# Multiagent Distributed Grid Scheduler

Victor Korneev , Dmitry Semenov, Andrey Kiselev
Nii “Kvant”, Moscow, Russia
Email: {korv@rdi-kvant.ru, sdvbox@gmail.com, a_v_k@rambler.ru}

Boris Shabanov, Pavel Telegin
Joint Supercomputer Center RAS, Leninsky pr. 32a, Moscow, 119991, Russia
Email: {shabanov, telegin }@jscc.ru

***Abstract*—An approach for resource scheduling based on multiagent model with distributed queue is discussed. Algorithms of functioning agents for distributed Grid scheduling are presented**

## I. Introduction

GRID-technologies [1] provide program solutions for creating Grids as certain class of networks. The main requirements for Grid environment in Joint Supercomputer Center RAS (JSCC RAS) are: raising the capacity of aggregate resources by eliminating the situation when some resources are idle while other resources are overloaded, and providing computational resources exceeding capacity of individual systems for execution of large scale parallel programs which can be efficiently implemented on several computational systems (CS).

Each CS contains control computer (CC). Control computers of all CS are integrated by network, they can transfer programs and data and execute remote programs.

Basic modules (BM or nodes) of several CS can be integrated with one or more high-speed networks like Infiniband for data transfers while program execution [2]. For each system in Grid a local batch system operates on CC. Usage of batch system in every CC allows using a single pool of computational modules. Batch system allocates jobs on nodes, terminates, delivers results to users and provides access control.

To include CSs under different administration into Grid without changing software and functioning the middleware software is used which implements functions of management system (MS) of Grid environment. MS of Grid enables submitting user’s jobs to single queue, running jobs on one or several CS using middleware, monitoring Grid environment, providing fault tolerance and access control.

Management systems based on centralized resource sharing model are most studied and implemented. In this model user jobs are submitted to a single queue, which is shared by all processors of parallel system. When the processor is free, it takes a job from the queue, or it is made by a system process, tracking the processors status. This model is used in metadispatcher in GridWay project [5, 6]. However in Grid with multiple CS and significantly different bandwidth between and inside CS it is impossible to achieve in reasonable time the complete and accurate description of the current state of resources and jobs. So, in large GRIDs it is necessary to use distributed metaschedulers based on distributed queuing system model. This paper describes an approach to release Content Addressable Network [4] as multiagent system to resource scheduling based on distributed single queue model and implementation its algorithms of functioning distributed Grid scheduler in JSCC RAS [7, 8]

The paper has the following structure. In second section architecture of Grid management system is presented. The third section discusses a several heuristic algorithms of decentralized job scheduling. The fourth section contains results of experiments on efficiency of proposed decentralized management system of Grid.

## II. Distributed Multiagent Metascheduler

Submitted user’s jobs must be registered in one of queues of distributed queuing system. Each queue is served by its own agent –local scheduler which accepts one of three decisions for each job: job can be scheduled for execution on resources of one or several CS, can be left in queue for further scheduling or transferred to another queue. Developing a distributed multiagent metascheduler, it is necessary on one hand to enable independent simultaneous scheduling of jobs in different queues by local schedulers, on the other hand usage of Grid resources must be coordinated.

Approach in this article suggests a resolution of this contradiction by allocation of Grid resource domain for each queue for independent scheduling by agent – local domain scheduler. If you allocate resources in separate domains of each CS in Grid and create one extra domain which includes resources of all CSs, then a hierarchy of queues is formed. In this hierarchy it is possible to coordinate the allocation of resources between jobs within the following algorithm of the agent with queues: the transfer of jobs between the queues of the lower level of the hierarchy is possible only through the upper level queue, which is used only for scheduling between lower level queues.

Each dedicated domain is managed by Grid CS component that is CS manager. Manager contains data structure necessary for local scheduler:

— information system (IS) containing resources table of managed domain and description of general Grid environment state;

— queue of jobs to be scheduled.

The basic functional processes of the manager are:

— its own local scheduler, which makes decisions on allocation of jobs on resources or transfer jobs to another queue based on IS data and queue state;

— process supporting current state of IS data to be coherent with IS of other schedulers;

— service processes which provide fault tolerance hierarchy of the managers and information security (protection against unauthorized access to resources).

CS manager, local scheduler of which makes decisions on allocation of jobs to CS resources we will call the manager M1 or 1-st level manager. Manager, scheduler of which allocates jobs between local schedulers, will be called M2 or the 2-nd level manager. M1 managers transfer jobs to batch system queue or to M2 manager. CC of each CS always executes a M1 manager. Number of M2 managers can be one or more depending on required reliability and throughput of management system of Grid. MS managers can run on CS control computers or on additional dedicated computers. Information links between managers form an acyclic graph. Managers are interacting using IP-addresses and port numbers.

Different algorithms can be used in local schedulers and MS managers: from solving optimization problems to heuristic algorithms, that allows taking into account specific heterogeneity of the Grid components.

A protocol for parallel resource allocation by hierarchy structure managers is suggested in [9]. For this protocol it is proved that there are no deadlocks caused by interlocking because of partial simultaneous allocation resources by different managers, and inability to continue jobs due to lack of resources for a job without releasing of resources by another job.

Hierarchy organization of CS managers in Grid allows:

— Ensure absence of deadlocks during distributed execution of scheduling algorithms and resources allocation;

— Take into account the specifics of managing heterogeneous objects, combining similar objects (CSs, domains) under control of single manager. Combining CSs to domain can be done using different similarity criteria: architecture, hardware and software platform, administration policy, geographical location, ownership of organization, etc.;

— Control MS managers in the same domain by single organization providing their support, and use scheduling algorithms common for given domain;

— Reduce number and variety of control object types for each manager, this simplifies formulation and implementation of management decision and reduces the uncertainty of complex multiprogramming case, determining and fixing the number of parameters for the higher level.

## III. Algorithms for Distributed Scheduling.

Let us consider that job can be executed on any CS from Grid and development of control solutions in the local manager uses two job parameters: required number of computational nodes and required time. In practice more parameters are used [8], but these two are sufficient for understanding the idea.

The following characteristics of jobs and batch systems are used for description of computational resources:

— Area of user job, it equals to product of requested number of nodes and requested time;

— Summary area of jobs on certain CS, it equals to sum of areas of jobs, which are queued or executed on this CS;

— Load, it equals to the ratio of the summary area of jobs on CS to the total number of computational nodes, which can execute user's programs. This characteristic describes mean time that nodes will be busy executing jobs;

— Upper load bound of CS, it limits CS load.

Current difference between bound and load is total area of jobs, which can be submitted to batch system on given CS.

M1 managers submit user jobs to batch system queue without exceeding upper load bound. It should be noted that in some cases, the batch system imposes a restriction on the maximum time user jobs. So, area of jobs which can be submitted is limited either by difference between bound and current load or by value equal to product of requested number of nodes to maximum allowed execution time for given batch system. Local scheduler takes into account this feature and will not schedule job with area exceeding this limit. Changing upper load bounds one can redistribute jobs between batch systems queues and CS managers queues.

As it is shown in [9], with appropriate objective function like deviation of load values of computational node from mean load value in local neighborhood, it is possible to minimize of the objective function through the development of local management decisions for CS load balancing.

Scheduling strategy is based on principle of making suboptimal decision in a coherent interaction between MS managers. Managers, which are distributed over a network, make local decisions, forming parts of global decision. Decision on resource allocation for user job is made only by MS manager controlling the given resource as it has most accurate information about allocated resource, that allows to make decision on the basis of actual data.

Let us explain using Fig. 1 how system of Grid managers functions. One can see Grid consisting of 7 CSs, each of CC executes manager M1i, i =1, . . . , 7 . Each M1i manager stores in its resource table a row contains value received from the batch system of corresponding CSi is recorded.

M2 managers have a set of enumerated logical channels ports, which link them with M1 and M2 managers. For each port M2 manager stores a raw in resource table with values of areas of jobs which can be submitted to batch systems, and their M1 managers can be reached by acyclic graph of logical channels from the given M2 manager. E.g., for M24 manager resource table raw for port 1 contains information about CS3 and CS4, in raw for port 2 information about CS1 and CS2, in raw for port 3 information about CS5, CS6 and CS7.

Termination of jobs, failure of CS and communication channels, recovery of CS and communication channels, connection of new CS to Grid result corresponding changes in resource tables. In general, each M2 manager contains complete information about Grid resources, but different managers have their own tables.

Jobs queue of MS manager uses FIFO strategy. Each time when resource table is changed or after a specified time interval manager attempts to schedule jobs from its queue. First of all list of CS in resource table is analyzed. In accordance with the applicable scheduling algorithm a CS with sufficient number of nodes for job execution is selected. If there are no appropriate CS found in resource table of MS manager, search is made by records corresponding to adjacent MS managers.

When requested resources found in system job is transferred to corresponding adjacent MS manager. When there are no resources available job goes to the end of MS manager queue, and manager allocates the next job. The set of all MS managers' queues forms a single global queue for aggregated resources of the Grid.

It is obvious that the transfer of jobs between managers may result infinite residence in queues of managers. To prevent this, a label is assigned to job, which allows or prohibits job rescheduling. If rescheduling is prohibited then local scheduler transfers job to manager which resources are allocated for job execution. Of course, this transfer is possible only if there is place in that M1 manager, otherwise job waits for possibility of transfer. Setting label prohibiting rescheduling may be result of excess of limit of reschedulings, or expiration of time interval of job stay in Grid.

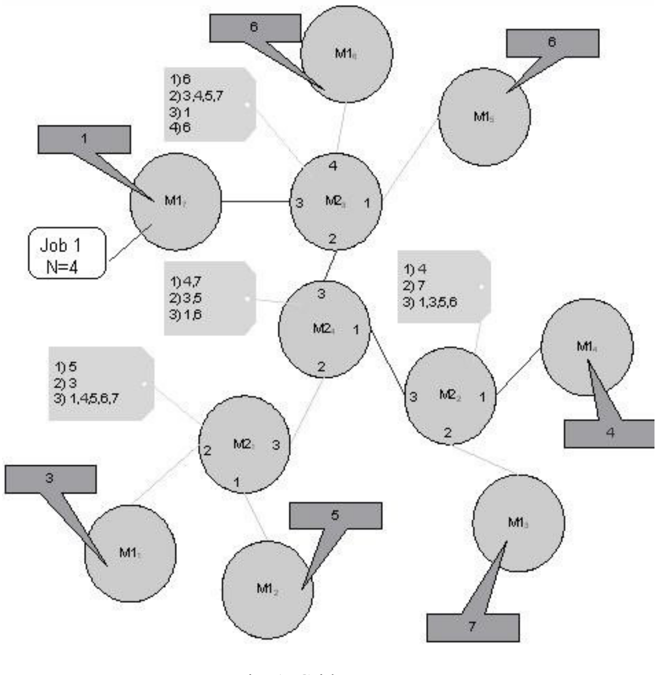

Fig. 1. Grid managers

The same label is used for scheduling parallel job on nodes of different CS. Parts of one job allocated on different CS are represented as separate labeled jobs, this guarantees reception of these jobs in the assigned CSs.

In some studies, particularly [10], of scheduling jobs in the tree-shaped queue algorithms leveling the number of jobs in queues are used. However, it seems that algorithms, taking into account the load of Grid resources, would be more appropriate.

The article investigates heuristic algorithms based on principles of minimal load and minimal sufficiency [3].

According to the minimal load principle a job is sent to the CS with least difference of upper load bound and load of CS, i.e. job is allocated to less busy system or in case of equal load to the system with minimum required number of units of a free resource. This algorithm provides dynamic leveling of computational load for CSs, so we call it balancing algorithm.

Alternative is usage of minimal sufficiency principle: job is scheduled to CS with sufficient for an immediate start job number of units of a free resource. Two modifications of minimal sufficiency algorithm were investigated. In the first modification jobs were assigned to Grid recourses in turn, in the second modification jobs were assigned when possible without regard to priority: as soon as required number of computational nodes became free in cluster CS, the first job able to be executed there was job was extracted from managers' queue. This allows to better load of CS, but order of jobs is violated.

Let us explain functioning of MS managers using example of scheduling on Grid in Fig. 1. In these examples allocation strategy with minimal sufficiency is shown.

Example 1. Job running on CS7 requires 4 resource units. Job will be transferred from M17 to M23 and later according to resource table to M24, then to M22 and finally to M14. M14 manager allocates job at CS4 which it controls. Job allocation on CS4 will change resource tables of managers M17, M23, M24 and M14 (number 4 will be excluded from all resource tables). Due to the fact that level 2 managers have aggregate information, decision on resource allocation can be taken only by level 1 manager or adjacent level 2 manager.

Example 2. Job requiring 8 resource units, running on any of the M1 managers will be suspended by the adjacent M2 until there is sufficient number of free nodes.

In managers' resource tables predicted values of free Grid resources for a given scale of time in the future can be formed. Managers can make decisions basing on these predicted values.

## IV. Investigation of Efficiency of Distributed Scheduling Algorithms.

A set of experiments was performed: once formed test jobs flow was fed to the Grid containing of two CS called next and neo, and operating under developed Grid environment [8].

Test flow consists of 2 parts, 25 jobs each and represents typical jobs flow for MVS-1000 system. In a separate series of experiments test the flow of jobs ran independently on each CS in the Grid.

In further experiments these flows were fed to the Grid management system simultaneously. In Table results of experiments are presented. In experiments A1-A5 balancing algorithm with different values of bound parameter.

In experiments B1 and B2 scheduling algorithm based on minimal sufficiency is used. In experiment B1 data jobs were allocated in turn, in B2 jobs were allocated whenever possible, without regard to priority.

TABLE 1.
EXPERIMENTAL RESULTS

| | CS | Number of jobs | Wall time |
|---|---|---|---|
| independent | next<br>neo | 25<br>25 | 02:43:54<br>05:3:58 |
| A1, no limit | next<br>neo | 42<br>8 | 03:47:35<br>03:36:18 |
| A2, bound = 80 | next<br>neo | 41<br>9 | 03:21:11<br>03: 18:29 |
| A3, bound = 70 | next<br>neo | 26<br>24 | 03:31:46<br>03: 04:30 |
| A3, bound = 60 | next<br>neo | 32<br>18 | 04:58:01<br>02:32:39 |
| A3, bound = 50 | next<br>neo | 29<br>21 | 03:14:21<br>04:23:49 |
| B1 in turn | next<br>neo | 33<br>17 | 03:38:32<br>03:54:27 |
| B2, no priority | next<br>neo | 28<br>22 | 02:56:17<br>03:06:31 |

In experiments A1 with infinite upper load bound jobs were allocated to resources without staying in managers' queues. It is needed to note that if jobs flow is distributed to two identical systems the way that there is the same load level at initial moment then times of execution of parts of jobs flow will vary because of differences of real execution times. On the "next" CS part of jobs flow was executed faster and CS was idle while the other CS was executing the rest of its jobs. This can be seen in results of experiments with infinite bound. However, it should be noted that due to more rational distribution of jobs (CS with more resources received greater part of flow) it was possible to reduce maximum of jobs processing times compared to execution of the flows on independent CSs.

In Table 1 one can see trend deterioration in quality of scheduling jobs flow with bound less than 80.With big bound values results of experiments tend to results with infinite bound. When bounds are lower it happens that jobs with small number of requested nodes and high requested time contribute significantly to the CS workload/ thus increasing load bound, while there are free nodes in CS. In cases like this idle time of nodes is high resulting summary time of executions.

It should also be noted that for all experiments MS [8] demonstrated stable operation both in normal mode and in high load mode: all jobs were scheduled and executed.

## REFERENCES

[1] Foster I., Kesselman C., Tsudik G., Tuecke S. A security architecture for computational grids // Proc. 5th ACM Conference on Computer and Communications Security. San Francisco: ACM Press, 1998. 83–92.

[2] Корнеев В. В. Вычислительные системы. М.: Гелиос АРВ, 2004.

[3] Корнеев В. В, Киселев А. В., Семенов Д. В., Сахаров И .Е. Управление метакомпьютерными системами // Открытые системы. 2005. № 2. 11–16.

[4] Lee J., Keleher P., and Sussman A. "Decentralized resource management for multi-core desktop grids," in 24th *IEEE International Parallel & Distributed Processing Symposium,* Atlanta, Georgia, USA, 2010.

[5] Богданов С. А., Коваленко В. Н., Хухлаев Е. В., Шорин О. Н. Метадиспетчер: реализация средствами метакомпьютерной системы Globus. Препринт ИПМ № 30. Москва, 2001.

[6] GridWay Metascheduler: Metascheduling Technologies for the Grid. URL: http://gridway.org.

[7] Савин Г. И., Корнеев В. В., Шабанов Б. М., Телегин П. Н., Семенов Д. В., Киселев А. В., Кузнецов А. В., Вдовикин О. И., Аладышев О. С., Овсянников А. П. Создание распределенной инфраструктуры для суперкомпьютерных приложений// Программные продукты и системы. 2008. № 2. 2–7.

[8] Руководство программиста грид (http://www.jscc.ru/informat/grid1.zip).

[9] Корнеев В. В. Архитектура вычислительных систем с программируемой структурой. Новосибирск: Наука, 1985. (http://andrei.klimov.net/reading/1985.Korneev.- .Arkhitektura.vychislitel'nykh.sistem.s.programmiruemoi.strukturoi.zip)

[10] Houle M., Symvonis A., Wood D. Dimension-exchange algorithms for token distribution on tree-connected architectures // J. of Parallel and Distributed Computing. 2004. № 64. 591–605.