Azerbaijan Journal of High Performance Computing, Vol 1, Issue 2, 2018, pp. 158-167
https://doi.org/10.32010/26166127.2018.1.2.158.167

Load Balancing in Distributed Exascale Computing Based on Process Requirements

Shirin Shahrabi[1], Faezeh Mollasalehi[1], Araz R Aliev[2], Ehsan Mousavi Khaneghah[1]

[1]Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran; Sh.Shahrabi@Shahed.ac.ir, Faezeh.Mollasalehi@Shahed.ac.ir, EMousavi@Shahed.ac.ir;

[2]Department of General and Applied Mathematics, HPC Advance Research Center, Azerbaijan State Oil and Industry University, Baku, Azerbaijan; alievaraz@azjhpc.org

*Correspondence: Shirin Shahrabi, Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran; Sh.Shahrabi@Shahed.ac.ir

Abstract

In distributed Exascale systems, the occurrence of a dynamic and interactive nature changes the workload of the system's computing elements. Because of this, the load balancer needs to collect information on the system state. Activating the load balancer increases the runtime of the scientific application. While analyzing the impact of the dynamic and interactive nature on the load balancer functionality, this paper also attempts to provide a mathematical definition for load balancer based on the concept of dynamic and interactive nature. This makes it possible to describe and examine the load balancer functionality by considering the impacts of the dynamic and interactive nature. As a result, decisions can be made on the behavior of the load balancer when a dynamic and interactive nature occurs. According to the mentioned operational function, this paper has analyzed the load balancer's behavior in processes with a dynamic and interactive nature. The introduced operational function for the load balancer in this paper enables the management element to separate the requests of processes with a dynamic and interactive nature and normal processes that leads to redistribution.

Keywords: distributed exascale computing system, load balancing, dynamic and interactive nature, runtime.

1. Introduction

One of the most important features of distributed Exascale systems, in comparison with traditional computing systems, is the concept of dynamic and interactive nature [1, 2, 3]. The occurrence of a dynamic and interactive nature in these types of computer systems is due to the lack of precise information on the nature of the scientific application and the executive structure to run it [4, 5, 6, 7]. To analyze the dynamic and interactive nature of the system and its impact on system's functionality, two patterns are usually used: the occurrence of the event and analyzing its impacts on the system and its constituent elements, or examining the process of the constituent elements' functionality and behavior and investigating that whether they are in line with the dynamic and interactive event or not. The most important impact and manifestation of the dynamic and interactive nature in distributed Exascale systems is on the system manager's functionality [2, 8, 9, 10]. In addition to being able to analyze and manage the dynamic and interactive nature, the system manager must also be able to make decisions on the effects of the dynamic and interactive nature on its own processes [2, 8].

Among the constituent elements of the system manager, the load balancer is one of the most important elements that is influenced by the occurrence of the dynamic and interactive nature [11]. One of the most important influences of the dynamic and interactive event is a change in the workload of systems that form the computing system [2, 12]. Due to the dynamic and interactive events, the system load can be changed in a way that the appropriate pattern (or patterns) for controlling the workload state may not be considered when designing the system [5, 6]. Based on the computing system's functionality as well as the scientific applications that are to be run by the computing system, the mechanisms or patterns used by the load balancer are normally determined and specified when it is designed [13, 14]. In distributed Exascale systems, due to the dynamic and interactive nature of computing processes and their answer structure, extracting the information on the application nature and the computing system functionality is not feasible [6]. Because of the lack of accurate information on the application nature and the system's functionality, in addition to normal runtime for redistribution activities, the load balancer requires some more time to perform activities related to establishing an appropriate answer structure to deal with the dynamic and interactive nature. This increase in the runtime of the load balancer activities will increase the runtime of the scientific application in the distributed Exascale system, which will violate the main objective of high performance computing systems (increased efficiency) [15].

To deal with this incident, the load balancer in distributed Exascale computing systems can use two strategies. In the first strategy, the load balancer does not influence the process of the dynamic and interactive event and allows it to occur. In this case, the load balancer manages the system workload by taking into account the system state after the occurrence of the event. In this method, the load balancer does not deal with the details of dynamic and interactive events; on the contrary, it requires to run more activities for redistribution [16]. In mechanisms designed according to this method, after the occurrence of the event, the load balancer creates an answer structure proportional to the system's workload after the event, and then, redistributes the load. The most important challenge of this approach is when the system workload follows a pattern that does not fit the mechanism (or mechanisms) used by the load balancer to redistribute it. This can be due to the fact that the occurrence of a dynamic and interactive nature has altered the system workload state and caused it not to be in accordance with the load balancing mechanism.

The second strategy is the functional and behavioral analysis of computing processes in distributed Exascale systems. In this method, the load balancer makes decisions on the computing processes, where the dynamic and interactive event is likely to happen, through examining the functional and behavioral trend of the computing processes as well as considering the triggers of dynamic and interactive behaviors.

Based on the analysis of the impact of the dynamic and interactive nature on the load balancer functionality, this paper introduces the function of load balancer functionality by taking into account the dynamic and interactive nature. It can be used as a benchmark for analyzing process requests. According to this function, it is possible to distinguish between the requests of processes with dynamic and interactive nature and normal requests that lead to redistribution. As a result, the load balancer can manage the process requests according to their nature. Thanks to this approach, the load balancer is able to analyze the process behavior and make decisions on the mechanisms used to redistribute the load based on the process behaviors.

2. *Related works*

Scientific applications have to be able to describe the activities of natural phenomena. They also need to be able to predict what will happen [5, 17]. Such computing applications require HPC systems, whose management element is able to support their requirements [5, 18]. The management element in distributed Exascale systems is able to execute scientific applications with higher complexities and face unpredictable requests [2, 8, 9, 19].

The purpose of using traditional computing systems was to examine a natural event for specific values in the shortest possible time. However, if we look at the nature of scientific and engineering applications that need Exascale systems, we will realize that the purpose of running these applications is to discover the rules governing a natural event [5]. One of the available solutions for HPC systems to support dynamic and interactive applications is to increase system scalability during runtime [20]. Paper [5] addresses the issue that how multiscale computing systems can optimally be used in current Petascale systems as well as HPC Exascale systems through increasing their capabilities.

DEEP project has developed a supercomputer architecture with a suitable software package and a series of massive optimization simulation applications. DEEP aims at providing scalability via using millions of processing cores to implement new scientific applications with Exascale-level computing performance [7, 21].

Exascale systems face different challenges such as Power, Extreme Concurrency, Limited Memory, Data Locality, and Resilience. In addition, as in HPC systems, where the resource management unit, especially the load balancing unit, plays a major role in the efficiency of these systems, it is also true for Exascale distributed systems and a proper load balancing mechanism is needed for effective implementation of the applications [22, 23].

In general, three main factors of load imbalance, sharing resources, and synchronization reduce the efficiency of parallel computations [24]. In [24], a framework is proposed for Exascale Multicore systems which takes a step towards solving these problems by presenting an optimal real-time method. Inspired by the theory of complex social networks, [24] suggests a new method for modeling the dynamic execution of an application is by constructing a dynamic application dependency graph.

In recent years, a large number of dynamic load balancing algorithms have been proposed [25, 26, 27]. These load balances use the migration of a work and its related data over the lifetime of the work [28]. Dynamic load balancing algorithms are divided into two categories: suitable to be uses in distributed Exascale systems or unsuitable. The majority of the proposed algorithms belong to the first category.

Considering the heterogeneity of the computing resources as well as the lag of communication link between the resources, a dynamic load balancing algorithm has been presented in [29]. This algorithm considers the workload migration as an important factor in calculating the load balancing cost. In [2], a dynamic distributed load balancing mechanism for Exascale computing systems is proposed. The proposed method covers the dynamic behavior of new issues. To estimate the overload of computing resources, this mechanism utilizes many parameters such as load migration and communication lag.

In [8], a resilient dashboard in the environment of load balancing algorithms is proposed to manage the load balancing process in real time. In this method, system's flexibility has been improved in order to adapt it to dynamic variables.

3. What Does Dynamic and Interactive Mean in Load Balancing?

The load balancer operates based on two spaces of resource attributes and process requirements [30, 31]. In its most general case, it must be able to implement equation 1 to perform its activities.

$$\mathit{Load\ Distribution}\colon [\mathit{Process}_{requirment}] \rightarrow [\mathit{Resource}_{Space}]\ \mathit{Therefore} \tag{1}$$

$$\mathit{Best}_{Load-Distribution} = \left[\begin{array}{c} \left[\nexists \mathrm{Process} \in \mathrm{HPC}_{\mathrm{Proces}} \overset{So\ Means}{\because} \ni \mathit{HPC}_{Process-Scheduling}\right] \\ \mathit{and} \\ [\mathit{Resource}_{Activity} = 100\%] \end{array}\right]$$

As seen in equation 1, the load balancer must be able to allocate the resources in the computing system to computing processes in such a way that: a) all computing resources in the system are at their maximum operating levels; b) the processes' time requirements are not violated and there is not a process that cannot be answered at the acceptable time due to being in the request queue [30, 31].

In distributed Exascale systems, both resource and process spaces can be changed over time [5, 10]. If the load balancer in a distributed Exascale system is operating based on the second strategy, based on a set of schemas, it should be able to make decisions on whether the occurrence of one of the three states of process creation, communication, and interaction will lead to a dynamic and interactive nature [10]. In load balancer's view, this means defining a set of indicators for analyzing the procedures of process activities and identifying processes that lead to dynamic and interactive occurrences, as well as analyzing the impacts of dynamic and interactive events on the system's workload state.

Since the system under study is a distributed Exascale system, the load balancer is implemented independently in each computing element, and it attempts to manage the processes in the local computing element [20].

If the St represents local and global processes at moment t in the local computing element, then the load balancer functionality can be described and formulated in the form of equation 2.

$$\text{Load Balancing}::<<S_t,\text{Process}_{\text{state}},\text{Resource}_{\text{state}}>,P_G,R_H,I_K,C,O,R> \tag{2}$$

Equation 2 tends to functionally describe the load balancer through studying the process behaviors and functionalities in distributed Exascale systems. This description would define a finite field on three spaces of $<S_t,\text{Process}_{\text{state}},\text{Resource}_{\text{state}}>$ by considering 5 characteristics for the process behavior and functionality and an R activity as well.

In equation 2, $\text{Process}_{\text{state}}$ represents the state of the global computing processes relative to the local computing element. This space indicates that what each global computing process requests from the local computing element. These requests can be in the form of a linked list. Given the possibility of defining the global activity, $\text{Process}_{\text{state}}$ space can be defined in terms of the concept of past performance and previous behaviors. If the distributed Exascale computing system is designed in a way that the $\text{Process}_{\text{state}}$ is created for each computing process from the moment the process is formed, this set will be initialized during the global activity which is being run on the system's computing elements and the process is part of it [21]. To initialize it, each process request of each computing element is stored in the form of a linked data structure. This data structure contains a section for storing the local system's state when answering the process, and another section for storing the process request from the local computing element.

In equation 2, $\text{Resource}_{\text{state}}$ represents the state of resources in the local system which are answering the requests of global computing processes. This space implies that every resource of the local computing element is capable of answering which requests. In order to be used to analyze the behavior and functionality of the resource and the process, $\text{Resource}_{\text{state}}$ space stores the request information as well as the pattern of answering the request in an array data structure. When a request is sent to the resource, the process owning the request stores the request information, including the request type, the process owning the request, the time and pattern of the creating the request, as well as the pattern used by the resource to answer the request [21]. The resource pattern for answering the request is a procedure that it takes to answer the request. Resource interactions with the process having the request, the time pattern allocated to the request, the constraints governing answering the request and the impact of answering the request, and most importantly, its influence on the resource state about the resource pattern for answering the request, are saved in the data structure of resource state by the process owing the resource.

In equation 2, P_G,R_H are the generating spaces of the processes and resources' behavior available in the local computing element. For each process and resource in the local computing element, a set of indices can be defined so that the process or resource state can be described based on them. Any changes in these indices will result in defining a new functionality or behavior for the process or resource. As e result, a reference set can be considered for all the possible functionalities and behaviors of the resource (or the process). R_H is a generating function which generates all the functionalities and behaviors of the resource reference set. P_G is a generating function which generates all the functionalities and behaviors of the process reference set.

In equation 2, I_K represents the generating space of the processes' dynamic and interactive nature. A set of states, which is a sub-set of the process functionalities and behaviors, can be considered for each process, and in case these states take place, dynamic and interactive nature will occur in the process. As stated in papers [10, 22], three states can be regarded for the dynamic and interactive nature.

In equation 2, C represents the influences that I_K states have on the states of P_G,R_H. For each element on I_K space, a mapping function can be defined which represents the impact of I_K space element on each element of P_G,R_H spaces. The defined mapping function may influence some of the elements of P_G,R_H spaces, or may not have any influences on them. Space C includes a set of the mentioned mappings.

In equation 2, space O is made up of two parts. The first part is for defining process descriptor indices, and the second one is for defining resource descriptor indices.

In this equation, R is the space for answering the process requests. Based on the mechanisms for process migration, resource discovery, changing the implementation priorities, and changing the resource functionality, the load balancer answers the process request depending on whether it is normal or dynamic and interactive. R describes the reaction activity in equation 3, which has been explained in section 1.2. Based on the schema extracted from the process behavior, this space answers the process request.

Transferring the processes that have caused the dynamic and interactive nature and disturbed the acceptable state of the load balancer, or changing the resource functionality, are two tools of answering the states which disturb the load balancer state [22]. These two strategies, and that which is used for what state, are determined by R.

4. Behavioral ExaLB

In distributed Exascale computing systems, both elements of process and resource can change the computing element state [1, 10, 32]. Changing the computing elements' state and their generating functions, or changing the resources' state and their generating functions will change the descriptor function of the computing element state. The change in the descriptor function will also change the function of the load balancer functionality.

To analyze the computing element's behavior and functionality, the analysis of the behavior and functionality of each of the two elements of the resource and process can be used. In this paper, the focus of the analysis is on the process. In distributed Exascale computing systems, the process has two types of behavior and functionality in the load balancer view.

The first type is normal behaviors, which are the changes in requirements and descriptor spaces of the process. Due to these behaviors, the load balancer answers the process requirements through the mechanisms of process migration, resource discovery, or changing the implementation priority [33, 34]. These requirements are taken into account when designing the answer structure. The mechanisms and patterns that are used by the load balancer, match these requirements, and when they occur, the load balancer has the mechanisms and patterns to manage them [34, 35].

The second type of process behaviors are dynamic and interactive [1, 2, 3]. They are not taken into account when designing the answer structure [5]. Their nature is such that decisions cannot be made on the pattern needed to deal with them until they occur. In equation 2, the generating space of dynamic and interactive behaviors, in load balancer's view, is I_K. As noted in [10], dynamic and interactive behaviors can raise from the process formation, communication, and interaction. The load balancer creates a graph which corresponds to the process request nature. The vertices of this graph are one of the five states of normal, formation, communication, not requiring the load balancer, or interaction. Each edge of this graph represents an event which has altered the process request nature in load balancer's view. iG graph can be shown in the form of i_G=(S,E), where S is the state based on which the process has activated or deactivated the load balancer, and E is an event which changes the process state and activates or deactivates the load balancer.

For every change in the state of the request nature graph relative to the load balancing element, the load balancer stores the information of the edge and the vertex in Y data structure. The information that involves transferring the process state into one of three states of formation, communication, and interaction is considered and the dynamic and interactive behavior trigger. For i_G graph, the load balancer can define IG matrix. IG matrix is defined on the space from Cartesian product of two generating spaces of R_H and P_G. The linear mapping operators defined on space C can be defined on IG matrix. According to the definition of space C, each mapping in space C maps each I_K element into each of two generating spaces of R_H and P_G based on equation 3.

$$\forall c \in C :: i_k \in I_K | \forall r_h \in R_H :: r_h \to r_h or\ r_h \to r'_h \ and\ \forall p_g \in P_G\, p_g \to p_g or\ p_g \to p'_g \qquad (3)$$

As seen in equation 3, each mapping in space C creates a linear transformation. Based on this linear transformation, each element of I_K space influences each element of R_H and P_G spaces and causes it to either become a new element or remain the same as before. In load balancer's view, this linear mapping makes sense when the dynamic and interactive behavior influences every process behavior or resource behavior, or the process or resource behavior in influenced by the dynamic and interactive behavior. If the dynamic and interactive behavior changes the process or resource behavior, the load balancer will seek to investigate whether the dynamic and interactive behavior is a stimulus or is itself the result of another behavior. To this end, if f is the defined stimulus on a linear transformation on space C, and $f(c) = 0$, then the transformation will be a dynamic and interactive behavior stimulus. $f(c) = 0$ means that the dynamic and interactive behavior stimulus aliquots behavior c.

5. Conclusion

In this paper, the dynamic and interactive nature and its impact on the load balancer functionality were discussed. Due to the occurrence of a dynamic and interactive nature in distributed Exascale systems, in addition to redistributing the load due to the occurrence of normal requests, the load balancer needs to be able to manage load balancing that results form dynamic and interactive requests as well. The nature of dynamic and interactive requests is different from the nature of normal requests. As a result, the mechanisms used to manage redistributions from dynamic and interactive requests will also vary. The nature of the dynamic and interactive requests is such that they transform the system and the load balancing state into an undefined state for the load balancer.

Consequently, the influences of the dynamic and interactive nature on the load balancer functionality need to be taken into account. Given the dynamic and interactive nature, the concept of global activity as well as the elements affecting the space defining the load balancer, this paper proposed the load balancer's operational function. This function describes the load balancer's functionality in both traditional computing systems and distributed Exascale systems. Redefining the function load balancer's functionality enables the load balancer to analyze the generating space of dynamic and interactive behaviors due to considering I_K space. This analysis enables the load balancer to, based on the presented model in this paper, make decisions on whether the process behavior will result in a dynamic and interactive nature or not. This will allow the load balancer to manage redistribution in distributed Exascale systems in accordance with the process behavior.

Reference

[1] Mousavi Khaneghah, E., Noorabad Ghahroodi, R., & Reyhani ShowkatAbad, A. (2018). A mathematical multi-dimensional mechanism to improve process migration efficiency in peer-to-peer computing environments. *Cogent Engineering, 5*(1), 1458434.

[2] Mirtaheri, S. L., & Grandinetti, L. (2017). Dynamic load balancing in distributed exascale computing systems. *Cluster Computing, 20*(4), 3677-3689.

[3] Barak, A., Drezner, Z., Levy, E., Lieber, M., & Shiloh, A. (2015). Resilient gossip algorithms for collecting online management information in exascale clusters. *Concurrency and Computation: Practice and Experience, 27*(17), 4797-4818.

[4] Reyle, C., Richard, J., Cambrésy, L., Deleuil, M., Pécontal, E., & Tresse, L. (2016). Perspectives in numerical astrophysics. *In the Annual meeting of the French Society of Astronomy and Astrophysics, SF2A-2016.*

[5] Alowayyed, S., Groen, D., Coveney, P. V., & Hoekstra, A. G. (2017). Multiscale computing in the exascale era. *Journal of Computational Science, 22,* 15-25.

[6] Innocenti, M. E., Johnson, A., Markidis, S., Amaya, J., Deca, J., Olshevsky, V., & Lapenta, G. (2017). Progress towards physics-based space weather forecasting with

exascale computing. *Advances in Engineering Software, 111,* 3-17.

[7] SEVENTH FRAMEWORK PROGRAMME Research Infrastructures. (n.d.). Retrieved from https://www.deep-projects.eu/images/materials/DEEP_ER_D1.1.pdf

[8] Mirtaheri, S. L., Fatemi, S. A., & Grandinetti, L. (2017). Adaptive Load Balancing Dashboard in Dynamic Distributed Systems. *Supercomputing Frontiers and Innovations, 4*(4), 34-49.

[9] Wang, K., Qiao, K., Sadooghi, I., Zhou, X., Li, T., Lang, M., & Raicu, I. (2016). Load-balanced and locality-aware scheduling for data-intensive workloads at extreme scales. *Concurrency and Computation: Practice and Experience, 28*(1), 70-94.

[10] Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N. (2018, February). Challenges of Process Migration to Support Distributed Exascale Computing Environment. In *Proceedings of the 2018 7th International Conference on Software and Computer Applications* (pp. 20-24). ACM.

[11] Amelina, N., Fradkov, A., Jiang, Y., & Vergados, D. J. (2015). Approximate consensus in stochastic networks with application to load balancing. *IEEE Transactions on Information Theory, 61*(4), 1739-1752.

[12] Klimentov, A., Buncic, P., De, K., Jha, S., Maeno, T., Mount, R., ... & Porter, R. J. (2015). Next generation workload management system for big data on heterogeneous distributed computing. In *Journal of Physics: Conference Series* (Vol. 608, No. 1, p. 012040). IOP Publishing.

[13] Hussain, H., Malik, S. U. R., Hameed, A., Khan, S. U., Bickler, G., Min-Allah, N., ... & Kolodziej, J. (2013). A survey on resource allocation in high performance distributed computing systems. *Parallel Computing, 39*(11), 709-736.

[14] Yang, C. T., Liu, J. C., Hsu, C. H., & Chou, W. L. (2014). On improvement of cloud virtual machine availability with virtualization fault tolerance mechanism. *The Journal of Supercomputing, 69*(3), 1103-1122.

[15] Kołodziej, J., Khan, S. U., Wang, L., Kisiel-Dorohinicki, M., Madani, S. A., Niewiadomska-Szynkiewicz, E., ... & Xu, C. Z. (2014). Security, energy, and performance-aware resource allocation mechanisms for computational grids. *Future Generation Computer Systems, 31,* 77-92.

[16] Skinner, B. F. (1953). *Science and human behavior* (No. 92904). Simon and Schuster.

[17] Fiore, S., Bakhouya, M., & Smari, W. W. (2018). On the road to exascale: Advances in High Performance Computing and Simulations—An overview and editorial. *Future Generation Computer Systems, 82,* 450-458. doi:10.1016/j.future.2018.01.034

[18] Straatsma, T. P., Antypas, K. B., & Williams, T. J. (2017). *Exascale Scientific Applications: Scalability and Performance Portability.* Chapman and Hall/CRC.

[19] Ghomi, E. J., Rahmani, A. M., & Qader, N. N. (2017). Load-balancing algorithms in cloud computing: a survey. *Journal of Network and Computer Applications, 88,* 50-71.

[20] Abraham, E., Bekas, C., Brandic, I., Genaim, S., Johnsen, E. B., Kondov, I., ... & Streit, A. (2015, September). Preparing HPC applications for exascale: Challenges and recommendations. In *Network-Based Information Systems (NBiS), 2015 18th International Conference on* (pp. 401-406). IEEE.

[21] Eicker, N., Lippert, T., Moschny, T., & Suarez, E. (2013, October). The deep

project-pursuing cluster-computing in the many-core era. In *2013 42nd International Conference on Parallel Processing (ICPP)* (pp. 885-892). IEEE.

[22] Singh, A., Juneja, D., & Malhotra, M. (2015). Autonomous agent based load balancing algorithm in cloud computing. *Procedia Computer Science, 45,* 832-841.

[23] Milani, A. S., & Navimipour, N. J. (2016). Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends. *Journal of Network and Computer Applications, 71,* 86-98.

[24] Xiao, Y., Xue, Y., Nazarian, S., & Bogdan, P. (2017, November). A load balancing inspired optimization framework for exascale multicore systems: a complex networks approach. In *Proceedings of the 36th International Conference on Computer-Aided Design* (pp. 217-224). IEEE Press.

[25] Márquez, C., César, E., & Sorribes, J. (2015, August). Graph-based automatic dynamic load balancing for HPC agent-based simulations. In *European Conference on Parallel Processing* (pp. 405-416). Springer, Cham.

[26] Marathe, A., Bailey, P. E., Lowenthal, D. K., Rountree, B., Schulz, M., & de Supinski, B. R. (2015, July). A run-time system for power-constrained HPC applications. In *International conference on high performance computing* (pp. 394-408). Springer, Cham.

[27] Panchal, B., Smaranika, P. (2018) An Efficient Dynamic Load Balancing Algorithm Using Machine Learning Technique in Cloud Environment. *International Journal of Scientific Research in Science, Engineering and Technology, 4(4),* 1184-1186.

[28] Jeannot, E., Mercier, G., & Tessier, F. (2016, November). Topology and affinity aware hierarchical and distributed load-balancing in Charm++. In *International Workshop on Communication Optimizations in HPC (COMHPC)* (pp. 63-72). IEEE.

[29] Soltani, N., & Sharifi, M. (2014). A Load Balancing Algorithm Based on Replication and Movement af Data Items for Dynamic Structured P2P Systems. *International Journal of Peer to Peer Networks, 5*(3), 15-32. doi:10.5121/ijp2p.2014.5302

[30] Sreenivas, V., Prathap, M., & Kemal, M. (2014, February). Load balancing techniques: Major challenge in Cloud Computing-a systematic review. In *2014 International Conference on Electronics and Communication Systems (ICECS)* (pp. 1-6). IEEE.

[31] Rahman, M., Iqbal, S., & Gao, J. (2014, April). Load balancer as a service in cloud computing. In *2014 IEEE 8th international symposium on service oriented system engineering (SOSE)* (pp. 204-211). IEEE.

[32] Singh, P., Baaga, P., & Gupta, S. (2016). Assorted Load Balancing Algorithms in Cloud Computing: A Survey. *International Journal of Computer Applications, 143(7).*

[33] Navimipour, N. J., & Milani, F. S. (2015). A comprehensive study of the resource discovery techniques in peer-to-peer networks. *Peer-to-Peer Networking and Applications, 8*(3), 474-492.

[34] Rathore, N., & Chana, I. (2014). Load balancing and job migration techniques in grid: a survey of recent trends. *Wireless personal communications, 79*(3), 2089-2125.

Submitted 27.07.2018
Accepted 16.10.2018