Azerbaijan Journal of High Performance Computing, Vol 1, Issue 1, 2018, pp. 20-41
https://doi.org/10.32010/26166127.2018.1.1.20.41

ExaMig Matrix: Process Migration based on Matrix Definition of Selecting Destination in Distributed Exascale Environments

Ehsan Mousavi Khaneghah[1], Amirhosein Reyhani ShowkatAbad[1], Nosratollah Shadnoush[2], Nigar Ismayilova[3], Reyhaneh Noorabad Ghahroodi[1], Elviz Ismayilov[4], Mohammad Saeed Nabati Saravani[1], Fatemeh Taheri Sarraf[1], Ali Soveizi[1]

[1] Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran, EMousavi@Shahed.ac.ir, Amirhoseinreyhani75@gmail.com, RNoorabad@hotmail.com, M.saeednabati@yahoo.com, yasaman.taheri76@yahoo.com
[2] Department of Management, Central Branch, Islamic Azad University, Tehran, Iran
[3] High Performance Computing Research Advance Center, Department of General and Applied Mathematics, Azerbaijan State Oil and Industry University, Baku, Azerbaijan, nigar.ismailova@asoiu.edu.az
[4] Department of General and Applied Mathematics, Azerbaijan State Oil and Industry University, Baku, Azerbaijan, elviz.ismailov@asoiu.edu.az,

*Correspondence: Ehsan Mousavi Khaneghah, Department of Computer Engineering, Faculty of Engineering, Shahed University, Tehran, Iran, EMousavi@Shahed.ac.ir

Abstract

In traditional computing system, load balancer, interim selecting the process, determine the destination computing node based on describing Indicators process status. In distributed Exascale computing system, due to the possibility of occurrence of a dynamic and interactive nature in execution time, it is possible. That the chosen destination computing node affected with dynamic and interactive nature so cannot be considered as a destination in process migration. This paper, by changing management approach in process migration. Consider process as an abstract element on the target computing node and calculates the impact of the factors the parameters affecting the process.
Considering the above factors make process migration manager able to create sets of computational node that can be considered as destination computing node.
In the event of a dynamic and interactive nature, in each element of the set, the process migration management, consider the effects of the factors affecting the activity of the process management and then re-weighs the computing element which make the above set. Using this mechanism allow the process migration management in case of dynamic and interactive nature occurrence in destination able to decide about changing on global activity execution so it is not necessary to recall load balancer manager in order to choose destination computing node. These subject louds to decrease execution time of process migration activity in distributed Exascale computing system.

Keywords: Process Migration, Distributed Exascale Computing Systems, ExaMig Matrix Mechanism, Dynamic and interactive Events

1. *Introduction*

The process migration has a task, based on the information received from the load distribution node, try to transfer the program code space, and also and the data space of a specified selected process [1]. In traditional computing systems, the process migration does not have duty to select the process and also the destination computing node [2]. The reason is the information consistency of the load balancer and the timing of the [implementation of] migration activities. In traditional computing systems, load balancer collect the information from the state of the computing node, especially the two computational node of source and destination, during the activities related to process migration is valid and there are no events during the above time which leads to a change in the status of the system and the effective nodes in the process migration execution activities [3, 4].

In Distributed Exascale computing systems, due to the definition of the concept of dynamic and interactive nature, it may leads that the system status descriptor be changed in the period of decision making about the status of the system and the execution of activities related to process migration [5, 6, 7, 8, 9]. In such a case, the information that the process migration is based on, may be invalid [5, 6]. Invalidity of information will affect the two concept of the transmitted node as well as the computing node of the destination. In this case, one of two policies ignoring the changes in the scope of the activity of the process migration or the changing the executer policy [10, 11].

In the policy of ignoring the changes, the process migration, regardless of dynamic and interactive event and also its effects on the system, proceed to continue the process migration trend [5, 12, 13], one of the failure state or end of occurred action. Then the node that has the ability to evaluate the system state descriptor indicators is to examine system. In traditional computing systems, as usual, load balancer has the ability to define and evaluate sets of indicators that describe the status of the system. This node may base on the new system status, attempts to undo changes or redistribute in system or confirm changes. The main advantage of this policy is the ease of implementation. On the contrary, the system may enter into situation that contradict the nature of the functioning of the computing systems [14, 15, 16].

Changing the executer policy, as respects the processor node is the process migration, so the procedure of review and implementation from the moment of calling the process migration, from load balancer delegate to process migration. In such a situation, the process migration should be able to decide based on a set of indicators on which process has the ability to be selected as a candidates of process migration, also this node should be able to decide what indicators should the destination machine have to satisfy both the condition of the migrated processes and the condition of the effects of the dynamic and interactive events on process the system and the condition of the time [17, 18, 19]. In changing the executer policy, two patterns can be used for the process migration to determine the general policy of identifying the two concepts mentioned above. In the first policy, each node of migration, and consequently each node of the beneficiary is analyzed by the process migration and decided upon it. However, this policy makes the process migration take careful decisions correct manageability of dynamic and interactive events. But on the contrary, requires high execution time. In this case policy, definition of a set of indicators and making decision based on these indicators. In second policy, it should be possible to describe the status of the nodes that influencing decision and the decision trend is based on a mathematical model, and based on the appropriate model with the status of the two nodes, the factors affecting the decision and decision process will be decided [5, 20].

In this article, by using matrix algebra, a mathematical model for describing the destination machines and the decision making process for selecting the destination machine in the process migration, is presented. This mathematical model provides the capability for process migration that can describe the effective indicators on the choice of destination machine and also redefine the decision making process based on the indicators. Also this mathematical model has the capability to describe the relationship between the above mentioned indicators.

2. Related Work

In traditional computing system, the system manager uses three different patterns to respond requests for processes [21, 22]. In the first pattern, the central element for answering the process requests is based on the resource discovery [22, 23]. In this method, resource discovery attempts to find resources for desired process outside the computing system. In the second pattern, the central element is the load balancer [23, 2]. In this way, the load balancer based on a set, called the Recourse State, describes the state of the computing node. Each request for the load balancer should be able to be expressed on the basis of a set of indices that are similar to the type of Resource state indices. When processes have an access request to a resource or resources in the local computing system that cannot meet the constraints required by the process to continue the process execution, the load balancer will redistribute the load and will call the process migration through migration mechanisms [24, 25]. In the third pattern, the process migration transmits a process to another computing node to meet the process requirements [26]. Each element of resource discovery and load balancer, based on information gathering, resource management and computing elements, are designed to operate at system level, despite the need to process `and to create a stable state, imposes additional load to the computing system. Therefore, reducing the number of times these units activated, can reduce the response time in addition to reducing system overhead [27, 28].

Each of the three mentioned patterns uses different mechanisms for carrying out the activity that the type and structure of the mechanisms are proportional to the architecture of the distributed system, the objectives and types of received requests [23, 24, 28]. Below is a review of some of the load distribution algorithms, resource exploration and process migration in distributed cluster, grid, peer to peer and cloud systems.

2.1. Load Balancing

In high performance computing systems, especially in distributed systems, the load distribution is one of the key elements in system management, which this process affects the performance of the computing system [29]. In [30], a kind of load distribution algorithm in cloud computing systems is introduced, which the load balancer receives a list of virtual machines and the JOB queue by scanning, and based on the information obtained, the request is made on a virtual machine properly mapped. If in this way the virtual machine is overloaded, a number of jobs will be transferred to other virtual machines. [31] Provides a mechanism for modifying the problem of throttled algorithms that do not take specific resources for tasks and are not suitable for heterogeneous virtual machine environments. In this mechanism, using clustering of virtual machines into groups in which virtual machines of a similar size, addresses the problem. This mechanism improves in wait time, runtime, turnaround time and throughput. [32] Proposes an autonomous agent-based load balancing mechanism that provides load balancing in cloud environments. This mechanism, by calculating the pre-load factor for virtual machines, is used to search for a virtual machine if the speed of the virtual machine approaches the threshold. Maintaining the information of the virtual machine of the candidate reduces the time of service. An algorithm called AMLB has been presented in [33], which collects and stores information about virtual machines, the number of received requests, at any given time, identifies the virtual machine with the least load and allocates the request to it. In this way, the virtual machine with higher processing capabilities is allocated more load. This algorithm improves response time. [34] An algorithm based on the method of estimating the end time of heterogeneous clouds is presented with the aim of improving processing time and response time. In this algorithm, a different level of timing is considered by calculating the average processing power of the virtual kernel.

In [35], a load distribution algorithm in cluster systems introduces. In this algorithm, instead of considering the queue length of the queue, it uses processor efficiency, processor queue length, network traffic, and memory efficiency to detect the load of each node. In this algorithm, after deciding on the transfer of the process, the process from the highly load to the lightly load computing node. In [36], a load distribution algorithm in grid systems is introduced. The algorithm gains load information from clusters and transfers it to a unit called GIC to store it. If the load transmitted exceeds the limit, that is, the amount of use of all cluster resources reaches a certain limit, the GIC unit will be notified. GIC transmits the load to another cluster by providing a list of other clusters that use the least.

2.2. Resource Discovery

One of the basic services in distributed computing systems such as grid computing system is the resource discovery, which should be able to provide relevant resources for requests [37]. In the classification of distributed resource algorithms in grid systems, [38] one or set of controllers finds appropriate resources in the server client architecture. In this mechanism, servers store information about the services and services they provide. When a request is received in relation to a specified service, the request is sent to the server and the appropriate resource is assigned to the request.

The paper [39] provides a sort of adaptive resource discovery mechanism and resource selection models in Grid. The mechanism is based on three models: the adaptation stretch model, the Adaptive and Model pull-pull adaptive. In the adaptation stretch model, the grid environment consists of several nodes, one of which is a synchronization node, in which a daemon is running. This daemon can collect dynamic information such as CPU speed, CPU load, and memory size from various remote nodes. Also, due to heavy traffic to the daemon, as soon as the grid environment becomes larger and larger, the search query process will waste a lot of time. In the adaptive pressure model, the Grid environment consists of several nodes, one of the nodes being used as the coordinator, the main server is running, and the other nodes in the environment collect local status information. The grid environment in this model consists of three layers: the main coordinator (layer 1), the aggregators (layer 2) and the grid node (layer 3).

In decentralized resource discovery mechanisms, [40, 41, 42, 43, 44, 45] they have come up with a flat-panel architecture that has been configured. Each user connects to a node, issues your application. The node responds to requests when it implements requests, or transfers requests to other nodes.

2.3. Process Migration

The process migration in distributed systems is responsible for managing the process transmission trend. There are various mechanisms in this scope. In [46], the process migration based on parameters decision and the system purpose, chooses one of four mechanisms: total copy, precopy, lazy copy, flushing, and designing the transfer process based on it. In [47], a migration mechanism is introduced based on a genetic algorithm that effectively mitigates the load imbalance in multiple processors. In this mechanism, the genetic algorithm is used to find the best destination for migrating the process. In [48, 49, 50], the process of migration has been proposed to improve efficiency and reliability. In this mechanism, each process is divided into several repetitions. Each copy is executed on a different server to calculate the amount of tolerance. In addition, any copy processor migrates to another server if it consumes more than one current server.

3. Basic concept

The process migration in the cluster and grid computing systems, does not play a role in any of the two concepts of candidate processes and candidate computing nodes as destinations. Both responsibilities are the responsibility of the load balancer. The load balancer, based on the single variable function defined by the central processing unit, decides on the candidate computing elements as the destination machine. When the load balancer attempts to select the candidate's computing node process transmission, this decision is made in the form of an abstract decision. In the abstract decision, the decision maker node summarizes all the factors affecting decision-making in a parameter (or decision parameters) and processes the trend of the decision in an abstract environment.

This leads to the following: a) during the decision-making trend, the role and influence of the factors effecting the decision will remain constant. B) Only the factors identified by the decision-maker affect the decision. In such a situation, the load balancer uses Eq. 1 to decide on the selection of the computing node as the destination candidate.

$$\forall \text{Machine}_j \in \{\text{Computing}_{\text{System}}\}, y = \text{Cpuusage}(t) |\; y + \text{Cpu usage}_{\text{process}_i} < 100 \text{ then select Machine}_j \qquad \text{Eq. 1}$$

As seen in Eq. 1, in computing systems such as Grid and Cluster, Process I, is chose by the load balancer for the process migration. This element obtains information from the data structural of the operating system or based on the information received from the corresponding units on the labor status of each candidate computational nodes. If, in the case of transferring the process to the computing node j, the load condition does not exceed 100%, it will start the transmission. Eq. 1 is an abstract formula, in which the effects of computing nodes and the role of process in global activity, does not considered.

This decision pattern in distributed Exascale systems, especially during the occurrence of a dynamic and interactive nature, cannot be used because, at the time of the occurrence of a dynamic and interactive nature, a concept known as impression and influential nodes is defined. Based on the concept of impression and influential elements, when dynamic and interactive event occurs, the set of factors on the decision-making trend of determining the candidate nodes for the process transmission is effective, and also what element is considered for the transfer of the process is impressed. Therefore, the Process migration should attempt to define two Influence and Impressionable spaces for each process. The two spaces mentioned, should also be defined for the destination computing node.

In distributed Exascale systems, unlike traditional computing systems, the core element of performing activities from the point of view of the process migration is global activity. In traditional computing systems, the concept of process is used by the process migration. The process migration defines its activities based on the concept of the process. The redefinition of the concept of the core element based on global activity makes it possible to: A) decisions made by the process migration, in contrast to Eq. 1, to withdraw from the abstract state. B) The destination machine is selected based on its functional role in completing a part (or all) of the global activity.

Consider that each global activity is a collection of related processes. These processes are not abstract and interact with each other. This causes the process migration, during the determination of the candidate's computing element for process transmission, requires the consideration of these interactions and communications.

If, at the moment t = Alpha, an event occurs in the system that the process migration selects process p for process migration, then the migrate set can be calculated based on all the computational elements that are allowed in Eq. 1. The members of the Migrate set are defined according to Eq. 2.

$$\forall \text{ Machine}_j \in \text{Eq. } 1_{\text{set}} |< \text{time}_{\text{conditation}}, \text{Dependency}_{\text{conditation}}, \text{Deversion}_{\text{conditatiom}} > \text{is Valid} \mid < \text{Time}_{\text{condition}}, \text{Type}_{\text{conditation}}, \text{Dependency}_{\text{condition}} > \text{ Then Machine}_j \in \text{Migrate}_{\text{set}} \qquad \text{Eq. 2}$$

Eq. 2 implies that each computing node j can be a member of the migrate set, which has A) the necessary condition for membership in the set, and after the process migration the load of destination node where acceptable. B) Time constraints, dependencies, and redirection of national activity. C) Time constraints, types, and dependencies in the destination machine to accept the computing process. Eq. 2, in contrast to Eq. 1, attempts to create a space of 3 * 3, in which both process and the destination computing node must be able to fit into an adaptable space. The element that connects the two three-dimensional spaces associated with each process and destination computing node is the time dimension.

Time from the viewpoint of the process means the consideration of the constraints governing the execution time of the process, and from the point of view of the destination computing node, means the constraints governing the implementation of processes related to global activities. In terms of the process, any concept in which the failure to perform an activity at a specified time would lead to failure of the execution process or Whether an activity occurs within a given time period leads to the activation of an incident, it is a time constraint. The timing constraints from the viewpoint of computing nodes include any computing node dependency in the sense of time, which in some way will affect the management of the global activities existing in the destination computing node.

The dependency constraint is also common between process and destination computing node. In the processes, the dependency constraint is the set of dependencies of the process to other processes and resources that fail to fulfill them will lead the process to fail its execution. In the destination computing node, the dependency constraint is any ability or inability that makes the execution of the process in the event of process migration on the destination computing node.

The redirection overrides the global activity on the process space, includes all constraints, limitations, capabilities, and features of the redirect the global activity from the source to destination computing node. This redirection of global activity makes the descriptor line of the computational process change. [4D] The type constraint is defined in distributed Exascale computing systems. This constraint stems from the patterns used by the distributed Exascale computing system manager for resource categorization. For example, in distributed Exascale computing systems, [52] uses the classification of resources based on the operating system classification model. The type constraint addresses whether the process transmission, from the source computing node to the destination computing node, has a functional definition or not? If the destination computing node has the ability to respond to the request, but the type of resource requested by the process has no comparative advantage and in the stable state of the computing system, the destination computing node is not considered as the source of the computation of the source response, in this case, the destination computing node is not capable of establishing the type constraint.

4. ExaMig Matrix

If, after the creation of the Migrate set, in the process p a dynamic and interactive event occurs, then the ExaMig Matrix process migration stops the current process migration trend and starts a new process migration, because the process p has be transferred to another process like ṕ. Another solution, known as the impact factor, is the extraction of the transformation effects of the process p into ṕ and the reflection of the effects on each three dimensions of the descriptor space of the process for the process migration. In such a situation, the most important challenge of the process migration is the process that, due to the changes in the process of turning the p to ṕ, each dimension of the descriptor space from the general pattern governing the dimension before the transformation of the p to ṕ or as a result of this becoming, then it has lost its nature and has become in another sense. For this purpose, the process migration introduces a conceptual process called the description of the next situation based on Eq. 3.

$$\text{Circumstance} = \sum_{\text{state}_i} (\text{Beta}|\text{Alpha}_{\text{state}_i})\text{Alpha}_{\text{state}_i} \qquad \text{Eq. 3}$$

In Eq. 3, if the process definition space is taken into account as a form of a three-dimensional space (T, D, D), then each process p is actually defined a space based on the multiplication of inner product of these three dimensions to each other. In this situation, the space w a sub-space on the influence of the inner-product of space (T, D, D) due to the occurrence of a dynamic and interactive nature, and Beta is a space of spaces (T, D, D) producer which the start process migration is defined by the process, that the circumstance is the best approximation of the beta by the w vector. Formula 3 describes the approximate description of the descriptor space of the process after the occurrence of a dynamic and interactive nature and its approximation to the process description space at the start of the migration process. Eq. 3 describes the approximate description of the descriptor space of the process after the occurrence of a dynamic and interactive event and its approximation to the process description space at the start of the process migration. If Circumstance exists, it means that the process migration is able to have an appropriate approximation for the process state after occurring in a dynamic and interactive event, otherwise, due to the lack of definition of the approximation, from viewpoint of the process migration, the process failed.

If the destination computing node has a dynamic and interactive nature and this dynamic and interactive nature causes the process transmission status be changed from the source to destination, then the process migration for each computational node in the Eq. 2 selected as the candidate computing node for the process transmission, calculates the Nominee coefficient based on Eq. 4.

$$\text{Nominee}(t) = [\beta_1\,(t) + B_2\text{Time}(t) + B_3\text{Running} - \text{ability}(t) + B_4\text{Dependency}(t) + B_5\text{Viability}(t) + B_6\text{Scalibility}(t)] * D(t) \qquad \text{Eq.4}$$

As is seen in Formula 4, the process migration will calculate the probability of the process migration in each computing node that is known as the destination candidate node.

$$\beta_1 = \left[\frac{\text{NumberAccepted Process}}{\text{Number Requested Process}}\right]_{t=\varepsilon}^{t=\gamma} \qquad \text{Eq. 5}$$

As is seen in Eq. 5, the coefficient β1 indicates the number of processes that apply to the computing node for process migration trend and has been migrated to the computing node, in relative to the number of processes that are used for the computational element, have applied for immigration - in the specified time frame [γ, ε]. The coefficient β1 is a value between zero and one, greater the probability of accepting the process of passing the process to the destination computing will be. β_i, which i index is between 2 and 6, indicates the importance of the index defined in [6] for the ability of the computing node to perform process migration during the occurrence of a dynamic and interactive nature. According to [6], to let a computing node be able to support during the occurrence of a dynamic and interactive nature, it is necessary to be able to include these indices in the computing node in which the dynamic and interactive event has occurred. Obviously, according to [6], the inability to calculate the indexes in a computing node which is a member of the migrate set, means that it is not possible to carry out global activities related to process migration in the computing node mentioned. β_i indicates the importance of index i for global activity, which is part of the candidate process for process migration. The coefficient β_i may be different for each global activity. The value of β_i , depends on the significance of the coefficient of the index defined for global activity in process related process migration. The coefficient D indicates the degree of distortion of the process involving the process migration of the destination computing node. The coefficient D is between 1 and 100, indicating that in a certain period of time, the dynamic and interactive nature of the occurrences in several percent of cases has resulted in the failure of global activity.

In distributed Exascale computing systems, unlike traditional computing systems, each process is not considered to be abstract in terms of the ExaMig Matrix process migration. From the ExaMig Matrix process migration point of view, each process has a set of interactions and communications with other processes that comprise a global activity, as well as processes belonging to other global activities that are executing on the computing node. This makes the ExaMig Matrix process migration, necessary to consider the concept of the beneficiary elements of the process. If the ExaMig Matrix process migration for the process Beta attempts to calculate Formula 4, then the set is called 2I, which contains effective and affecting the Beta process. The ExaMig Matrix process migration for each member of 2I sets, calculates Eq. 4.

Considering the calculation of Eq. 4 for the Beta process and set of 2I processes, the ExaMig Matrix process migration uses Eq. 6 to examine the influence and impressionable of elements of the 2I set on the Beta process.

$$\begin{aligned}\overrightarrow{\mathrm{Nom\imath nee}}(\mathrm{t})_{\mathrm{Beta}} &= [\mathrm{Indicator}_{\mathrm{matrix}} * \beta_{\mathrm{matrix}}] * \mathrm{D(t)}_{\mathrm{matrix}} \mid \mathrm{Indicator}_{\mathrm{matrix}} \\ &= \begin{bmatrix} 1 & \mathrm{I}_{21} & \cdots & \mathrm{I}_{61} \\ \vdots & \vdots & \vdots & \vdots \\ 1 & \mathrm{I}_{2k} & \cdots & \mathrm{I}_{6k} \end{bmatrix} \text{ and } \beta_{\mathrm{matrix}} = \begin{bmatrix} \beta_1 \\ \vdots \\ \beta_k \end{bmatrix} \text{ and } \mathrm{D(t)}_{\mathrm{matrix}} \\ &= [\mathrm{D}_1(\mathrm{t}) \quad \ldots \quad \mathrm{D}_6] \qquad \text{Eq. 6}\end{aligned}$$

As can be seen in Eq. 6, in distributed Exascale computing systems, for each Beta process, maping the values of the invertible variable $\overrightarrow{\mathrm{Nom\imath nee}}(\mathrm{t})_{\mathrm{Beta}}$ is available. The $\overrightarrow{\mathrm{Nom\imath nee}}(\mathrm{t})_{\mathrm{Beta}}$ vector is a 6*6 vector. This vector illustrates the effects of the elements of the 2I set on the process migration of the Beta process during the occurrence of a dynamic and interactive nature. In formula 6, the $\mathrm{Indicator}_{\mathrm{matrix}}$ indicates the effect of each element of set 2I on a specific and definite element j, in which the j can be $[\mathrm{Time(t)}, \mathrm{Running-ability(t)}, \mathrm{Dependency(t)}, \mathrm{Viability(t)}, \mathrm{Scalibility(t)}]$. each member of I_{nk} set can be calculated from the Eq. 7.

$$\begin{aligned}\mathrm{I}_{\mathrm{nk}} = \lim_{\mathrm{t}\to\mathrm{Stable}} &\text{ influence } (\text{Machine q} \to \text{Variable j}) \mid \mathrm{q} \in 2\mathrm{I}, \mathrm{j} \\ &\in [\mathrm{Time(t)}, \mathrm{Running} \\ &- \mathrm{ability(t)}, \mathrm{Dependency(t)}, \mathrm{Viability(t)}, \mathrm{Scalibility(t)}] \qquad \text{Eq. 7}\end{aligned}$$

As seen in Eq. 7, the effect of the computing node q on the variable j in the state in which the system is in stable state, is calculated and considered as the value of

I_{nk}. The effect of the machine q on the j variable is that the computing node q depends on has an effect on the variable j or not. If the value of I_{nk} is more the specified value Z, the Z value is depends on the nature of the program executing on the distributed Exascale computing system, then the influence of computing node q on the variable j is strong and has influences on the process migration; otherwise the computing node q will be omitted from the 2I set.

In Eq. 6, each element of the β_{matrix} indicates the significance of coefficient of the computing element's ability to perform process migration during the occurrence of the dynamic and interactive nature of the vector form. each element of the β_{matrix} is calculated on the basis of this, if any of the j indicators contained in each computing node can be assigned to a vector. In this vector, the vector size is equal to the weight of the index and its direction, is the as the process migration in the computing node. At the moment of the creation of a distributed Exascale computing system, this vector is positive for all indexes, and over time and according to the process migration, the direction and amount of these vectors varies. The weights of the vectors with each other and with the Beta process are the components of the elements of β_{matrix}.

In Eq. 6, each element of the $D(t)_{matrix}$ indicates the refractive index of the computing node for performing process migration during the occurrence of a dynamic and interactive nature in the form of vectors. The value of each element of the $D(t)_{matrix}$matrix is equal to the weighted error rate of the vector of process migration impediments with each other and with the Beta process.

The $\overrightarrow{Nominee(t)}_{Beta}$ is calculated as the effective coefficient of the computing node containing the Beta process, which is the member of the Migrate set. The process migration for each constituent process which is the member of migrate set, begins to calculate $\overrightarrow{Nominee(t)}_{Beta}$ vector. Then, by considering the orientation of process migration from the indicators mentioned in [6], compares the vectors and select the destination computing node.

5. Evalution

To evaluate the proposed mechanism for selecting the destination node in distributed Exascale computing systems [7, 9], a peer to peer computing system is used. In this computational system, the system manager uses the notion of areas for system management, this make it possible for this kind of computational system to be four regions that are commensurate with the four main sources defined by the operating system. In this type of computing system, global activity concept is used to examine impressionable and influence concept on the chosen process for process migration trend.

In order to evaluate the proposed framework, two types of global activity have been executed MM5 [51, 52] and WRF [51, 52] software tools that require distributed Exascale computing and processing system, each of these two software systems use available computational resources based on their own global activities. So, there are two types of executing global activities in the system. Each computing element of the system can be involved in execution of one or two global activity at each time. On the other hand, each computing node can execute one or more parts of global activity in a time unite. These part of global activities are not dependent.

System formation contains 140 computing nodes, which allow the computational system considered to be test bed as a broad system for each of the two software system. The mentioned software typically runs on less than 140 computing node, makes it possible to analyze software state at a time when they are on a large system.

Due to the nature of distributed Exascale systems and the necessity of defining the initial computational system, 30 computing nodes were considered as primary computational system. These 30 computational nodes are in accordance with the basic requirement of the computational processes associated with the applied scientific and practical program to create dynamic and interactive events on machine No.7 specific version of system management software has been used, in which the process migration uses the ExaMig matrix mechanism. The machine No.7 was chosen because of its participation in the most execution time of scientific program of both program global activities. Hardware configuration of this computing machine is the same as the other computing machines in the system. If, for every global activity, the activity page as expressed in [53] is considered, in the majority of the time for execution of scientific and applied programs, computational node NO.7 is the intersection point of both pages corresponds to global activity. Each other computing node can also be selected as the considering computing node. In computing node NO.7 the process migration is able to manage the process creation connection and interactions with system environment, which leads to dynamic and interactive nature. For this purpose, management element stated in No.7 node.

Is able to manage process that are not present in the global activity structure.

Manage the relationship between two processes which create the global activity that are not considered in the basic structure of the global activity.

The computing node NO.7, related to on other machines process of two applied scientific and practical software. The No.7 machine management is able to manage connection between applied process and global activity and also the corresponding process is out of the computational system that does not consider in structure of global activity responsibility, the system and consequently No.7 computing node, has been investigated in 50 time unite. In figure No.1 the number of occurrence related to the conventional process migration and also process migration at dynamic and interactive occurrence time in one of the migrate sat element in No.7 computing node.

As seen in figure1, conventional, in computing node No.7, on arrange of 2/0 processing request occurs. In computing node No.7, in each examining time unite, among the processes related to the two global activities, there are 2/8 processes that contains some requests which cannot be responded in computing node Nom.7 but there is a computing node that able to respond to those requests. On the other hand, in computing node nom.7 after formation of the migrate set, on average 1.5 requests per time unite become dynamic and interactive. From migration management point of view, in 56% of cases, after the migration set formation for decision making about the destination computing node, a dynamic and interactive nature occurs at least on one computing node and make the process migration management to used ExaMig matrix mechanism. The dynamic and interactive occurrence shows the need of defining ExaMig matrix mechanism in 56% of migrate set formation by process migration.

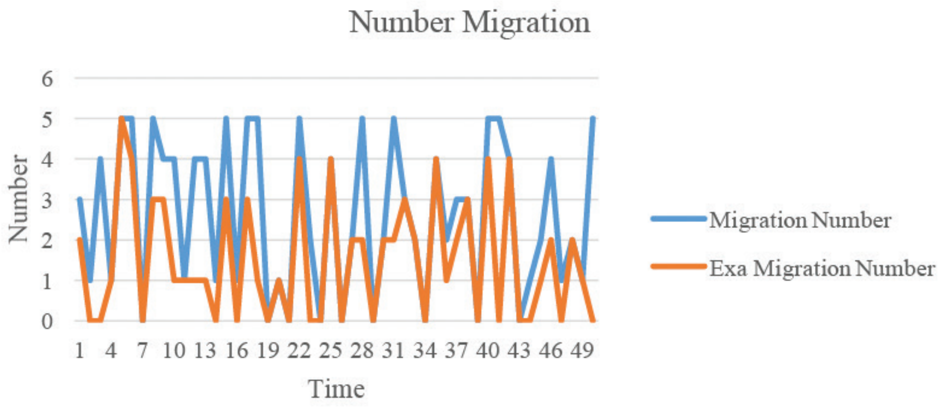

Figure 1. The number of dynamic and interactive events in computing node number 7

Incident has led to the dynamic and interactive process migration. From the process migration viewpoint, in 35% of the examining time, computing node No.7, has not been involved in a dynamic and interactive event in any element of the migrate set so the ExaMig mechanism has not been activated. If computing node No.7 test repeats for many times, it is seen in 35% of examining time no incident leads to ExaMig activation the number is changing among 20 up to 43 percent. The reason for this acceptable spectrum is not the specific and practical software nature. The scientific and practical program and its global activity was the same in all experiments, especially when the number of test repetition increases and the system inters the equilibrium state and follow a steady pattern. The reason for this is the choice of machines that of member of migrate set. In the other words, the dynamic and interactive occurrence where the system test is in its equilibrium state, in 39.5% of the time depends on the ExaMig process migration management use what kind of algorithm to choose migrate set computation node.

As seen in figure1, the operational nature of processes related to global activity in computing node No.7 is such that in12 times computing node No.7 checking, 100%. Common process migration request converted to dynamic and interactive process migration. By considering the scientific and practical programs execution it is concluded, from 12 times, 18 times due to the creation of a new process, 3 times due to the communication and interaction between processes and one time due to the interaction with system environment effect, in computing node such as 5, 15 and 25, as one of the migrate set member. Involved in dynamic and interactive nature what is important in this regard, is that ExaMig mechanism does not consider the dynamic and interactive creation reason in migrate set member to re-decide on destination computing node.

The ExaMig investigate the migrate set status based on formula 6 after dynamic and interactive nature, the r-weighs the migrate set members.

In figure2, the experiment has been reviewed in two situations, in the first situation, the process migration management in the time of dynamic and interactive occurrence in one or more migrate set member uses the ExaMig mechanism and in the second situation it uses re-building the migrate set or reload balancing. Each execution time of both situations at system considering time is shown, as shown in fig two, average time required to run the ExaMig matrix mechanism at any considering time is about 51.38 and the average time required to run the load balancing is about 52.92 time unite.

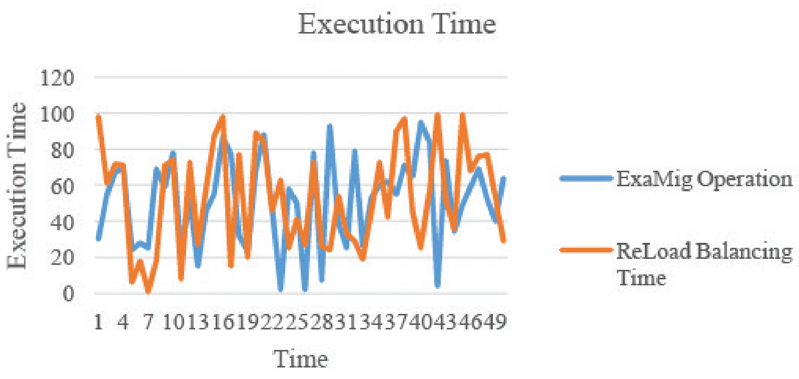

Figure 2. the time execution of ExaMig Matrix mechanism and Reload Balancing Approach

From the process migration management point of view this means in case of dynamic and interactive nature occurrence for each of migrate set member the required time to create new migrate set or re-weighing the current migrate set member with considering the dynamic inter active effect on set member that dynamic and interactive occurs in, tack two time unites the major functional difference of the ExaMig process migration in using matrix mechanism or reload is the required information for making decision and create migrate set . if the computing system has the fixed pattern of hardware and software information like time unit 5-7the required time to execute reload mechanism is low and matrix mechanism execution time is incomparable with that generally when changing frequency of system resource status variables and also the five process migration management decision making variable are low and system stay in equilibrium condition mechanisms like reload higher performance and less execution time than matrix mechanism on the other hand in a situation where the changing frequency of resources and process migration management decision making performance are high and function 4 and 6 completely depend on time concept – the matrix mechanism functionality due to the re- weighing migrate set members and their dis changing in this situation is lower accumulation. In the reload mechanism- the members of the migrate set may be different from those of the previous migrate set. This is due to the accumulation of new information about the system state and computing node. However- while the matrix mechanism does not change the migrate set member- their weighs change to be selected as the destination computing node. On the other hand –the matrix mechanism is able to after calculating formula 6 and calculating each strata by formula 7- make decision about the destination node based on each $\overrightarrow{\mathrm{Nominee}(t)}_{\mathrm{Beta}}$ strata. In the matrix mechanism, according to the $\overrightarrow{\mathrm{Nominee}(t)}_{\mathrm{Beta}}$ dimensions, the decision space for choosing the destination computing node change from a traditional one-dimensional space based on the central processing unit capacity to a 6*k dimensional space that showing the combinations and effects of the five dimensional process migration decision making on each other.

In fig.3 the number of computing node candidate and the migrate set member are shown in both the conventional process migration and matrix.

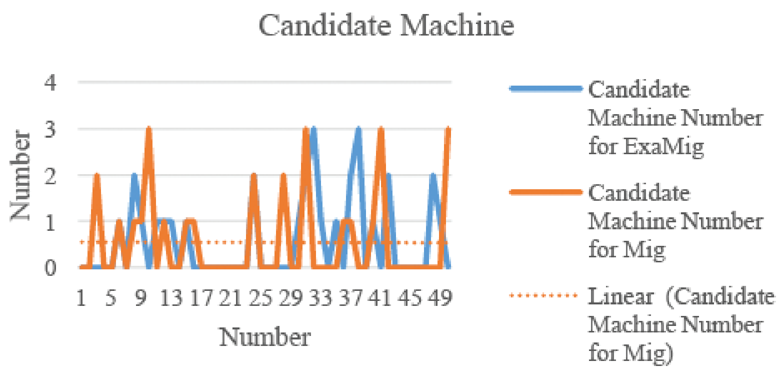

Figure 3. the number of candidate computing node chosen by ExaMig Matrix mechanism

As seen in fig.3, conventionally 0.62 computing node selected as process migration destination computing node candidate in each system investigating time unit. Difference between numbers of occurrence leads to process migration destination is the absence of an appropriate element for managing process migration. The reason for this subject in the system management nature, in system management, if process requests access to a resource that local computing node lacks the ability to respond, a process migration management is called. Process migration management create migrate set based on information of operation history data structure and also decision making parameter status and the constraints and condition of process request. If the each of the three factors above information is not enough or the process request constraints and conditions not appropriate for any computing node, system management calls the resource discovery. There for process migration is not feasible because difference between the computing node candidate number and the number of occurrences leading to the process migration activation and migrate set creation, so it change to resource discovery trend.

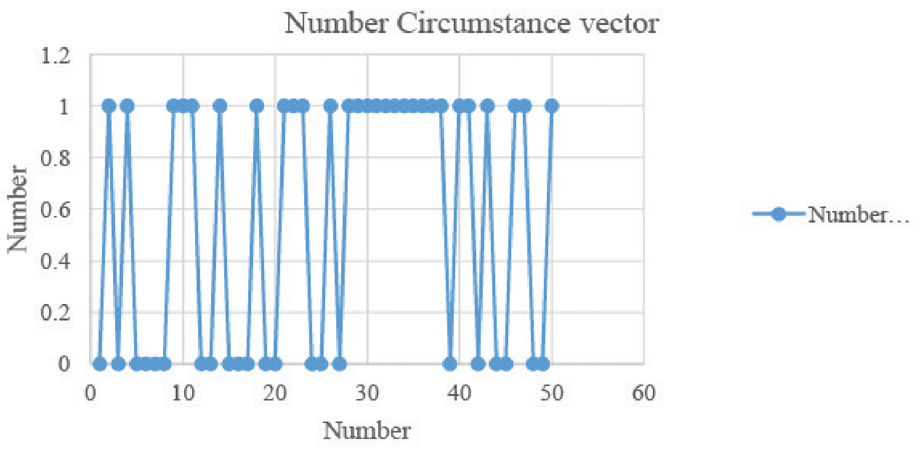

Figure 4. Calculating the $Circumstance$ vector in ExaMig Matrix process migration

In Figure 4, the Circumstance vector creation status for the process 114 is displayed in computing node num7. Process 114, in computing node num7, is randomly selected and any other computing process can be selected. In the test shown in Figure 4, the Circumstance vector computation condition for Process # 114, which at the time of process migration is of a dynamic and interactive nature, is displayed for 50 times the experiment.

As seen in Figure 4, the ExaMig Matrix process migration, in the 56% of events found a Circumstance vector. From ExaMig Matrix's process migration point of view, this means that in 56 percent of the cases, the process migration, after the occurrence of a dynamic and interactive event, is able to recreate the vector corresponding to the process and extract the process vector image in each of the three dimensions (T, D, D) and the definition of the space w for the process.

The reason for this is, how to form wi vectors. The wi vectors are obtained from the process state of the computing node that are membered in candidate set in each of the three dimensions (T, D, D). Generally, the nature of processes executing on a computing system is such that the change in the resultant state of the computational processes does not occur immediately. The nature of the computational processes of the scientific and software executing on the computing system at the time of the system review, employs a pattern of stability and no immediate change. This leads to the fact that, except at certain times, the status of the wi vectors of the computational elements of the candidate set member is stable and not subject to change. Therefore, the pattern of finding or not finding the Circumstance approximation vector for successive experiments is usually constant and does not change the state of the computational processes existing in the computational node of the candidate set member.

The instantaneous changes shown in Fig. 4 are due to the change in the candidate set elements and, consequently, the change of the wi vectors relative to the computing elements of the candidate member set. This is due to the change in the position of the candidate's computing nodes. The change in the status of the nodes of the candidate's set can be due to the change in the members of the candidate's set as a result of the fulfillment or non-fulfillment of the conditions expressed in Eq. 2 or the change in the executive status of the candidate member nodes.

6. *Discussion*

The activation of the process migration means that failure to execute the process in the source computational node, and the computational process must be transferred to another computing node, to continue the process of running the global activity. On the other hand, in distributed Exascale computing systems, there is a possibility of a dynamic and interactive nature at any moment in the process of implementing a global activity. This dynamic and interactive nature can occur from the moment that process starts transferring from source to destination. In such a situation, the process migration should have a mechanism for managing and controlling the occurrence of a dynamic and interactive nature and also preventing the failure of the process migration. The inability of the process migration to perform the two tasks mentioned above until the load balancer is activated again, needs to gather information about the new status of the system. For this purpose, the process migration must be able to change the axial node from process to global activity. Changing the axial element of the process migration from process to global activity ,will change e the duty of process migration from process transmission to changing the execution trend of global activity.

The axial node changing makes it in addition to the traditional definition of the process migration, as the process application management node for global activities, when one (or more than one) process of global activity is in the local computing node has a request that the local computing node lacks the ability to respond to, but the computing system has node that have the ability to respond to requests, also to be used. Changing the axial node, the process migration from process to global activity, cause that the node has the ability to manage and control dynamic and interactive event during its implementation. Dynamic and interactive event during execution of the process related to the process migration, the three elements, source, destination and process migration management were identified as the main node and beneficiaries of the activities related to the process migration. Dynamic and interactive nature can occur in any of the three mentioned elements. In this article, investigates the status of the process migration activities at the time of occurrence of the dynamic and interactive nature of the destination node and after start of process migration, as well as the occurrence of a dynamic and interactive nature for the process at the time of selection as a immigrant process. The occurrence of a dynamic and interactive nature in the destination node, during the process of implementing the process migration, increase the probability of failure in the implementation of this activity. The occurrence of a dynamic and interactive nature in the destination node, it may create a situation that does not allow to run immigrant process at destination.

The occurrence of a dynamic and interactive nature in the process selected as an immigrant process, may cause the immigrant process requirements to be change so that the target computing node fails to meet the requirements at the process. In this case also process migration will fail.

Increasing the chances of failure during source discovery activities will increase the time cost of executing process management operations in distributed Exascale computing systems. The nature of the activities of the process migration is in a way which at the time of the transfer of the process from the source node to the destination computing node, immigrant process and set of process that interact and communicate with processes in standby mode. This situation means increasing the time of implementation scientific and practical and increasing the response time. On the other hand, at the time of the activities related to the process migration just this node is interacting with the relevant nodes, including the process and the computational nodes of the source and destination. The process changes and the computing node due to the occurrence of a dynamic and interactive nature during the execution of activities related to the process migration by this node can be analyzed and extracted. Therefore, if the process migration can, based on a pattern derived from the properties of activity, describe and analyze the process and the computational node to be able to change when dynamic and interactive nature occurs. Defining a set of nodes as the target computing node, as well as redefining them based on properties that effect on process activities, will reduce the probability of failure of global activities.

Changing the axial node, the process migration management node, caused that from the point of view(sight)of the process migration management node, the migrating processes, in the target computing node are based on a three-dimensional space, and the target computing node is also described based on a three dimensional space. The description of the nodes of the three dimensional process space causes the process to leave an abstract concept and to be considered part of a global activity. This makes it possible for the process node, two concepts of the influential node and impressionable node space. Defining the two spaces makes it possible for the process node transfer time from the source computing node to the target computing node, if any of the effective nodes on the process definition space are changed, provided this change causes the process request space differ from the initial request space of the process. In this case, the process migration is to retrieve the other computing node as the destination. On the other hand, the definition of the two influential and impressionable space on the process makes it possible, in the event of a dynamic and interactive nature in the process during transfer process, the process migration as an element that is associated with the process and controls it, can put the impacts of the dynamic and interactive nature of the process on other nodes of computing system management, which are working with the elements of the collection. Defining the process space based on the three concepts of time constraints, dependency and also redirection of activity, it implies the concepts of acceptable nodes as the destination of the process. These two dimensions represent the process requirements to continue their work. Dependences indicates the effect of two influential and impressionable space on the process. This dimension, explicitly states how the process has some patterns in terms influence of the nodes of the beneficiary or effect on other nodes. Dependence dimension, in fact, it's the mapping of the effective nodes space and remapping of influential nodes of the process. The aforementioned dimension indicates the elements that the process need them to carry out its actions, as well as the nodes that need the process to continue their activity.

Redirect activity dimension, is taken from the concept of global activity. This dimension indicates the feature that computation node of the destination should have, possibility to redirect global activity and implement part of global activity in the target computing node. This dimension implicitly expresses the constraints that are govern in process migration trend. Three dimensional process describing space from the sight of process migration, describing the requirements of the process, considering that the process is part of global activity. The definition of the target computing node space is also described based on the three dimensional space, time constraints, type and dependence. Type dimension, indicates the capabilities of the target computing node for carrying out activities. The time constraint is defined as the constraint of sharing the two definition space of the target computing node and process from sight of the process migration, it explains whether the computing node has the ability to execute the process. The dependency constraint indicates the effects of two effective and impression space on the computational node space. This dimension indicates which node to carry out their activities require the presence of a computing node, and which node must be present .This feature is normally set by the load balancer management.

In formula number three, if the process definition space is considered in the form of a three dimensional space (T, D, D), in this case expression each process p is actually a defined space based on the three dimensions in each other. In this situation, we can consider the space w a sub-space of inner products of space elements. (T, D, D) due to the occurrence of a dynamic and interactive nature and Beta is a space of process migration is defined based on it. In this case, the circumstance is the best approximation of the Beta by the w vector Eq. 3, describes the approximate of the descriptor space of the process state after the occurrence of a dynamic and interactive nature and its approximation to the process description space at the time of migration begins. If circumstance exits, this means that the process migration is able to have an appropriate approximation for the process state after occurrence of a dynamic and interactive nature and otherwise, due to a lack of process migration fail.

In this paper, for managing the occurrence of a dynamic and interactive nature in the immigrant process, during the implementation of the activities related to the process migration, based on the vector algebra begins to describe the system status.

7. Conclusion

In distributed Exascale computing systems, the process migration, in addition to the task of transferring the process from the destination-to-source computing element, is responsible for creating a new accountability structure for the computing process of the global activity participant. In this type of computational system, the process migration is attempting to create a new accountability structure in the computing system for a process that has not the ability to respond to its requests in the destination computing node. During the transmission trend, from the destination-to-source computing element, a dynamic and interactive nature can occur. A dynamic and interactive nature can occur in either of the three elements of the computing: element of the destination, the source and migration process, or in more than one beneficiary nodes. The most important consequence of the occurrence of a dynamic and interactive nature during the execution of activities related to the process migration is to increase the response time and increase the probability of failure of the process migration trend. In the ExaMig Matrix mechanism, by changing the axis element, from the process to the element of global activity provides a model for dealing with and managing the occurrence of the dynamic and interactive nature of the process element during the process migration trend. In this mechanism, by defining the set of candidate computing elements as the target computing node, in addition, reducing the probability of failure of the process migration trend, if there is a dynamic and interactive nature in the process element, it is possible to define new responsiveness structures for the process. In the ExaMig Matrix mechanism to manage and control the occurrence of a dynamic and interactive nature in the destination computing node, while using the notion of computing elements of the target candidate, concepts such as impressionable and influential nodes on the computing system, coefficient of acceptance and refractive index are considered. In each computing node of the destination candidate, the definitions and indicators for executing and executing process migration from the viewpoint of the process migration is considered, taking into account the above concepts, after the occurrence of a dynamic and interactive nature. Take place Based on the fact that the processor migration trend is based on which feature and indicator, it is decided which computing node of the candidate member is capable of executing the process. This makes the process migration, in addition to having a mechanism, able to control and manage the occurrence of a dynamic and interactive nature in the computing system at the time of execution, without requiring a recall of the load distribution, it is able to prevent the failure of activities related to process migration.

References

[1]. Ahmed, Khalid, et al. “Resource manager for managing the sharing of resources among multiple workloads in a distributed computing environment.” U.S. Patent No. 9,632,827. 25 Apr. 2017

[2]. Shah, Syed Asif Raza, Amol Hindurao Jaikar, and Seo-Young Noh. “A performance analysis of precopy, postcopy and hybrid live VM migration algorithms in scientific cloud computing environment.” *High Performance Computing & Simulation (HPCS), 2015 International Conference on.* IEEE, 2015.

[3]. Garcia, G., Octavio, J., Nafarrate A.R. Collaborative agents for distributed load management in cloud data centers using live migration of virtual machines. *IEEE transactions on services computing* 8(6), 916-929.

[4]. Chen C. (2015) Energy-efficient fault-tolerant data storage and processing in mobile cloud. *IEEE Transactions on cloud computing* 3(1), 28-41.

[5]. Mousavi Khaneghah, E., Reyhaneh, N.G., Amirhosein, R.S. (2018) A

mathematical multi-dimensional mechanism to improve process migration efficiency in peer-to-peer computing environments. *Cogent Engineering,* 5(1), 1458434.

[6]. Khaneghah, E.M., Amirhosein R.S., Reyhaneh N.G. (2018) Challenges of Process Migration to Support Distributed Exascale Computing Environment. *Proceedings of the 7th International Conference on Software and Computer Applications.*

[7]. Khaneghah, E.M. (2017) PMamut: runtime flexible resource management framework in scalable distributed system based on nature of request, demand and supply and federalism. *U.S. Patent No. 9,613,312.*

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

[9]. Sharifi, M., Seyedeh, L.M., 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.

[10]. Jiang, Y. (2016) A survey of task allocation and load balancing in distributed systems." *IEEE Transactions on Parallel and Distributed Systems,* 27(2), 585-599.

[11]. Thakor, D., Bankim, P. (2018) Performance Measurement and Evaluation of Pluggable to Scheduler Dynamic Load Balancing Algorithm (P2S_DLB) in Distributed Computing Environment. *Advanced Computational and Communication Paradigms,* 319-329.

[12]. Noshy, M., Abdelhameed, I., Hesham, A. A. (2018) Optimization of live virtual machine migration in cloud computing: A survey and future directions. *Journal of Network and Computer Applications.*

[13]. Lim, D.J., Timothy, R.A., Shott, T. (2015) Technological forecasting of supercomputer development: The March to Exascale computing. *Omega 51:* 128-135.

[14]. Pickartz, S. (2016) "Application migration in HPC—A driver of the exascale era? *Proceedings of the International Conference on High Performance Computing & Simulation.*

[15]. Healy, P. (2016) Single system image: A survey. *Journal of Parallel and Distributed Computing 90,* 35-51.

[16]. Patil, S.S., Arpita N.G. (2017) Dynamic Load Balancing Using Periodically Load Collection with Past Experience Policy on Linux Cluster System. *Am. J. Math. Comput. Model 2(2),* 60-75.

[17]. Khaneghah, E.M. (2011) An efficient live process migration approach for high performance cluster computing systems. *Innovative Computing Technology*. 362-373.

[18]. Varadarajan, S., Ruscio, J. (28 November 2017) Transparent check pointing and process migration in a distributed system. *U.S. Patent No. 9,830,095.*

[19]. Cabello, U. (2014) Fault tolerance in heterogeneous multi-cluster systems through a task migration mechanism. *Proceedings of the 11th International Conference on Electrical Engineering, Computing Science and Automatic Control.*

[20]. Holmbacka, S. (2014) A task migration mechanism for distributed many-core operating systems. *The Journal of Supercomputing 68(3),* 1141-1162.

[21]. Tai, J. (2014) Load balancing for cluster systems under heavy-tailed and temporal dependent workloads. *Simulation Modelling Practice and Theory,* 44, 63-77.

[22]. Qureshi, M.B. (2014) Survey on grid resource allocation mechanisms. *Journal*

of Grid Computing, 12(2), 399-441.

[23]. Pooranian, Z. (2015) An efficient meta-heuristic algorithm for grid computing. *Journal of Combinatorial Optimization,* 30(3), 413-434.

[24]. Siar, H., Kourosh, K., Chronopoulos, A.T. (2015) An effective game theoretic static load balancing applied to distributed computing. *Cluster Computing,* 18(4), 1609-1623.

[25]. Liu, Y. (2017) DeMS: A hybrid scheme of task scheduling and load balancing in computing clusters. *Journal of Network and Computer Applications* 83 (2017): 213-220.

[26]. Kumar, A., Vishnu, V., Krishnakumar, A., Kumar, N. (2018) Efficient performance upsurge in live migration with downturn in the migration time and downtime. *Cluster Computing,* 1(11).

[27]. Dam, S. (2018) An Ant-Colony-Based Meta-Heuristic Approach for Load Balancing in Cloud Computing. *Applied Computational Intelligence and Soft Computing in Engineering.* 204-232.

[28]. 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.

[29]. Yevmenkin, Maksim, et al. “Load-balancing cluster.” U.S. Patent No. 8,886,814. 11 Nov. 2014.

[30]. Ahmed, T., Singh, Y. (2012) Analytic study of load balancing techniques using tool cloud analyst. *International Journal of Engineering Research and Applications, 2(2),* 1027-1030.

[31]. Kapoor, S., Chetna D. (2015) Cluster based load balancing in cloud computing. *Proceedings of the Eighth International Conference on Contemporary Computing (IC3).*

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

[33]. Jena, S.R., Ahmad Z. (2013) Response time minimization of different load balancing algorithms in cloud computing environment. *International Journal of Computer Applications* 69 (17).

[34]. Chien, N.K., Nguyen H.S., Ho D. L. Load balancing algorithm based on estimating finish time of services in cloud computing. *18th International Conference on Advanced Communication Technology (ICACT).*

[35]. Werstein, P., Hailing, S., Huang, Z. (2006) Load balancing in a cluster computer. *Seventh International Conference on Parallel and Distributed Computing, Applications and Technologies.*

[36]. Suri, P.K., Singh, M. (2010) An efficient decentralized load balancing algorithm for grid. *Proceedings of the 2nd International Advance Computing Conference.*

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

[38]. Krauter, K., Buyya, R., Maheswaran, M. (2002) A taxonomy and survey of grid resource management systems for distributed computing. *Software: Practice and Experience 32(2),* 135-164.

[39]. Kovvur, R.M.R. (2010) Adaptive resource discovery models and resource

selection in grids. *Proceedings of the 1st international conference on parallel distributed and grid computing.*

[40]. Iamnitchi, A., Foster, I. (2001) On fully decentralized resource discovery in grid environments." *International Workshop on Grid Computing.* Berlin: Springer.

[41]. Iamnitchi, A., Foster, I. (2004) A peer-to-peer approach to resource location in grid environments. *Grid resource management.* 413-429.

[42]. Torkestani, J.A. (2012) A distributed resource discovery algorithm for P2P grids. *Journal of Network and Computer Applications, 35(6),* 2028-2036.

[43]. Asghari, S., Navimipour, N.J. (2018) Resource discovery in the peer to peer networks using an inverted ant colony optimization algorithm. *Peer-to-Peer Networking and Applications,* 1-14.

[44]. 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.

[45]. Govindarajan, K., Kumar, V.S., Somasundaram, T.S. (2017) A distributed cloud resource management framework for High-Performance Computing (HPC) applications. *Proceedings of the Eighth International Conference on Advanced Computing.*

[46]. Sandhya, S., Revathi, S., Cauvery, N.K. (2016) Performance Analysis and Comparative Study of Process Migration Using Genetic Algorithm. *International Journal of Science, Engineering and Technology Research, 5(11),* 3179-3183.

[47]. Duolikun, D. (2015) Energy-aware Migration and Replication of Processes in a Cluster. *Proceedings of the 10th International Conference on Broadband and Wireless Computing, Communication and Applications.*

[48]. Patel, M., Chaudhary, P., Garg, S. (2018) Improved pre-copy algorithm using statistical prediction and compression model for efficient live memory migration. *International Journal of High Performance Computing and Networking,* 11(1), 55-65.

[49]. Bloch, T., Sridaran, R., Prashanth, C. S. R. (2018) Understanding Live Migration Techniques Intended for Resource Interference Minimization in Virtualized Cloud Environment. *Big Data Analytics.* 487-497.

[50]. Zhang, F. A Survey on Virtual Machine Migration: Challenges, Techniques, and Open Issues. *IEEE Communications Surveys & Tutorials 20(2),* 1206-1243.

[51]. Goga, K. (2018) Performance of WRF Cloud Resolving Simulations with Data Assimilation on Public Cloud and HPC Environments. *Conference on Complex, Intelligent, and Software Intensive Systems.*

[52]. Kartsios, S. (2017) The Role of Heat Extinction Depth Concept to Fire Behavior: An Application to WRF-SFIRE Model. *Perspectives on Atmospheric Sciences.* 137-142.

[53]. Mirtaheri, S.L. (2013) Four-dimensional model for describing the status of peers in peer-to-peer distributed systems. *Turkish Journal of Electrical Engineering & Computer Sciences, 21(6),* 1646-1664.

Submitted 03.02.2018
Accepted 05.05.2018