this, as described above for Module B, is the situations when there is no any free resource with the required performance or lack of free resources in general, which are also the specific of the Module C. However, for Module C the number of such situations is smaller, since between brokers in the Module C there are horizontal links that allow them to share resources. Due to this, the total time for tasks set execution is reduced.

E. Analysis of the DCS Resources Utilization

An important indicator of the DCS is the resource utilization, since the idle resources have a negative impact on the DCS performance.

Table 5. The DCS resource utilization for the different RCS modules and different sets of tasks

Resources category / Number of tasks	Category IV 300-400 MIPS	Category III 401-500 MIPS	Category II 501-600 MIPS	Category I 601-700 MIPS	Average utilization of the all DCS resources (%)
Module A					
100 tasks	32.78%	44.07%	53.25%	65.03%	48.78%
250 tasks	36.17%	47.45%	55.29%	67.63%	51.64%
500 tasks	36.53%	49.00%	56.83%	69.02%	52.85%
1000 tasks	37.34%	49.24%	57.76%	69.48%	53.46%
Module B					
100 tasks	21.20%	48.99%	54.03%	65.01%	47.31%
250 tasks	22.03%	49.24%	57.85%	68.80%	49.48%
500 tasks	25.35%	49.67%	59.42%	69.50%	50.99%
1000 tasks	24.43%	54.33%	59.20%	70.60%	52.14%
Module C					
100 tasks	18.19%	61.67%	72.43%	71.06%	55.84%
250 tasks	17.78%	66.50%	77.57%	77.70%	59.89%
500 tasks	18.14%	68.54%	82.28%	81.61%	62.64%
1000 tasks	18.51%	70.49%	82.81%	82.97%	63.70%

During the experiments described above, the data were collected on the time of the resources utilization by the full time of DCS functioning. The working time of the resource was considered to be the time from the moment when the subtask was transferred to it for execution until this task will be completed, and the rest of the time this resource was considered to be unused. During the experiments, we define the total working time and total idle time for each resources, then we determine the resources utilization. According to the received data, there was determined the average resources utilization (in %) by categories for each of the RCS modules (Table 5).

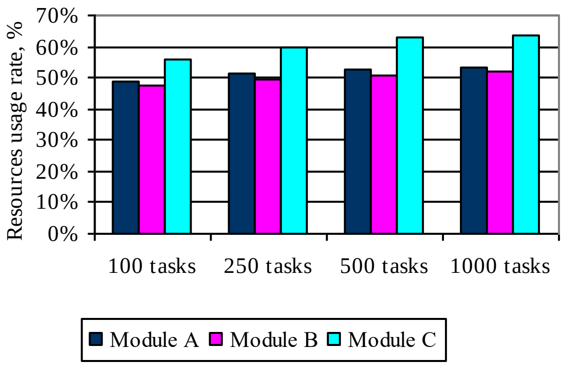

Fig.6. The average resource utilization (%) for the different RCS modules

Fig. 6 shows the histograms of the average resource utilization for the various RCS modules.

As can be seen from Table 5 and Fig. 6 for Module C, the average resources utilization is higher than for Modules A and B on 15% - 18%. This is because in the Module A resources are idle in the time when they are connected to the community, but there is no any ready-to-perform subtask for them. For the Module B and Module C the idle is the part of resources that is reserved for requests for the additional resources of tasks that are currently executing. However, for Module C, the idle time is lower, since resources are transferred between brokers, and this fact increases the number of tasks the subtasks of which can be allocated to execute on a computing resource. These experimental results also confirm the facts described above, in particular, the reduction of the execution time for the set of tasks for Module C.

VII. Conclusion

In this paper we proposed the implementation of the RCS module based on the network-centric approach. We perform a comparative analysis of the parameters of the suggested RCS module with RCS modules based on the hierarchical approach and on the basis of the computing resources communities formation for the tasks execution.

As a result of the conducted experiments, we found that the RCS module on the network-centric approach allows us to low as much as 2 times approximately the waiting time of task in DCS. Also, for this RCS module, the execution time for the tasks set was reduced by an average of 18.5% compared to other modules. The experiments showed an average increase of the resource utilization rate with this module, on 17%.

Another parameter that was analyzed during the research is the number of information messages exchanges during the tasks allocation and execution.

As revealed in the result of study, the number of message exchanges per task for RCS module on the basis of the network-centric approach is lower by 61% than for RCS module based on the formation of computing resources communities for the task. But it should be noted that these results are received with a simple algorithm for the resources selection, and in the case of implementing of the another (more complex) algorithm, the results may differ significantly from those obtained.

This study took into account only the resource performance when selecting the resources for the task, but did not take into account a number of the other parameters, for example, the volume of input and output data of the task, the data transmission rate in the communication channels of the network that connects the computing resources of the DCS. The study of these parameters is the future work. Also, further work is to investigate the effect of the task parameters on the parameters of the DCS functioning and the applying of the more complex algorithms for the resource allocation for RCS module based on the network-centric approach.

References

[1] Z. Hu, V. Mukhin, O. Barabash, Y. Kornaga, O. Herasymenko, «Analytical Assessment of Security Level of Distributed and Scalable Computer Systems», *Int. J. of Intelligent Syst. and Applicat*., vol. 8, no. 12, p. 57 – 64, Dec. 2016. doi: 10.5815/ijisa.2016.12.07.

[2] V.Mukhin, "Adaptive approach to safety control and security system modification in computer systems and networks" , *Proc. of the 5th IEEE International Workshop on Intelligent Data Acquisition and Advanced Computing Systems: Technology and Applications, IDAACS'2009. Rende (Cosenza)*, Italy, 21 -23 Sept. 2009, pp.212–217.

[3] H. Hussain, S. Ur R. Malik, A. Hameed, S. Ullah Khan, G. Bickler, N. Min-Allah, M. B. Qureshi, L. Zhang, W. Yongji, N. Ghani, J. Kolodziej, A. Y. Zomaya, Ch.-Zh. Xu, P. Balaji, A. Vishnu, F. Pinel, J. E. Pecero, D. Kliazovich, P. Bouvry, H. Li, L. Wang, D. Chen and A. Rayes, "Survey on Resource Allocation in High Performance Distributed Computing Systems", *Parallel Computing*, vol. 39, issue 11, pp.709-736, 2013.

[4] M. B. Qureshi, M. M. Dehnavi, N. Min-Allah, M.S. Qureshi, H. Hussain, I. Rentifis, N. Tziritas, T. Loukopoulos, S. U. Khan, Ch.-Zh. Xu and A. Y. Zomaya, "Survey on Grid Resource Allocation Mechanisms", *Journal of Grid Computing*, vol. 12, issue 2, pp.399-441, 2014.

[5] S. P. Singh, S. Ch. Sharma, "A Particle Swarm Optimization Approach for Energy Efficient Clustering in Wireless Sensor Networks", *International Journal of Intelligent Systems and Applications(IJISA)*, vol.9, no.6, pp.66-74, 2017. doi: 10.5815/ijisa.2017.06.07.

[6] Y. Zhu and L. M. Ni, *A Survey on Grid Scheduling Systems* [Online], Technical Report SJTU_CS_TR_200309001, Department of Computer Science and Engineering, Shanghai Jiao Tong University, 2013, 41p. Available at: http://www.cs.sjtu.edu.cn/~yzh u/reports/SJTU_CS_TR_200309001.pdf. (last access date 09.06.2017).

[7] D. P. Vidyarthi, B. K. Sarker, A. K. Tripathi and L. T. Yang, *Scheduling in Distributed Computing Systems: Analysis, Design & Models (A Research Monography)*, Springer Science+Business Media: LLC, USA, 2009, 302p.

[8] Bh. V. Chawda, J. M. Patel, "Investigating Performance of Various Natural Computing Algorithms", *International Journal of Intelligent Systems and Applications(IJISA)*, vol.9, no.1, pp.46-59, 2017. doi: 10.5815/ijisa.2017.01.05.

[9] M. Rahman, R. Ranjan, R. Buyya, "Decentralization in Distributed Systems: Challenges, Technologies, and Opportunities", in *Advancements in Distributed Computing and Internet Technologies: Trends and Issues*, A.-S. K. Pathan, M. Pathan and H. Y. Lee, Eds. Information Science Reference (an imprint of IGI Global), USA, 2012, pp.386-399.

[10] P.-O. Östberg, D. Espling and E. Elmroth, "Decentralized Scalable Fairshare Scheduling", *Future Generation Computer Systems*, vol. 29, issue 1, pp.130-143, 2013. doi:10.1016/j.future.2012.06.001.

[11] Y. Huang, N. Bessis, P. Norringtonb, P. Kuonend and B. Hirsbrunner, "Exploring Decentralized Dynamic Scheduling for Grids and Clouds Using the Community-aware Scheduling Algorithm", *Future Generation Computer Systems*, vol. 29, issue 1, pp.402-415, 2013. doi:10.1016/j.future.2011.05.006.

[12] A. Kalyaev and Ia. Korovin, "Adaptive Multiagent Organization of the Distributed Computations", *AASRI*

Procedia 6, p.49-58, May 2014, doi:10.1016/j.aasri.2014.05.008.

[13] D. S. Alberts, J. J. Garstka and F. P. Stein, *Network Centric Warfare: Developing and Leveraging Information Superiority*, 2nd Edition (Revised), CCRP publication series, 1999, 294p.

[14] V. Muliukha, A. Ilyashenko, V. Zaborovsky and A. Lukashin, “Cyber-physical approach to the network-centric robotics control task”, AIP Conference Proceedings, 2016, 4p. doi: 10.1063/1.4965420.

[15] S. R. Ditmeyer, *Network-centric Railway Operations: Utilizing Intelligent Railway Systems* [Online], Journal of Transportation Law, Logistics and Policy, vol. 77, number 3, 28p., 2010. Available at: http://www.transportation.northwestern.edu/docs/2011/2011.03.15.Ditmeyer_Paper.pdf. (last access date 09.06.2017).

[16] R. Buyya and M. Murshed, “GridSim: A Toolkit for the Modeling and Simulation of Distributed Resource Management and Scheduling for Grid Computing” [Online], *The Journal of Concurrency and Computation: Practice and Experience (CCPE). – Wiley Press*, 37p., 2002. Available at: http://arxiv.org/abs/cs/0203019. (last access date 09.06.2017).

Authors’ Profiles

Zhengbing Hu: Associated Professor of School of Educational Information Technology, Huazhong Normal University, PhD.

M. Sc (2002), PhD (2006) from the National Technical University of Ukraine Kiev Polytechnic Institute”.

Postdoctor (2008), Huazhong University of Science and Technology, China.

Honorary Associate Researcher (2012), Hong Kong University, Hong Kong.

Major interest: computer science and technology applications, artificial intelligence, network security, communications, data processing, cloud computing, education technology.

Vadym Mukhin: Professor of computer systems department of National Technical University of Ukraine “Kiev Polytechnic Institute”, Doct. of Sc.

Born on November 1, 1971. M. Sc. (1994), PhD (1997), Doct. of Sc. (2015) from the National Technical University of Ukraine “Kiev Polytechnic Institute”; Assoc. Prof. (2000), Professor (2015) of computer systems department.

Major interest: the security of distributed computer systems and risk analysis; design of the information security systems; mechanisms for the adaptive security control in distributed computing systems; the security policy development for the computer systems and networks.

Yaroslav Kornaga: Assoc. professor of computer systems department of National Technical University of Ukraine “Kiev Polytechnic Institute”, PhD.

Born on January 1, 1982. M. Sc. (2005), PhD (2015), from State University of Telecommunications; Assoc. Prof. (2015) of techical cybernetics department.

Major interest: the security of distributed database and risk analysis; design of the distributed database; mechanisms for the adaptive security control in distributed database; the security policy development for distributed database.

Oksana Herasymenko: Assist. professor of networking and Internet technologies department, Taras Shevchenko National University of Kiev.

Born at 1983 in Borzna town, Chernihiv region, Ukraine. M. Sc. in computer system and networks (2006) from Chernihiv State Technological University, Ukraine.

Major interests: resource control in distributed computing system and data mining. Now she is preparing her Ph.D. thesis in information technologies.

Yaroslav Lavrenko: Assoc. professor of dynamics and strength of machines and strength of materials department of National Technical University of Ukraine “Kiev Polytechnic Institute”, PhD.

Born on March 15, 1983. M. Sc. (2006), PhD (2014), from National Technical University of Ukraine “Kiev Polytechnic Institute”; Assoc. Prof. (2015) of dynamics and strength of machines and strength of materials department.

Major interest: Dynamics and durability of high rotation system, vibrations.

How to cite this paper: Zhengbing Hu, Vadym Mukhin, Yaroslav Kornaga, Yaroslav Lavrenko, Oksana Herasymenko,"Distributed Computer System Resources Control Mechanism Based on Network-Centric Approach", International Journal of Intelligent Systems and Applications(IJISA), Vol.9, No.7, pp.41-51, 2017. DOI: 10.5815/ijisa.2017.07.05