Supercomputer Efficiency: Complex Approach Inspired by Lomonosov-2 History Evaluation

Sergei Leonenkov[1,2], Sergey Zhumatiy[1]
1 Research Computing Center, Lomonosov Moscow State University, Russia
2 Faculty of Computational Mathematics and Cybernetics, Lomonosov Moscow State University, Russia
{leonenkov, serg}@parallel.ru

Abstract. These days the number of supercomputer users and the jobs they execute is rapidly growing, especially for supercomputers, providing computing time to external users. Supercomputers and their computing time are highly expensive, so their efficiency is crucial for both users and owners. There are several ways to increase operational efficiency, however, in most cases it involves a trade-off between efficiency metrics. This brings about a need to define "efficiency" in each specific case. We use the historical data from two largest Russian supercomputers to create a number of metrics in order to provide the definition of resource management "efficiency". The data from both Lomonosov and Lomonosov-2 supercomputers consists of over one year history of job executions. Lomonosov and Lomonosov-2 efficiency in terms of CPU hours utilization is considerably high, nevertheless, our global goal is to offer the way to maintain or improve this metric when maximizing others examined in the paper.

Keywords: High-Performance Computing; Resource Management; Supercomputer Job Scheduling Efficiency.

1 Introduction

The average performance of TOP500 List supercomputers is growing every year, along with the number of users and their tasks, which are growing even faster. For instance, the same situation occurs in the Research Computing Center of Moscow State University (RCC MSU), where two largest CIS systems are installed: Lomonosov and Lomonosov-2.

Lomonosov supercomputer, which shows theoretical peak performance of 1.7 Pflop/s and Linpack performance of 0.901 Pflop/s, increases the base of its users every year, since 2014 the number of users has grown by many times [3,4]. For the last year it maintained an average level of CPU hours utilization at 92.3 percent. In turn, Lomonosov-2 supercomputer with theoretical peak performance of 4.946 Pflop/s and Linpack performance of 2.478 Pflop/s does not have such a longstanding history of use, but the available usage history data shows that during last year over 1000 users

launched over 200000 tasks, while utilization of CPU hours was 88.7 percent [8]. Thus, task of increasing CPU hours utilization is very important for RCC MSU.

To secure RCC MSU clusters efficiency we have researched Lomonosov and Lomonosov-2 historical data of using Simple Linux Utility for Resource Management (SLURM) to analyze current status [1]. Within the framework of this research, we were able to develop our definition of supercomputer efficiency based on limitations of RCC MSU and selected list of metrics.

The paper organized as follows. Section 2 is devoted to Lomonosov and Lomonosov-2 supercomputers SLURM statistics review. Section 3 describes main trends of RCC MSU supercomputer usage. Section 4 provides an approach to define supercomputer resource management efficiency and represents list of efficiency metrics used. Our conclusions and future plans are drawn in the final section.

2 Background

Lomonosov and Lomonosov-2 supercomputers are two core high performance computing systems of Moscow State University. Both provide access for various scientific research groups whose number has already exceeded 900. As a rule 1000 jobs are processed every day and requested by 50-70 different users in general. There are approximately 3000 active accounts and this number is still growing every year. To cope with the constantly growing workload a SLURM (Simple Linux Utility for Resource Management) system and its implementation of Backfill scheduling algorithm are used.

2.1 SLURM

SLURM is a highly scalable, fault-tolerant cluster resource manager and job scheduler for big computational systems.SLURM cluster manager is used on many supercomputers specified in Top500 rating, including half of supercomputers stated in the top 10 (as per November 2013). The system is available under GNU GPL V2 license and well-documented. [1].

SLURM is based on the hierarchical model of supercomputer management systems. SLURM is designed for heterogeneous clusters with up to 10 million processors possible. It is successfully used on a supercomputer with more than 98000 nodes. Those who use a supercomputer managed via SLURM can set up to 1000 jobs for execution per second. Manager can perform up to 500 jobs per second (depending on the system configuration and equipment).

System administrator can set logical configuration of computing system, which would be supported by SLURM, flexibly and vary a set of cluster parameters easily by changing a configuration file or using appropriate commands, also external schedulers are allowed.

2.2 Lomonosov and Lomonosov-2 supercomputers

Lomonosov supercomputer consists of 12346 CPUs (mainly Intel Xeon X5570 2.93GHz), but even this amount is not enough to provide all of MSU research groups with necessary computation power. All CPUs are physically separated into groups of 4 or 6, called nodes. A selection of nodes may be combined into a logical group, which is called partition and includes a queue of incoming jobs. Partitions can be limited by, for example, indicating users who can use them, size of a job or processing time limits. Each partition focuses on the specific needs of users. Current configuration of the Lomonosov supercomputer includes 8 separate sections, each consisting of 1 to 4096 nodes [3-4].

We have researched Lomonosov supercomputer usage statistics from March 2014 to March 2017. During this time, users submitted overall 828389 jobs. Column 2 of Table 1 shows jobs state distribution gathered by SLURM over considered period. It should be mentioned that statuses cancelled, completed, failed and timeout do not imply CPU hours losses in most cases, it is mainly depends on program architecture. Node_fail job status represents that job terminated due to failure of one or more allocated nodes.

Table 1. Lomonosov supercomputer jobs statistics over researched period.

Status	Number of jobs (Lom)	Number of jobs (Lom-2)
Cancelled jobs	93758	37582
Completed jobs	384990	125648
Failed jobs	267550	33847
Node_fail jobs:	4423	2000
Timeout jobs:	77668	6201

Lomonosov-2 is top Russian supercomputer built by T-Platforms. Lomonosov-2 is an Intel Xeon/FDR InfiniBand cluster, accelerated with NVIDIA Tesla K40s and Tesla P100 GPUs [8]. Lomonosov-2 supercomputer consist of 1696 CPUs (Intel Haswell-EP E5-2697v3, 2.6 GHz, 14 cores and Intel Xeon Xeon Gold 6126 2.60GHz, 12 cores). Current configuration of the Lomonosov-2 supercomputer includes 4 separate partitions: “compute”, “low_io”, “tesla” and “test”, which consists of 1120, 384, 160 and 32 nodes respectively [7]. We have researched Lomonosov-2 supercomputer usage statistics from August 2016 to August 2017. During this time, users submitted overall 205278 jobs. Column 3 of Table 1 shows jobs state distribution gathered by SLURM over considered period. Next section provides more detailed analysis of Lomonosov and Lomonosov-2 supercomputers usage history and examines different efficiency metrics.

3 Usage history analysis

Data was gathered using Octotron [9] and SLURM sacct plugin. We developed SLURM usage history analysis and visualization tool. All metrics computations and figures in this section was made in this tool.

Figure 1 shows CPU time utilization of Lomonosov-2 supercomputer over one year period from August 2016 to August 2017. It is defined by the ratio of the number of nodes that execute users' jobs (black line) to the number of available nodes at a given moment of time (blue line).

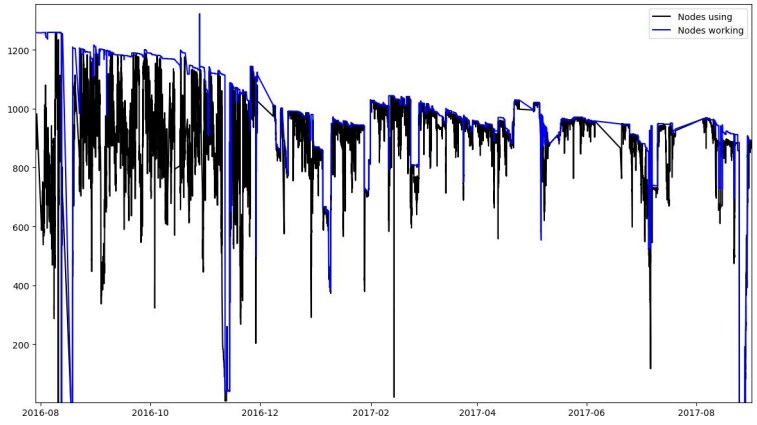

Fig. 1. Lomonosov-2 supercomputer CPU time utilization.

The overall CPU time utilization rate over this period amounted to 88,7%. The graph illustrates that before the year 2017 CPU time utilization was lower than the average due to the fact that the majority of users got an access to Lomonosov-2 supercomputer in January 2017. The average number of users raised at this time from approximately 3 to more than ten users per time unit. The efficiency of Lomonosov-2 supercomputer scheduling in terms of CPU time utilization starting from January 2017 almost reached the maximum. As a consequence, any further optimization of scheduling techniques must be based on other efficiency metrics that will allow to improve supercomputer users' experience.

For accurate analysis the Lomonosov supercomputer statistics was also examined on the same period (only for queue "regular4"). Note that the maximum number of available nodes for this period was only around 2000 (not 4096) because of maintenance works. CPU time utilization over the period totaled over 92%.

Owing to the fact that Lomonosov supercomputer operates for many years, it has a significantly wider user base than Lomonosov-2 supercomputer. Another valuable

observation that at the given moment of time the number of users in the queue is much more than the number of users whose jobs are on execution. This tendency represents the future development of user base and distribution of user jobs on queue and on execution of Lomonosov-2 supercomputer.

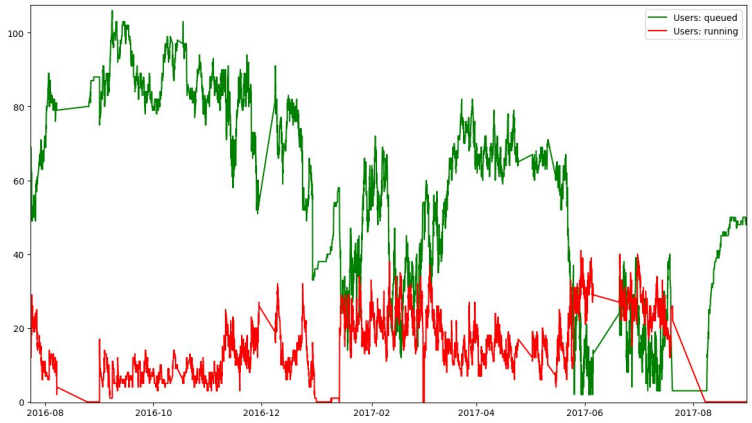

Fig. 2. Lomonosov-2 supercomputer number of jobs queued or running.

In spite of the high utilization of CPU time, RCC MSU supercomputers does not always use the most efficient settings for scheduling algorithms for majority of users. After analyzing historical data, we selected metrics that will help to assess more precisely the user's experience on the supercomputer.

4 Supercomputers' Scheduling Efficiency

Because of such considerable load, RCC MSU has faced a challenge of SLURM's settings scarcity. For instance, one of the serious problems of CPU time scheduling is the fact that each user's waiting time is too long, even when using a job scheduler based on Backfill algorithm. This causes a common problem: each user queues several jobs (often with the purpose of program debug) and waits. When the first job of the user is started, top of the queue is already occupied by this user's jobs, so the majority of other users are waiting for one. This issue cannot be solved by putting a limitation on the number of user's running jobs, as the efficiency of the queue may decrease significantly if it contains many short and/or small jobs.

Due to this limitations we have already tested some optimizations of Backfill algorithm. For example, we added individual CPU hours limit, this decision showed great performance on historical data. This solution has been tested compared to standard SLURM backfill plugin on historical data queue "regular4" Lomonosov supercomputer, and showed good acceleration for first job start time of each user (up

to ~9% compared to standard SLURM backfill plugin and average acceleration is ~9.5 minutes per week for first job start time of each user), while the total start time of jobs almost has not been increased (only 0.2% compared to standard SLURM backfill plugin and with an average 10 seconds start time delay for all users jobs, except first user's jobs). [10]

4.1 Jobs packing quality

Let's introduce some terms. A strip with fixed width H, which shows resources utilization of a computing system in time, is set (H - number of cluster nodes). The strip has an XY coordinate system (X corresponds to time, Y - number of nodes). In the strip we set a slot W with length T, which represents a time interval. Slot start coordinate is the coordinate of its bottom left angle (X_0, Y_0). (Fig. 3)

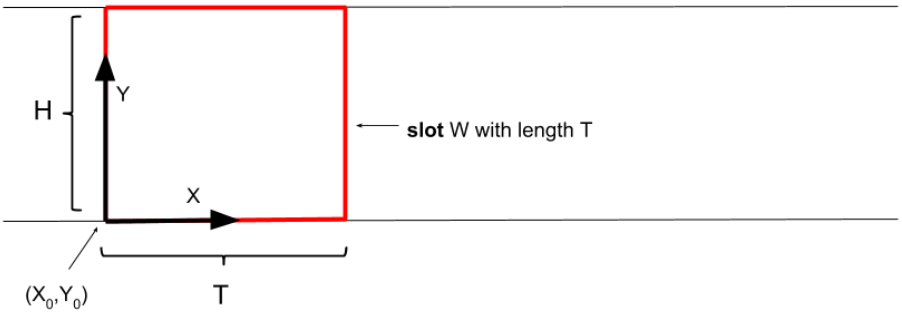

Fig. 3. Lomonosov supercomputer number of users, whose jobs queued or running.

Job is a user's program that has two states: it is either in a queue or is being executed in computing resources.

Job is a set of elements: Ji = {X_i, T_i, H_i, R_i, U_i, Q_i} (Fig. 4), where

X_i — execution start time of a task in computing resources;

T_i — time length of job execution in computing resources;

H_i — number of computing nodes required to execute a task;

R_i — non-empty setup of j pairs (y_{ij} , h_{ij}), which describes task allocation in nodes as a rectangle with bottom left angle coordinate (Xi, yij), T_i execution time and h_{ij} number of nodes such that $\sum_j h_{ij} = H_i$; (Fig. 4, 5)

U_i — identifier of a user associated with a task;

Q_i — queueing time.

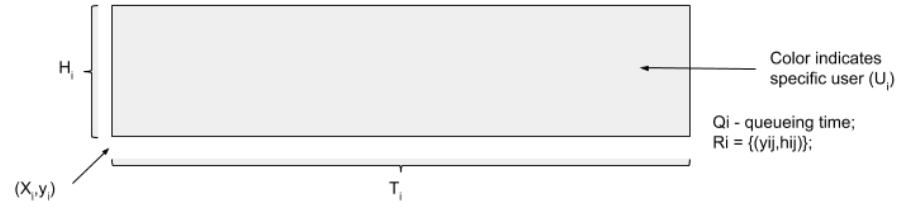

Fig. 4. Job

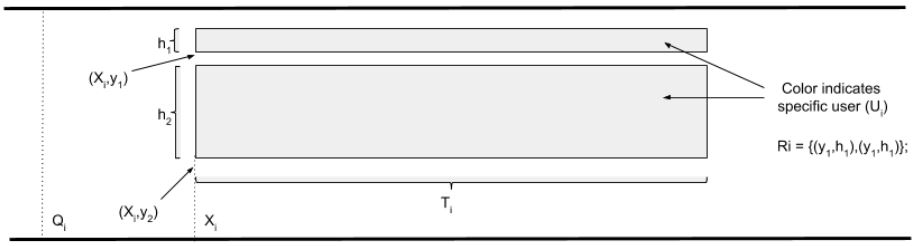

Fig. 5. Job J_i in a strip and R_i decomposition example.

Notations interpretation: a job is represented as a rectangle with set coordinates of a bottom left angle, defined size (H_i and T_i are rectangle's sizes on Y and X axes respectively) and color (corresponds to user identifier), decomposition of R_i among nodes. (Fig. 5)

Let's consider two setups of jobs:

1. Z_{start} is a setup of jobs executing in the $X_i = X_0$ moment of time. (Fig. 6)
2. Z_{queue} is a setup of queued jobs, for which X_i coordinates and R_i decomposition are not set and $Q_i \leq X_0 + T$. (Fig. 7)

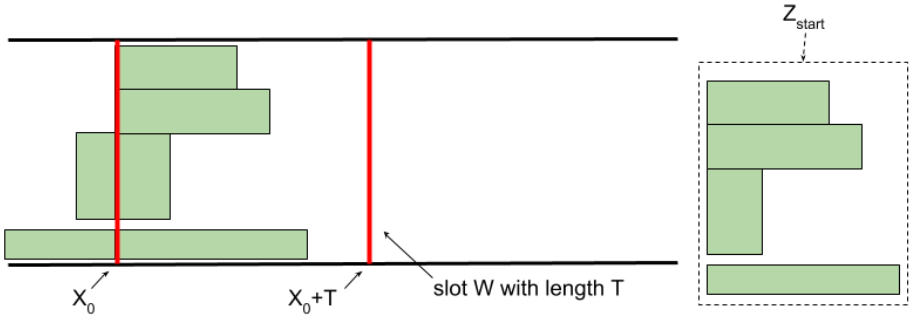

Fig. 6. Z_{start} setup and an example of its position in the W slot.

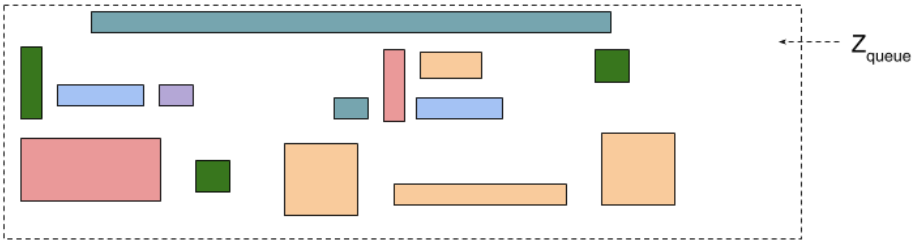

Fig. 7. Z_{queue} setup

Jobs packing in the W slot can be defined as a setup Z, which contains all the tasks from Z_{start} and 0 or more tasks from Z_{queue} with set coordinates X_i and decompositions of R_i, where:

1. $\forall\ X_i \in Z_{queue} \Rightarrow X_0 < X_i \leq X_0 + T$;
2. $\forall\ X_i \geq \max(X_0, Q_i)$;
3. all tasks in the Z pack do not intersect and lie in the strip.

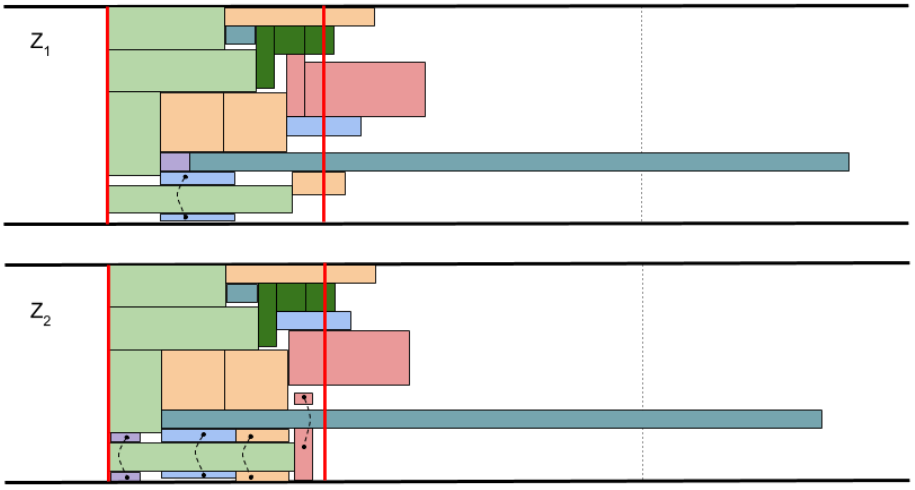

Fig. 8. Examples of two different packs Z_1 and Z_2

Let's clarify properties of a pack and a limitation imposed on its composition. We suppose that composite parts of tasks from Z_{start} cannot be changed and that in the process of packing for any task J_i from Z_{queue} it is allowed to break each particular rectangle from R_i into number of rectangles in such way that:

1. sizes of all the rectangles from R_i on the Y-axis add up to H_i, i.o. $\sum_j h_{ij} = H_i$;
2. X-coordinate of the left side of all rectangles is equal to X_i;
3. sizes of all the rectangles on the X-axis is equal to T_i;

4. it is not allowed to rotate rectangles.

Packing quality loss function is defined as a function of Z pack and slot parameters. Let us denote it by Opt(Z,W).

Definition of the problem: the objective is to find a Z_{end} jobs pack from job setups Z_{start} and Z_{queue} in slot W reaching the minimum value of packing quality loss function Opt(Z,W).

4.2 Efficiency metrics and its quality loss functions

Basing on the conducted research, we offer a set of metrics, which allows to consider the task of CPU hours scheduling efficiency more comprehensively, and a formula, which provides a means of comparing different settings of any scheduling algorithms. A list of metrics that should be considered when determining the operational efficiency of RCC MSU was formulated based on the historical data from supercomputers Lomonosov and Lomonosov-2 is presented below.

Let's introduce methods for evaluating the quality loss function of a Z pack. Each of proposed metrics represents some quality characteristic of supercomputer's resource management.

Given that

- UserNum(Z) — number of users, whose tasks belong to Z pack,
- UserJob(i) — a set of jobs of i^{th} user in Z pack,
- Class(A,B) — a set of jobs, for which H_i belongs to set half-interval [A, B).

Most widely used resource management quality characteristic is utilization of computing nodes. The main idea is to minimize the number of free resources.

$$\text{Opt(Z,W)=Utilization(Z,W)} = 1 - \sum_{i=1}^{|Z|} (H_i * (min(T,\ X_i + T_i) - X_i)/(H * T);$$

Another useful metric is average start time of the first job of users in slot W. The objective is to minimize the average distance from a job of each color, for which X_i - X_0 is minimal among jobs of this color, to the beginning of the slot W – X_0.

$$\text{Opt(Z,W)=FUJStartTime(Z,W)=} \sum_{user=1}^{UserNum} min_{j \subset UserJobs(user)} (X_j - Q_j)/UserNum(Z);$$

Next two metrics represents number of running jobs (minimizing the number of not running jobs from Z_{queue}) and number of users, whose jobs from Z_{queue} were started in slot W (minimizing the number of users, whose jobs from Z_{queue} were not started in slot W).

$$\text{Opt(Z,W)} = \text{StartedJobs(Z,W)} = (|Z_{start}| + |Z_{queue}| - |Z|)/(|Z_{queue}|);$$

$$\text{Opt(Z,W)=StartedUsers(Z,W)=UserNum}(Z_{start}) + \text{UserNum}(Z_{queue}) - \text{UserNum(Z)};$$

Last metric shows average start time of jobs belonging to a specific class (minimizing the average distance from each job, which size $H_i \in [A, B)$, to the beginning of the slot W – X_0).

Opt(Z,W)=AVGStartTime(Z,W,Class)= $\sum_{i \subset Class} (X_j - Q_j)/|Class|$;

Finally, we propose supercomputer's resource management efficiency definition and a related formula, which represents supercomputer scheduling efficiency based on normalized values of proposed metrics. $PriorityCoefficient$ variables indicate priorities of each metrics, it should be predefined due to analysis made on supercomputer administration needs. $MetricsValue_i$ is one of five represented metrics.

$$Efficiency = \sum_{i=1..5} (PriorityCoefficient_i * MetricsValue_i), \text{ where}$$

$$\sum_{i=1..5} (PriorityCoefficient_i) = 1.$$

Proposed formula is based on operational statistics of RCC MSU supercomputers, so in other cases metrics can be chosen in different way (depending on goals of supercomputer centres) and added to this formula. The only limitation is that $MetricsValue_i$ should be normalized on interval [0,1].

5 Future Work

Future work lies in field of designing fast online algorithm for proposed "efficiency" function minimum search. All future research will be conducted on Lomonosov-2 and Lomonosov supercomputers.

Acknowledgments. This material is based upon the work supported by Russian Foundation for Basic Research. (Agreement N 17-07-00664 A).

References

1. M. Jette, M. G. (2003). SLURM: Simple Linux Utility for Resource Management. Proceedings of ClusterWorld Conference and Expo. San-Jose, California
2. Slurm Workload Manager (2015), http://slurm.schedmd.com/slurm.html
3. V. Sadovnichy, A. T. (2013). "Lomonosov": Supercomputing at Moscow State University. In Contemporary High Performance Computing: From Petascale toward Exascale (pp. 283-307)
4. Lomonosov — T-Platforms (2015) Retrieved from http://www.top500.org/system/177421
5. D. Lipari (2012). The SLURM Scheduler Design. http://slurm.schedmd.com/slurm_ug_2012/SUG-2012-Scheduling.pdf
6. M. Jones (2012). Optimization of resource management using supercomputers SLURM. Retrieved from http://www.ibm.com/developerworks/ru/library/l-slurm-utility/
7. Lomonosov-2 supercomputer configuration (2018) Retrieved from http://users.parallel.ru/wiki/pages/22-config
8. Lomonosov-2 supercomputer on TOP50 list (2018) Retrieved from http://top50.supercomputers.ru/?page=stat&sub=ext&id=593

9. Antonov Alexander, Nikitenko Dmitry, Shvets Pavel, Sobolev Sergey, Stefanov Konstantin, Voevodin Vadim, Voevodin Vladimir, Zhumatiy Sergey (2015) An Approach for Ensuring Reliable Functioning of a Supercomputer Based on a Formal Model. Parallel Processing and Applied Mathematics. 11th International Conference, PPAM 2015, Krakow, Poland, September 6-9, 2015. Revised Selected Papers, Part I.
10. Leonenkov S., Zhumatiy S.: Introducing New Backfill-based Scheduler for SLURM Resource Manager, Procedia Computer Science. Volume 66, 2015, (pp 661-669)