Efficiency Analysis of Decentralized Grid Scheduling with Job Migration and Replication

Mikhail Kurnosov
A.V. Rzhanov Institute of Semiconductor Physics SB RAS A.V. Rzhanov Institute of Semiconductor Physics SB RAS
13 Lavrentyev ave, Novosibirsk, Russia
+7 383 330 56 26
mkurnosov@isp.nsc.ru

Alexey Paznikov
13 Lavrentyev ave, Novosibirsk, Russia
+7 383 330 56 26
apaznikov@isp.nsc.ru

ABSTRACT
In this paper, we consider the resource management problem in geographically-distributed computer systems. Decentralized scheduling algorithms are proposed: locally optimal algorithm (LOS), algorithm with the job replication (RS), algorithm with the job migration (MS) and algorithm based on the combined approach (MRS). The algorithms were investigated on the multicenter computer system. We also compare efficiency of decentralized scheduler GBroker with centralized scheduler GridWay.

Categories and Subject Descriptors
C1.4 [Processor Architectures]: Parallel Architectures – Distributed architectures; C2.4 [Computer-Communication Networks]: Distributed Systems – Grid computing.

General Terms
Algorithms, Management, Performance, Experimentation.

Keywords
Grid scheduling, geographically-distributed computer systems, decentralized scheduling.

1. INTRODUCTION
Today at the solving large problems in the realm of science and technique geographically-distributed computer systems (CS) are widely used. These CS are represented as large collections of distributed computational units (subsystems), interacting via local and global area networks (including Internet) [1]. Multiclusters and Grid systems belong to the geographically-distributed CS.

Grid scheduling system is an essential part of the Grid and currently is an active research area. Scheduling process generally includes resource discovering, filtering, scheduling according to the scheduling policies and job submission to the certain subsystems. During these stages we have to consider the structure and workload variable nature.

In geographically-distributed computer systems each of subsystems is running under the control of the resource management system (RMS: TORQUE, Altair PBS Pro, SLURM, etc), which supports jobs queues and allocates the resources (processor cores) for them.

Centralized approach for the scheduling in geographically-distributed CS involves the functioning in the system of the centralized scheduler maintaining the global job queue. The failure of that scheduler will cause an unavailability of the entire system. Besides that, in the case of exploitation these tools in large-scale CS, time costs for resources searching are increasing severely.

The major centralized schedulers of parallel programs in geographically-distributed CS are GridWay [2], AppLeS [3], GrADS [4], Nimrod/G [5], Condor-G [6]. GridWay is the most widespread scheduler in Grid systems. The object in it is minimizing of jobs service time. GridWay supports migration between subsystems. AppLeS realizes scheduling at the application level and so requires special user’s knowledge about the system configuration. This fact reduces the universality of the scheduler. GrADS as like as GridWay supports job migration and realizes scheduling of Grid workflows. Nimrod/G through the economical models provides the trade-off between resource providers (subsystems) and resource consumers (users). Condor-G can schedule jobs which may be presented as directed acyclic graphs (DAG).

In the case of decentralized scheduling there is collective of schedulers functioning in the distributed CS. These schedulers support distributed jobs queue and jointly take a decision about resource allocation. This allows achieving CS robustness, i.e. the ability to continue functioning at some subsystems failures.

For some classes of distributed CS there were created effective decentralized scheduling algorithms. However the applicability of these methods is limited for Grid systems. The algorithms don’t consider bandwidth of the channels between resources and queues on the subsystems.

The paper suggests three new scheduling algorithms, based on job replication (RS), job migration (MS) and combined approach (MRS). The approaches realized in these algorithms allow better consider workload dynamicity and network performance (as compared with locally optimal scheduling (LOS)). Also the influence of schedulers’ local neighborhoods structures on the scheduling efficiency is studied.
2. DECENTRALIZED SCHEDULING OF PARALLEL JOBS IN GEOGRAPHICALLY-DISTRIBUTED CS

Let there be a Grid system, consisting of H subsystems; N is the total number of elementary computers (EC) in the system. The EC is the computer resource unit, destined for the parallel process execution (e.g. CPU core). Denote \( n_i \) the number of EC of members of the subsystem \( i \in S = \{ 1, 2, ..., H \} \); \( c_i \) is the number of non-used EC on the subsystem \( i \); \( q_i \) is the jobs number in the subsystem’s i queue; \( t_{ij} = t(i, j, m) \) is the time of message with size \( m \) pass between subsystems \( i, j \in S ((i, j, m) = s) \). We consider that all subsystems are interconnected by communication network.

There are a local RMS and a decentralized scheduler on each subsystem. Collective of schedulers is represented in the form of a directed graph \( G(S, E) \), in which vertices correspond to schedulers and edges are the logical connections between them (Figure 1).

The being of the edge \((i, j) \in E\) means that scheduler \( i \) may send jobs to the scheduler \( j \). A set of vertices \( j \) adjacent to the vertex \( i \) forms its local neighborhood \( L(i) = \{j \in S | (i, j) \in E\} \).

Figure 1. Example of schedulers’ local neighborhoods:
\( H = 4, L(1) = \{2, 3\}, L(2) = \{1, 4\}, L(3) = \{1, 4\}, L(4) = \{2, 3\} \)

User submits the task to the scheduler \( i \). The task contains the parallel program, input files and resource request, which contains the rank \( r \) of the program (number of parallel processes), sizes \( z_1, z_2, ..., z_k \) of stage-in files (\( z = \text{byte} \)) and numbers \( h_1, h_2, ..., h_k \) of subsystems on which the corresponding files are located (\( h \in S \)). Scheduler (accordingly to the algorithm implemented in it) searches the (sub)optimal subsystem \( j^* \in L(i) \cup \{i\} \) (or subsystems \( j^*_1, j^*_2, ..., j^*_m \) ) from its local neighborhood.

3. DECENTRALIZED ALGORITHMS OF JOB SCHEDULING

Four decentralized algorithms of parallel jobs scheduling in geographically-distributed CS are proposed. Each algorithm describes functioning of the scheduler \( i \) on submitting the job to its queue. At the initial stage, all the algorithms assume the request of the scheduler \( i \) to the monitoring service and obtaining from it current values of parameters \( t_{ij}, c_p, s_p, q_j \in n_j \) \( (j \in L(i) \cup \{i\}) \). After that the set \( S(i) = \{j| n_j > r, j \in L(i) \cup \{i\} \} \) of feasible subsystems is built up. These systems have count of EC not less than required.

3.1 Locally optimal scheduling algorithm (LOS)

The algorithm selects locally optimal subsystem.

1. The subsystem \( j^* \) with the minimal value \( F(j), j \in S(i) \) is selected from the local neighborhood \( S(i) \) of scheduler \( i \).

\[
F(j) = \frac{t_j + \frac{c_{\max}}{w_j}}{t_{\max}} + \frac{c_j}{w_{\max}}, \quad \text{if } c_j < r \text{ or } q_j > 0,
\]

\[
F(j) = \frac{t_j}{t_{\max}}, \quad \text{else.}
\]

Here \( t_j = \sum_{l=1}^{z} (h_l, j, z_l) \) is the time of delivery job files to the subsystem \( j; \ t_{\max} = \max_{j \in S(i)} t_j; \ c_{\max} = \max_{j \in S(i)} c_j; \ w_j = q_j / n_j \) is the number of jobs in queue, corresponding to one EC of subsystem \( j; \ w_{\max} = \max_{j \in S(i)} w_j \). Function \( F(j) \) takes into account time of staging-in and subsystems’ workload.

2. Job is submitted to the local RMS’s queue of subsystem \( j^* \) and then the staging the job’s files to that subsystem is performed.

Ranking of subsystems by object function \( F(j) \) allows to consider the time of staging-in and relative workload of subsystems. But the variation of the workload may cause the job service time extension. The lack is resources are allocated statically for the whole job’s lifetime. Therefore workload increasing or performance drop may cause the job service time increasing.

3.2 Scheduling based on job replication (RS)

This algorithm bases on submitting the job to the multiple subsystems. The replication scheme allows taking into consideration the variable nature of computational resources.

1. \( m \) subsystems \( j^*_1, j^*_2, ..., j^*_m \) are chosen from the local neighborhood \( S(i) \) in nondecreasing order of function \( F(j) \).

2. The job is submitted simultaneously to the queues of local RMSs of subsystems \( j^*_1, j^*_2, ..., j^*_m \), after that the staging is performed to these subsystems.

3. With the interval \( \Delta_t \), scheduler \( i \) checks the job’s state on subsystems \( j^*_1, j^*_2, ..., j^*_m \) and determines the subsystem \( j^* \) on which the job was executed before other subsystems.

4. The job is removed from the local RMSs of subsystems, distinct from the \( j^* \).

From our point of view, RS may be useful for the high priority (urgent) jobs, because it well consider the difference in queues workload on the subsystems. But in common cases we expect the
higher load of network and file system, which may become the bottlenecks. The increase of stage-in time will affect the job service delay.

3.3 Scheduling based on job migration (MS)
Periodic resource search for jobs in scheduler queue is realized in this algorithm.
1. From the local neighborhood S(i) of scheduler i, the subsystem j* with minimal value F(j), j ∈ S(i) is chosen.
2. The job is submitted to the local RMS queue of subsystem j* and then the files are delivered to that subsystem.
3. The scheduler i with interval Δi performs the procedure LOS of searching subsystem j' for the job in scheduler’s j* queue.
4. If one of the founded subsystems satisfies the condition F(j*) – F(j') > ε then the job migration to the queue of subsystem j* is performed (the job is removed from the scheduler’s j* queue).

The main feature of MS that it combines the advantages of locally optimal scheduling and replication scheduling. It takes into account the changes of workload and at the same time doesn’t burden the network or file system. But there is no way to predict when the job will be executed on the subsystems so unwanted migrations are possible.

3.4 Scheduling based on both job migration and job replication (MRS)
The algorithm is based on combination of two approaches. The job is replicated to the several subsystems and can migrate between the queues. It realizes the trade-off between the advantages and shortcomings of replication and migration approaches.

It’s important to note that computational complexity of searching of subsystems with proposed algorithms doesn’t depends on number H of subsystems, because the searching performs only within the schedulers’ local neighborhoods. This guarantees the applicability of the created algorithms in large-scale geographically-distributed CS.

4. SOFTWARE PACKAGE GBROKER OF DECENTRALIZED SCHEDULING
All proposed algorithms are implemented in the software package GBroker [7] of decentralized scheduling of parallel jobs in geographically-distributed CS (Figure 2). The package was developed in the Center of Parallel Computational Technologies of Siberian State University of Telecommunication and Information Sciences (CPCT SibSUTIS) with the Computer Systems Laboratory of A.V. Rzhano Institute of Semiconductor Physics Siberian Branch of Russian Academy of Sciences (ISP SB RAS). It includes the scheduler GBroker, interface module GCClient and monitoring systems NetMon and DCSMon. Module GBroker realizes algorithms of decentralized scheduling, interacting with the local RMS by means of system GRAM of Globus Toolkit. DCSMon is responsible for the information about computational resources of subsystems in the scheduler’s local neighborhood. NetMon provides the information about the performance of network channels between subsystems.
Rank $r$ of each job was selected from the set $\{1, 2, 4, 8\}$ randomly with uniform distribution.

Denote $t_k$ is a time of the job $k \in \{1, 2, ..., M\}$ submit to the scheduler, $t'_k$ — start time of the job $k$ execution, $t''_k$ — completion time of the job $k$ execution. Let $\tau$ is the total service time of the flow of $M$ jobs.

For the efficiency of scheduling algorithms evaluation we used the following indicators: system bandwidth $B$, mean job service time $T$ and mean time $W$ of job waiting in the queue.

$$B = \frac{M}{\tau}, \quad T = \frac{1}{M} \sum_{k=1}^{M} (t'_k - t_k), \quad W = \frac{1}{M} \sum_{k=1}^{M} (t''_k - t_k)$$

### 5.1 Experimental analysis of the algorithms

Figures 4 and 5 depict the results of comparison of algorithms LOS, RS, MS, MRS efficiency at service of job flow of $M = 200$ jobs. Job flow arrived to the subsystem Xeon80. Local neighborhoods of schedulers had a structure of complete graph.

The system bandwidth for the algorithm RS for $m \in \{2, 3\}$ is more than bandwidth for the algorithm LOS. Performance degradation for large values of $m$ is connected with the increasing loading of communication channels at large input files staging-in to several subsystems. Network and file system are the bottlenecks in this case and may cause severe performance degradation.

LOS is not so ineffective in case of large jobs flows, which almost uniformly load the subsystems. But when we have the high difference in workload it may be quite inappropriate. In that case RS is much better.

In the algorithms MS and MRS searching subsystem interval was $\Delta = 30$ s, migration criterion was $\varepsilon = 0.2$. The least mean service time of jobs and mean waiting time in queue were achieved at using algorithms LOS and MS (see Figure 5). Larger system bandwidth was obtained for the MS. This is due to the fact that migration provides workload balancing in case of non-uniform distribution of jobs.

Algorithms RS and MRS are recommended for using only in case of low intensity of the job flows or small size of the input data. Also it could be used for the urgent jobs with the highest priority. In the case of high intensity of job flow, the time of input data delivery increases significantly due to the increasing of network channels and network file systems of segments loading. It leads to the bandwidth decreasing and job service time (waiting time) increasing for all the scheduling algorithms. This effect increases dramatically with the flux level. For MRS all the benefit from replication is reduced because of the negative effect from the bottlenecks.

---

**Figure 4. Efficiency comparison of algorithms LOS and RS**
5.2 Choice of structure of scheduler’ local neighbors

The investigation of local neighborhoods structures influence to the scheduling efficiency was conducted. In the experiments the flows of $M = 50$ jobs were submitted simultaneously to all the schedulers. Local neighborhoods in form of complete graphs and the perspective structures such as rings, meshes, stars, 2D torus and $D_2$-graphs are considered [1].

Figure 6 depicts the influence of local neighborhoods structure to the service efficiency of job flows by algorithm MS. High values of bandwidth besides the complete graph were obtained for the configuration on the basis of 2D torus, mesh and $D_2$-graph. For the latest two structures were obtained the highest values. Using of non-complete structures at local neighborhood configuration doesn’t leads to significant decrease of scheduling efficiency. These structures may be formed in the absence of direct links of communication between some subsystems, for example in case of some subsystems failures. $ND$ torus and $D_2$-graphs may be recommended as the structures of local neighborhoods.

On the whole the structures with small diameter (average diameter) may be used for the structures of local neighborhoods of decentralized schedulers. Also the vertices in the graphs must have multiple adjacent vertices which form its local neighborhoods.

The main shortcoming of the known perspective structures is that they are uniform. But it’s known that jobs flows to different subsystems differ from each other, hence the workload and migration rates may be also different. All these considerations tend to suggest that non-uniform structures may be more effective and algorithms of searching these structures are required.

5.3 Comparative analysis of GBroker and GridWay

We compared the efficiency of job flows service by centralized scheduler GridWay and developed decentralized scheduler GBroker.

Two experiments were conducted for the scheduler GBroker. During the first one the flow of $M = 300$ jobs was submitted to the subsystem Xeon80. In the second experiment distributed job service was modeled: equal flows of $M = 50$ jobs were simultaneously submitted to the all 6 schedulers’ queues. 2D torus and complete graph were used as the structures of logical connections of schedulers. MS was used as the scheduling algorithm. Package GridWay was installed on the segment Xeon80 and configured in accordance with the recommendations of the developers.

The Figure 7 shows that the bandwidth of scheduler GBroker at several job flows service exceeds the bandwidth of scheduler GridWay. Mean service time and mean waiting time are close to GridWay and insignificantly increase in case of centralized service.

Mean service and waiting times grow almost linearly with the flux level. This is due to the growth of the queues on the subsystems. In contrast, bandwidth quickly comes to the saturation state and remains almost unchanged. We consider this is due to the lack of the downtime connected with vacant resources.

We can conclude that decentralized scheduling is no less effective than centralized one and moreover provides robustness of large-scale computer systems. When using decentralized scheduling it’s preferable to submit jobs to all the schedulers functioning on the subsystems.
6. CONCLUSION

As compared with centralized approach proposed algorithms of decentralized scheduling significantly decrease the complexity of the search of resources and provide robustness of geographically-distributed CS. In comparison with centralized scheduler
GridWay higher throughput is achieved (with the using of algorithm MS) with comparable values of mean job service time. Migration scheme realized in algorithms MS, MRS can be effectively used for system bandwidth increasing (as compared with LOS). At job replication to the several subsystems (algorithms RS, MRS) the overheads for the files staging are increase. This fact leads to the job service time increase.

Developed package GBroker is free. For the decentralized scheduling organization in geographically-distributed CS it’s enough to install the package GBroker on each subsystem and configure the schedulers in accordance with the acquired recommendations. In the configuration of local neighborhoods of schedulers the structures with small diameter are recommended.

Our currently research area is the algorithms and software tools of effective scheduling of Grid workflows. Also we will implement the forecasting methods in network and subsystem monitoring. Furthermore we will focus on developing algorithms of generation of optimal local neighborhoods structures of decentralized schedulers.

7. REFERENCES