UDK 621.391 DOI: 10.20998/2413-3000.2017.1225.17

S. V. LISTROVOY, K. A. TRUBCHANINOVA, V. O. BRYKSIN, M. S. KURTSEV

A UNIFORM PROCEDURE OF A SYSTEM RESOURCES INTERACTION IN DISTRIBUTED COMPUTER MEDIA

Детально описана процедура розподілення ресурсів на основі принципу роздільного розподілення задач та на основі методу групової вибірки. Запропонована процедура взаємодії усіх ресурсів, яка дозволяє організувати процес планування виконання завдань в розподілених обчислювальних системах. Також запропонована уніфікована процедура організації взаємодії усіх ресурсів розподіленої обчислювальної системи, яка дозволяє з єдиних позицій організувати процес планування.

Ключові слова: розподілені обчислювальні системи, планування, розподілення завдань, розподілення ресурсів.

Подробно описана процедура распределения ресурсов на основе принципа раздельного распределения задач и на основе метода групповой выборки. Предложена процедура взаимодействия всех ресурсов, которая позволяет организовывать процесс планирования выполнения заданий в распределенных вычислительных системах. Также предложена унифицированная процедура организации взаимодействия всех ресурсов распределенной вычислительной системы, которая позволяет с единых позиций организовать процесс планирования.

Ключевые слова: распределенные вычислительные системы, планирование, распределение заданий, распределение ресурсов.

Nowadays railways present the processes of the existing information systems and their integration into a uniform automated control system of railway freight transportation. These processes must be based on the uniform architecture, system construction which takes into account current and future information technologies such as grid technologies and cloud calculation technologies. One of the main problems of introducing these technologies is the increased time while planning resources distribution when fulfilling tasks, that appear in distributed control system. It is important to develop uniform approaches to the solution of this problem. A uniform procedure of all the resources interaction that allows organising planning tasks in distributed computer systems in the given article. The procedure of resources distribution on the base on the principle of task distribution and cluster sampling method has been also described.

Keywords: distributed computing systems, scheduling, task allocation, resource allocation.

Introduction. At present, in all the branches of economics including a transportation industry, cardinal changes take place in the field of information technologies and systems. Thus, according to the Ukrainian transport policy to 2020, the principal implementation directions are the availability and quality support of transport services owing to the development of integrated data systems for control, monitoring and identification of cargoes and containers. Nowadays, the existing railway transport data systems are under upgrading and integrating thereof into a united automated cargo carriage control system of Ukrainian railway lines. The processes are to be based on a unified architecture of system integration taking into account the peculiarities of modern and advanced information technologies, such as Grid and cloud computing methods. One of the main problems of realisation of these methods is the problem of resource allocation scheduling when processing the tasks in a distributed control system. So, the development of the universal general approaches to solving the problem seems to be of current importance.

Problem definition and formalisation. As shown in articles [1,2], in general, the given problem in a distributed arbitrary computer environment may be presented as a tuple:

$$\begin{aligned}&Schedule=(T, R, ComLinkThroughout,\\&MappingEvent, F(Matching), SchedMethod,\\&ResChar, TaskChar, StratSchedul(Focus),\\&StratExec, ObjectFunc, Mode, NoLevel),\end{aligned}$$

where T – global stream task set;

R – resource set;

$ComLinkThroughout$ – set of communication links and their task-resource throughput;

$MappingEvent$ – time of task mapping onto resource;

$F(Matching)$ – the function of task mapping onto resources;

$ResChar$ – a characteristic set of resources R;

$TaskChar$ – a characteristic set of tasks T;

$StratSchedul(Focus)$ – strategy set of task scheduling;

$StratExec$ – the strategy set of task execution onto resources;

$ObjectFunc$ – set of objective functions;

$Mode$ – set of different scheduling modes;

$NoLevel$ – scheduling level.

The set T finds a set of global task stream entered for processing into a distributed computing system (DCS): $T=(T_1, T_2, T_3, \dots, T_m)$.

The set R defines a set of all the DCS resources: $R=(R_1, R_2, R_3, \dots, R_n)$.

Time of tasks mapping $MappingEvent$ onto resources is specified by a minimal number of tasks for scheduling determined by a number of available resources and/or minimal time for monitoring the system resource and task state.

The mapping function F of tasks T onto resources R of the system presents the compliance matrix $F(Matching): T\times R\times ComLinkThroughout \rightarrow R^{+}$, where $Matching$ is a compliance matrix of scheduled tasks T with DCS resources R taking into account throughput of a set of communication links $ComLinkThroughout$ between the scheduled tasks and resources of DCS.

The set of scheduling methods $SchedMethod$ defines the scheduling methods of tasks T onto resources R. The set of task scheduling strategies $StratSchedul(Focus)$ defines the scheduling strategies described by the tuple $StratSchedul(Focus) = (SystemOrient, UserOrient)$, where $SystemOrient$ is a system-oriented scheduling strategy oriented (focused) on improvement of the productivity and throughput of DCS (meta scheduler level of task streams); $UserOrient$ is task scheduling strategy oriented (focused) on users (application layer (local resource manager or scheduler)).The strategy set $StratExec$ of task T executing on resources R allocated for them from the set described by the tuple $StratExec = (Deadline, Budget)$, where $Deadline$ - a directive time of task execution; $Budget$ - restriction to budget (cost) of task execution. The set of objective functions $ObjectFunc$ characterizing the DCS operation described in the tuple

$$ObjectFunc = (Utilization, LoadBalance, Time(Cost), Makespan, Penalty),$$

where $Utilization(NoResources, ResUtiliz)$ is a number of used resources $NoResources$ and utilization factor value $ResUtiliz$ of the system resources R; $LoadBalance$ - load balancing of resources R; $Time(Cost)$ - time (cost) of executing the global queue tasks; $Makespan$ - maximal time of completing the execution of global queue tasks T on resources R; $Penalty$ - penalty when exceeding the allowed runtime or directive time of task execution. The set of scheduling modes of task execution $Mode$ in DCS described by the tuple $Mode = (Batch, Online)$, $Batch$ - batch mode of task execution; $Online$ - scheduling mode and task execution directly after their entering into DCS for processing. A set of scheduling levels of task execution $NoLevel$, described by a tuple $NoLevel = (Global, Local)$, where $Global$ - the global level of task scheduling – meta scheduler level of task streams; $Local$ - local level of task scheduling – local scheduler level (schedulers of local resource control systems) of local task batches. The set of $ResChar$ defines a set of system resource R characteristics described by the tuple:

$$ResChar = (NoCores, NoProcessors, MinDisk, MinMemory, OS, MinCPUFreq, MaxCPUFreq, MinCPUPower, MaxPUPower, Architecture, ComLinkThroughout\ (R_j)),$$

where $NoCores$ - number of processing cores of resource;

$NoProcessors$ - number of resource processors;

$Disk$ - resource disk size;

$Memory$ - resource random access memory size;

OS - type of operating system used onto resource;

$MinCPUFreq$ - minimal resource processor unit frequency;

$MaxCPUFreq$ - maximal resource processor unit frequency;

$MinCPUPower$ - the minimal processor unit power consumption;

$MaxPUPower$ - maximal processor unit power consumption;

$Architecture$ - resource architecture type;

$ComLinkThroughout\ (R_j)$ - communication line throughput to resource R_j.

The set $TaskChar$ defines a set of characteristics of the system tasks T described by the tuple:

$$TaskChar = (TaskResourceRequirements(TaskType, NoCores, minDisk, minMemory, Software, Hardware, OS), walltime, deadline, budget),$$

where $TaskResourceRequirements$ - resource task requirements; $TaskType$ - task type: Parallel, Serial; $NoCores$ - the required core number for execution; $minDisk$ - minimal size of disk memory; $minMemory$ - minimal size of random access memory; $Software$ - the required software; $Hardware$ - the required hardware; OS - the required operating system; $walltime$ - the expected task execution time onto resource; $deadline$ - directive time of task execution; $budget$ - cost restriction (budget) of task execution, determined from the formula $cost(walltime) + penalty(deadline) \le budget$.

The general concept of establishing the dispatching control in Grid. In the system of optimal scheduling of resource allocation in a regional level Grid, from the viewpoint of upgrading the efficiency and flexibility of using the resources, it is recommended using a four-level hierarchical controlling and scheduling structure, which combines centralised and decentralised modes. The first level of control should ensure the coordination of resource and task reallocation among the groups of global network regions; the second level of control is to ensure the required service quality of online communities dynamically changing in the network within the region enabling the network resources to be used at the most [3,4,5].

Irrespective of the first one the third level, on the basis of the common task pool, puts into effect the task reallocation among the clusters and task interchange among themselves. And the fourth level schedules the task processing inside the cluster. For this purpose, it is advisable to have a central control with the main functions of servicing the common task pool and coordinating the operation of the second-level execution managers, practically without interfering in the process of their scheduling but supplying the second-level execution managers with information for scheduling. The second-level execution managers select themselves the tasks from the common pool, may send them to each other and forward directly to a particular cluster for solving. When the online communities are made up and the communities present their tasks to the central control, the last offers at a determined price the connection of extra network resources that are free in the network.

At that, it is supposed that those, who wish to submit their resources, inform the central control, which takes the functions of an intermediary to submit the extra resources to the second-level execution managers. The intermediary activity of the central control let the second-level execution managers be free of analysing and processing the state of all the network resources and be fully occupied with scheduling the resource allocation through a common task pool in the network.

At the interregional level, the central controls of every region interact with each other forming the distributed central control of the global network appropriate to the fourth level of control generating the data streams of resource state in global network regions [6,7,8]. Let's consider the principles of Grid system generation, where distribution of tasks is affected in heterogeneous computing environment, and in general they are to solve such problems as definition of functional purpose of every Grid architecture component, definition of general principles of the Grid component interaction, building the mathematical support and software that ensures effective and reliable functioning of Grid systems. Let's consider the Grid system as a heterogeneous environment where every resource R_i is characterised by a characteristic vector $(\alpha_1^{R_i}, \alpha_2^{R_i}, \ldots, \alpha_q^{R_i})$ where every task З_i will also be specified by characteristics $(\alpha_1^{\text{З}_i}, \alpha_2^{\text{З}_i}, \ldots, \alpha_q^{\text{З}_i})$. The task З_i may be executed onto the resource R_i if inequalities $\alpha_1^{R_i} \le \alpha_1^{\text{З}_i}; \alpha_2^{R_i} \le \alpha_2^{\text{З}_i}; \ldots, \alpha_q^{R_i} \le \alpha_q^{\text{З}_i}$ are solved.

Let's consider the distribution of tasks based on the principle of a shared problem allocation, refer to Fig.1, but at that the distribution with the help of an execution manager will not be affected statically [9,10], as is done in the known shared task allocation maps, but dynamically and uninterruptedly on the basis of the following procedure *D*.

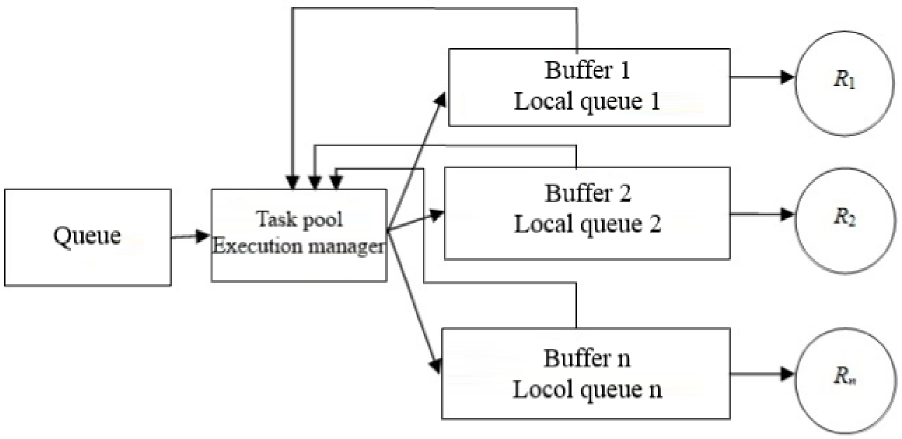

Fig. 1 – Shared task allocation in heterogeneous environment

Procedure *D*.

On the basis of the task and free resource data, a correspondence table is set up in the execution manager, the resources $\{R_i\}$ correspond to the table columns, the problems to the bars from the task queue. At the crossing of the bar and the column, we'll put 'one' when comparing $(\alpha_1^{R_i}, \alpha_2^{R_i}, \ldots, \alpha_q^{R_i})$ and $(\alpha_1^{\text{З}_i}, \alpha_2^{\text{З}_i}, \ldots, \alpha_q^{\text{З}_i})$ it runs out that the problem may be executed on an available free resource and 'zero' if this is not the case. In the future, while scheduling, it is suggested that one of the following procedures should be used. The first one is based on solving the least covering problem (LCP), we find a minimal number of resources, where the formed task queue may be executed, and send the tasks for solving to the dedicated resources. Then, the queue is replenished with new tasks and the procedure iterates, the tasks with zero bars in the table are sent back into the queue and the administrator is informed about the impossibility of executing the particular task onto one resource; the second procedure is based on the utilisation.

A mathematical model of that particular procedure is the task of linear Boolean programming $L_t = \sum_{j=1}^{n} x_j(t_k) \to \min$ with the restrictions $\sum_{j=1}^{n} \beta_{ji} x_j(t_k) \ge 1;\ i = \overline{(1, m)}$; $\beta_{ji} \in \{0,1\}$; $x_j(t_k) \in \{0,1\}$, where m – number of tasks subject to scheduling, n – number of the system resources available and free at the scheduling moment t_k $[T_0, T_n]$. The scheduling is realised within the time interval $[T_0, T_n]$, where T_0 – scheduling start time, T_n – scheduling end time of tasks. The problem may be considered as the problem of evaluating the maximal number of columns in a Boolean matrix *B* covering all the bars of the particular matrix, the elements thereof in the context of solving the scheduling problem are interpreted as follows: the distributed computer system resources available and free at the scheduling moment correspond to the columns, and the tasks, which are subject to scheduling and to be solved onto those resources, correspond to the bars. The peculiarity is that the order of 'ones' in the matrix *B* is changed dynamically.

The second procedure is based on the batch sample method, where several tasks out of the task queue are served at a time. Selected are the tasks requiring different resource types for implementation, and the sum of their priorities should be maximal. In the case of availability of equal tasks, 'the oldest' are selected. Thus, the largest number of tasks that use different types of resources are required to be selected out of the queue, and the sum of the selected task priorities should be maximal. Moreover, the tendency of the sum of the selected task priorities to maximal is the main criterion when selecting the tasks out of the queue. Let's assume that the task requiring the resource of a particular type may be executed onto any resource of this type. As a matter of principle, a particular set of resources may be required for executing the task Зk. Assume that $\{\vec{X}\}$ is a set of all the variants of task selection out of the queue, $\vec{X}$ is one of the variants of task selection, and $\vec{X}=\{x_1,x_2,\ldots,x_k,\ldots,x_p\}$, where $k=\overline{1..p}$, p – a number of tasks in the queue, x_k – Boolean variable equal to 1, if the task $З_k$ is selected in this variant, and to 0 if not, C_k is a priority of $З_k$ task. To characterise the sum of the selected task priorities we use the functional $F=\sum_{k=1}^{p}C_k x_k \to \max$. Assume, that A_{kg} is a Boolean variable equal to 1 if the task $З_k$ uses the resource R_g, and to 0 if not. B_g is the number of resources of the particular type R_g. Then, from the condition, that at any time any resource may be used for executing the task, we get M restrictions of the type: $\sum_{k=1}^{p}A_{kg}x_k \le B_g$, $g=\overline{1..M}$. Hence, we need to get $\vec{X}$ a sample out of the set $\{\vec{X}\}$, where the functional will be maximal when satisfying all the restrictions. We have got the linear programming problem with Boolean variables. The request queue with such a formal characterization is enabled by stages.

Every stage consists of detecting the optimal sample $\vec{X}$, its servicing and changing the functional and restrictions with consideration for changes in a queue after servicing the sample. The effective implementation methods of both the procedures are presented in articles [1–3]. The dynamic procedure D forms local queues ensuring maximal loading of every resource. As any Grid system is potentially a heterogeneous environment, let's consider the application possibilities of that particular procedure for scheduling the task execution in it. In the Grid system it is advisable to have two basic levels, i.e. the servers of analysis of the resource state and the clients' tasks, which should execute the analysis of the clients' tasks direct-connected to a server and the analysis of the resources onto which the clients' tasks may be executed. The servers of this type actually initiate the stream of Grid problems in the network. At that, it is advisable to divide the network into zones within which the control servers may monitor free resources of a group of clusters $\{K_i\}$, Fig. 2, connected to this particular control server in T time not exceeding a certain admissible time $T_д$. The servers exchange the information on free resources availability in their own zones, Fig. 3.

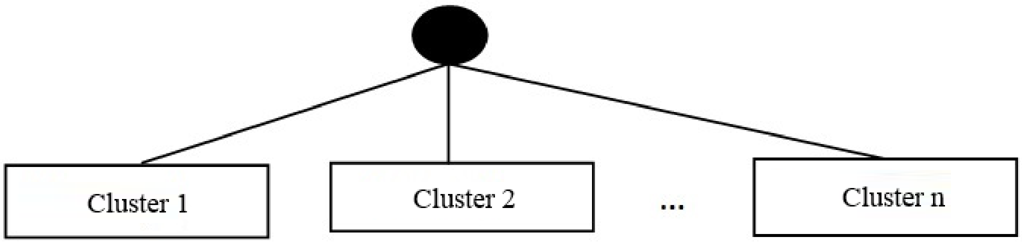

Fig. 2 – Zone $\{K_i\}$ clusters which may be inquired in $T \le T_д$ time

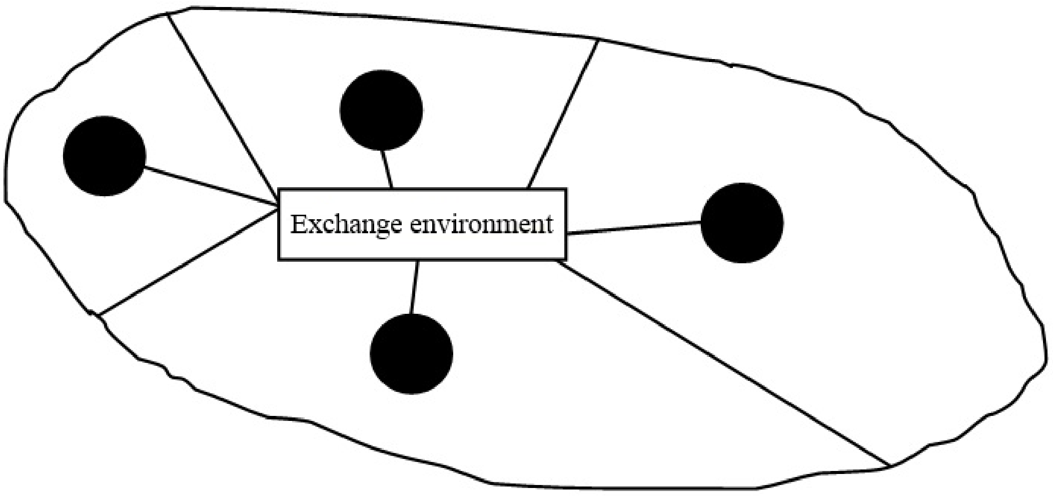

Fig. 3 – Interaction of control servers

If there are no free cluster resources in a server zone, then the tasks are passed for solution to a control center with a maximal number of free resources in clusters $\{K_i\}$ of its zone, at that, the priority of these tasks is to be reasonably increased for avoiding the situation when any tasks may remain unsolved for a long time.

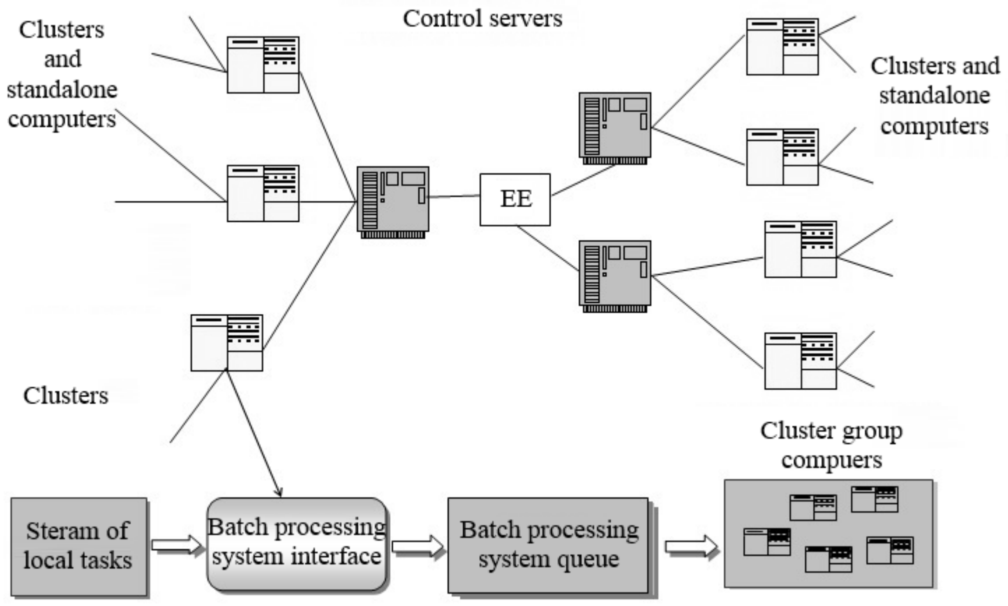
Clusters and standalone computers
Control servers
Clusters and standalone computers
EE
Clusters
Cluster group compuers
Steram of local tasks
Batch processing system interface
Batch processing system queue

Fig. 4 – Two-level Grid architecture

The problem queue is in the control server buffer, at that, every problem is characterised by the vector of indices $(\alpha_1^{3_i}, \alpha_2^{3_i}, \ldots, \alpha_q^{3_i})$ as the storage space required for solving the problem, requirements to the operating resource system, the agreed price for a solution, and etc. Any cluster $\{K_i\}$ presents data to a control server of its zone, data of available free resources and their characteristics in the same format as for problems $(\alpha_1^{R_i}, \alpha_2^{R_i}, \ldots, \alpha_q^{R_i})$. A task pool is formed from the server buffer; the pool is required to be sent into clusters $\{K_i\}$ for execution. Based on the data of problems and free resources of clusters, a table is formed in the control centre database of the particular zone. The table columns are zone clusters $\{K_i\}$, the bars are the problems out of the formed task pool. At the intersection of a bar and a column, we'll put 'one' if the result of the comparison $(\alpha_1^{3_i}, \alpha_2^{3_i}, \ldots, \alpha_q^{3_i})$ and $(\alpha_1^{R_i}, \alpha_2^{R_i}, \ldots, \alpha_q^{R_i})$ shows that the particular problem may be executed onto the available free resource and 'zero' if this is not the case. Further, when solving the problem with the use of one of the suggested procedures, we find a minimal number of clusters, where the formed task pool may be executed, and send it to selected clusters for execution.

Then the next task pool is selected out of the buffer and the procedure repeats, those tasks that have the corresponding zero bars in the table are sent for solution to a nearby zone. When forming the task pool out of a buffer, preference as to selection is given to those tasks that got through a large number of control centers. The sets of clusters $\{K_i\}$ of any zone form the second level. Thus, a two-level system is presented in Fig.4. At the first level the control servers of the centers exchange the information on free resource availability in the information exchange environment (EE) and if there are no free resources in a server zone, then it sends the task under solution to a control server, where the zone has the largest number of free resources. At the second level there are two competitive streams, i.e. the Grid task stream and the local task stream.

Generalised upper-level model of allocating the resources in GRID system based on procedure D

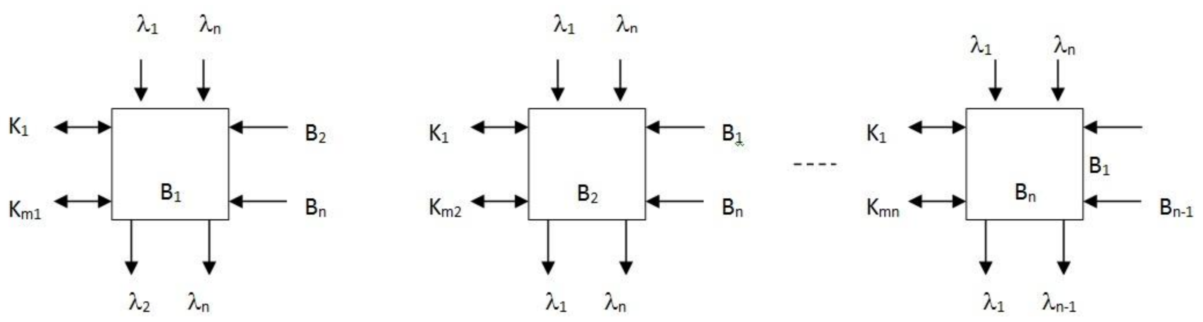

Fig. 5 – Interaction model of upper-level brokers

The loading brokers $\{B_i\}$ are composed on the basis of a correspondence table stored in a database (DB). The table columns correspond to free resource types and their quantity, and the table bars correspond to the types of the problems that may be solved onto these resource types. If the number of tasks in the correspondence table column exceeds the number of resources of the appropriate type, then only the tasks with maximal total priority are left in the correspondence table. The peculiarity of heterogeneous system operation is that some tasks may be executed only onto the particular resource types and other tasks onto some types of resources, and while locating the tasks arbitrary in a queue as per the correspondence table, a competition of some tasks to one resource may result in inequality of resource loading and a considerable delay in execution of competitive tasks.

The tasks are desired to be distributed in a queue at a previous step in such a way that at the next step the maximum number of free resources may be prepared for executing the next subset of tasks. For that, a common queue is suggested to be converted to local ones on the basis of an interrupted execution of *D* procedure, i.e. at any step of scheduling a local queue the minimal quantity of brokers is defined, that may be suitable for execution of all the tasks located in a task pool by an execution manager and the tasks are queued in a broker group that may solve all pool of tasks. Moreover, first a broker queue is filled as more tasks will be executed onto it and if the queue starts exceeding a buffer capacity, then the tasks are shifted to a next broker of this group capable of executing the tasks. If such the brokers are not available in the group, then any broker being capable of executing this task type is selected as per the correspondence table, i.e. the procedure *D* has been executed. If the task may be executed only onto that particular broker, where a local queue exceeds the buffer capacity, then the task is sent back to the task pool. The queues to clusters $\{K_i\}$ connected to broker B_i are formed in the same way on the basis of uninterrupted execution of the procedure *D*. Let's consider the upper-level broker model in details. Every broker at the same time is a analysis center of tasks entering in the Grid system, see Fig. 6.

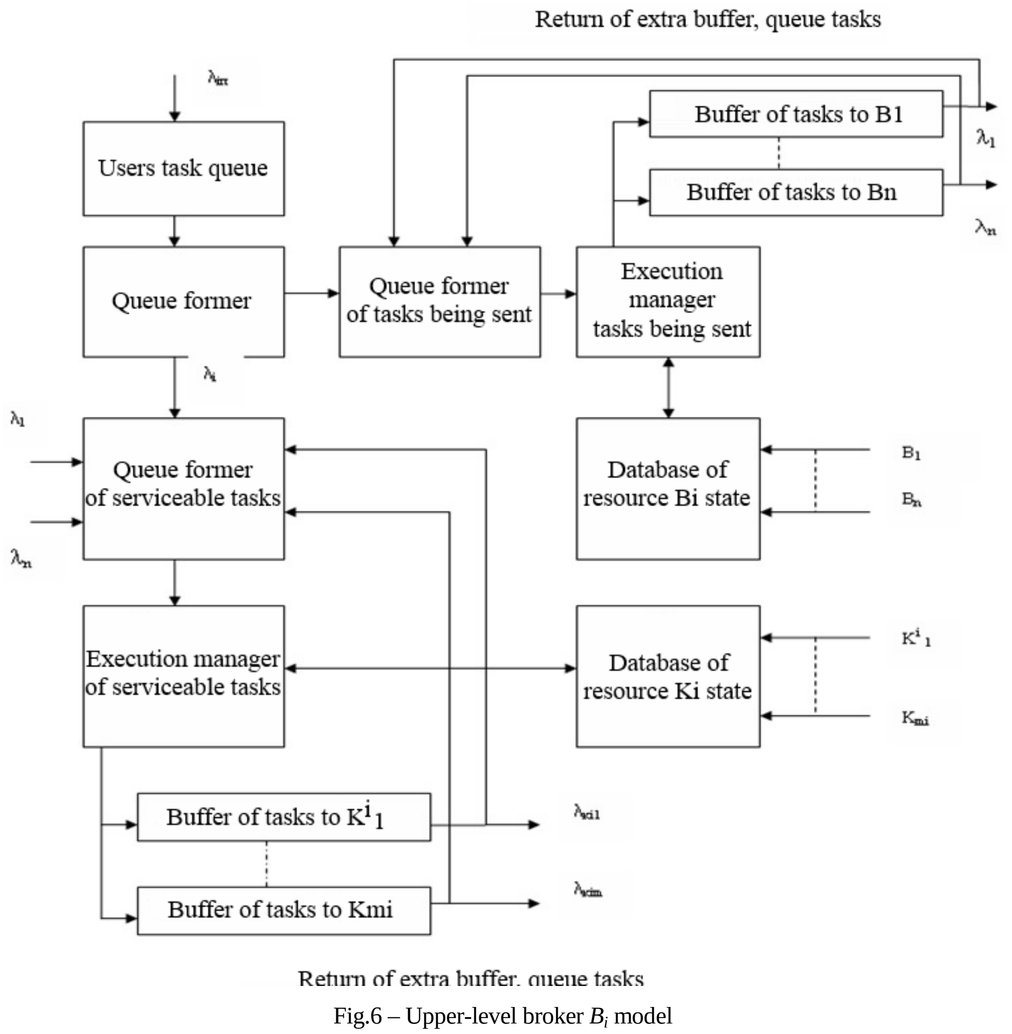
Return of extra buffer, queue tasks
λ_{in}
Users task queue
Queue former
Queue former of tasks being sent
Execution manager tasks being sent
Buffer of tasks to B1
Buffer of tasks to Bn
λ_1
λ_n
λ_i
λ_1
λ_n
Queue former of serviceable tasks
Database of resource Bi state
B_1
B_n
Execution manager of serviceable tasks
Database of resource Ki state
K^i_1
K_{mi}
Buffer of tasks to K^i_1
Buffer of tasks to Kmi
λ_{ki1}
λ_{kim}
Return of extra buffer, queue tasks

Fig.6 – Upper-level broker B_i model

The tasks entered in the center are divided into two queues: the queue of tasks for solving with the clusters connected to this control center, and the queue of tasks that may not be executed onto the resources of the clusters connected to this control center because of characteristics of resources and types of solvable problems. Both of the queues are formed similar and independently on the procedure *D* model basis.

Conclusion. So, the article presents a unified procedure of all the resources interoperation in a distributed system enabling the scheduling process of task execution in distributed computing systems to be initiated from unified positions.

References (transliterated)

1. Minukhin S. V. *Models and methods for solving planning problems in distributed computing systems.* Kharkiv, Shedraya usadba plus, 2014. 323 p.
2. Ponomarenko V. S., Listrovoy S. V., Minukhin S. V., Znahur S. V. *Methods and models of resource planning in GRID-systems.* Kharkiv, KhNUE, 2008. 407 p.

3. Listrovoy S. V., Minukhin S. V. Approach and model of resource allocation planning in Grid. *International Scientific and Technical Journal "Problems of Management and Informatics"*. 2012, no. 5, pp. 65–82.
4. Papadimitriu H., Staygits K. *Combinatorial optimization. Algorithms and complexity*. Moskow, Mir, 1985. 509 p.
5. Kesselman C. Foster, Tuecke S. The Anatomy of the Grid: Enabling Scalable Virtual Organizations. *International J. Supercomputer Applications*. 2001, vol. 15 (3),. Available at: http://www.globus.org/alliance/publications/papers/anatomy.pdf.
6. Listrovoy S.V., GUL A. Yu. «Method of Minimum Covering Problem Solution on the Basis of Rank Approach»// *Engineering Simulation*. 1999. vol.17. pp. 73–89.
7. Listrovoy S. V., Minukhin S. V. General approach to solving optimization problems in distributed computing systems and the theory of building intelligent systems. *International Scientific and Technical Journal "Problems of Management and Informatics"*. 2010, no. 2, pp. 65–82.
8. Listrovoy S. V., Minukhin S. V. General Approach to Solving Optimization Problems in Distributed Cjmputing Sysntems and Theory of Intelligence Systems Construction. *Journal of automation and information sciences*. 2010, vol. 42, no. 3, pp. 30–46
9. Listrovoy S. V., Golubnichiy D. Yu., Listrovaya E. S. Solution method on the basis of rank approach for integer linear problems with boolean variables. *Engineering Simulation*. 1999, vol.16, pp. 707–725.
10. Listrovoy S. V., Tretjak V. F., Listrovaya A. S. Parallel algorithms of calculation process optimization for the boolean programming problems. *Engineering Simulation*. 1999, vol.16, pp. 569–579.

Поступила (received) 14.02.2017

Бібліографічні описи / Библиографические описания / Bibliographic descriptions

Уніфікована процедура організації взаємодії ресурсів системи в розподілених обчислювальних середовищах / С. В. Лістровий, К. А. Трубчанінова, В. О. Бриксін, М. С. Курцев // Вісник НТУ «ХПІ». Серія: Стратегічне управління, управління портфелями, програмами та проектами. – Х. : НТУ «ХПІ», 2017. – № 3 (1225). – С. 101–107. – Бібліогр.: 10 назв. – ISSN 2311–4738.

Унифицированная процедура организации взаимодействия ресурсов системы в распределенных вычислительных средах / С. В. Листровой, К. А. Трубчанинова, В. А. Брыксин, М. С. Курцев // Вісник НТУ «ХПІ». Серія: Стратегічне управління, управління портфелями, програмами та проектами. – Харків : НТУ «ХПІ», 2017. – № 3 (1225). – С. 101–107. – Библиогр.: 10 назв. – ISSN 2311–4738.

A uniform procedure of a system resources interaction in distributed computer media / S. V. Listrovoy, K. A. Trubchaninova, V. O. Bryksin, M. S. Kurtsev // Bulletin of NTU "KhPI". Series: Strategic management, portfolio, program and project management. – Kharkiv : NTU "KhPI", 2017. – No 3 (1225). – P. 101–107. – Bibliogr.: 10. – ISSN 2311-4738.

Відомості про авторів / Сведения об авторах / About the Authors

Лістровий Сергій Володимирович – доктор технічних наук, професор, Український державний університет залізничного транспорту, професор кафедри спеціалізованих комп’ютерних систем; тел.: (050) 935–50–42; e–mail: om1@yandex.ru.

Листровой Сергей Владимирович – доктор технических наук, профессор, Украинский государственный университет железнодорожного транспорта, профессор кафедры специализированных компьютерных систем; тел.: (050) 935–50–42; e–mail: om1@yandex.ru.

Listrovoy Serhiy Volodymyrovych – Doctor of Technical Sciences, Professor, Ukrainian State University of Railway Transport, Full Professor at the Department of Specialized computer systems; tel.: (050) 935–50–42; e–mail: om1@yandex.ru.

Трубчанінова Карина Артурівна – кандидат технічних наук, доцент, Український державний університет залізничного транспорту, доцент кафедри транспортного зв’язку; тел.: (050) 637–43–26; e–mail: TKA2@ukr.net.

Трубчанинова Карина Артуровна – кандидат технических наук, доцент, Украинский государственный университет железнодорожного транспорта, доцент кафедры транспортной связи; тел.: (050) 637–43–26; e–mail: TKA2@ukr.net.

Trubchaninova Karyna Arturivna – Candidate of Technical Sciences (Ph. D.), Docent, Ukrainian State University of Railway Transport, Associate Professor at the Department of Transport communication; tel.: (050) 637–43–26; e–mail: TKA2@ukr.net.

Бриксін Володимир Олександрович – кандидат технічних наук, Український державний університет залізничного транспорту, старший викладач кафедри інформаційних технологій; тел.: (095) 591–22–79.

Брыксин Владимир Александрович – кандидат технических наук, Украинский государственный университет железнодорожного транспорта, старший преподаватель кафедры информационных технологий; тел.: (095) 591–22–79.

Bryksin Volodymy Oleksandrovych – Candidate of Technical Sciences (Ph. D.), Ukrainian State University of Railway Transport, Associate Professor at the Department of Information Technology; tel.: (095) 591–22–79.

Курцев Максим Сергійович – Український державний університет залізничного транспорту, аспірант кафедри спеціалізованих комп’ютерних систем; тел.: (050) 302–99–12; e–mail: kurtsev_m@ukr.net.

Курцев Максим Сергеевич – Украинский государственный университет железнодорожного транспорта, аспирант кафедры специализированных компьютерных систем; тел.: (050) 302–99–12; e–mail: kurtsev_m@ukr.net.

Kurtsev Maksym Serhiyovych – Ukrainian State University of Railway Transport, graduate student at the Department of Specialized computer systems; tel.: (050) 302–99–12; e–mail: kurtsev_m@ukr.net.