Modern Applied Science; Vol. 9, No. 5; 2015
ISSN 1913-1844 E-ISSN 1913-1852
Published by Canadian Center of Science and Education

Decomposition Analysis and Machine Learning in a Workflow-Forecast Approach to the Task Scheduling Problem for High-Loaded Distributed Systems

Andrey Vladimirovich Gritsenko[1], Nikita Georgievich Demurchev[2], Vladimir Vyacheslavovich Kopytov[2] & Andrey Olegovich Shulgin[2]

[1] Department of Information Technologies, North-Caucasus Federal University, Stavropol, Russian Federation

[2] Infocom-S, Stavropol, Russian Federation

Correspondence: Andrey Vladimirovich Gritsenko, Department of Information Technologies, North-Caucasus Federal University, Stavropol, Russian Federation.

Received: December 29, 2014 Accepted: January 3, 2015 Online Published: March 25, 2015

doi:10.5539/mas.v9n5p38 URL: http://dx.doi.org/10.5539/mas.v9n5p38

Abstract

The aim of this paper is to provide a description of machine learning based scheduling approach for high-loaded distributed systems that have patterns of tasks/queries that occur recurrently in workflow. The core of this approach is to predict the future workflow of the system depending on previous tasks/queries using supervised learning. First of all, the workflow is analyzed using hierarchical clustering to reveal sets of tasks/queries. Revealed sets of tasks/queries then undergo restructuring to represent patterns of recurrent tasks/queries. Later these patterns become the object of the forecasting process performed using neural network. Information on predicted tasks/queries is used by the resource management system (RMS) to perform efficient schedule. To estimate the performance of the described method it was at first realized as a module of the simulation tool Alea that models the work of high-performance distributed systems and then compared with other state-of-the-art scheduling algorithms. The simulation was produced for two datasets: in one of the experiments the proposed method showed best results, and in the other it was inferior to just a single method, though it was much better than commonly used standard scheduling algorithms.

Keywords: task scheduling, machine learning, supervised learning, hierarchical clustering, workflow prediction, resource management systems

1. Introduction

The field of high-performance computing (HPC) is nowadays presented by a vast class of distributed computing systems. The main criterion to estimate the productivity of HPC systems has always traditionally been the time required to complete a full set of tasks, called makespan. It is obvious that values of such estimation criteria depend not on the productivity of the distributed computing system itself, but on the efficiency of task management systems as well. A task management system is a tool that schedules the execution of users' tasks on the HPC system's resources. There is a set of criteria besides makespan that is used to assess the efficiency of task management systems and scheduling algorithms as well.

Considering that there a lot of types of distributed HPC systems for some of them other objectives rather than makespan can be the most significant. For example, for users of academic-purpose distributed computing systems the objective that presents time users wait for their tasks to begin to execute can be even more important than makespan, especially for a certain group of users – students, who are time-limited to utilize HPC system's resources. The same requirements may be also imposed by users to other types of distributed systems and servers. For example, nowadays as a result of and along with the ubiquitous distribution of different kinds of smartphones, tablets and other mobile gadgets the percentage of high-loaded mobile systems increases as well among overall number of distributed systems and servers. It becomes obvious that the crucial parameter by which users assess the work of such systems is the timeliness of the execution of users' tasks and queries, i.e. the task or query begins to execute as soon as it is submitted.

There are two possible ways to reduce the amount of time a user waits before his task or query is completed, not considering the time of execution itself. The first one is 'quantitative' – to increase the amount of resources. Thus, whenever a user submits a task or a query there are always available resources to execute the task/query. Obviously, it is not an appropriate solution primarily because of the expensiveness of distributed systems' and centers' equipment. The second solution is to try to forecast the submission of a task/query so the resources required for its execution could be freed beforehand.

There has been done some research in the area of forecasting the future state of the HPC system's workload. In this area scientists have mainly focused on two lines of research:

1) prediction of the time a user waits before his task begins to execute (Li, 2008; Smith et al., 2004; Yuan et al., 2008);
2) prediction of the completion time of local tasks in GRID computing systems (Akioka and Muraoka, 2004; Halimon and Smirnov, 2009; Li et al., 2004).

Evidently, results on the first direction could be used directly to reduce the waiting time – time a user waits before his task or query begins to execute. While results on the second line of research may look appropriate for the problem of forecasting the workload of a distributed HPC system, apparently it is much easier to predict a workload for a single user then for a number of users. As it would be shown in Section 2.2 standard forecasting methods could not be applied for such a complex workload.

Taking into account all of the above the following can be stated: the scientific topicality of the development of a method to forecast the future workload of a distributed HPC system and an algorithm to use this prediction in a schedule to increase the efficiency of this system is well founded. In terms of practice the topicality could be grounded as follows: the share of academic-purpose distributed computing systems reaches 17.4% amongst TOP500 supercomputer sites (15.6% in performance scale) that makes this group of distributed HPC systems a valuable object of research.

2. Methodology

In this section we would first describe the preliminary analysis of workloads of two distributed HPC systems. This analysis reveals the presence of long-term memory. In Section 2.2 we attempted to use several standard forecasting methods: autoregressive integrated moving average (ARIMA), group-method data handling (GMDH) and singular spectral analysis (SSA). Section 2.3 is devoted to the description of the decomposition process of the workload to reveal patterns of recurrent tasks, based on the hierarchical clustering method. In Section 2.4 we describe the implementation of a neural network to forecast for each of the revealed patterns when and whether a new task would be submitted.

To approve the proposed forecasting method we used workloads of two distributed HPC systems. The first workload contains 17365 tasks submitted during 6 months to 14 computing nodes of the cluster system Zewura in the Czech National Grid Infrastructure MetaCentrum with overall amount of CPUs equivalent to 806 and performance equivalent to 14.7 TFlops/s (MetaCentrum, 2014). The second workload contains 202876 submitted tasks during 42 months to 4692 computing nodes of the 3rd most powerful academic-purpose HPC system in the world Lomonosov with overall amount of CPUs equivalent to 39920 and performance equivalent to 901.9 Tflops/s (Lomonosov, 2014). Workloads are named Zewura and Lomonosov respectively by the names of their distributed HPC systems.

2.1 Analysis of the Workloads

The analysis of academic-purpose clusters' workloads including Zewura described in (Klusacek and Rudova, 2010) reveals a great amount of periodically submitted tasks with similar features: resource requirements, priority and duration (Figure 1). The subsequent inquiry of the Zewura workload made it possible to ascertain that patterns of recurrent tasks were submitted by a certain group of users – students. Though it was impossible to perform the similar examination of the Lomonosov workload due to its uniform ungrouped structure, we have made a hypothesis, that workloads of academic-purpose high-performance systems could be characterized as having such patterns of recurrent tasks.

It becomes obvious that seriate task submissions allow to use predictive methods to forecast the future workflow of the cluster system. First of all, to perform any of the predictive algorithms a list of submitted tasks should be presented as time series. Time series contain the following information about tasks – submission time, priority and resource requirements: number of requested nodes and processor time.

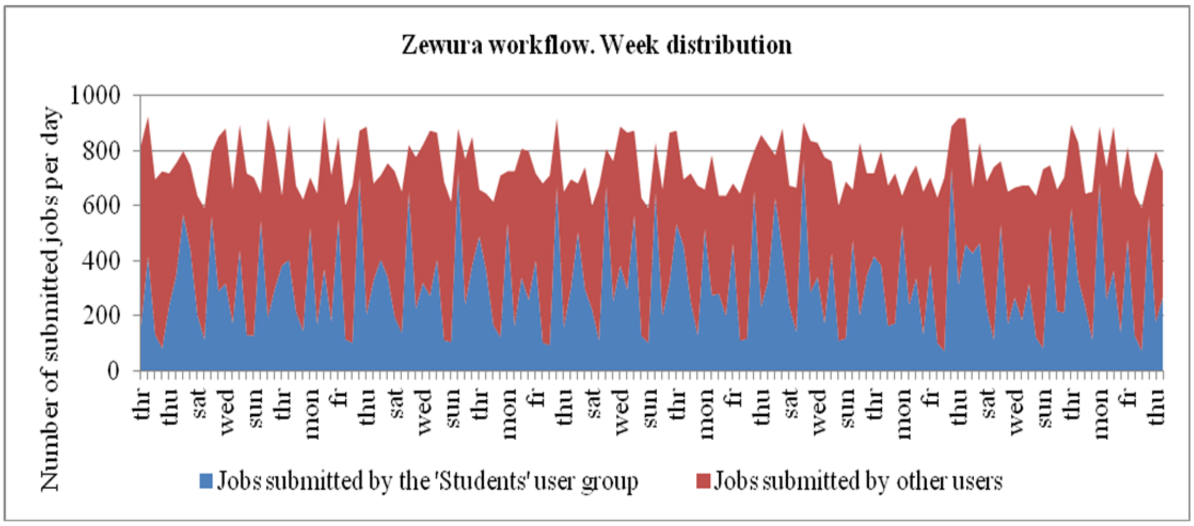


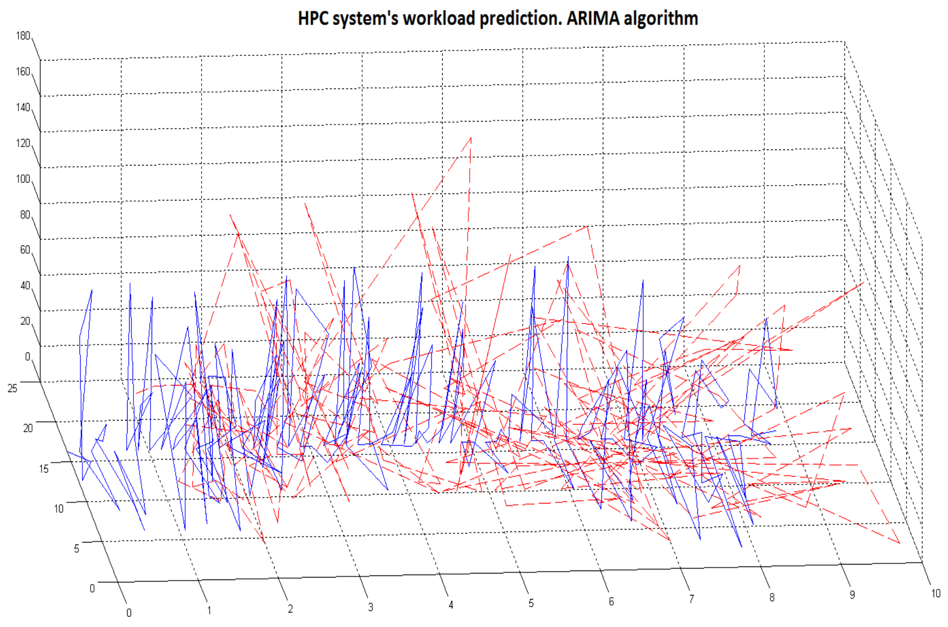
Figure 1. Recurring patterns of the tasks submitted by the users of the Student user group observed in the Zewura workflow

To obtain more reasons for using forecasting methods to predict the future workload for academic-purpose high-performance systems the Hurst exponent (Hurst et al., 1965) for both Zewura and Lomonosov workflow time series was computed. The Hurst exponent is used as a measure of long-term memory of time series. In other words it determines the rate at which autocorrelation function of time series decreases as the lag between pairs of values increases. The presence of long-term memory that corresponds to the value of Hurst exponent $H \in (0.5;1)$ in turn makes the prognosis of future tasks highly reliable. The process of Hurst coefficient computation for both Zewura and Lomonosov workloads is described in (Tebueva at al., 2011). The value of the Hurst coefficient for the Zewura workflow is equivalent to 0.714731 and for the Lomonosov workload – 0.69814, thereby verifying the appropriateness of the implementation of prediction algorithms.

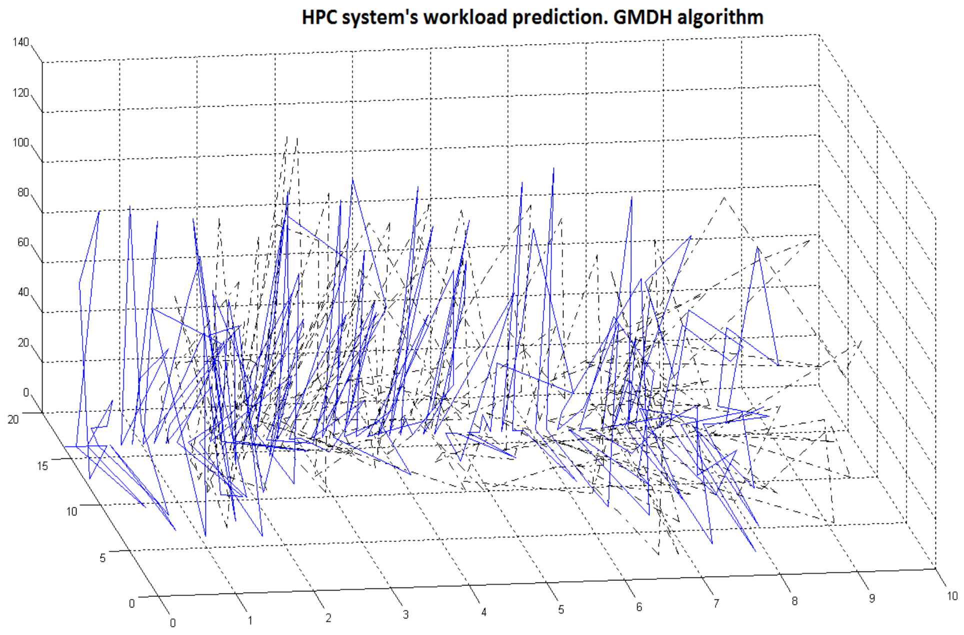
2.2 The Comparative Analysis of Using Different Forecasting Methods

Research described in (Gritsenko and Shulgin, 2011) is devoted to the appliance of ARIMA (Box et al., 1994) and GMDH (Madala and Ivakhenko, 1994) methods as well as SSA (Golyandina and Stepanov, 2005) forecasting algorithm in order to gain a proper forecast for the Zewura workload. Figure 2 shows a great inaccuracy of the results when using algorithms mentioned above.

(a)


HPC system's workload prediction. ARIMA algorithm
180
160
140
120
100
80
60
40
20
0
25
20
15
10
5
0
0
1
2
3
4
5
6
7
8
9
10

(b)


HPC system's workload prediction. GMDH algorithm
140
120
100
80
60
40
20
0
20
15
10
5
0
0
1
2
3
4
5
6
7
8
9
10

(c)

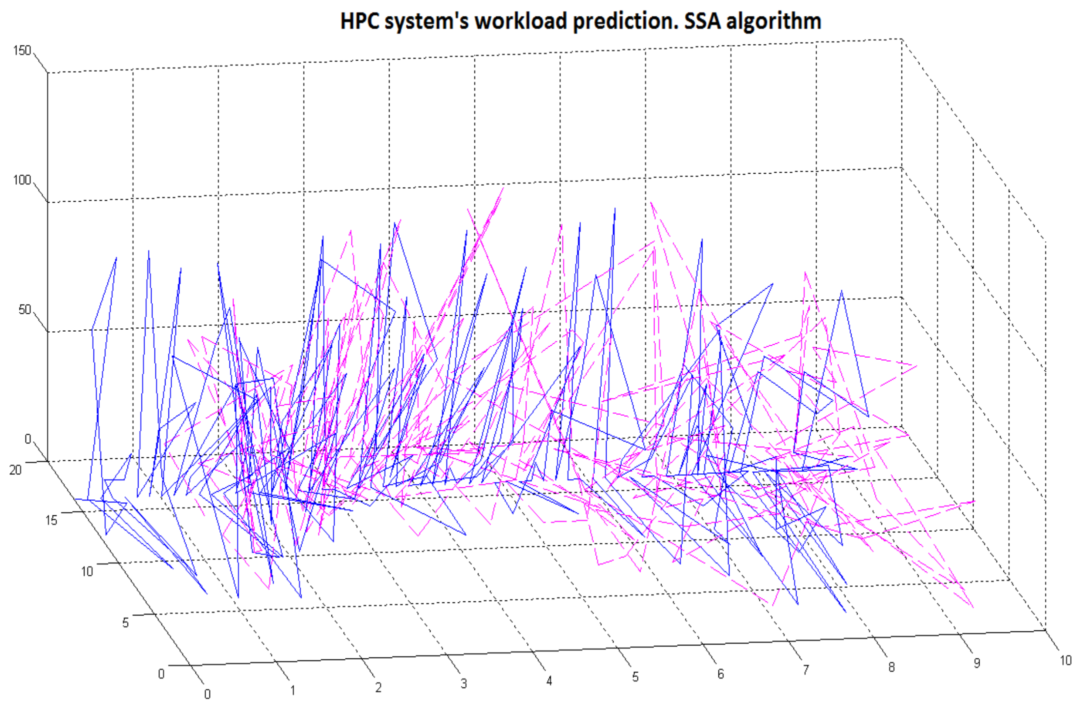


Figure 2. Prediction inaccuracy of different forecasting algorithms: (a) ARIMA method – red line, (b) GMDH method – black line, (c) SSA metnod – magenta line. X axis (length) represents time in seconds x10^4, Y axis (higth) – task execution time in hours, and Z axis (depth) – the number of requested nodes. For more convenience the original data is displayed as blue line on every plot

Observing plots from the Figure 2 it can be concluded that there is a contradiction. On the one hand, the analysis in (Tebueva et al., 2011) showed that both Zewura and Lomonosov workloads possess long-term memory and thus should be appropriate for a successful forecasting. On the other hand, the attempts to use different prediction methods failed as the results are absolutely insufficient. To solve the arisen contradiction we proposed to decompose time series to reveal essential seriated components.

2.3 Forming Patterns of Recurrent Tasks

As it was stated above, the tasks that are recurrently submitted by users have similar task features: submission time (with regard to the hour/day of the week/day of the month/etc.), execution time, resource requirements, priority. Considering this we have drawn the following conclusion: if recurrent tasks that belong to the same pattern have similar features, then represented as points in the multidimensional space they would be very close to each other. If we subject a workload of any academic-purpose distributed computing system to the clustering process, we would obviously get high-dense clusters that represent patterns of recurrent tasks. As we do not know beforehand the number of patterns of recurrent tasks we can not use standard clustering methods; instead we should implement agglomerative hierarchical clustering method.

At the very beginning the agglomerative hierarchical clustering methods represents all the tasks in the workload space as separate clusters. Then the iterative process begins: at each step the nearest two clusters merge. To decide what clusters are the nearest a measure of dissimilarity is required. This measure could be considered as the combination of an appropriate metric (a measure of distance between pairs of points) and a linkage criterion. A linkage criterion specifies the dissimilarity of clusters as a function of the pairwise distances of points in the clusters.

Among different options of metrics we have chosen Mahalanobis distance (McLachlan, 1992) as it considers not only the distance between the points but the shape of clusters as well:

$$d(a,b) = \sqrt{(a-b)^T S^{-1} (a-b)} \tag{1}$$

In Equation 1 *S* represents a covariance matrix.

To estimate the distance between two clusters we used Unweighted Pair Group Method with Arithmetic Mean (UPGMA). According to this method the distance between any two clusters *A* and *B* is taken to be the average of all distances between pairs of objects *x* in *A* and *y* in *B*, that is, the mean distance between elements of each cluster:

$$\frac{1}{|A| \cdot |B|} \sum_{x \ni A} \sum_{y \ni B} d(x, y) \tag{2}$$

Obviously, we are not interested to wait until UPGMA agglomerative hierarchical clustering method finishes it execution, as it stop when all of the initial data points are merged into a single cluster. On the contrary, our goal is to obtain high-dense clusters that represent patterns of recurrent tasks. To reach this goal we have added a stopping criterion to the original UPGMA method: if the value of the metric for a newly merged cluster exceeds more than two times the value of the metric for any of the merging clusters, then such clusters are not combined and excluded from the further consideration. The modified UPGMA stops when there are no clusters left to merge.

We interpret the obtained clusters of data point as patterns of recurrent tasks by ordering them in ascending order of their submission time. We also compute for each pattern its period as the mean difference between submission times of two adjacent tasks.

2.4 Forecasting Patterns of Recurrent Tasks

When we get the patterns of recurrent tasks we can now perform the prediction for each pattern individually. Apparently, standard forecasting methods like ones mentioned in Section 2.2 are not suitable for this mission, as they answer the question "*What* will appear in the next instant of time?" We have proposed to prolong patterns of recurrent tasks in the future infinitely using its period and an 'averaged task'. The 'averaged task' is a task whose features are computed as mean values of the corresponding features of the tasks that currently belong to the pattern. That allows us rephrasing the question. Now we can forecast the future state of workloads of academic-purpose distributed HPC systems by answering the following question "*Will* a certain task appear in a definite instant of time?"

To answer this question we have proposed to implement a neural network as its standard output signal could be either 0 or 1 (Werbos, 1974). As the input signal we have used not only submitting parameters of tasks like priority, resource requirements, execution time and user group's name, we have also added the position of the task in the pattern as well as the features that describe the pattern itself and: pattern's period, pattern's length so far (i.e. the amount of tasks that have already occurred in the workload). To prevent unnecessary executions of the neural network we have added an exclusive criterion: once for a certain pattern the number of false predictions exceeds 20% of the current pattern's length, all the tasks of this pattern are excluded from the further consideration.

There are several possible ways of acting when we get a prediction result. First of all, one can implement a binary decision-making process: if the output signal of a neural network is less than 0.5 then the task is ignored, otherwise the required for its execution resources are reserved. On the other hand, one can treat the output as the probability of the task's appearance in the workload and act according to this probability. For example, if the output value is between 0.8 and 1 then it is highly probable that this task will be submitted by a user and thus the resources for this task reserved 'firmly' – this reservation could not be undone. If the output is between 0 and 0.4 then it is highly probable that the task would not occur in the workload and it is ignored. If the output is between 0.4 and 0.5 then a 'soft' reservation is made – the required resources are reserved until a task with a higher priority occurs in the workload, then the reservation is undone.

In our experiments with the Zewura and Lomonosov workloads we have implemented the first mentioned option – binary decision-making process. Figure 3 depicts the accuracy of the results of this implementation of the proposed forecasting approach in comparison to the mentioned above forecasting methods making it possible to evaluate the precision of these methods by sight.

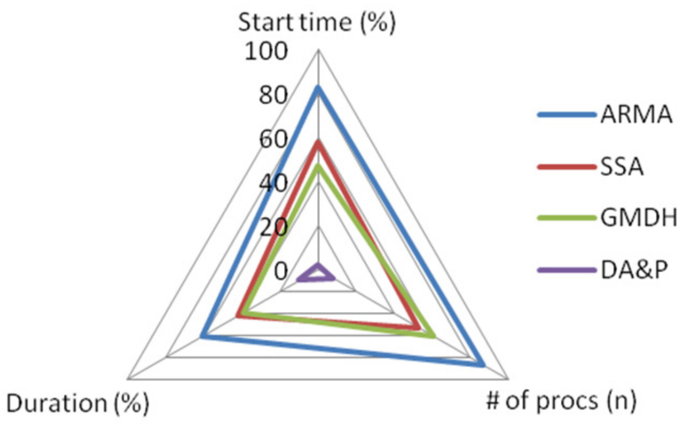


Figure 3. Diagram of the mean absolute deviation of the different forecasting methods' results

3. Experiments and Results

As it can be concluded from the Section 2.4 our proposed method only affects those tasks that belong to any pattern of recurrent tasks. Accordingly, our proposed method DAP could not be used as a separate scheduling method, it could be implemented only as an additional software to other existing scheduling algorithms. In our experiments we used DAP method in the combination with a routine scheduling method – backfilling method.

In the experiments we used the following architecture of the neural network as a result of the performed analysis: it has 5 hidden layers with 10 neurons each. Table 1 contains the results of this comparative analysis of different architectures: *a* – 1 hidden layer with 10 neurons; *b* – 5 hidden layers with 10 neurons each; *c* – 1 hidden layer wit 25 neurons; *d* – 10 hidden layers with 10 neurons each.

Table 1. The efficiency of a neural network training

	Objective function value				Prediction accuracy (%)			
Architecture	*a*	*b*	*c*	*d*	*a*	*b*	*c*	*d*
Zewura	2.48	1.69	2.24	1.52	70.4	95.04	84.88	96.13
Lomonosov	2.49	1.09	1.97	1.33	58.08	92.13	80.8	91.4

The prediction accuracy was computed using *F1*-metric:

$$F_1 = 2 \cdot \frac{Precision \cdot Recall}{Precision + Recall} \tag{3}$$

In Equation 3, precision expresses the ratio between true positive predictions and the overall positive predictions. Recall expresses the ratio between true positive predictions and the overall positive occurrences.

When training the neural network we used a standard objective function:

$$J(\Theta) = -\frac{1}{m}\left[\sum_{k=1}^{m} y^k \log\left(h_\Theta\left(x^k\right)\right) + \left(1-y^k\right)\log\left(1-h_\Theta\left(x^k\right)\right)\right] + \frac{\lambda}{2m}\sum_{l=1}^{L-1}\sum_{i=1}^{s_l}\sum_{j=1}^{s_{l+1}}\left(\Theta_{ji}^{(l)}\right)^2 \tag{4}$$

In Equation 4, m – amount of the input elements, y^k – real output for the kth input element; $h_\Theta\left(x^k\right)$ – produced output for the kth input element; L – number of layers; S_l – number of neurons in the layer *l*, excluding biased neurons; $\Theta_{ji}^{(l)}$ – mapping coefficient from the ith neuron of the lth layer to the jth neuron of the (*l*+1)th layer.

There is a vast number of different objective functions to estimate the efficiency of a task management systems.

Amongst this number we have chosen 3 objective functions that allow to evaluate the efficiency of a task management system from different points of view:

- Makespan

Makespan is an essential objective function that is defined as the completion time of the last task in the workload. In This criterion is turned out to be unbiased as it does not depend on the execution order of tasks. This objective assesses the work of a task management system from the administrator's point of view.

- Computational resource utilization

Instead of estimating the efficiency of a schedule calculating merely the percentage of used CPUs, it is more reasonably to compute the resource utilization using the Equation 5 that does not consider situations when there are not enough submitted jobs to use all of the available resources:

$$Resource\ utilization = \frac{CPU_{active}}{\min\left(CPU_{available}; CPU_{requested}\right)} \tag{5}$$

Computational resource utilization makes it possible to assess the efficiency of a schedule in respect to the economic aspect of the scheduling problem.

- Slowdown

Slowdown objective function is a dimensionless quantity that is calculated as shown in the Equation 6:

$$Slowdown = \sum_{Tasks} \frac{FinishTime - SubmitTime}{FinishTime - StartTime} \tag{6}$$

Slowdown involves both wait time (difference between start time and submission time) and response time (difference between finish time and submission time). Its advantage in comparison to these objectives is that slowdown considers processing time of each task, thus decreasing the influence of small tasks being in the queue for a long time.

The detailed description of the multi-criteria evaluation process of task management systems is given in (Gritsenko, 2014). Table 2 contains the binary comparison matrix for the three mentioned objectives. Weights of these criteria correspond with the explanation given in the Section 1: the timeliness of the execution of users' tasks and queries becomes the crucial parameter by which the efficiency of distributed high-loaded systems is estimated, and could be explained by means of the following reasons.

- From the point of view of the users the slowdown objective is much more important than the resource usage as no one wants its task to be in the queue for a long time.
- The aim of the distributed computing system is to complete all the tasks in the queue as possible in a short period of time therefore the makespan is as significant criterion as the slowdown, besides frequently these objective functions correlate in a way where decrease of the slowdown results in the decrease of the makespan.
- On the other hand, taking into consideration high cost of the distributed systems' equipment the value of the resource utilization objective increases drastically making its influence on the overall schedule efficiency rating more significant.
-

Table 2. Binary comparison matrix

Criteria	Makespan	Slowdown	Resource usage	∑
Makespan	–	0.5	0	0.5
Slowdown	0.5	–	1	1.5
Resource usage	1	0	–	1

To estimate the performance of the proposed approach for the task scheduling problem a grid simulation system Alea 3.1 (Rudova & Klusacek, 2010) was used. This software allows executing a various number of scheduling algorithms of both queue-based and schedule-based approaches. The main distinguish between these approaches is that queue-based algorithms perform dynamic scheduling when a schedule is constructed every time a new task is submitted, while algorithms of the schedule-based approach require information about all tasks in the workflow at the moment of scheduling, in other words perform a static scheduling. This feature of the schedule-based approach makes it impossible to use its algorithms in real distributed computing systems. Though schedule-based algorithms can be used as relevant estimators in simulators, since average values of the objective functions for schedule-based methods are often much better than those for queue-based methods.

To perform an exhaustive and reliable comparative study the following algorithms were specified along with the proposed algorithm DAP:

- Queue-based: first-come-first-served (FCFS), smallest-job-first (SmalJF), conservative backfilling (Cons BF), aggressive /easy backfilling, last-come-first-served (LCFS), shortest-job-first (SJF), first-fit, earliest-deadline-first (EDF), algorithms used in PBS-Pro (Nitzberg et al. 2004) resource management system;
- Schedule-based: earliest-suitable-gap (ESG), best-gap, tabu-search.

The results of the execution of different algorithms scheduling Zewura and Lomonosov workloads are presented in the tables 3, 4, 5 and 6. System usage and slowdown are dimentionless quantities and makespan is measured in seconds.

Table 3. Absolute values of the objective functions for different scheduling algorithms. Zewura Workload

Criteria	Scheduling algorithms													
	FCFS	SmalJF	Cons BF	Easy BF	LCFS	SJF	First Fit	PBSPro	BSG	ESG	DAP	Best Gap	Tabu Search	EDF
Makespan	16630898	16410022	14977165	15224472	15754907	16159979	16109705	15214316	15365688	15074443	15223893	15372390	15698837	16630898
System usage	77.016	85.044	90.989	87.931	82.744	86.088	79.82	88.3	86.653	92.642	92.284	92.749	83.738	77.016
Slow down	10916.3	956.993	493.165	1111.56	1904.01	1184.29	6960.19	465.714	576.358	261.779	257.016	238.452	1933.40	10916.3

Table 4. Absolute values of the objective functions for different scheduling algorithms. Lomonosov workload

Criteria	Scheduling algorithms													
	FCFS	SmalJF	Cons BF	Easy BF	LCFS	SJF	First Fit	PBSPro	BSG	ESG	DAP	Best Gap	Tabu Search	EDF
Makespan	107100566	107108258	107098023	107097817	107099830	107101707	107100548	107099792	107100516	107098164	107098060	107098124	107100516	107100566
System usage	98.45	99.027	98.753	98.547	98.69	98.974	98.785	99.131	98.997	98.89	98.965	99.051	98.623	98.45
Slow down	612.998	69.3385	212.874	151.005	208.288	254.501	596.212	186.287	289.059	76.4019	88.2647	90.6587	309.628	612.998

The comparison process described in (Gritsenko, 2014) can be divided into the following steps:

1) Compute max and min values of objectives: for every objective the biggest and the smallest values are determined on the set of specified algorithms.
2) It is natural that one algorithm may be superior as compared to the other algorithm by one objective, and be inferior when compared by another objective. To simplify the evaluation process we compute relative estimations for objectives: for each algorithm a relative estimation is calculated as a ratio of the difference between algorithm's objective value and the min value of this objective to the difference between max and min values of the same objective. This estimation shows the proximity of the evaluated algorithm to the best algorithm on a set of objectives.
3) Global estimations: the next step is to compose a square matrix with the number of columns and rows equals to the number of compared algorithms. Each element of the matrix represents the ratio of the overall superiority of the row algorithm on the column algorithm to the overall superiority of the column algorithm on the row algorithm.
4) Final estimations: to define which of the compared scheduling algorithms performs the most efficient

schedule on a set of specified objective functions one should calculate a main eigen vector for the defined prefence matrix. An algorithm that relate to the biggest element of the main eigen vector is the desired algorithm.

Table 5. Relative global estimations for most efficient algorithms. Zewura workload

	Scheduling algorithms				Main eigen vector
	Cons BF	DAP	BestGap	TabuSearch	
Cons BF	1	0.0535	0.1468	0.2	0.038
DAP	18.7039	1	13.9913	1.9021	0.9474
BestGap	6.8126	0.0715	1	0.5029	0.1391
TabuSearch	5	0.5258	1.9883	1	0.2856

Table 6. Relative global estimations for most efficient algorithms. Lomonosov workload

	Scheduling algorithms						Main eigen vector
	SmallestJF	Easy BF	PBS-pro	DAP	BestGap	TabuSearch	
SmallestJF	1	3.7388	2.5707	0.6727	0.7145	0.5195	0.1813
Easy BF	0.2675	1	0.5471	0.0108	0.0077	0.0089	0.0143
PBS-pro	0.3890	1.8278	1	0.2775	0.2121	0.1049	0.0585
DAP	1.4865	92.9277	3.6042	1	1.1406	0.6587	0.4249
BestGap	1.3997	130.6599	4.7149	0.8768	1	0.2293	0.4442
TabuSearch	1.9249	111.3494	9.5369	1.5181	4.36066	1	0.7653

3. Discussion

As it can be observed from the result tables above the proposed algorithm described in this article offers a great performance compatible to the performance of the schedule-based methods and superior to the ones of the queue-based methods.

Nevertheless, in order to implement the proposed decomposition analysis and machine learning based algorithm properly and to guarantee the superiority of the proposed algorithm, the distributed computing systems must meet some requirements. The main requirement consists of the following. The distributed high-load system must possess patterns of recurrent tasks or queries. Amongst the systems that definitely possess patterns of recurrent tasks are academic-purpose distributed high-performance computing systems, as well as distributed high-loaded mobile systems. Though in this work only experiments with the workloads of academic-purpose distributed computing systems have been executed and depicted in future works workload of other systems that are characterized by the presence of patterns of recurring tasks or queries.

The list of minor restrictions placed on distributed HPC systems involves homogeneous architecture of these systems and non-preemptive job scheduling. Although, these restrictions could be easily get round. In modern cluster systems with heterogeneous architecture there are certain queue associated with certain type of computational resources. Consequently each heterogeneous system could be considered as a set of several homogeneous systems that at first glance allow to evade the restriction. On the other hand the strong heterogeneity is essential to grid systems as well as preemptive job scheduling that is crucial to these systems. Taking into account all of the above it could be stated that redemption of both homogeneous and non-preemptive restrictions would expand the area of application on distributed high-loaded systems with different principles of operation and architectures, including grid systems.

There is also some future research to be done to make the proposed algorithm widely applicable for the task scheduling problem. The main research issue inherent to the proposed scheduling algorithm is covered in the learning paradigm of the implemented learning algorithms: both decomposition analysis based on the hierarchical clustering and prediction realized as neural network use batch learning. In other words, these algorithms have access to the entire dataset at once. As a result, in terms of batch learning paradigm it is impossible to reveal patterns of recurrent tasks that have not been presented in the initial training dataset. To overcome this problem the learning paradigm should be changed to the online learning, though this change in terms of distributed high-loaded systems may cause the increase of the computational complexity of the used algorithms. Therefore the additional research is required to find the trade-off between the computational complexity and the online learning advantages.

Acknowledgments

Research was done in the course of research scientific work under the name of: "The development of cross-platform mobile application engineering technology with the specified contours of integration with the purpose to increase functional and resource efficiency of corporate information systems" (unique identifier of applied scientific research is RFMEFI57614X0066) as part of the realization of federal special-purpose program "Research and development in priority areas of the elaboration of Russian scientific technology complex for 2014-2020 years".

The Zewura workload log was graciously provided by the Czech National Grid Infrastructure Metacentrum.

The Lomonosov workload was gained gathering information on submitted tasks via monitoring the official site that shows current state of the cluster system Lomonosov.

Alea 3.1 software is the result of the research intent no.0021622419 (ministry of education, youth and sports of the Czech Republic) and the grant no. 201/07/0205 (grant agency of the Czech Republic). The owner of the result is Masaryk University, a public high school, ID: 00216224.

References

Akioka S., & Muraoka, Y. (2004). *Extended forecast of CPU and network load on computational GRID.* IEEE International Symposium on Cluster Computing and Grid, pp. 765-772. http://dx.doi.org/10.1109/CCGrid.2004.1336711

Box, G., Jenkins, G. M., & Reinsel, G. C. (1994). Time series analysis: Forecasting and control. Prentice-Hall.

Golyandina, N., & Stepanov, D. (2005). SSA-based approaches to analysis and forecast of multidimensional time series. In proceedings of the 5th St.Petersburg workshop on simulation.

Gritsenko, A. (2014). A comparison model of scheduling and resource distribution algorithms for large scale computing systems. *In the world of scientific discoveries, 10*(58), 80-96. ISSN: 2330-9288. http://dx.doi.org/10.12731/wsd-2014-10-6

Gritsenko, A., & Shulgin, A. (2011). Applying of forecasting methods in job management and resource allocation algorithms in high performance systems issue investigation. *In the World of Scientific Discoveries, 8*(20), 43-55.

Halimon, V. I., & Smirnov, A. V. (2009). Optimization of distributed computational processes in corporate nets. Proceedings of XXII International Scientific Conference: Mathematical methods in technique and technologies.

Hurst, H. E., Black, R. P., & Simaika, Y. M. (1965). Long-term storage: An experimental study. London.

Klusacek, D., & Rudova, H. (2010). The importance of complete data sets for job scheduling simulations. Job Scheduling Strategies for Parallel Processing, pp. 132-153. http://dx.doi.org/10.1007/978-3-642-16505-4_8

Li, H. (2008). Workload characterization, modeling, and prediction in grid computing. PhD thesis, ASCI Graduate School, University of Leiden.

Li, H., Groep, D., Templon, J., & Wolters L. (2004). *Predicting job start times on clusters.* IEEE International Symposium on Cluster Computing and Grid, pp. 301-308.

Lomonosov supercomputer online state. Retrieved August 5, 2014, from http://t60-2.parallel.ru/cgi-bin/cleo-viz.cgi

Madala, H. R., & Ivakhnenko, A. G. (1994). Inductive learning algorithms for complex systems modeling. CRC press, Boca Raton.

McLachlan, G. J. (1992). Discriminant analysis and statistical pattern recognition. Wiley Interscience. http://dx.doi.org/10.1002/0471725293

Nitzberg, B., Schopf, J. M., & Jones, J. P. (2004). PBS-Pro: GRID computing and scheduling attributes, GRID resource management. ISBN 1-4020-7575-8.

Rudova, H., & Klusacek, D. (2010). Alea 2 - job scheduling simulator. In proceedings of the 3rd international ICST conference on simulation tools and techniques. ISBN: 978-963-9799-87-5.

Smith, W., Foster, I., & Taylor V. (2004). Predicting application run times with historical information. *Journal of*

Parallel and Distributed Computing, 64(9), 1007-1016. http://dx.doi.org/10.1016/j.jpdc.2004.06.008

Tebueva, F., Gritsenko, A., & Rusakov D. (2011). The method of calculating classification indices of time series of evolutional processes with long-term correlations. *Vestnik SGU, 75*(4), 39-43. ISSN: 1998-6383.

The national grid infrastructure Metacentrum. Retrieved September 15, 2014, from http://www.metacentrum.cz/en/

Werbos, P. J. (1974). Beyond regression: new tools for prediction and analysis in the behavioral sciences. Phd thesis, Harvard University.

Yuan, Y., Wu Y., Yang, G., & Zheng, W. (2008). Adaptive hydrid model for long-term load prediction in computational grid. IEEE International Symposium on Cluster Computing and Grid, pp. 340-347. http://dx.doi.org/10.1109/CCGRID.2008.60