Azerbaijan Journal of High Performance Computing, Vol 1, Issue 2, 2018, pp. 168-178
https://doi.org/10.32010/26166127.2018.1.2.168.178

Challenges of Resource Discovery to Support Distributed Exascale Computing Environment

Elham Adibi, Ehsan Mousavi Khaneghah

Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran; Elham.Adibi@shahed.ac.ir, EMousavi@shahed.ac.ir

*Correspondence: Elham Adibi, Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran, Elham.Adibi@shahed.ac.ir

Abstract

The resource discovery management unit (RD) in distributed Exascale systems needs to be able to manage the occurrence of dynamic and interactive events in the requesting process when running activities related to RD. The occurrence of dynamic and interactive events in the requesting process causes some challenges in the execution trend of activities related to RD. The trend of execution activities related to RD will fail if there is no management and control on the occurrence of dynamic and interactive events. This paper first explores and describes the concept of dynamic and interactive nature in distributed Exascale systems. In addition, it attempts to show the influences of occurrence the dynamic and interactive events in the computing elements requesting the resource as well as on RD functionality. In addition, this paper aims at describing the challenges of RD to be used in distributed Exascale systems. To achieve this, changes need to be made in the generating space and RD functionality. Taking into account the solutions presented in this paper, RD has the potential to be used in distributed Exascale systems by remaining compatible with traditional computing systems.

Keywords: exascale computing, resource discovery, dynamic and interactive nature.

1. Introduction

In traditional computing systems such as cluster systems, the nature of the system is such that the computing element is not added to or removed from the system during runtime. Therefore, the number of computing elements is constant in the system [1]. In these types of computing systems, the system is normally running a scientific and engineering application. Based on the information received from the user or based on the data structure of processes' initialization, the resource management unit in cluster systems has detailed information on the state of processes and their requirements. The implementation of resource management unit in cluster systems is carried out in a centralized manner. This type of implementation causes resource management unit to have a clear view of the features of resources in the system as well as the nature of computational processes being run in the system. If a process needs a resource, the centralized manager will try to find the most suitable resource in the system [2].

By the increased need for high computing and processing power and a change in the nature of scientific applications, we witness the formation of Exascale distributed computing systems [3]. In distributed Exascale computing systems, the nature of scientific applications is the same as the nature of scientific applications being run on cluster computing systems. The difference between the two systems is that a distributed Exascale system can execute more than one scientific application. Implementing more than one scientific application will make it necessary to define the concept of the global activity [4]. The global activity is defined for each scientific application that requires a high performance computing system.

In the global activity state, there may be states in which a local operating system cannot answer the requests of a process (which is part of the global activity). In this case, RD will be called [5]. The task of this unit is to find a computing element that is capable of meeting the needs of computational process. The overall mechanism used in distributed computing systems is based on the fact that the process requests access to a resource that may not exist in the local computing system. RD vices the responsibility of the requesting process and finds a computing component beyond the system limits through scalability or linking with other systems. This unit can answer the request of the process. Activities related to RD should be performed in the shortest possible time as much as possible [6, 7].

In scientific and engineering applications that need distributed Exascale systems, the nature of the application is such that its requirements change over the runtime. This creates the concept of dynamic and interactive nature in computational processes. The dynamic and interactive nature means that during executing a global activity, due to the occurrence of an event, the process requirement will change or a new requirement will be created. The reason is establishing links and interactions between computational processes inside and outside the system. The created requirements have not been taken into account at the time of designing the computing system [8]. Given the above explanations, it can be concluded that all the processes' requirements to continue the trend of running a global activity are not clear at the beginning of the application execution, and as a result, the need for RD will become more important.

In addition to performing traditional activities related to managing and executing computational processes, based on flexible structures, the resource management unit in distributed Exascacle computing systems should be able to create an answering structure by considering the dynamic nature and interactive nature of computational processes. Furthermore, it should be able to expand the computing system to respond to new requests [9].

From resource management's perspective, the occurrence of a dynamic and interactive nature in computational processes means creating a request or a state which the system has no facilities to answer it. Through the concept of scalability (or connections with other systems), the resource management unit typically tries to find a computing element that can answer the request by using the available resources in this element [10]. Because of this, the role of RD in such systems has higher executive importance compared to traditional distributed computing systems. In distributed Exascale computing systems, a dynamic and interactive nature may occur in computational processes at any moment of the execution trend of the scientific application. This is also true when running activities related to RD. In these types of computing systems, some states may occur in which RD may be activated for any reasons. Once it is activated, the state of the computing elements influencing the execution trend of activities related to RD may face a dynamic and interactive nature. In addition of examining the concept of the dynamic and interactive nature in processes requiring Exascale systems, this paper has investigated and analyzed the effects of this concept on RD functionality. In this regard, it examines the challenges of RD in distributed Exascale systems.

2. Related works

Applications that require high processing power or high data volumes, such as weather forecasting, financial analysis, and other calculations like these, often cannot be done by a machine. Peer-to-peer systems try to help performing such applications by collecting and collaborating between hardware and software resources [11, 12]. In fact, peer-to-peer computing systems provide an infrastructure to gain access to widespread computing capabilities [13].

The goal of peer-to-peer computing systems is to create co-operations among resources to solve a problem requiring high processing power. This operation is carried out in a dynamic environment which is built by several virtual organizations [14]. Scientific and engineering applications that require HPC systems are such that all the features required by the application cannot be extracted at the start-up. Also, there is no accurate view on the nature of process requirements during system's lifetime. For this reason, computing systems are required that can answer to the new requirements of the processes over the system's lifetime through scalability and connections with other systems. Paper [15] explains the most important features of Peer-to-Peer computing systems.

The reason for scalability is answering a process that cannot be answered by the local operating system. This operation is done by establishing connections with other computing elements and working with them [16]. The request must be answered by finding a computing element which can also meet the process requirements. In distributed computing systems, this activity is known as resource discovery (RD). RD finds resource that is able to answer the request of a process by observing the rights of gaining access to a resource. In papers [5, 17], RD and how it operates are explained.

Resource discovery is part of the resource management system. The implementation of the system's resource management unit and all its sub-units means increasing the application's runtime. Hence, RD-related activities should also be carried out at the appropriate time by observing the limitations defined in the request [18].

Due to the decentralized nature of distributed computing systems, the processes in the system start a computing activity. Each computational process may have some requests that cannot be answered by the local operating system. In this case, load balancer or RD will be activated. A set of processes that perform a computing activity with each other is called a global activity. Due to the dynamic and interactive nature of processes participating in distributed Exascale computing systems, more sophisticated management structures than traditional computing systems are required to be created. The resource management system should be able to manage such computing systems by employing flexible and dynamic structures. In addition to describing the global activity, paper [8] has raised part of the challenges of resource management system in distributed Exascale computing systems.

Paper [19] proposes a two-phase mechanism for performing activities related to RD. This method is capable of being used in Peer-to-Peer computing systems, taking into account the scalability and dynamic features of such systems. There are some resources in Peer-to-Peer computing systems whose features change over time. Therefore, in order to discover an appropriate resource, in addition to considering static properties, mechanisms need to be used to consider changes made in the system. The method introduced in this paper is to determine a weight for each static and dynamic resource in the system. If a user asks for a resource, the computing element, as the super peer, finds resources that are capable of answering the request. Resources are ranked based on the defined mechanisms, and the most appropriate resource is assigned to the requestor.

Paper [20] has proposed a framework for RD in response to dynamic requests in unstructured Peer-to-Peer systems. In this paper, five components are defined which are in charge of performing tasks such as information gathering, decision making, describing machines' states, finding a resource in response to requests in both normal and dynamic situations, as well as load balancer. If the computational process has a dynamic request, it will send its request to other computing elements. Each computing element that receives the request, measures its state based on a mechanism. The mechanism accurately estimates the machine's state before and after answering the request. As a result, the computing element makes decisions on its ability and inability to answer the request. Evaluations have shown that RD-related activities by this framework can successfully be performed at a high rate.

3. The influence of dynamic and interactive nature of computational processes on resource discovery

In traditional computing systems such as cluster systems, grid systems and Peer-to-Peer computing systems, the system is running a scientific application that examines the rules governing a natural event. Running the application is dependent on a computing activity or a specific state. This will allow the system designer to have accurate information on answering structures when designing the system. In cluster systems, resource management unit provides the application with response structures at the time of system designing, and in grid and peer-to-peer computing systems at the time of system implementation. However, scientific applications which require distributed Exascale computing systems, the nature of the application is such that the information on answering structures cannot be extracted when the system is designed. This is due to the occurrence of a dynamic and interactive nature in the processes of these types of scientific applications [8, 9].

The dynamic and interactive nature of the computational processes of distributed Exascale computing systems is due to the occurrence of one (or more than one) of the following three states: a) There is a process in the system that creates a new process which is not defined at the time of designing the computing system. This results from the formation of a new concept or a new variable in the scientific and engineering application that is being run by the computing system. b) The process has inter-process connections and interactions with another process within the computing system, which is not defined at the time of designing the computing system. This results from the formation of a new connection in the scientific and engineering application that is being run by the computing system. c) The process has inter-process connections and interactions with another process outside the computing system, which are not defined at the time of system designing. This is due to defining a new variable in the scientific and engineering application that is being run by the computing system [21, 22].

Resource discovery, as part of the resource management unit, is influenced by the occurrence of a dynamic and interactive nature in computational processes. This influence on the function of RD can be discussed in two areas. In the first area, there is a process in the system that does not have access to a resource which is outside the system. If the local operating system is not able to respond, RD will be called [23]. The occurrence of dynamic and interactive events in computational processes increases the frequency of calling RD. In the second area, RD is activated in one way or the other, and this unit will be provided with <Request, Limitation> at the moment of activation. The basic activity of RD is to find a resource in accordance with the request of the process beyond the system limits. RD is looking for the requested computing element by considering the constraints determined by the requesting process.

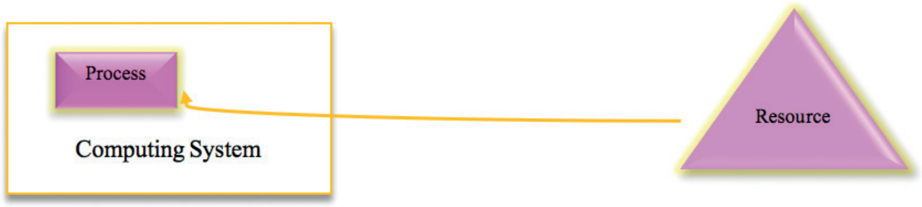

Figure 1. RD trend in computing systems

As seen in Fig.1, after receiving <Request, Limitation>, RD attempts to quit the system and starts searching in the system environment. RD is activated at t = Alpha and finds the resource at t = Beta. Also, allocates the resource to the requesting process at t = Omega. In this case, the process may become dynamic and interactive at any time in [Alpha, Omega] period. The influence set can be defined for each RD trend, which are processes in the system that affect <Request, Limitation>. The dynamic and interactive nature can occur in each of the processes of the Influence set during this timeslot. In this situation, the space that is used by RD to search for a resource is a two-dimensional space in the form of <Request, Limitation> that may be different from the system's actual space. This difference arises from the occurrence of a dynamic and interactive event and its influence on RD functionality. In this case, RD may seek to find a resource with the conditions mentioned in Limitation, which does not fulfill the request of the process. The influence of the dynamic and interactive nature on RD functionality is called the View Influence. As seen in Fig.1, a dynamic and interactive nature may occur in the process owning the requested resource during [Beta, Omega] timeslot. In this case, the functionality and nature of the resource found by RD differs from the functionality and the actual nature of the process requested a resource. This difference may be such that the computing element containing the resource fails to satisfy the limitations requested by the process requesting the resource. Such an influence by the dynamic and interactive event on RD functionality is called "backward influence".

As shown in Fig.1, the resource found by RD is located outside the system's limits. Hence, the computing element containing the resource has no obligations to continue the response procedure. During the [Beta, Omega] timeslot, when t ≠ Omega, the computing element which contains the requested resource may quit the response procedure due to the system's dynamic nature. In this situation, RD has failed to successfully complete its related activities. All the activities performed from t = Alpha until t = Failure are among the time wasted by RD. This influence is considered as "Time Influence".

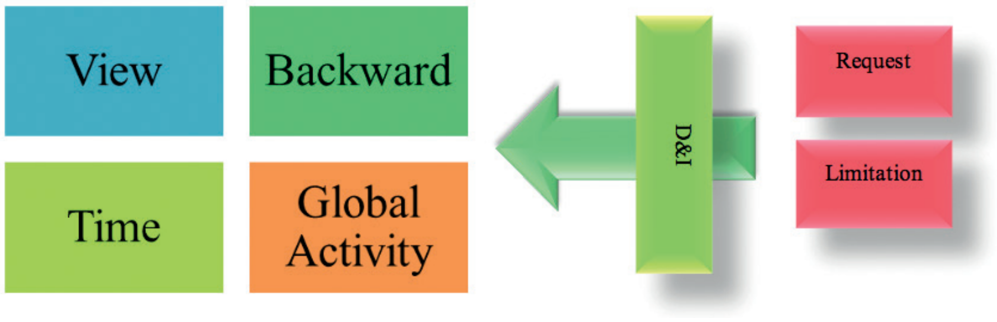

Figure 2. the influence of the dynamic and interactive nature on RD functionality.

In traditional computing systems, the process is the central component of doing management activities. However, in distributed Exascale computing systems, due to the dynamic and interactive nature of computational processes, the central component is the global activity. In these systems, the constituent units of the resource management system are controlling the global activity in order to implement it in the shortest possible time. The occurrence of a dynamic and interactive event in computational processes will change the global activity functionality. By changing the attributes of the global activity, RD needs to consider new conditions or to support new features. This influence of the dynamic and interactive nature on RD functionality is called "Global Activity Influence". Fig.2 shows the influence of the dynamic and interactive nature of the computational processes on RD functionality.

As can be seen in Fig.2, regardless of the type of the dynamic and interactive event, its occurrence will result in one of the influences of Global activity, Time, Backward, or View in the system. In traditional computing systems, RD cannot manage these four influences. Given the space generator of RD and how it works, it is impossible to redefine and describe these four influences in some way or the other. Furthermore, the operators described on the spaces that describe RD are not capable of managing and controlling the events leading to the formation of the four influences or their results.

4. Proposition

In traditional computing systems, RD is described in the form of <Request, Resource, scalability, permission, <finding >. In these systems, the task of RD is to find a resource. To do this, it finds a computing element outside the system which can answer the request of the process via sharing resources [24]. Two key components in activities related to RD in traditional computing systems are in the form of <Process, Resource>. RD management unit should manage the process and the resource establishes a link between the <Process, Resource> [21, 22] so that it can answer the request of the process. Due to the presence of the resource and its influence on activities related to RD, it is necessary to consider the limitations governing the resource. The task of RD is to map f (RD) : <Request, Limitation> ⊠ <Answer, Permission>. In traditional computing systems, Processstate does not change from the time the RD management unit is activated until the request of the process is answered. Lack of changes in Processstate means no changes in the requirements and constraints of the request of the process [25]. Additionally, in this type of computing system, Resource and Permission features will not change from the moment the RD management unit is called until the resource utilization is completed by the process.

Since the activities related to RD in traditional computing systems are based on a two-dimensional space of request and resource, this space lacks the ability to consider the concept of global activity. Global activity is a linked data structure consists of a set of processes and resources allocated to processes in various computing elements implementing them [26]. This definition cannot be stored in any of the two main spaces of resource and requests. As a result, the traditional RD is unable to define and understand the effects of Global Activity.

Defining the request space in traditional RD is such that after receiving the request, it does not re-examine the request during the answering procedure and updating the limitation space. Because of this, RD cannot redefine the influences of View based on its generating spaces. Due to this inability, RD lacks operators to deal with dynamic and interactive events that lead to the creation of View. The finding operator in traditional RD has no connections with the current state of the requesting process as well as the Limitation space. The reason is that the Limitation space in the functional function of RD is created and initialized only when RD starts the activity.

The concept of permission means obtaining permission to gain access to the resource found by RD [27]. After obtaining access rights to the resource, RD does not store any kind of information on the resource state. In traditional computing systems, the state of the computing element containing the resource is only examined at the time of discovery and no information on the state of the process owning the resource is stored. This also applies to the requesting computing element. The reason is defining the RD based on the spaces of <Request, Resource> and defining the Permission operator only at the time of resource discovery.

The nature of defining RD is such that there is no item in space generator or RD operators to guarantee performing and the continuing its related activity. This is more complex in Exascale distributed systems due to the different nature of the computational processes. Because the operations of permission, finding, and scalability concepts may be correct, but the dynamic and interactive event happens in the computational process from when the RD activity starts up until the resource is allocated to the requesting process. The occurrence of the dynamic and interactive nature in computational processes does not affect the operation of any of the above concepts, but causes the computing element containing the resource to leave the answering procedure or changes the limitations governing the request.

5. Argument

The challenges for RD to manage the dynamic and interactive events in distributed Exascale systems are due to: a) the inability to redefine the four noted influences on RD generating spaces, and b) the inability of this unit's operators to control and manage these influences. To manage and control the effects of Global Activity, RD requires to change the pivotal components of <Process, Resource>. To manage this influence, resource and request sets need to be added as the new generating space of RD. On the other hand, the request of the process is part of a global activity that is created by a process or (in particular cases, more than one process) which is a member of the global activity. Therefore, the request can be removed from the generating space of RD. The task of RD in traditional computing systems is to find a resource. However, in distributed Exascale computing systems, the main task of RD is to create an answering structure based on the resources out of the system. Hence, RD needs to define operators in order to create a new answering structure. Now, RD has to analyze the nature of the request to create a new answering structure so that it can make decisions on forming a new answering structure based on the nature of the request as well as its governing limitations. The new answering structure implicitly includes the RD trend.

To manage and control View challenges, RD needs to define the mechanisms to link with the requesting process and considering the Processstate. From the RD's perspective, there should be some mechanisms to update the Limitation state of the requesting process and to analyze the state of the requesting process. In addition to taking into account Processstate, RD should also be able to expand the finding space. In traditional computing systems, the concept of finding in RD only emphasizes on finding the resource. However, the resource management system in distributed Exascale computing systems must be able to definitely answer the request of the process through the answering structure created by RD. These capabilities can be presented in the form of the mechanisms used by RD to manage the failure. Moreover, the expansion of the finding operator implies that the request is adaptable to the resources examined by RD. In adaptability, the requesting process may request access to a resource that RD cannot find a computing element that has all the features requested by the process. Among the resources examined by RD, there may be one or more resources that have the highest degree of adaptability to the request of the process. As it is noted, the occurrence of a dynamic and interactive nature in the requesting process and the occurrence of the View change cause changes in the Requeststate as well as the Processstate. In this case, since RD tries to collect the information on the computing elements being examined, using the concept of to extend the finding operator can reduce the probability of the failure of activities related to RD.

To manage and control backward challenges, RD should have mechanisms to check the state of the process containing the resource. From RD viewpoint, this means defining a system that includes three components of the requesting process, the process owner requested resource and RD itself. In this system, two activities of add or remove are defined for each component, and each activity must be performed in such a way that information on the activity can be available to other components. If RD can support the system, then changing the state of the process owner requested resource will change the system state, and consequently, the activity related to RD. This system is active until the end of the answering procedure. In addition, based on the data structure, RD must be able to have information on the resources examined during the RD trend. This information helps to determine the priority levels for RD requests by extending the finding operator and considering the concept of Answerable and True as the sub-sections of the finding operator.

Resource discovery in distributed Exascale computing systems should be able to efficiently guarantee performing the relevant activities and not failing in the RD trend. Due to the uncertain nature of distributed systems, providing such a guarantee is very difficult. Therefore, RD in these types of systems should be able to define the concept of failure and failure management. Failure is not defined in the indicators defining RD in modern computing systems. Hence, RD needs to be able to define failure based on the generating space that describes RD to manage Time influences.

6. Conclusion

The task of RD is to find the resource that is requested by the process that outside of the system. This unit must be able to resolve the challenges resulted from the occurrence of the dynamic and interactive event in the requesting process. To manage and control these challenges, RD must be able to extend its pivotal activity from finding a resource to creating answering structure. This change in RD manager's functionality alters the generating space of RD from a five-dimensional resource-based space into a ten-dimensional global activity-based space. Because of this change in the dimensions of the generating space, in addition to considering the Processstate, RD forms an answering structure. Moreover, it can take into account two concepts of request similarity, and controlling and managing the request failure. Extending the generating and functional space of RD makes it necessary to consider several separate sections for RD. In addition to supporting the normal activities related to RD, it should be able to support the dynamic and interactive events in the computational processes based on at least two matching and exploring sections.

References

[1] Barak, A., La'adan, O., & Shiloh, A. (1999). Scalable cluster computing with MOSIX for Linux. *Proc. 5-th Annual Linux Expo, 100.*

[2] Verma, A., Pedrosa, L., Korupolu, M., Oppenheimer, D., Tune, E., & Wilkes, J. (2015, April). Large-scale cluster management at Google with Borg. In *Proceedings of the Tenth European Conference on Computer Systems* (p. 18). ACM.

[3] Reed, D. A., & Dongarra, J. (2015). Exascale computing and big data. *Communications of the ACM, 58*(7), 56-68.

[4] Khaneghah, E. M. (2017). *U.S. Patent No. 9,613,312.* Washington, DC: U.S. Patent and Trademark Office.

[5] Zarrin, J., Aguiar, R. L., & Barraca, J. P. (2018). Resource discovery for distributed computing systems: A comprehensive survey. *Journal of Parallel and Distributed Computing, 113,* 127-166.

[6] Ghebleh, R., & Ghaffari, A. (2017). A Multi-criteria Method for Resource Discovery in Distributed Systems Using Deductive Fuzzy System. *International Journal of Fuzzy Systems, 19(6),* 1829-1839.

[7] Arab, M. N., Mirtaheri, S. L., Khaneghah, E. M., Sharifi, M., & Mohammadkhani, M. (2011, September). Improving learning-based request forwarding in resource discovery through load-awareness. In *International Conference on Data Management in Grid and P2P Systems* (pp. 73-82). Springer, Berlin, Heidelberg.

[8] 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.

[9] Mousavi Khaneghah, E., Mirtaheri, S. L., Sharifi, M., & Minaei Bidgoli, B. (2014). Modeling and analysis of access transparency and scalability in P2P distributed systems. *International Journal o*f Communication Systems, 27(10), 2190-2214.

[10] Silberschatz, A., Galvin, P. B., & Gagne, G. (2014). *Operating system concepts essentials.* John Wiley & Sons, Inc..

[11] Brömmel, D., Suarez, E., Orth, B., Graf, S., Detert, U., Pleiter, D., ... & Lippert, T. (2014). Paving the road towards pre-Exascale supercomputing. In *NIC Symposium 2014* (No. FZJ-2014-01327). Jülich Supercomputing Center.

[12] Mallon, A. D., Lippert, T., Beltran, V., Affinito, F., Jaure, S., Merx, H., ... & Eick-

er, N. (2013). Programming model and application porting to the Dynamical Exascale Entry Platform (DEEP). In *Proceedings of the Exascale Applications and Software Conference, Edinburgh, Scotland, UK.*

[13] Pouya, I., Pronk, S., Lundborg, M., & Lindahl, E. (2017). Copernicus, a hybrid dataflow and peer-to-peer scientific computing platform for efficient large-scale ensemble sampling. *Future Generation Computer Systems, 71,* 18-31.

[14] Kaur, K., & Rai, A. K. (2014). A comparative analysis: Grid, cluster and cloud computing. *International Journal of Advanced Research in Computer and Communication Engineering, 3*(3), 5730-5734.

[15] Khaneghah, E. M., & Sharifi, M. (2014). AMRC: an algebraic model for reconfiguration of high performance cluster computing systems at runtime. *The Journal of Supercomputing, 67*(1), 1-30.

[16] Khaneghah, E. M., & Ghoreishi, S. A. (2017, June). CGUW: A system software for heterogeneous IPC mechanism in grid computing environments. In *2017 International Conference on Engineering, Technology and Innovation (ICE/ITMC)* (pp. 58-62). IEEE.

[17] 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.

[18] Qureshi, M. B., Dehnavi, M. M., Min-Allah, N., Qureshi, M. S., Hussain, H., Rentifis, I., ... & Zomaya, A. Y. (2014). Survey on grid resource allocation mechanisms. *Journal of Grid Computing, 12*(2), 399-441.

[19] Kaur, M., & Kadam, S. S. (2017). Discovery of resources using MADM approaches for parallel and distributed computing. *International Journal Engineering science and technology, 20*(3), 1013-1024.

[20] Mirtaheri, S. L., & Sharifi, M. (2014). An efficient resource discovery framework for pure unstructured peer-to-peer systems. *Computer Networks, 59,* 213-226.

[21] Kashyian, M., Mirtaheri, S. L., & Khaneghah, E. M. (2008, September). Portable inter process communication programming. In *The Second International Conference on Advanced Engineering Computing and Applications in Sciences, 2008. ADVCOMP'08* (pp. 181-186). IEEE.

[22] Sharifi, M., Mirtaheri, S. L., & Khaneghah, E. M. (2010). A dynamic framework for integrated management of all types of resources in P2P systems. *The Journal of Supercomputing, 52*(2), 149-170.

[23] Wu, J. (2017). *Distributed system design.* CRC press.

[24] Bisbee, S. F., Moskowitz, J. J., Becker, K. F., Peterson, E. K., & Twaddell, G. W. (2017). *U.S. Patent Application No. 13/369,112.*

[25] 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.

[26] Hesselink, L., Rizal, D., & Bjornson, E. S. (2015). *U.S. Patent No. 9,191,443.* Washington, DC: U.S. Patent and Trademark Office.

Submitted 16.06.2018
Accepted 18.10.2018