A Mathematical Multi-Dimensional Mechanism to improve Process Migration Efficiency in Peer-to-Peer Computing Environments

Ehsan Mousavi Khaneghah, Reyhaneh Noorabad Ghahroodi, Amirhosein Reyhani ShowkatAbad

Publisher: Cogent OA

Journal: *Cogent Engineering*

DOI: http://doi.org/10.1080/23311916.2018.1458434

A Mathematical Multi-Dimensional Mechanism to improve Process Migration Efficiency in Peer-to-Peer Computing Environments

Ehsan Mousavi Khaneghah, Reyhaneh Noorabad Ghahroodi, Amirhosein Reyhani ShowkatAbad

Ehsan Mousavi Khaneghah[1], Reyhaneh Noorabad Ghahroodi[1], Amirhosein Reyhani ShowkatAbad[1]

[1]Department of Computer Engineering, Faculty Engineering, Shahed University, Tehran, Iran

EMousavi@Shahed.ac.ir, rnoorabad@hotmail.com, amirhoseinreyhani75@gmail.com

Abstract

In high-performance computing systems, choosing a suitable mechanism for migrating the processes has direct effects on the performance of the system. In traditional systems like clusters, process transmission in process migration management is done using a single mechanism. By changing the requirements of computing systems, open systems like grid and peer-to-peer systems are created. In these systems, the system management does not have a complete view of the nature of all processes due to scalability. Therefore, using a fixed mechanism may decrease the performance of the system. This paper represents a mathematical model for choice process migration mechanism using vector definition of global activities and migration mechanisms. This model helps in the management of process migration by considering the system's situation and also based on processes types begin to choose a process migration strategy. The result of evaluation indicates the validity of this mathematical model. The required information for this modeling can be extracted from the data structures of the operating system.

Keywords: High-Performance Computing Systems, Peer-to-Peer Computing, Process Migration, Scalability, Global Activity, Distributed Peer-to-Peer Computing Systems

1. Introduction

In traditional high-performance computing (HPC) systems, the system management has complete information about all machines and properties of whole running processes due to the closure of system based on system thinking theory [1,2,3]. Therefore, when the load distribution is impaired based on the available information about entire processes and available resources the load balancer decides to displace some processes to reload balancing. After making decisions about swiping, the process migration proceeds to transmit a process, and a new load-balancing pattern is built [4]. The main result of running the process migration is to get a new pattern of load balancing for achieving high-performance computing [5, 6].

Process migration is the transfer of a part or the entire running process from one node to another under the designed conditions. Any system using process migration should have the capability of running the process on each of the systems' node. In general, the HPC systems use a process migration for process transition and its management [3, 7, 8].

In cluster computing systems, selecting a suitable mechanism for process migration has a direct effect on the functionality and performance of the system [8, 9, 10]. If the process migration is not suitable for the computing system's properties such as candidate process and the available resource properties, then there is an increase in the process migration time. It should be noted that in cluster computing systems, the system computes only one job [11].

The intended purpose of the computing job is a process that a cluster computing system is designed to handle. To complete this activity, the system begins to create a set of processes that can be treated by the computing nodes. These created processes have plenty of communications among them and have resource dependencies that make these processes different from those executing on the local machine. On the other hand, the computing nodes in a cluster computing system are fixed during runtime; hence, it cannot be decreased or increased [1, 10]. Usually, cluster management comprises of these two concepts, and the work of process migration management is to select a uniting mechanism for process migration [12].

The process migration mechanism is selected by the special properties of the job activity (like interprocess communication type, resource dependency) and the special properties of the resource like conditions of servicing to the local and global processes. After the selection of a suitable mechanism, the transfer of process from the source machine to destination computer takes place. It was seen that if the chosen mechanism was not based on the named properties, the migration time and probability of transmission failure increased, whereas increasing process migration time may decrease the performance of the system [13].

On the other hand, when a process is selected as a candidate for migration, the process state is changed to suspend state [8, 10], i.e., all communications between the process and local as well as global processes are denied. Some mechanisms aim to reduce the freeze time, but this may increase the load volume and ruling traffic among the network. If the CPU spends time on the non-computing activity operations such as process migration, then the performance of the system decreases [13].

The process migration mechanism should be compatible with process type and the situation of resources that are being used during transmissions or resources that play a role in process migration. As only one computing job can be defined in cluster systems, all processes that exist in the system are related to this activity. In this situation, each determined property of computing activity can be expanded to all processes. If the required properties that determine the process migration mechanism are defined then they can be used for each global process. On the other hand, the available resources do not change in the cluster system, so the extracted information is always valid [1, 14].

This application can be run by changing the concept and nature of scientific applications. In other words, high-performance computing can be extended to open computing like Grid and Cloud computing. Scalability and openness are the main results of this evaluation. Thus, it can be said that open computing systems are cluster computing systems with the features of distributed systems like scalability and interconnections [14, 15].

In peer-to-peer computing systems, the system manager does not have a complete and exact view of all the available resources and nature of all the running processes due to scalability and the ability to define new global activities [16, 17, 18] at the system level.

If load distribution impairs, then scalability makes the system unable to be used as a single mechanism for process migration. According to changing the resource, the process migration management in this kind of systems requires gathering information about the source and destination machines for each process migration operation. On the other hand, this issue makes changing of resources available during the time of migration while this feature is not available in cluster systems [1, 10, 16, 18].

The most challenging issue in process migration is the concept of executing more than one global activity in a system. In a peer-to-peer computing system, the ability to process more than one global activity is available [14, 16, 18]; this indicates that in each specified moment during the whole processing time, more than one global activity's execution can be observed. This issue will create plenty of process groups, which are related to a global activity.

Each of the created groups may have different properties of process migration. These differences and the scalability of peer-to-peer computing systems do not let us select a united mechanism for process migrations, unlike cluster computing systems, which use a single and united mechanism for process migration. Regarding the special properties of each global activity and the global activity creator groups, a unit, and special mechanism must be determined.

In peer-to-peer computing systems, each global activity creates a pseudo cluster computing system [15]. This means, when a global activity begins its execution, groups of membered machines based on the ruling policy and their abilities begin processing. These sets of machines can be named as a cluster computing system that processes a determined global activity. The most significant issue is the appearance of a machine in a logical cluster system; this can make a machine to be a member of more than one logical cluster system. In a peer-to-peer system, some machines can be a member of more than one logical cluster; in this situation, the machines are

considered as logical sets. A logical machine is a set of abilities, which a machine exploits to execute a determined global activity [19, 20].

The peer-to-peer computing systems allow the use of a mathematical model for indicating the process migration parameters and modeling the mechanisms. These properties are used for indicating which parameters should be considered as a suitable mechanism for process migration. Meanwhile, in this paper, we define the required parameters for process migration, and also the basic process migration mechanisms that are used in cluster systems.

In high performance computing systems, the nature of the system is in a way that, only one global activity [16, 17, 18] is able to run and execute. The designer of the computing system, in this type of computing system, based on the nature of the application (the behaviors and the patterns of the application at a runtime) as well as the nature of the mechanism used for the load distribution, has determined an appropriate mechanism for process migration. This mechanism, figure out by the conditions of implementation of the program and the constraints governing on processes of the program builder, including time constraints, as well as the size of processes in the system, among the existing conventional mechanisms, these parameters will determine a suitable mechanism for process migration in traditional computing systems. While computing systems designed to run and execute more complex scientific and applications such as peer-to-peer systems and distribution systems and distributed Exascale systems, it is possible to run more than one global activity during the system's whole life.

Running more than one global activity causes the state of a computing node of a system member to occur at a given moment at which the computing system manager is executing more than one global activity. The processes which contain a global activity, usually have common characteristics in the context of constraints, time constraints, and size constraints. There may be a situation in the system that the system's manager, based on the decision of the load distribution, requires the migration of two (or more than two) global computational processes, each of which belongs to a global activity, Therefore, in this setting, the process migration should transfer two (or more than two) processes that have constraints and time constraints as well as different sizes based on a process migration unit mechanism. This will cause either the functionality of the process migration to be reduced or the operation of the process migration be failed.

The traditional HPC systems proceed a single global activity, this makes the ability of choosing a suitable-single mechanism for executing the process migration, although due to more than one defined global activity, in newly stated systems such as Distributed Exascale systems and Peer to Peer Computing systems, a single mechanism would not meet the systems requirements, this issue makes all the processes be more complicated.

In traditional computing systems, the system has been initiated to run and execute a single global activity for a long period of time, since this will help the process migration to explore and choose the best and most suitable mechanism for proceeding the process migration. In these systems, the best mechanism can also be defined by the given information about the process by the user; the pattern to rebalance the load of the system which is used by the load balancing manager also can influence the selected mechanism.

In newly stated systems such as Distributed Exascale systems and Peer to Peer computing systems, since more than one global activities are proceeding, the computing nodes are contributed to proceed more than one process; this issue makes the process migration to explore a suitable mechanism which is adapted to only one of the contributed process.

In computing systems designed to execute complex scientific and applications, the system should normally be compatible with traditional computing systems. The reason of this issue refers to the nature of the scientific and applied programs that run on these types of computing systems. Thus, defining a single mechanism as process migration mechanism leads to the inability of the computing system to execute traditional computing programs. The process migration must be able to choose a mechanism, based on the traditional processes defined for process migration, and based on a set of indicators, which one of these mechanisms has the most adaptive nature of the process.

The mathematical model represented in this paper enables the process migration management to choose a suitable mechanism by matching the calculated vector for the global activity and the vectors of each process migration mechanism. The nature of process properties and process migration modeling is selected in a way that can be implemented by the available information in operating system's data structure. In this type of computing systems, traditional applications are used as kernels of complex applications and scientific applications.

2. Related work

In this part, we have studied related works on improving and enhancing the performance of process migration. All available activities can be classified into three classes [21, 22, 23, 24, 25, 26, 27, 28, 29].

Class I: These are the operating systems that use multi-strategies for process transmission (migration). For example, RHODOS, which is based on microkernel [21] and message passing. Since RHODOS can manage multiple strategies, each state uses an algorithm that has the best effect on the improvement of the performance of system [23]. The logical design of process migration management in this operating system is such that it uses Pre_Copy, Total_Copy, and COR strategies based on performance and allowed time parameters [8, 23].

MACH, which is also based on microkernel [24], is another example of Class I operating systems. It has two tools for process transmission that enlists one of these two with a different pattern. One tool is the Simple Migration Server (SMS) that uses COR strategy for process migration. The other tool is the Optimal Migration Server (OMS) that provides user-level migration and supports COR, Total_Copy, and Pre_Copy strategies [24].

The Class I operating systems contain more than one algorithm for the process migration manager. The benchmark for these systems to select a specific algorithm is the time of process migration trend. In these systems, the process migration manager chooses one of the migration algorithms based on the efficiency and time allowed for the suspended process. In the systems listed in this category, the centrality of deciding whether the selected mechanism is the only time

remaining process execution, or not. In these systems, the migration mechanism tries to manage the implementation process by modifying the migration algorithm so that the process would be executed and completed at the specified time in the destination machine itself. The process migration manager considers the process intended for migration to be an abstract process that lacks communication and interaction with system and system environments and is the only standard that governs that time.

Class II: These mechanisms are a combination of some strategies. They combine some strategies to extract a united method for enhancing the performance and decreasing the disadvantages. Some examples of these strategies have been described in previous studies [25, 27].

In [25] uses a combination of Flushing, Post_Copy, and Pre_Copy for process transmission so as to achieve a logical cost and extract beneficial properties of each strategy [25]. This mechanism tries to make a general strategy by gathering all outstanding features of the three mentioned strategies. This mechanism is implemented in three phases: Pre-migration phase, migration, and post-migration. In the first phase, all address spaces of a process are transmitted to the file server. In the next phase, there are two processes which shown the migrated process and the state of the process are sent from source (host) to destination, and the process will continue running on the destination machine. In the third phase, information of address space that is private for the process is sent to the destination machine and other information is sent to file server. The migration time of this mechanism is similar to Lazy_Copy. Besides Pre_Copy and Total_Copy strategies, this mechanism has no residual dependencies [8].

Studies [27, 28] have represented a GALAXY distributed operating system that uses a combination of Demand_Paging and Pre_Copy mechanisms. During process transmission, the process is still running on the source machine, and the uninvolved memory pages are transmitted to the destination. The process is then turned into a suspend state, and the state of the process is transmitted to the destination. After that, the memory pages involved are transmitted. After completion of the mentioned steps, the process will resume on the destination machine.

In this class, there are systems that try to create a mechanism by combining traditional mechanisms for reducing process suspended time. In these systems, the created algorithms try to govern the disadvantages of a traditional algorithm with the advantages of another algorithm. This indicates that the criteria governing the algorithms used in this class system are the same as that used in the traditional process of migration. In the systems created based on these mechanisms, concepts such as dependency, inter-process communication type, and the ability to execute global activity among the computing nodes are considered.

Class III: This class includes the mechanisms that are newly designed based on four basic strategies to reduce their disadvantages. We can name Pre-record Algorithm [29] as a good example of this class. This paper represents an algorithm named Pre_Record, which is designed based on page faults, dependencies, and process suspending time indicators. Pre_Record transmits the predicted demanded pages by calculating the process runtime difference on the source and destination machines (node). This operation continues until the process is running on the source machine [29]. The operation ends when the process address space is finished and the

process is suspended. During the runtime, the process can access and observe some parts of address space, memory pages, stacks, and files. When the process continues running on the destination machine, it can rapidly access and observe that address space. The source stores memory pages and will send them after transmitting the kernel of address space. This algorithm is categorized by short suspended time and medium network traffic [29].

In this class, system parameters such as suspending time, dependency degree, and process transmission errors are considered. Although the existing systems in this category are related to dependency challenges, the challenges of communication and interactive interaction are also considered in a number of cases. The most important challenge for these systems is to focus on the pursuit of a single national activity. The formation of global activities in open computing systems makes it possible. In addition to the challenges, we need to consider the pattern of resource use.

2.1 Migration mechanism and its important parameters

Due to the importance of process transmission in distributed systems such as clusters, grids, peer-to-peer and cloud computing systems, different strategies are used for process migration. Total_Copy, Lazy_Copy, Pre_Copy, and flushing are four basic strategies in cluster computing system [3, 8, 25, 30].

2.1.1 Total_Copy

One of the most common strategies used for process migration is Total_Copy [8]. Amoeba [31] and Condor [32] are the operating systems that use this algorithm for process migration. Figure 1 represents how this strategy transmits a process.

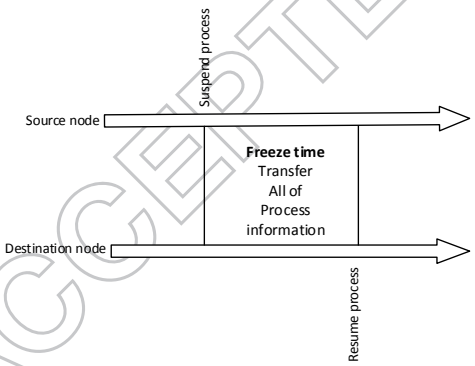


Figure 1: Process migration using Total_Copy strategy [35]

In Figure 1, at first, the process is suspended. The source machine transfers the process control and execution state and then dispatches the communication links and any other buffered messages. The source machine transfers the file descriptors, dirty file cache blocks, and pages

related to stack, code, and heap to the destination machine. Hereafter, all process information is deleted from the source machine and the migrated process resume execution on the destination node. In this strategy, sending all the process information is always preferred to continue the process executions [8].

One of the benefits of this strategy is the simplicity of implementation. When all the process information is transmitted, it will completely omit the residual dependencies. In terms of the independence factor, this strategy supports the fault tolerance feature. This means that if the source machine makes any mistake, then the migrated process can independently continue the execution without having any distribution. It is an efficient algorithm for memory as it can transmit and release the memory from source machine [25].

Transmitting all address space will increase the freeze time and delay the response time of the migrated processes to other processes. Also, it is completely clear that the more is the address space, the more time will be spent. In message-based systems, high long response time will disturb the whole system. While transmitting the whole address space in suspended time, no changes will be observed in memory pages during the migration [8, 27].

2.1.2 Pre_Copy

This strategy was first used and designed in V operating system [33] for decreasing the freeze time in Total_Copy strategy. Since all the interprocess communications in V operating system is performed by message passing, Total_Copy could not respond the system and process requirements. In this strategy, while the process is running on the source machine, some parts of address space are transmitted [8, 32]. Figure 2 explains how this strategy transmits the process?

Due to this strategy, source machine sends codes, stacks, and modified pages to the destination machine while the process is still running on it. The modified pages belong to the stack, but whenever the code changed, the modified pages are sent to the destination machine. The information is sent continuously until the number of pages stands over a threshold. After the process freezes, the source machine transfers the File descriptors, dirty file cache blocks, and all other pages related to stack, code, and heap to the destination machine. Hereafter, the migrated process resumes the execution on the destination machine [8, 32].

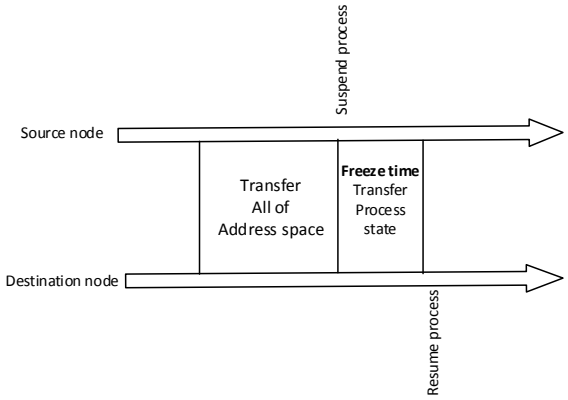


Figure 2: Process transmission using Pre_Copy strategy [34, 35, 36, 37]

As shown in Figure 2, this strategy decreases the freeze time rather than Total_Copy. This reduces the interprocess communication fault significantly. When the migration process finishes, the process becomes completely independent of the source machine. Depending on the memory accessibility pattern, some memory pages may be sent to destination machine because these pages are modified during the execution time. If these pages increase consumedly (so much), the action of sending pages will never reach the set threshold, resulting in the failure of the migration process. Although this strategy uses less freeze time than Total_Copy, it has total migration time compared to Total_Copy strategy. This issue will add an overhead and decrease the ability of process migration for supporting the load balancing. It also supports the fault tolerance as all address spaces are transferred [38]. If the address space size increases, the time of migration will increase too. The execution of this strategy is completely independent of the type of inter-process communication.

2.1.3 Lazy_Copy

Lazy_Copy strategies were first used in Accent operating systems [39] for transferring the minimum size of address space. This strategy is an instance of demand paging approach that should support remote paging in the system [39]. This strategy just transfers the process execution state and will send the address space on demand [8, 32]. Figure 3 shows the steps of Lazy_Copy strategy transmission.

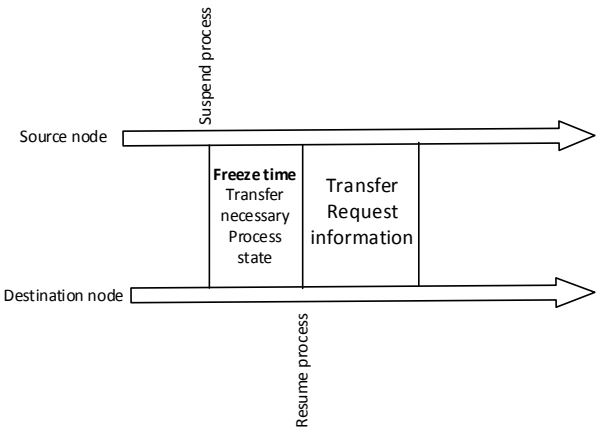


Figure 3: Process transmission using Pre_Copy algorithm [34, 40]

As Figure 3 indicates, the first process will be suspended. Execution and controlling states with some parts of address space, file descriptors, and dirty file cache blocks will be sent to the destination machine, but in this strategy, code and the stack will not be transferred. After this transmission, the migrated process will resume on the destination machine. If any page fault event occurs during the execution, the needed codes will be sent to the destination. The destination machine can send a request, and the required codes and stacks can be received from the source machine.

In this strategy, the amount of transferred data is decreased, and freeze time will be less than that of Pre_Copy strategy that prevents the transferring of the unneeded memory pages. This will reduce the overhead and increase the performance of the load balancing. This strategy is completely independent of the process size and modifies the memory pages that will not disturb the migration functionality [8].

Residual dependency is one of the disadvantages of this strategy. The address space must be able to access until the migrated process execution finishes on the destination machine. This disadvantage will turn into a bottleneck when the migrated process had many transmissions. In this case, a hierarchal page fault is needed to receive the demanded memory page, and if the source machine mistook the strategy, it will indeed result in its failure [8, 32].

2.1.4 Flushing

Flushing strategy is the first strategy to use a third entity for transferring. It was first used in Sprite operating system [41], a file-based system. In this strategy, a file server is added to the source and destination machine. The main aim of Sprite operating system for representing this strategy was to overcome residual dependency [8].

This strategy uses the concept of virtual memory in which it completely depends on how an operating system defines the virtual memory [42]. In figure 4, the transmission steps are shown.

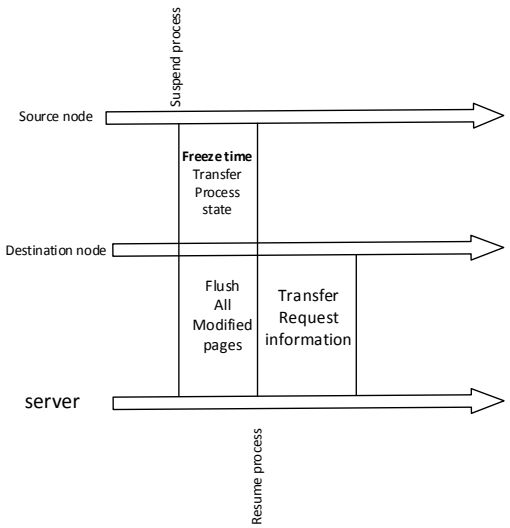


Figure 4: Process transmission using Flushing strategy [32, 34]

As Figure 4 shows, the process is first suspended, and then the execution and controlling states with brief information about processes, such as communicational links, buffered messages, file descriptors, and dirty file cache blocks are sent to the destination machine. However, no parts of the code, heap, and stack are involved in this transmission. All memory pages including the information in the stack, heap, and cache block will be flushed from source machine to destination. When "continue message" is sent to the destination machine, the process resumes the execution. However, when a page fault happens, the migrated process rapidly receives the demanded code pages from the source machine followed by the receiving ability of the heap and stack pages.

This strategy has no residual dependency, and the freezing time significantly decreases compared to the Total_Copy strategy. Flushing strategy is suitable for systems that use file passing as inter-process communications. Hence, if this strategy is applied to operating systems based on message passing when the freeze time is too short, it will not perform well, and the transmission may have many overheads [8, 32]. On the other hand, using this strategy in operating systems that are using File passing as interprocess communication requires implementation of the concepts and mechanism related to file.

For speeding up the accessibility of memory pages while using a file server, the file system must be optimal. As Sprite operating system uses file passing-based interprocess communication, the accessibility of files increases and page faults respond faster. This strategy is independent of the

process sizes, and the modified memory pages for process migration or the fault tolerance feature is supported in such systems.

3. Basic concepts

In this part, the basic concepts of peer-to-peer computing systems have been studied using the process migration mechanism and the global activity concept. Studying the concept of global activity enhances the ability to perceive the nature of cluster computing systems compared to peer-to-peer computing systems. On the other hand, a mathematical model based on the global activity concept in peer-to-peer systems is going to be presented by studying the process of migration mechanisms.

3.1 Process indicator and parameter types influencing process migration mechanism

Choosing the right unit with the duty of determining the properties and type of candidate of the migrating process is a crucial task. There are two common approaches for decide the unit; these approaches consider factors like when and where. The term 'where' is used for destination and 'when' indicates the exact time of process migration. In the first approach, the process migration time and destination machine indicate the parameters of the process type, which perform the duty of a load balancer [8]. In the second approach, the process migration management is the determining factor of these parameters. The ruling policy beyond the process migration management indicates the approach of this decision [4, 31]. Along with determining and choosing the approach, the considered unit must be able to perform the following tasks

A. The considered unit must be able to bring up a set of indicators as migration parameters, and they must be determined based on the process properties. These parameters must indicate whether a process is required to migrate or not.
B. The considered unit must be able to determine the execution time of migration and indicate which events or process migration parameters are required to check whether a process should migrate or not [43].
C. The considered unit must also indicate the acceptable resource properties in process migration. This unit must be able to describe the membered machines in the system based on their acceptable resource properties for a successful computing activity. The acceptable resource parameters should be in a way that the process migration management can map the properties and migration parameters among the mentioned sets, which determines the migration destination [1].

In the cluster computing systems, the load balancer does A, B, and C duties. In these systems, the load balancer selects acceptable response time as one of the process migration parameters. In these computing systems, the process properties will not change during the running time as all processes belong to a global activity. On the other hand, the duty of computing systems (Duty C) is fixed and does not change during the system's running time because of the non-scalability of these systems. The load balancer, based on static or dynamic policies, selects the load

distribution time [43, 44] while the process properties change during the running time in peer-to-peer computing systems. The changes in the properties of the process are completely based on the scalability and ability to define more than one global activity in peer-to-peer computing systems [1]. So, the migration parameters can also change during the running time in peer-to-peer computing systems, and the changes in computing resources affect the set of acceptable properties for the resources [15].

In peer-to-peer computing systems, changes in both acceptable resource properties and migration parameters result in process migration management and load balancer that are beneficial during the process migration trend. The load balancer must concentrate on extracting acceptable resource properties that can be extracted by the resource discovery. The process of migration management must concentrate on the parameters set for migration. Related to the concept of peer-to-peer computing systems like cluster computing systems, the duty of determining the migration time can be passed onto the load balancer [8].

By selecting the mentioned scenario, the process migration management must determine the required parameters to:

- Decide whether the process is required to migrate or not.
- Decide which strategy is suitable for the process of migration in a specific global function among all the available strategies in cluster computing system.

The definitions of the process migration parameters must be based on the properties of the global activities based on the idea that these parameters should be applied to global processes.

By studying the process migration strategies [8, 30] and load balancing mechanisms used in cluster computing systems [5], the below parameters are finalized based on the type of processes and process transmission trend. Furthermore, these parameters used in process migration management can be studied under two classifications.

Class I includes the main properties during the execution time. These properties have meaning and concept just during the execution time and cannot be evaluated in other instances. These parameters can also change during the running times [45].

I. Interprocess communication number (IPC#): It describes the number of inter-process communications of a process with other processes.
II. Priority: It indicates the first and foremost execution of a process in a global activity rather than other processes.
III. $Memory_{Request}$: It indicates the required memory of a process for completing its execution.
IV. $Resource_{Dependency}$: It illustrates which local or global resource is required by the process.

Class II includes the intrinsic properties that are completely fixed and do not change over time. These properties are constant at all times [45].

I. Process size: The number of memory pages allocated to the process

II. IPC_{type}: Its intended propose is to determine the kind of communications among processes. In this paper, the IPC_{type} is limited to three types, i.e., file passing, message passing, and distributed shared memory
III. Transparency: It indicates the process that is called by a nickname or address. When named by its address, the process is fully transparent.
IV. Reliability: It addresses whether the process can execute on all systems or not, and whether the system is based on processing machine?

The set of these parameters is known as process descriptors.

On the other hand, the process migration management should be able to describe the available process migration mechanism in cluster computing based on the set of parameters. This issue helps the system in determining the mechanism based on factors required in the migration mechanisms in the cluster system.

Migration parameters were studied by Medina et al. [8], who classified the parameters into three classes.

Class I includes time-related parameters, migration costs, and process-related properties. The parameters that are important in choosing the process migration strategy because of the main reason of creating different processes include the reducing freeze time [4, 8]. They are defined below:

I. $Initial_{time}$: The time spent during the process starting from migration until it again resumes its execution on the destination machine.
II. $Total_{time}$: The time spent during the process starting from migration until the whole information of the process has been sent to the destination machine.
III. $Freeze_{time}$: It is the duration when the process is suspended and the migrated process cannot respond to the requirements.
IV. $Migrate_{cost}$: It refers to the costs of transferring the state of a process.
V. Memory $Migrate_{cost}$: It refers to the cost demand paging from process address space.
VI. IPC_{type}: It chooses the strategy that is both suitable and influential.

Class II involves locality properties, which indicate the dependency of the migrated process to the source machine. These properties are commonly named as “residual dependency,” and it means that some parts of process information will stay on the source machine even when the process migration is completed. In this case, this machine can recover the process using the remaining information on the source machine whenever the destination machine fails [8].

Class III includes the properties that are related to its transfer in accordance with the operating system’s aim. In other words, the main aim of using a determined strategy is to achieve transparency and reliability during process migration [31, 46].

Process migration management must be able to describe the process descriptors in terms of the global process parameters. In this case, both process descriptor and process migration mechanisms must be the same and can be defined by the global computing activities. Process

migration management should be decided on the basis of the parameters of global activities, which require an exact view of the global activity and its ruling properties.

3.2 Global activity

In peer-to-peer distributed computing systems, each computing node of the system (member) begins to initiate a computational activity due to the decentralized nature of the distributed system. This computational activity may have some requirements that cannot be fulfilled by the local computing start-up machines. However, elements of the peer distributed computing counterpart systems are based on the mechanism, which discovers the resource and development of the system to meet the demands of the created computational activity [47, 48]. In this case, the page for computational elements responds at the system level and requests for processes in this computational activity. The set of processes created in the verdict is related to the computational activity and is known as a global activity. Each member of the Global Activities page carries out the part of the activity related to the global activity with an intention to complete it within the page.

For illuminating this issue, a peer-to-peer distributed computing system is considered [16, 17]. The computational element "a" can start to compute a computational activity. The existence of the set of requests in computational activity causes the resource management to discover the resource that is capable of answering requests. This further causes the collection of processes related to the computing activity to be transferred to the new computing element in order to continue the process, and add a new element in the page of the computational activity created in the machine. Creating computing pages for the global activity using all the processes related to the activity is completed. According to the transmitted processes, these can be implemented in parallel with each other. Therefore, a set of processes related to a global activity is running in a page on global activity at a given time.

Processes related to global computing on a global activity page are the parts of the activity. These processes result when a part or all of a global activity is running on a computing element, and there is a process (or processes) in which the local computing element is capable of answering the requests of that process (or processes). Moreover, the system management element is based on its policies and has discovered the resource and developed a computational system to respond to the process request (or processes). In this situation, the system management element adds to the system's new machine to move the processes requiring a new resource into the computing machine. Moreover, the processes are transferred to the new computing machines due to the redistribution of the process by the migration mechanism. If 'W' represents the set of all processes related to the global activity 'I' in the system, then the member processes of W have certain properties that make it possible to distinguish the local processes in the system. Eq. 1 can express the generating space of the computational processes.

$$Global_{Activity}:(Process, P_{Remain}, IPC, Usage, Size, Resource_{dependency}, time < LB, Migration, RD >) \quad \text{Eq.1}$$

As seen in Eq. 1, the generating space of the computational processes is defined by the two sets of $Process$,P_{Remain}, which are of the type of data structures defined on the basis of the process. These two sets are the kind of process whose information can be extracted from the data structure of the process. As stated in the generating computing process space, [the] three RD, Migration, and LD operators create the global process spaces based on [the] two sets of processes mentioned before. According to Eq. 1, the global process spaces are described by five properties, i.e., IPC, $Usage$, time, $Resource_{dependency}$, and $Size$. These five properties are considered by the nature of computational processes in terms of the three LB, Migration, and RD operators that create the space for computational processes. The IPC feature is a vector property, which indicates the situation of communication between processes (processes) forming a global activity. $Usage$ Ffeature is a scalar property, which indicates the average time usage of a CPU that comprise global activity in computing elements. $Resource_{dependency}$ feature is a vector property that indicates the relationship between each process with the required resources for the global computing process. The $Size$ feature is also a scalar property that indicates the size of the global computing process.

The concept of $Usage$ from process migration management viewpoint and the load balancer viewpoint is different. From the load balancer viewpoint, $Usage$ of a global process is the time usage of central processing unit (or also the network bandwidth or memory allocation). From the process migration management viewpoint, the concept of $Usage$ is the modified memory pages on the source machine. In process migration management viewpoint, the $Usage$ concept includes the network bandwidth, central processing unit's allocation of source machine, and also the execution of system call on the destination machine. When a global process begins to modify a memory page in source machine, it means that all processes use destination processor, source processor, and the network bandwidth.

The $Resource_{dependency}$ concept from load balancer view is the resources required to continue the execution. From process migration management's viewpoint, the $Resource_{dependency}$ concept emphasizes on the failed process transmission as to whether it can continue the execution or no. The intended purpose of this issue is that if the source machine fails during the migration then whether it can continue the execution of the migrated process on the destination machine.

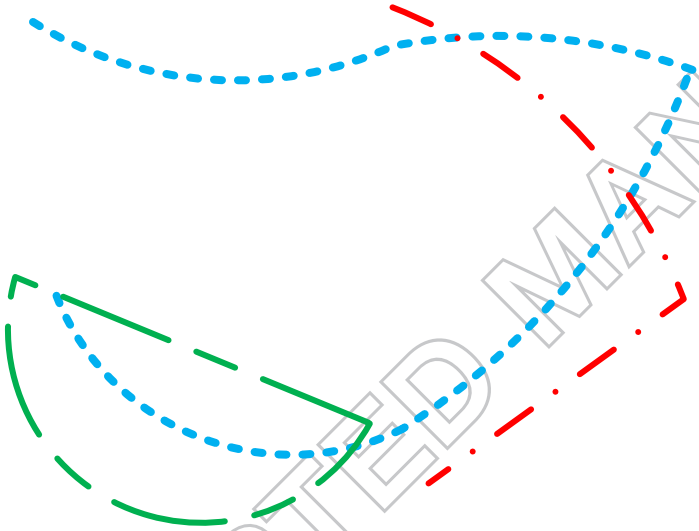
The Time feature is a vector property that indicates the effective times on the process time. The space that governs time is a composite space. In Eq. 2, it describes the space of Time.

$Time: < time_{space}, time_{global}, time_{local}, < Do, Waiting >, < Migrate, communicate \gg$ Eq.2

The concept of Time, in Eq 2, defines the three spaces of the process time and the time of global operations. The operators of activity, process waiting, process migration, and process communication can be defined as the concept of Time.

Consider computer A is a member of peer-to-peer distributed computing systems [16, 17]. According to the capabilities of the machine and the policies for creating a page related to the global activities of the system management element, the computational element “an” is present at a specific moment in more than one global activity page. Cluster systems have only one computational element, which can create global computational activity. In other words, if Machine A is defined in the cluster system, then it is only present on the page of the global activity. The presence of a machine in more than one global activity enables it to run on a computational element in more than one type of processes. This causes the system management element in Machine A to have the global computing management mechanisms appropriate to the nature of global computational processes. One of these global computing operations is the process migration mechanisms.

Figure 5 indicates the position of the computing element “an,” which is shown at a given moment in the peer-to-peer system in the context of global activities.



computational element of the system member of the peer-to-peer system begins partly in Machine A and ends in the starting phase of the global activity.

If the distribution status changes in the peer-to-peer system at $t = t_z + \beta$, then it could be due to the beginning or end of the global activity, and consequently the change can be observed in the system status from the load balancer or system expansion.

Now Eq.3 and Eq.4 are based on Eq.1 and the set of indicators describing the process and migration indicators.

$$Migration_{unit_{Process}}:\Bigg[\&\Big[\Big(IPC\#_{time}^{\substack{t=Next\ Migration\ Load \\ t=Migration_{load}}}\rightarrow IPC_{vector}\Big)as\ scaler\Big]\&$$
$$\Big[\overbrace{Average}^{}_{\text{Priority}}\rightarrow \text{Time}_{\text{vector}}\ as\ scaler\Big]\&$$
$$\Big[Memory_{request_{Size=Request\#}^{Direction=}}\overbrace{vector}^{}\rightarrow Resource_{dependency}\ as\ vector\Big]\&$$
$$\big[Resource_{Dependency}\rightarrow Reource_{dependency}as\ Creator\big]\&\&[Size\rightarrow size\ as\ creator]$$
$$\&\big[IPC_{type}\rightarrow IPC_{vector}as\ vector\big]\&$$
$$\big[Transparecny\rightarrow Resource_{dependency}as\ vector\big]\&\&$$
$$\big[Relability\rightarrow Resource_{Dependency}as\ vector\big]\Bigg]\ Eq.3$$

As described in Eq.3, it maps the process descriptors to the process generator space. Information about the process descriptor parameters can be extracted from the structure of the kernel operating system. The process migration for each global computing process attempts to create a data structure corresponding to the generating space of global computing processes. In Eq.3, transparency is in the form of a vector; hence, if it is equal to the processing address then it is also equal to the negative vector, and if it is equal to the process name then it is equal to a positive vector. In the case of a reliable variable, if the process is dependent on the machine then it is equal to the negative vector, and if it is independent of the machine then it is a positive vector.

If IPC type variable is identical in both origin and destination machines, it is equal to a positive vector; if not, then the vector is negative. A negative unit vector will be implemented for each source machine memory requirement, and a positive unit vector will be implemented for each memory requirement during the process migration to the destination machine. The process migration management, after executing Eq.3, creates three vectors and two scalar values for each computational process. If the information about each of the three vectors or two scalar values of the computing process generating space in Map, as shown in Eq.3, is not complete, then the migration management will obtain the process of the information from the corresponding data structure.

$$Migration_{unit_{mechanism}}:\Big[\Big[\big(Initial_{time},\text{total}_{\text{time}},\text{Freeze}_{\text{time}}\big)\rightarrow time_{vector}\, as\ vector\Big]\&\ \Big[\big(Migrate_{cost},\, Memory\ Migrate_{cost}\big)\rightarrow Resource_{dependency}\, as\ vector\Big]\&\ \Big[IPC_{type}\rightarrow IPC\ as\ vector\Big]\Big]\ Eq.4$$

In Eq.4, the process migration management maps the migration parameters to the process generator space. In Eq.4, the three variables $Initial_{time}$,$\text{total}_{\text{time}}$,$\text{Freeze}_{\text{time}}$ are mapped to the time vector space. These three variables are actually vectors with a size equal to the weight of the vectors, and their direction is always in the direction of the negative unit vector. The two variables, i.e., $Migrate_{cost}$, $Memory\ Migrate_{cost}$variables are mapped to the $\text{Resource}_{\text{Dependency}}$ vector space, which is also a vector with weights equal to the size of the two variables, and their direction is also always in the direction of the negative unit vector. The migration management is a process for shaping the generating space of the characteristics of global computational processes for each cluster process migration mechanisms, in addition to the usage of the mapping Eq.4, the information contained in Section 3.3 is also used.

The process migration management, based on Eq.3 and Eq.4 and the information in section 3.3, will create two generating spaces. The first generating space is equal to that of the global computational process. It is created for each computational process. The second generating space is equal to that of the migration mechanisms and is calculated for each of the traditional process migration mechanisms.

3.3 Summarizing mechanisms based on the defined parameter

Table 1 summarizes the mentioned strategies based on five parameters: process size, interprocess communication type, resource dependency, usage, and time bounding.

Table 1. Modified using algorithms in clustering systems based on properties of process activity generating space

	Process size [45,8]	IPC type [44,8]	$\text{Resource}_{\text{dependency}}$	Usage	Migration time [8]	Time
Total_Copy	Is important	No dependency	No dependency [8]	Not important	high	There are time bounding based network limitations [8]

Pre_Copy	Is important	No dependency	No dependency [8]	Is important	Very high	Time bounding based on network limitation [8]
Lazy_Copy	Not important	No dependency	Depends on source machine [34]	Not important	Low	Time bounding based on data's dependency [34]
Flushing	Not important	Depends on IPC type	No dependency [34]	Not important	Moderate	Time bounding based on data's dependency [34]

Table 1 indicates the usage parameter as mentioned in 3.2. The process migration view includes network's bandwidth and CPU usage [49]. In process migration mechanisms, these two parameters count the modified pages on source machine [8, 50]. When a process requires modifying the accessibility on the source machine in process migration mechanisms, the performance of this activity is based on the central processing unit and network' bandwidth. As mentioned in 3.3.2, the Pre_Copy strategy still executes on the source machine and also uses these parameters due to the transmitting process to the destination machine [49, 50].

The IPC parameter indicates the type of interprocess communication that is used in the system. Flushing strategy requires a system that supports the concept of the file for process migration. If the strategy is used on the systems that support the file concept, it demands special file mechanisms [39].

Parameter Resource $_{dependency}$ from process migration view indicates the dependency of the process to the source machine. One of the aspects of this parameter from process migration view is the ability to continue the execution of the process when the accessibility of source machine fails. Some process type relations to the global activities are in a way that makes the process dependent on the source machine. In this case, this strategy should perform the migration process in the peer-to-peer system in such a way that the process dependencies can be managed. In Flushing strategy, the moderate and management features will be advanced. Total_Copy and Lazy_Copy strategy, according to all information transmission, must note the process size. The process migration time depends on the transferred information between the suspending time and the execution time. Pre_Copy strategy, while transferring all address space and multiple sending modified pages, need much more time for migration [50].

4. MDPM mechanism

Due to current load state, load balancer decides to reload balance in peer-to-peer-distributed computing systems. In these computing systems, process migration management should decide and select a suitable mechanism, which is based on the five parameters related to generating space of the global process. To overcome this issue, the process migration must be able to describe the mechanisms of process migration based on Eq.4. In addition, process migration must use the information mentioned in part 3.3 related to the process generator space. The main reasons for emphasizing on migration mechanisms are the concepts of global activity and peer-to-peer computing systems. The P2P system discussed in this paper is computational type, and thus, the aim of these systems is to execute each global activity in minimum time. For example, each indicated global activity mentioned in Figure 5 initiates a computing clustering system that can be implemented in both centralized and decentralized manner, and is related to the global activity concept based on the computing region creation mechanisms.

Computing region management uses the clustering management law to manage its own activities. So, adjusting and matching the migration mechanism related to the global process properties must be based on a powerful mathematical model. In this paper, vector algebra is used as the mathematical model. By using this powerful tool, the process migration management begins to adjust and match the properties of the sets of process descriptor and migration parameters to select the best process migration mechanism for global computing process. This paper uses vector algebra for describing the five elements in a process generator space as well as the mentioned mapping.

As mentioned in Eq.1, the process generator space is described by IPC, Usage, time, $\text{Resource}_{\text{dependency}}$, and size of a process. Eq.3 explains how the process migration management starts mapping the process descriptor parameters to the process generator space. Other required information for building process descriptor can be extracted from the data structure of the operating system. As per Eq.4 and part 3.3, the process migration management begins to build a new data structure for each computing region based on the previously discussed process migration parameters. So, if the load balancing became unbalanced in time $t = RLB_j$, then the process migration base on W and V sets (Eq.5 and Eq. 6) starts to reload the balance of the system. So if the load distribution becomes unbalanced, the process migration management is decided by considering the V and W sets (Eq.5 and Eq.6).

$$W = Global\ Process_{description} \mid \forall i \in Global\ Process_{description} \therefore GPD_i = \left[\begin{matrix}\{Global\ Activity_{Creator} Set\ ,\ Global\ Activity_{remain} Set\},\\ \{Size_{GDP}, Resource_{dependency_{GDP}}, time_{GDP}\ set, IPC_{GDP} Set, Usage_{GDP}\}, Migration\end{matrix}\right] \quad \text{Eq.5}$$

Eq.5 explains the description of the data structure of each global computing process based on process generator space parameters and the ability to execute the migrated process. Each data structure has three vectors ($Resource_{dependency_{GDP}}$, IPC_{GDP}, and $Usage_{GDP}$) and two scalar variables ($Size_{GDP}$ and $Usage_{GDP}$).

$$V = Mechanism_{desciption} | \forall i \in Mechanism_{desciption} \therefore MD_i = \left[\begin{matrix} \{Initial\ Process_{State}, Next\ State\ Process_{state}\}, \\ \left\{ \begin{matrix} Process_{size} Range, Resource_{dependency}\ Condition\ Set, time_{condition}\ set, IPC_{Condition} Set, \\ Usage_{condition} Range \end{matrix} \right\}, \\ Migration \end{matrix} \right]$$

Eq.6

Eq.6 indicates the data structure description of each migration mechanism based on five parameters of process generator space for the migration.

In Eq.6, the intended propose of the data structure calculation for each process migration mechanism is based on process generator space and process migration management controlling duty, and is the special implementation of each process migration mechanism. In this paper, four implementations of process migration mechanisms are discussed.

Table 2. List of Mechanism implementations

	Name of Mechanism	Implementation name	Comments
	Total Copy	Amoeba [24]	• Does not support virtual memory concepts • Is not a real-time operating system • User level process migration • Microkernel design • Location independent addressing
	Pre-Copy	System V [33]	• Supports virtual memory concepts • Suitable strategy for real-time activities
	Lazy_Copy	Accent [38]	• A low-cost migration using strategy • Virtual memory stays at source machine • Supports remote paging concepts
	Flushing	Sprite [40]	• The selected IPC is file passing • The main aim is to

			prove reliability • Emphasizes on transparency degree and functionality

As indicated in Table 2, only one implementation is considered for each process migration mechanism. The intended purpose of studying these mechanisms is to know about the process of implementing the mechanism.

The $Usage_{GDP}$. variable is a runtime variable, whose size is measured and calculated by the process management. For extracting this variable, process migration management can calculate this variable by using the process for average allocated times. This process is a member of a global activity based on the available information extracted from the local operating system's data structure. In MDPM mechanism, process migration management stores the process information and the average allocated time to each process of the global activity in operating system's data structures.

Generator space, as demonstrated on Eq. 6, is defined on Initial $Process_{State}$, next s_{tate}, $Process_{state}$ sets. The first set indicates the state of the available process before starting the migration operation, whereas the second set indicates the state of the available process after the migration operation has been done. The state of the process explains the state of load balancing in the system, allocated time to the process, and the waiting time of the process on the local machine. This data structure is described by five process generator space parameters. The $Size_{GDP}$ variable indicates which related mechanism can be matched to which size of the processes. Table 3 shows the span of suitable process size for each mechanism.

Table 3. Process Size limitation for each mechanism

Implementation name	**Size Condition**	**Comments**
Total copy	There is a limitation in the size of the process	Process size depends on the acceptable migration time Process size depends on the network communicating limitations
Pre Copy	There is a limitation in the size of the process	Process size depends on the network communicating limitations
Lazy Copy	There are no limitations in the size of the process	Transferring depends on the data dependency

Flushing	There are no limitations in the size of the process	Inter-processing communication is the priority

As indicated in Table 3, Total_Copy and Pre_Copy mechanisms have size limitations (bounding) as an important parameter. In these two mechanisms, the process size limitation bounding is dependent on the whole program size and bandwidth limitations for the network. Also, the time limitation bounding in these two mechanisms depends on the acceptable migration time. Considering Omega as the time of each migration and Teta as the total acceptable migration time, 'Size*Omega<<Teta' must be satisfied to use one of these two mechanisms for process migration. In Total_Copy and Pre_Copy, process migration mechanisms and process size have direct effects on acceptable migration time. In Lazy_Copy and Flushing strategies, process size limitation bounding has less priority. In these two mechanisms, data dependency and inter process-communicating pattern determine size parameter. In Lazy_Copy mechanism, dependency pattern between program data characterizes size parameter limitation, whereas in Flushing mechanism, the interprocess communication indicates the maximum size and spanning ability of the processes. In these two mechanisms, i.e., Total_Copy and Pre_Copy, the process migration time must be less than the total acceptable migration time.

$Resource_{dependency}\ Condition\ Set$Variable is a vector, which indicates the conditions of the mechanism about the resource dependency and another process dependency. This vector can be calculated for each process *m* in which the process can be a computing one or the vice of a mechanism. In each of V and W sets, $Resource_{dependency_m}$ is a finite vector, and it can be calculated from the summation base in Eq.7 as shown below.

$$Resource_{dependency_m} = \left[Resource_{dependency_{IPC_m}} + Resource_{dependency_{Memory_m}} + Resource_{dependency_{IO_m}} + Resource_{dependency_{file_m}}\right] \quad Eq.7$$

In Eq.7, $Resource_{dependency_m}$ vector is calculated by the summation of the process *m* dependent on each resource elements, i.e., memory, input, and output files. Each of the four creating vectors of vector $Resource_{dependency_m}$, is an orthogonal vector. Considering the period $[RLB_i, RLB_j]$ of time, in which the load balancing state does not change, the $Resource_{dependency_m}$ vector is calculated from Eq.7. For each device of mechanism '*m'* or process '*m,'* the $Resource_{dependency_m}$ vector is only calculated once. If the load balancing state has been changed, then this vector must be recalculated.

For each $Resource_{dependency_{m_i}}$ in which *i* can be any of file, memory, process and I/O resources, if process or the vice-process were in a way that requires resource *i* in source machine for continuing the execution, the orthogonal vector with $Resource_{dependency_{m_i}}$ vector must be added to a vector that has the same direction with the negative unit vector. The length of this vector is equal to the required time for executing this process in resource 'i' on the source machine. If the process 'm' requires to access resource 'i' on the destination machine, then the orthogonal vector with $Resource_{dependency_{m_i}}$ vector must be added to a vector that has the same direction with the positive unit vector. The length of this vector is same as that of the $Resource_{dependency_{m_i}}$ vector in the source machine.

In peer-to-peer computing systems, the reason for doing process transmission by process migration management is to reduce the response time. This system is related to the global activity and must be executed in short time as possible. In peer-to-peer computing systems, if the long queue of created processes or the CPU is busy, the process migration must rebalance the load of the system. In other words, the main aim of the process migration is to achieve a computing unit. For each global computing process, β vector can be defined as an ideal computing vector. Like $urce_{dependency_m}$, this vector is an additional vector that can be calculated from Eq.8.

$$\beta = [\beta_{Process} + \beta_{IO} + \beta_{file} + \beta_{memory}] \qquad Eq.8$$

Vector β indicates the requirements of a global process of the local operating system, which the global process executes. The direction of this vector is always positive and has the same direction with positive unit vector having the length equal to all the required time of each resource from the local operating system. In an ideal condition, the local operating system can respond to all the requirements of the element created by β vector; however, in this case, the global computing process will be completed on the local machine. The information about the β vector can be extracted from each of created moments, and the migration time can be transferred from the source machine to the destination machine. This data extraction is accessible by the data structures of the required process.

The length of the $Resource_{dependency_{m_i}}$ vector in $[RLB_i, RLB_j]$ period is always changing. This issue changes the $Resource_{dependency_m}$ vector during the mentioned period, which happens during the $t = RLB_j$ moment, where the load of the system remains unbalanced. The process migration management requires the information about the $Resource_{dependency_m}$ vector for each global process. For estimating the $Resource_{dependency_m}$ in the $[RLB_i, RLB_j]$ period, the ALPHA vector based on Eq. 9 is being represented:

$$Alpha_{Resource_{dependency_k}} = \left[\sum_k (\beta_k | Resource_{dependency_k}) Resource_{dependency_k}\right]_{RLB_i}^{RLB_j} \overbrace{\therefore}^{To\ somehow\ that}$$

$k \in \{Memory, file, IO, Process\}$ Eq.9

Eq. 9 tries to find a good estimation for β vector by creating vectors of$Resource_{dependency_m}$. ALPHA vector indicates how the $Resource_{dependency_m}$ vector was during the$[RLB_i, RLB_j]$? Period. This title is affected by the β vector, indicating the ideal state of the requirements of process 'm' on the local machine. This vector indicates which resources must be available in what capacity for the successful execution of the process '*m'* in the local machine. On the other hand, $Resource_{dependency_m}$ vector indicates the resource dependency of process '*m'* on different resources of source and destination machine. If the estimation of β vector by the $Resource_{dependency_m}$ vector was in such a way that it reduced the size of $\left\|\beta - Alpha_{Resource_{dependency_m}}\right\|$, then the resource dependency of process '*m'* on the local machine can respond to all the requirements of the global process '*m.'* If the value of $\left\|\beta - Alpha_{Resource_{dependency_m}}\right\|$ was a big number, then the resource dependency of process '*m'* is in a way that the local resources cannot respond to its requirements.

The Alpha vector describes the $Resource_{dependency_m}$ vector situation during the $[RLB_i, RLB_j]$ period. Process '*m,'* whether created on the local machine or in $t = RLB_i$, is transferred to the local machine' paying special attention to this issue is so necessary. For the process migration management unit, the situation of the $Resource_{dependency_m}$ vector during the $[RLB_i, RLB_j]$period is important because the process situation before the $t = RLB_i$, was studied in the previous situation of enabling the process migration management unit.

The IPC vector can be calculated by Eq. 10 by using the summation of its constituting orthogonal vectors.

$$IPC_m = (IPC_{global} + IPC_{local} + IPC_{OS})\ Eq.10$$

Eq. 10 indicates that the IPC_m vector is affected by its constituting vectors. In this vector space, IPC_{global} indicates the connection between process 'm' with other global process - IPC_{local} vector, which further indicates the connections and communication of process '*m'* with the local processes except the local operating system. The IPC_{OS} demonstrates the connections and communications of the process '*m'* with the local operating system. By changing the constituting vectors of the IPC_mvector during the $[RLB_i, RLB_j]$ period, the estimation of the vector from Eq. 9 cannot be described. The $Alpha_{IPC_m}$ vector is introduced, which estimates the IPC_m vector during the mentioned period of time.

According to Eq. 2, the time vector can be calculated by the summation of its constituting orthogonal vectors. Eq. 11 indicates the time vector:

$$Time_m = \left(Time_{local_m} + Time_{global_m} + time_{operation_m}\right)$$
$$\left| time_{operation_m} = \left(Time_{migration_m} + Time_{loadbalancing_m} + time_{RD_m} + time_{Allocation_m}\right)\right| tim$$
$$e_{migration_m} = \left(freeze_{time_m} + Initial_{time_m} + Migrate_{time_m}\right) \text{ Eq.11}$$

Eq.11 demonstrates the time vector of the process '*m,'* indicating the rolling situation of the time of the process '*m'* during the $[RLB_i, RLB_j]$period on the local machine. $Time_{local_m}$ Sshows the time duration of the process '*m'* in the local machine. The weight of this vector is equal to the allocated time of the process *m* in the local machine, and its direction is always the same as that of negative unit vector. The $Time_{global_m}$ vector indicates the time situation of process *m* in a system that the global activity is executing.

The weight of this vector is equal to the time that process *m* requires for completing its executions, and its direction is the same as that of the positive unit vector. $Time_{operation_m}$ Vector is the summation of the load balancing time, migration, discovery, and the allocation time of a resource. All their directions are the same as that of the negative unit vector, and the weight is equal to the required time for completing their own executions during the $[RLB_i, RLB_j]$ period. The $time_{migration_m}$ vector is a summation of three vectors.

The direction of $freeze_{time_m}$' $Initial_{time_m}$, and the $Migrate_{time_m}$ vectors are similar to that of the negative unit vector, whereas their weight is equal to the allocated time for completing their execution by the process migration management. As the estimation presented for $Resourc e_{dependency_m}$ in Eq.9, we represent another estimation of the time vector. $Alpha_{time_m}$ Vector is an estimation of the process '*m'* vector during the $[RLB_i, RLB_j]$ period.

According to the represented information about the *Usage* and *size* variables, the given information in the Table 3 and the estimated vectors in the t= RLB_j, the process migration management for each global process and mechanisms in Table1 has the set of information < $Alph a_{Resource_{dependency_m}}$, $Alpha_{IPC_m}$, $Alpha_{time_m}$, $Usage, Size$ > . The calculated information are about the situation of the global process and mapping state to each mentioned process migration mechanisms in cluster computing system during the $[RLB_i, RLB_j]$ period of time.

Considering the process migration management for process migration operation, it selects the global process ξ. In this case, the process migration management uses Eq. 12, and the information was given in the Table 1, which can decide which process migration mechanism is suitable for the current migration operation.

$(\forall \xi \in Global\ Activity\ and\ \gamma \in Cluster\ Migration\ mechanism)$:

$$W_X.\left[\frac{\left(\left|\overrightarrow{Alhpa_{X_\xi}}\right|.\left|\overrightarrow{Alhpa_{X_\gamma}}\right|.\cos\rho_x\right)}{\left|\overrightarrow{Alhpa_{X_\xi}}\right|.\left|\overrightarrow{Alhpa_{X_\gamma}}\right|}\right] \therefore X \in \{Resource_{dependency}, IPC, Time\}\ Eq.12$$

Eq.12 begins to calculate the angle along $\overrightarrow{Alhpa_{X_\gamma}}$ and $\overrightarrow{Alhpa_{X_\xi}}$ vectors for each $urce_{dependency}$, IPC, and $Time$ vector property. In the Eq.12, $\cos\rho_x$, is a variable between 0 and 1. If we consider only one property of set X, then it is affected by the process migration operation. The value of the $\cos\rho_x$ is 0, which means that the γ mechanism is the most suitable for process migration of global process ξ. $\overrightarrow{Alhpa_{X_\gamma}}$ and $\overrightarrow{Alhpa_{X_\xi}}$ vectors have the same directions. If $\cos\rho_x$ has the value besides zero, the W_X factor indicates whether the γ mechanism is suitable for this migration operation or not. The W_X factor can be calculated by the Eq.13.

$$\mathrm{W_x} = \frac{X_{Operation\ time}}{\xi_{operation\ time}}\ Eq.13$$

The W_X factor indicates the importance of the property X for the execution of the global process. For calculating this factor, we can use the kernel data structures of the operating system. Moreover about the IPC property, the W_X factor indicates the required time for executing the IPC divided by the whole execution of the process ξ in the local machine. About the *Time* property, the $\mathrm{W_x}$ factor indicates the allocation time to global process ξ for migration. The W_X factor of all the information about the process migration of global process ξ are available, then the process migration management divides the average required time for process migration as a type of the ξ global process within the whole execution time for the processing of the global process ξ, and finally calculates the W_X factor. If the information about the ξ types of global process are not available, the W_X factor is considered to be equal to 1. About the $Resourc\ e_{dependency}$property, the W_X factor indicates the number of responded resources by the local operating system divided by the whole number of resources denoted by the global process ξ.

The result of Eq.12 is a number between 0 and 1. Being a part of the result of Eq. 12 means the mechanism γ is not suitable for the process migration ofξ. Although Eq. 12 determines whether the mechanism γ is suitable for process migration of ξ or not by only considering one property, this equation considers that there is only one property in the set X that determines the suitability of the mechanism. This issue happens when the process migration management considers all five $Alpha_{Resource_{dependency_m}}, Alpha_{IPC_m}, Alpha_{time_m}, Usage, Size >$ properties for the process migration using the mechanism γ. Eq. 14 represents a global function for deciding the suitability of each mechanism.

$$\boldsymbol{\vartheta} = \frac{\left(\sum_i \boldsymbol{W_{X_i}} \cos \boldsymbol{\rho_{X_i}} + \frac{Size}{Max\ Size_{Range}}.\frac{Usage}{Max\ Usage_{range}}\right)}{5} | X \in \{Resource_{dependency}, IPC, Time\}\ Eq.14$$

In Eq. 14, $\boldsymbol{\vartheta}$ variable is a scalar variable and has the value between 0 and 1. If the value of the variable $\boldsymbol{\vartheta}$ was zero, the γ mechanism is the most suitable mechanism for this migration operation, and if the value was 1 then the γ mechanism is not suitable for this migration operation. Process migration management can decide the suitability of mechanism γ for the process ξ. By using the Eq.14, a threshold can be represented for process migration of the process ξ during the mechanism γ. This issue can be set by the manager and calculated by the results of evaluation.

5. Evaluation

In this paper, a distributed peer-to-peer computing system [16, 17], which can create computing regions for managing the special resource, was used to evaluate the represented mechanism. The computing regions are made logically and include numerous machines have the capability of responding to I/O, process, file or memory requirements for many global activities.

Due to kernel-level implementation, the process manager mentioned in [16, 17] is involved in the collection and gathering of information about the functionality, nature, and behavior of the process. This information enables the MDPM mechanism to select the suitable mechanism for process migration. The [16, 17] operating system has extended data structure compared to than the UNIX system V [51]. The data structures of [16, 17] operating system can store vector-based information and thus, this process can define in the vector-based model. The Oasis concept in [16, 17] enables the complete and pure history of each process.

For evaluating the MDPM mechanism, the computing regions in this peer-to-peer computing system [16, 17] are involved in the processing of three global activities concurrently. A total of 40 machines are involved in this computing region to process MM5 [52], WRF [53], and Charm [54] applications.

The MM5 and WRF programs as the mainstream software of the Meteorology and Charm software are considered as one of the main applications of the Molecular Dynamic domain. Each of the listed software applications needs to be responded as quickly as possible. Due to the high utilization of these software systems, various traditional cluster computing systems have been configured to run them; the results of the traditional cluster computing applications are easily accessible. The governing model of the behavior of the computing processes of the named software is accessible due to multiple performances. In the case of the MM5 software, due to the existence of information about the various implementations of this software, it can be accurate about the status of its processes at runtime. Conventionally, the applied science and technology applications in the field of meteorology are used as scientific and applied software used to evaluate the function of high performance computing systems. Most of the software in this domain either emulate the MM5 pattern or the WRF pattern. The reason why Charm software is considered is its scientific and practical nature. This software is considered as one of the most important softwares in the field of Molecular Dynamic, which is used by various fields of

science, and is considered as an example of scientific and applied software. The distributed peer-to-peer computing system is running on the named software simultaneously.

The process migration management involves four basic strategies: Total_Copy, Pre_Copy, Lazy_Copy, and Flushing.

The main purpose of the evaluation in this paper is to examine using a single mechanism or using a multiple mechanism like MDPM for process migration in Distributed Exascale or in distributed peer to peer computing systems. The main purpose of any kind of computing system is to run the scientific and applied program in the shortest possible time. The time required to execute the activities related to the element of computing system management in the decree is the time to increase the time of the implementation of the scientific and applied program. Among the components of the computing system management, the time needed to execute activities related to the process management migration element is not due to the fact that, during the execution of activities related to this element, there is no access to one (or more than one) process. More tangibly, it increases the time of the scientific and applied program. In addition, due to the fact that there is no possibility of achieving one (or more than one process) during the implementation of the processes related to the process migration management element, it may be a problem in the implementation of other existing processes. The system will increase the time to run the scientific and applied program. Therefore, a mechanism for the process migration is a suitable process that can be transferred in its shortest time based on the nature of the process and the proposed indicators to describe the process status.

In the MDPM mechanism, based on the formulas describing the status of the process migration engine (Equation 6), as well as the generalized computational process state descriptor formula (Equation 5) about which mechanism is used by the four mechanisms used in traditional computing systems, decisions are made to migrate processes that are most adapted to each other. So the most important variable to be considered in the experiments in this section is the time it takes to transfer the process.

In this evaluation, the computing regions have been studied in two states.

In the first state, the process migration management uses only the Lazy_Copy mechanism for process migration. Although the process migration unit can use all the four basic mechanisms for process migration operation, we have selected the Lazy_Copy mechanism because of its performance. The Lazy_Copy mechanism has also been chosen because of the commons of using this algorithm among clustering systems while processing named programs. It should also be noted that the Lazy_Copy mechanism includes the concepts of Total_Copy and Pre_Copy mechanisms. In the Flushing mechanism, the time of responding in inter-process communication may be included as the process migration time.

The reason for focusing on the Lazy_Copy mechanism is the use of this mechanism in traditional computing systems. This mechanism has the features of other triple mechanisms. On the other hand, the main goal of this mechanism is to reduce the response time. For this reason, in this paper, in the first state the considered mechanism is Lazy_Copy mechanism. However, the peer-

to-peer manager [16, 17] has the ability to perform the first state with each of the four mechanisms used in traditional computing systems.

In the second state, the MDPM mechanism has been used by the process migration management in this computing region.

The experiment is based on the actual function of the process migration. The experiment attempts to examine a situation in the system in which there is an occurrence that results in the activation of the process of migration management element. For this purpose, there is a situation in the system that a computing nodes are running only one or two global activities (corresponding to the application MM5 and WRF) and the third overall activity (corresponding to the Charm application) while the system is started. Starting the third global activity will lead the load of the system be unbalanced. This change in load status of the system causes the load distribution to invoke the process migration. The process migration in the first scenario uses only the Lazy_Copy mechanism to transfer any process regardless of process characteristics. The process transmitted by the process migration can belong to any of the three global activities associated with the applications MM5, WRF, and Charm. In the second scenario, according to Equation 5, it decides on the nature of the process, and given that the system describes the process migration mechanisms based on Equation 6, it is about what mechanism is suitable for the transfer of the process, to make the most suitable choice.

In time $t=t_{\varepsilon}$, the load distribution of the system is unbalanced because of the creation of a new global activity. Due to the reducing response time policy in the system level, the process migration management begins to select a set of processes required for process migration. The activities of creating new global activities in the computing region and unbalancing the system's load balance occurred for 25 times.

The reason for the repetition of the test 25 times is the nature of the system's stability. If the test is repeated for 25 times, then the conditions governing the test are in the state of equilibrium. In a number of repetitions 25 times, almost all patterns of process migration are studied. The reason for this is due to the number of computational member machines and the pattern that governs the computing processes of MM5, Charm, and WRF software in clustering systems. if there is only one execution of computing processes of MM5, Charm, and WRF software on the experimented computing region, all stated about the process migration are studied.

Each time, the load balancer in the first state and load balancer with the process migration management in the second state began to select the migration required processes. For evaluating the performance of Lazy_Copy and MDPM mechanisms, we have studied machine number 4.

In order to evaluate the function of the MDPM mechanism in relation to the Lazy_Copy mechanism, based on the two scenarios, each computing node as the member of computing region can be studied. On the basis of a random mechanism, one of the member systems of the system, where it is possible to turn the machine's situation, is chosen from the performer of two general activities to three global activities. In this experiment, based on the random mechanism and according to the stated condition, machine number 4 was selected.

Figure 6 indicates the number of global processes and the number of selected processes for migration.

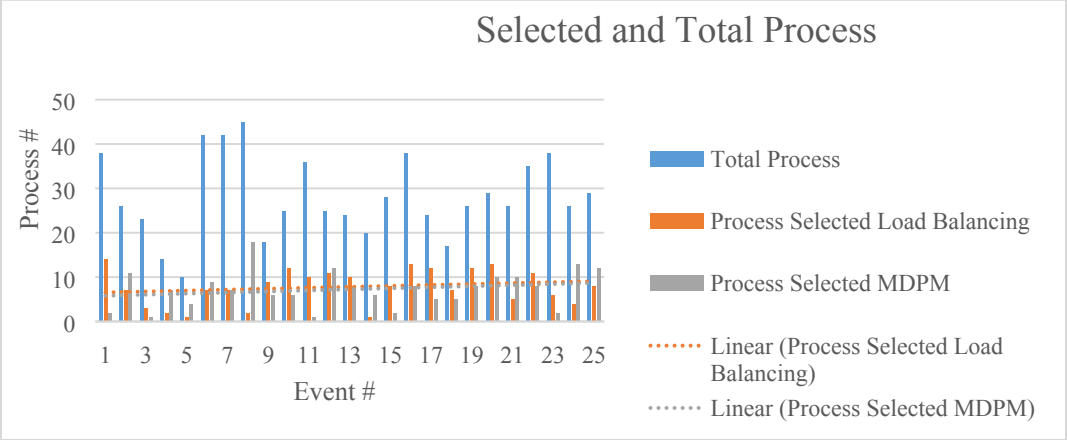


Figure 6: The number of global processes on machine 4 and the number of selected processes.

Figure 6 shows the number of processes selected by each of the two Lazy_Copy and MDPM mechanisms for the process migration, as compared to the total computing processes present in machine number 4. On average, at each retest, the MDPM mechanism, 7.24 processes, and the Lazy_Copy mechanism consider 7.8 processes were chosen for process migration as candidate processes. On average, 28 global activities are running at each test in machine number 4. This means that the Lazy_Copy mechanism typically has 27.8% of the processes and the MDPM mechanism, accounting for 25.7% of the running computing process as the candidate processes for process migration. The reason for this, is the mathematical model, determining the procedure as a candidate for process migration in the MDPM mechanism. In the MDPM mechanism, based on Equation 5, each computational process is described based on five effective parameters in the process transmission. At the designing time of the system, based on Equation 6, each of the four traditional process migration mechanisms has been redesigned according to the effective parameters in the process migration and based on the concept of global activity. The use of two Equation 5 and 6 makes the MDPM mechanism, for each of the 28 computing processes, characterized by a precise description of a) based on five parameters affecting process migration, b) based on the role of the process in the global activity. Then, the MDPM mechanism decides on what kind of process migration mechanism is appropriate based on Equation 12 and 14. The reason of the difference in the number of candidate processes between the MDPM and Lazy_Copy mechanism, refers to the nature of these two. In the Lazy_Copy mechanism, during the transmission, the control states and process execution are transferred and the pages of the address space are not transmitted until page-fault occurs. Therefore, it can be argued that the initial decision criterion of this mechanism is influenced by the size of the control and execution state of the process. This is while the MDPM mechanism uses a five-variable function to decide whether a process is being nominated for transmission or not?

As Figure 6 demonstrates, when the number of studies increases, both MDPM and Lazy_Copy mechanisms select an equal number of processes for migration depending on the load balancer. In both mechanisms, the load balancer should rebalance the load of the system according to the

new global activity's creations. Due to the reducing response time policy, those processes with more waiting time are selected.

As figure 6 indicates, the 25-repeat test of the standard deviation of the MDPM mechanism, 4.15 and the Lazy_Copy mechanism, is 4.02. The reason for this is the nature of the implementation of the MDPM mechanism. The MDPM mechanism based on information gathered, global activities' data structure initiates a model governing the five parameters that affect process migration. The MDPM mechanism, based on this information, defines process migration patterns. This is in the sense that the MDPM mechanism, based on the information given above, indicates that if the status of five-parameter parameters describing the state of the global computational process is in line with what pattern, then decide what kind of process migration mechanism is suitable for the selected process based on decision-making structures filled before. As the number of experiments increases, the standard deviation of the MDPM mechanism decreases. The reason for this is the nature of repetitive activities in scientific and applied applications such as MM5, WRF and Charm. This suggests that: a) If a specific pattern cannot be found in the context of the five-dimensional parameters affecting process migration, b) the repeatability of activities related to the program, then the efficiency of the MDPM mechanism decreases.

In this evaluation, processes numbered 117 and 225 on machine 4 have been studied.

To examine the time independent variable, one can consider the status of each candidate process for process transfer by the process migration manager element. In this paper, a random mechanism was used to select the process, therefor based on the random mechanism, processes 117 and 225 were selected. The process numbering is based on the specific ID assigned to the global activities by the computing system manager.

Processes numbered 117 and 225 are involved in two separate global activities.

Process number 117, is related to the scientific and practical application MM5, and process number 225 is related to the scientific and applied program Charm.

In Figure 7, the situations and resource requirements of resource dependency have been studied for the two named processes.

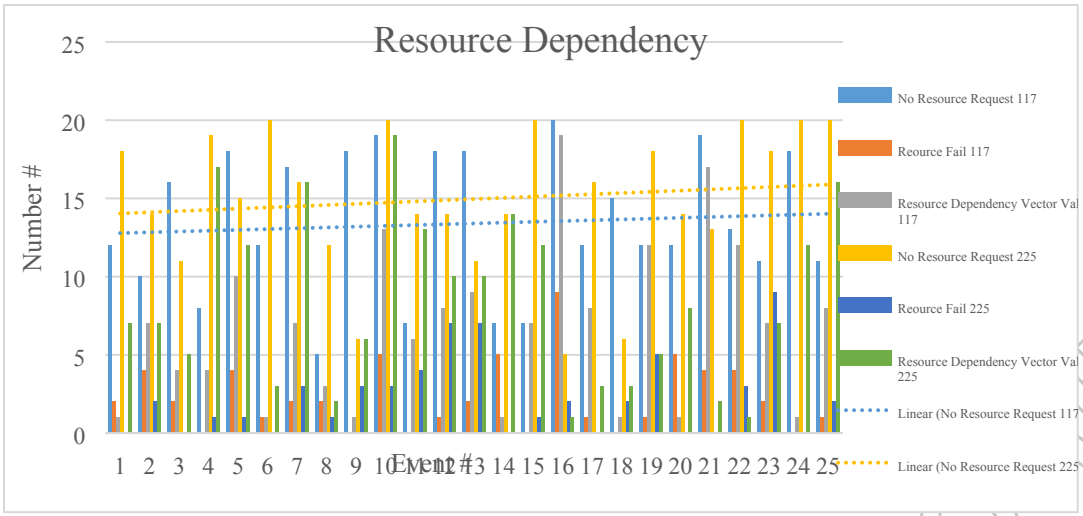


Figure 7: Resource dependency's situations of processes 117 and 225

One of the most important challenges in process migration that causes the process migration trend fails is the change in the dependencies of the process being transmitted to the source machine. The test results shown in Figure 7 are for the MDPM and Lazy_Copy mechanism to check this situation. When the process is chosen for being transferred in a computing node of the system, except in the Total_Copy mechanism, in other mechanisms, the process dependency to the source machine is considered as a bottleneck to continue the process of running the activity.

In mechanisms used in traditional computing systems such as Lazy_Copy, Pre_Copy and Flushing, due to the lack of one-time process transmission and the gradual transfer of process from the source machine to the destination machine, a concept known as the status of source machine became important. In traditional cluster computing systems, the probability of occurrence of a change in the state of the source machine, except for the failure of the source machine, is very low. The reason of this issue is the functionality of the load distribution and process migration. In the traditional cluster computing systems, at the starting time of the process migration trend, the load distribution, due to having a precise view of the status of resources and processes in the system, manages to prevent events that cause the process state to be changed in the local computing node. On the other hand, in traditional cluster computing systems, only one and in a number of situations two variables of process size and time are considered as factors affecting process migration. The existence of a limited number of factors influencing process migration makes the load distribution be able to create controlling structures on the named factors, when the process migration trend is being done.

In distributed Exascale systems, the high probability of the occurrence of a dynamic and interactive nature and the lack of precise view on the status of processes and resources in a system by the load distribution may lead to inform a situation, causing the status of the source machine be changed. Changing the status of source machine may happen in two ways: a) changing the status of the process; b) changing the status of the factors affecting the process; dynamic and interactive nature of processes may lead the state of the process to be changed, this change may let the process migration be failed or need to change the process migration

mechanism or even failing to meet the requirements of the process in the chosen destination machine. In changing the status of factors affecting process migration, it is also possible that each of the five factors affecting process migration is changed in such a way that each of the three situations mentioned is created. In the examination shown in Fig. 7, the occurrence of each of the three situations in a distributed peer-to-peer system is examined.

In peer-to-peer computing systems, conventionally, the destination machine is also part of the computing system, so the possibility of the occurring the dynamic and interactive nature among the process (or processes) always exists. The occurrence of a dynamic and interactive nature in the destination machine change the state of the computing nodes, and consequently the state of the processes and resources in the computing nodes. In the destination machine, the occurrence of a dynamic and interactive nature causes the state of the processes to interact with the source machine or source owner node. In any case, the occurrence of a dynamic and interactive nature may cause the selection of the destination machine to be violated for process migration, or the computing node does not have the capability to execute the transferred process, which itself results in the creation of a process migration array. From the point of view of the process migration, each destination machine is described based on five factors that affect process migration. When the status and operation of the processes in the destination machine changes, from the perspective of the process migration, these changes became important when it changes each of the five factors affecting the process migration. This change in status may lead to the lack of meaningful process migration or the need to change the process migration mechanism. In the test shown in Figure 7, in addition to examining the three modes of the target machine, we examine the occurrence of each of the two situations in peer to peer computing systems and also distributed Exascale system.

As Figure 7 indicates, the patterns of resource requirements for both the processes are completely different. In the same period, the number of resource requirements of process 117 is more than that of process 225. On the other hand, at the beginning of the evaluation, the number of resource requirements of process 117, which the local operating system could not respond, was more than this type of requirements of process 225. While the number of resource requirements of process 117 during the time had decreased however time elapsed and more times overloads. The number of request process 117 that cannot respond to the local operating system reduced compared to that of process 225. Similarly, the sizes of resource dependency vectors of both the processes 117 and 225 are different. In a statistical experiment, the resource dependency vector of process 117 was found to be smaller than that of process 255. In experiment 19, the resource dependency vector was equal for both the processes, and after this experiment, the resource dependency vector of process 117 became larger.

For analyzing this situation, we need to check the resource requirements of processes 117 and 225. The other available global processes in the system, until the experiment 19, required access to the process 117 or shared data with this process placed in the local machine. After experiment 19, the ruling pattern changed in our peer-to-peer distributed computing system and the processes related to the process 225 located on the local machine. The application of this change

affected the MDPM mechanism. This mechanism decreased and increased the size of resource dependency vectors of process 117 and process 225, respectively.

During the test shown in Figure 7, in Process 117, in six times from the 25-time test, the request to access the resource in the target machine was not failed, so if the process migration used the Lazy_Copy mechanism, this mechanism was carried out successfully due to the fact that the destination machine was not changed. In 19 other experiments, the occurrence of a dynamic and interactive nature in the source machine has made the process 117, requiring access to new resources. In this 19-time experiment, the occurrence of a dynamic and interactive nature by the process migration has made it possible for the five factors affecting the process migration to change on the source machine. The test shown in Figure 7, about process 117, illustrates the situation where the change of factors does not cause the process migration to be canceled or the selected destination machine changes. This test also shows situations of the process migration mechanism that needs to be changed. The summation of five factors affecting the process migration in the source machine, after the occurrence of a dynamic and interactive nature, has become such that it is no longer possible to use the Lazy_Copy mechanism. In peer to peer system used for the experiment, the concept of computing region is used. In each computing region, the member nodes of the region have a relative advantage in responding to requests for processes in a given type of resource. In experiment number 17, the occurrence of a dynamic and interactive nature has caused a sequential change in the source of the machine related to processor number 117. Using the Lazy_Copy mechanism makes hierarchical page faults, so the process migration mechanism has changed to flushing.

In each of the 19 trials related to Process 117 shown in Figure 7, the occurrence of a dynamic and interactive nature by Process 117 causes the dependence vector of process 117 to be modified and changed.

In the experiment shown in Figure 7, about the process 225, 25 times the test, the condition of changing the destination machine has not happened in eight times of repetition. Stability the status of the destination machine has made the Lazy_Copy mechanism run smoothly. In 17 tests, the occurrence of a dynamic and interactive nature in the selected destination machine for the 225 process has led to apply changes on five effective factors in the process migration in the destination machine. Changing the factors affecting process migration in the destination machine causes the Lazy_Copy mechanism to be challenged. For example, in Test No. 12, there is a process in a destination machine that has a dynamic and interactive nature. Changing the status of the five factors causes the four requests to be created by process 225, which changes the dependency vector of process 225. This change of the dependency vector of process 225 causes the Lazy_Copy mechanism not be able to be used. One of these requests makes the 225 process need to transfer the entire address space. In this case, the Lazy_Copy mechanism cannot be used as an efficient mechanism. The MDPM mechanism, in this situation, by changing the process migration mechanism from Lazy_Copy to Total_Copy, allowed the entire address space of process 225 to be transferred

As shown in Fig. 7, two processes No. 117 and 225, in the number of requests, as well as the number of fail-access requests in access to resources when the number of experiments increases, the named processes are able to obey a unite mechanism. The only difference between the two processes 117 and 225 is the dependency vector of these two processes.

The time of execution of process 117was high before experiment 19, but for process 225, the execution time was high after experiment 19. This means, when the resource dependency increases, the Lazy_Copy mechanism cannot response properly. This occurs when the MDPM mechanism was used in the Flushing mechanism before experiment 19 and Lazy_Copy mechanism was used after experiment 19 for process 117.

Figure 8 indicates the execution time of process 117.

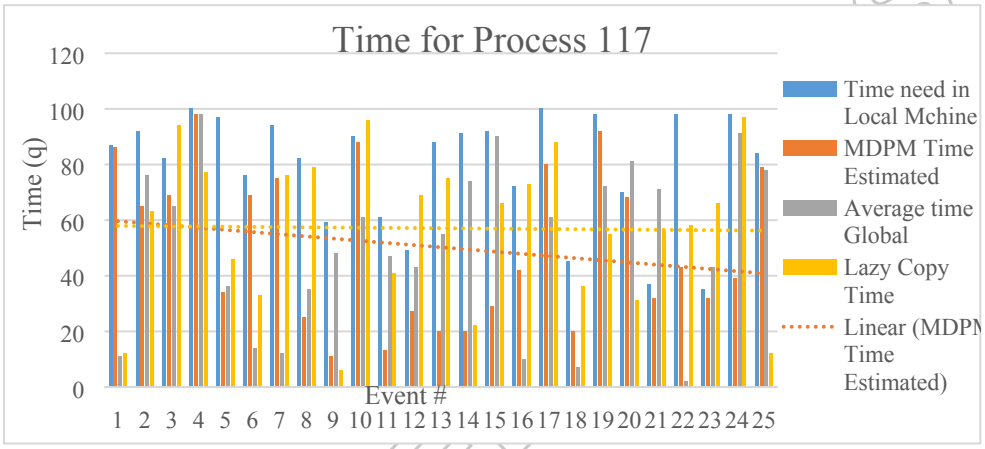


Figure 8: Time of processing process 117 for each 25 times

Figure 8 illustrates the execution time of Lazy_Copy, the time needed for process migration using the MDPM mechanism, the average time of execution of process 117 in computing system before executing on the local machine, and the time needed for completing the process on the local machine.

In a test with the results shown in Figure 8, Process 117 is considered as an example of managed processes by both the MDPM and Lazy_Copy mechanisms. The reason for choosing process 117 is because of having information about its implementation in the previous test, and any other process in the system can also be studied. In this experiment, four indicators have been analyzed and evaluated. Indicators of the time estimation about the program in the local machine, of the 117 process in the total national activity, the estimated migration time by the MDPM and Lazy_Copy mechanism in this experiment have been discussed.

In eight-time repetition from the whole experiment, the time of execution MDPM mechanism was more than that of the Lazy_Copy mechanism. During this repetition, the MDPM mechanism needed an average of 35 more time units compared to the Lazy_Copy mechanism. The most

significant title about these eight experiments is their distribution type among the whole experiment. In the 8-times repetition, five repetitions were before the experiment 11. This is because the MDPM mechanism needs to gather information from the data structures of the kernel operating system for deciding a suitable mechanism to transfer a global process.

The Lazy_Copy mechanism is not a history based mechanism and does not gather information. This mechanism, for each process migration, ignores all previous operations. In comparison, the MDPM mechanism, according to the concept of global activity, stores related information on the data structures of processes and uses this gathered information for each process migration requirements. The Flushing mechanism was the primary mechanism chosen by the MDPM mechanism, but the MDPM mechanism has no information about the process 117. During the experiments 11 to 19 and 22 to 24, the execution time of MDPM was less than that of the Lazy_Copy mechanism. This is because the selected mechanism was equal to the Lazy_Copy or Flushing mechanism after the experiment 19.

Another reason for this issue is the gathered information about the destination machine and previous migration operations. In experiments 20 and 21, both mechanisms use the same strategies; the reason is the system's situation, dependency of process 117 on other processes, and the process pattern state, which is related to the global activities of which process 117 is also a member.

As shown in Figure 8, the MDPM mechanism needs less time for process migration compared to Lazy_Copy mechanism. At the beginning of the experiment, the MDPM mechanism required more time than the Lazy_Copy mechanism because of the MDPM requirements of gathering information about the process 117 and also the analysis of global activity's state. In the MDPM mechanism, the time of process migration reduces over time because the mechanism stated in the equilibrium situation extracted the ruler pattern on global activities that involves process 117.

Extracting the ruler pattern enables the mechanism to decide a suitable mechanism for process 117. The differences in execution times between MDPM and Lazy_Copy mechanisms on the later experiment is because of the concept of Lazy_Copy and Flushing strategies and also the decision of the MDPM mechanism about the destination machine.

As the results indicated, the MDPM mechanism is tangibly based on the available information collected by the operating system. If the operating system failed to store the history of operations and the MDPM failed to learn from the gathered information, then the performance and the functionality of this mechanism would be similar to that of classes I and II as mentioned earlier in the related works.

6. Discussion

Unlike the cluster computing systems, the peer-to-peer computing systems can define and execute more than one global activity. In cluster computing systems, all computing processes are involved in one global activity, indicating similar properties of all the global activities in those processes. In this paper, we have studied the defined global activities in peer-to-peer computing

systems that indicate that all global processes can be defined by the process generator space defined on two sets of processes, five special properties, and three types of operators. In this paper, the special properties of a process are mapped to the five dimensions of the process generator space.

Using a fixed strategy for process migration is the most challenging title in cluster computing systems. This is due to the disadvantages of a mechanism and reduced performance of the process migration management. Due to the concept of being involved in more than one global activity in peer-to-peer computing systems, the process migration management has more flexibility in choosing and using a suitable strategy for process migration. It can also change the current using strategy based on the requirements. This paper represented MDPM mechanism that can extract properties of a process and then select the best mechanism for process migration using the gathered information.

For selecting a suitable strategy, the matching mechanism is used in this paper. By matching the process properties with the process migration and the properties of the mechanism based on descriptor vectors and scalar variables, the MDPM mechanism selects a strategy. However, in peer-to-peer computing systems, decisions made by only considering the properties of the strategy cannot guarantee the performance of a mechanism. In cluster computing systems, the use of each strategy and the system's properties are considered to be important; so, there is a requirement of a flexible algorithm. MDPM is a flexible mechanism that first selects a suitable process for migration by the load balancer and migration manager, then MDPM mechanism, by considering, process descriptors, a mechanism based on the process generator space, and system situation begins to match the process space and the mechanism space and select a suitable strategy.

The MDPM mechanism selects more processes for migration compared to other mechanisms in cluster computing systems. This is because of the difference in the global activity definitions among the represented mechanism and other available mechanisms. In cluster computing systems, using each of four basic strategies, the load balancer selects one or more processes. The process management migrate the process or processes based on the determining source and destination machines by the load balancer. From the load balancing view, each process is described in the form of <Waiting Time, Computing Need Time>, and each computing machine is described in the form <CPU Usage, <Memory Usage, Network Bandwidth>>, although in many systems <Memory Usage, Network Bandwidth> elements are ignored. This type of consideration makes the criterion for selecting a process from the load balancer view, which could be either the usage of the CPU and the process waiting time.

In MDPM mechanism, unlike the four basic strategies, the load balancer only selects the process, and the process migration management selects a process for migration. In this mechanism, process migration management describes each process by Eq.1. In other words, five properties describe each process. Describing a process based on the five types of properties enable the MDPM mechanism to decide about the ability of the process migration.

If the load balancer decided to rebalance the load of a peer-to-peer computing system, the process migration management of a process could determine the destination machine according to five properties. Three of these properties are defined as vectors, and two of them are scalar variables.

The use of two scalar variables allows the process migration to decide a suitable mechanism and decrease the number of destination candidate machines based on the size and usage parameters. On the other hand, the two vector properties, using vector algebra, make an accurate description and select the best strategy for process migration. The ability to determine the descriptors vector of destination machine is similar to the description of the state of the immigrant process. In Eq.12, for each of these properties, vector parameters, in particular, the weight property, are considered. Process migration management, based on the weight of parameters and the importance of running the global activity trend, decides to migrate a process. On the other hand, the description of the global process by migration management allows this unit to decide on the continuing of the process activity based on the capability of destination machine. The global process definition based on the five parameters are compared with the parameters used by load balancer in cluster systems, more process for migration at the load redistribution time. This model makes it possible to determine the destination machine based on the process features.

The parameters used in MDPM mechanism enables the process migration management to use the advantages of all the four basic mechanisms. The MDPM mechanism using the concept of global activity and the complete required process vectors to create the description status vectors of the process. In clustering systems, the using mechanisms for each occurrence that results in transmission, the process migration management begins to migrate the process without mattering the process is a member of global activity or a member of a local process. The used mechanism in clustering systems is not based on history while the MDPM mechanism defines the migration only for the global computing activities and considers all computing process a member of global activity. Therefore, the information about migrating operation is stored in the computing global activity's data structures. This information storing increases the primitive migration time, but in the sequence of migrations, the migration time decreases. The MDPM mechanism uses the stored information in computing global activity's data structures to match the best migration strategy with the concept of the process nature. MDPM mechanism fails if the nature of the global activity is such that the number of times of migration is small in some total management activities of the process.

The MDPM mechanism, unlike basic strategies used for process migration, considers the local machine situation and the environment of the system. Basic strategies used in clustering systems, consider a process as a lonely process. In these mechanisms, the process migration management considers there is only a process in source machine that must be transferred. For performing this operation, the process may be stated in the frozen state or may transfer only the critical parts of a process, and when a page fault happens, transmits other residual parts of the process. The main reason for using this pattern is the process definition in process migration management view. In this definition, a process is studied only based on the time's data structures and the main memory

The MDPM mechanism, the process definition is in a way that includes interprocess communications in local machine and the environment of a local machine. The MDPM mechanism uses five parameters to describe the process communications and the concept of global activity for studying the communications between the process and the local machine environment. In the MDPM mechanism, processes are not an abstract concept, but a member of global activity. IPC, Time and Resource dependency features have direct effects on global activity concept and ideal vector calculations. Due to this issue MDPM, mechanism considers the process, global activity conditions and process effects on the environment.

The MDPM mechanism also uses the Equilibrium situation concept. Using the Equilibrium situation concept will decrease the required time for gathering the parameters for process migration. The result of the evaluation indicates that, while using the MDPM mechanism, the peer-to-peer computing system turned into the Equilibrium situation and the mechanism has enough information about the processes that are the member of a global activity, even by using fixed strategy needs less time for process migration. According to this reason, MDPM mechanism is not efficient in peer-to-peer computing systems that the membered processes of global activities are few or the needed time for processing them is much shorter than the system is running time.

As the results of the evaluation confirm, the migration time will be different even when MDPM mechanism uses a single mechanism for process migration. The reason for this issue is a difference of system's situation and communicated process.

The basic strategies used in the computing clustering systems try to reduce the Residual dependency, unlike the MDPM mechanism that may keep up the resource dependency in some conditions. Global activity's definition in MDPM mechanism from process migration management's view is the reason for keeping up this dependency. In the MDPM mechanism, from the process migration management's view, a transmitting process is still a member of a global activity or may have communications among other processes and global activities.

7. Conclusion

In distributed computing systems, unlike cluster, computing systems can execute more than one global activities. Executing one global activity in clustering systems causes to be able by extracting the properties, limitations and the ability of the system, to choose a determined process migration mechanism. In these systems unlike the distributed computing systems, the ruling concept of global activities are almost the same. This title makes the system be able to by choosing one process migration mechanism, transfers all process migrations. On the other hand, related to the differences between being members of the processes in distributed systems, a single mechanism cannot be used for all process migrations. This article represented the MDPM mechanism that by using a mathematical pattern, tries to describe process migration mechanisms and process properties. The MDPM mechanism uses vector algebra to indicate the adaptation of the process migration mechanism and the process properties.

This makes it possible to use a mechanism that is most in line with the computing features of the processor for process migration. In the MDPM mechanism, unlike traditional process migration, the choice of five features as a description space for global activities and process migration mechanisms makes the process migration, in addition to having a more precise description of each mechanism and any computing process will be able to consider the interactions between the process and the computational system in choosing the process migration mechanism. In the MDPM mechanism presented in this paper, the computing process during process migration is not considered as an abstract concept, but the definition of the process and the migration mechanism based on the five characteristics, makes the process migration and the computational process a part of a global activity, and its interactions and communication with the environment and system should also be taken into account during migration. In addition to choosing an appropriate process for process migration in distributed computing systems such as grid and peer-to-peer systems, this mechanism also allows for consideration of the characteristics of the process migration destination of the computing process based on the process description. In computing systems, a scientifically applied program is required for its implementation, and the lifespan of the program in this type of computing system is longer than the other systems, and it is possible to check the system in a state of provides a balance over a longer period. This capability enables the MDPM process migration to be used in distributed Exascale computing systems.

References

[1]. Hussain, Hameed, et al. "A survey on resource allocation in high performance distributed computing systems." *Parallel Computing* 39.11 (2013): 709-736.

[2]. Tzeng, Chao-Wen, Shi-Yu Huang, and Pei-Ying Chao. "Parameterized all-digital PLL architecture and its compiler to support easy process migration." *IEEE Transactions on Very Large Scale Integration (VLSI) Systems* 22.3 (2014): 621-630.

[3]. Barak, A., and A. Shiloh. "The MOSIX cluster operating system for distributed computing on Linux clusters, multi-clusters and clouds." (2013).

[4]. Jianjun, S. H. E. N., et al. "Hypervisor level distributed load-balancing." U.S. Patent No. 9,037,719. 19 May 2015.

[5]. Desai, Tushar, and Jignesh Prajapati. "A survey of various load balancing techniques and challenges in cloud computing." *International Journal of Scientific & Technology Research* 2.11 (2013): 158-161.

[6]. Rajan, Rajesh George, and V. Jeyakrishnan. "A survey on load balancing in cloud computing environments." *International Journal of Advanced Research in Computer and Communication Engineering* 2.12 (2013): 4726-4728.

[7]. Barak, Amnon, Alexander Margolin, and Amnon Shiloh. "Automatic resource-centric process migration for MPI." *European MPI Users' Group Meeting*. Springer, Berlin, Heidelberg, 2012.

[8]. Medina, Violeta, and Juan Manuel García. "A survey of migration mechanisms of virtual machines." *ACM Computing Surveys (CSUR)* 46.3 (2014): 30.

[9]. Sharifian, Hamid, and Mohsen Sharifi. "Network ram based process migration for hpc clusters." (2013): 47-53.

[10]. Khorandi, Sina Mahmoodi, et al. "Local Robustness: A Process Migration Criterion in HPC Clusters." *Innovative Computing Technology*. Springer, Berlin, Heidelberg, 2011. 374-382.

[11]. Luntovskyy, Andriy, and Josef Spillner. "Evolution of Clustering and Parallel Computing." *Architectural Transformations in Network Services and Distributed Systems*. Springer Vieweg, Wiesbaden, 2017. 45-76.

[12]. Egwutuoha, Ifeanyi P., et al. "A survey of fault tolerance mechanisms and checkpoint/restart implementations for high performance computing systems." *The Journal of Supercomputing* 65.3 (2013): 1302-1326.

[13]. Svärd, Petter, et al. "Principles and performance characteristics of algorithms for live VM migration." *ACM SIGOPS Operating Systems Review* 49.1 (2015): 142-155.

[14]. Adams, Joel C., et al. "Budget Beowulfs: A showcase of inexpensive clusters for teaching PDC." *Proceedings of the 46th ACM Technical Symposium on Computer Science Education*. ACM, 2015.

[15]. Javanmardi, Saeed, et al. "Fr trust: a fuzzy reputation–based model for trust management in semantic p2p grids." *International Journal of Grid and Utility Computing* 6.1 (2014): 57-66.

[16]. Khaneghah, Ehsan Mousavi. "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. 4 Apr. 2017.

[17]. Sharifi, Mohsen, Seyedeh Leili Mirtaheri, and Ehsan Mousavi Khaneghah. "A dynamic framework for integrated management of all types of resources in P2P systems." *The Journal of Supercomputing* 52.2 (2010): 149-170.

[18]. Khaneghah, Ehsan Mousavi, and Mohsen Sharifi. "AMRC: an algebraic model for reconfiguration of high performance cluster computing systems at runtime." *The Journal of Supercomputing* 67.1 (2014): 1-30.

[19]. Navimipour, Nima Jafari, and Farnaz Sharifi Milani. "A comprehensive study of the resource discovery techniques in peer-to-peer networks." *Peer-to-Peer Networking and Applications* 8.3 (2015): 474-492.

[20]. Sharifi, Mohsen, et al. "Process Management Reviewed." (2011).

[21]. Healy, Philip, et al. "Single system image: A survey." *Journal of Parallel and Distributed Computing* 90 (2016): 35-51.

[22]. Setiawan, Iwan, and Eko Murdyantoro. "Commodity cluster using single system image based on Linux/Kerrighed for high-performance computing." *Information Technology, Computer, and Electrical Engineering (ICITACEE), 2016 3rd International Conference on*. IEEE, 2016.

[23]. Rathore, Neeraj, and Inderveer Chana. "Load balancing and job migration techniques in grid: a survey of recent trends." *Wireless personal communications* 79.3 (2014): 2089-2125.

[24]. Milojičić, Dejan. *Load distribution: Implementation for the Mach microkernel*. Springer-Verlag, 2013.
[25]. Zarrabi, Amirreza. "A generic process migration algorithm." *International Journal of Distributed and Parallel Systems* 3.5 (2012): 29.
[26]. Sandhya, S., N. Usha, and N. K. Cauvery. "Load Based Migration Based on Virtualization Using Genetic Algorithm." *Emerging Research in Computing, Information, Communication and Applications*. Springer, New Delhi, 2016. 303-310.
[27]. Thakkar, Nirali, and Anand Pandya. "Process Migration in Heterogeneous Systems."
[28]. Bahena, Vıctor Rodrıguez. "Embedded Distributed Systems: A Case of Study with Clear Linux Project for Intel R Architecture." (2014).
[29]. Zhongyuan, Shan, et al. "Use Pre-record Algorithm to Improve Process Migration Efficiency." *Distributed Computing and Applications for Business Engineering and Science (DCABES), 2015 14th International Symposium on*. IEEE, 2015.
[30]. Liu, Haikun, et al. "Performance and energy modeling for live migration of virtual machines." *Cluster computing* 16.2 (2013): 249-264.
[31]. Ziwisky, Michael W. *A message-passing, thread-migrating operating system for a non-cache-coherent many-core architecture*. Marquette University, 2012.
[32]. Patel, Drashti. "Process migration and load balancing." *INTERNATIONAL JOURNAL OF RESEARCH IN ADVANCE ENGINEERING* 1.1 (2015): 25-30.
[33]. Litton, James, et al. "Light-Weight Contexts: An OS Abstraction for Safety and Performance." *OSDI*. 2016.
[34]. Chen, Song-Yi. "Migrating processes in distributed computing systems." (2013).
[35]. Hsu, Ching-Hsien, et al. "An adaptive pre-copy strategy for virtual machine live migration." *International Conference on Internet of Vehicles*. Springer, Cham, 2014.
[36]. Patel, Minal, Sanjay Chaudhary, and Sanjay Garg. "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 (2018): 55-65.
[37]. Lei, Zhou, et al. "A Novel Hybrid-Copy Algorithm for Live Migration of Virtual Machine." *Future Internet* 9.3 (2017): 37.
[38]. Egwutuoha, Ifeanyi P., et al. "Cost-oriented proactive fault tolerance approach to high performance computing (HPC) in the cloud." *International Journal of Parallel, Emergent and Distributed Systems* 29.4 (2014): 363-378.
[39]. Zarrabi, Amirreza, Khairulmizam Samsudin, and Wan Azizun Wan Adnan. "Linux support for fast transparent general purpose Checkpoint/Restart of multithreaded processes in loadable kernel module." *Journal of grid computing* 11.2 (2013): 187-210.
[40]. Kashyap, Sanidhya, Jaspal Singh Dhillon, and Suresh Purini. "Rlc-a reliable approach to fast and efficient live migration of virtual machines in the clouds." *Cloud Computing (CLOUD), 2014 IEEE 7th International Conference on*. IEEE, 2014.
[41]. Peters, Eric C., et al. "Computer system and process for transferring multiple high bandwidth streams of data between multiple storage units and multiple applications in a scalable and reliable manner." U.S. Patent No. 9,152,647. 6 Oct. 2015.
[42]. Vyas, Ravindra A., et al. "Load balancing using process migration for linux based distributed system." *Issues and Challenges in Intelligent Computing Techniques (ICICT), 2014 International Conference on*. IEEE, 2014.
[43]. Jain, Navendu, et al. "Process migration in data center networks." U.S. Patent No. 9,619,297. 11 Apr. 2017.
[44]. Milani, Alireza Sadeghi, and Nima Jafari Navimipour. "Load balancing mechanisms and techniques in the cloud environments: Systematic literature review and future trends." *Journal of Network and Computer Applications* 71 (2016): 86-98.
[45]. Katyal, Mayanka, and Atul Mishra. "A comparative study of load balancing algorithms in cloud computing environment." *arXiv preprint arXiv:1403.6918* (2014).
[46]. Tanenbaum, Andrew S., and Herbert Bos. Modern operating systems. Prentice Hall Press, 2014.
[47]. Alghamdi, Turki G., Robson Eduardo De Grande, and Azzedine Boukerche. "Enhancing Load Balancing Efficiency Based on Migration Delay for Large-Scale Distributed Simulations." *Proceedings of the 19th International Symposium on Distributed Simulation and Real Time Applications*. IEEE Press, 2015.

[48]. Navimipour, Nima Jafari, et al. "Resource discovery mechanisms in grid systems: A survey." *Journal of Network and Computer Applications* 41 (2014): 389-410.
[49]. Areas, Application. "HiPEAC 2016 Conference Programme."
[50]. Cano, José, et al. "Dynamic process migration in heterogeneous ROS-based environments." *Advanced Robotics (ICAR), 2015 International Conference on*. IEEE, 2015.
[51]. Spinellis, Diomidis. "A repository of Unix history and evolution." *Empirical Software Engineering* 22.3 (2017): 1372-1404.
[52]. Liu, C. H. "Application of grids, clouds and high-performance computing in research of urbanization." *International Symposium on Grids and Clouds, ISGC 2015*. 2015.
[53]. Krishnan, S. P. T., et al. "Performance Characterisation and Evaluation of WRF Model on Cloud and HPC Architectures." *High Performance Computing and Communications, 2014 IEEE 6th Intl Symp on Cyberspace Safety and Security, 2014 IEEE 11th Intl Conf on Embedded Software and Syst (HPCC, CSS, ICESS), 2014 IEEE Intl Conf on*. IEEE, 2014.
[54]. Mendygral, P. J., et al. "WOMBAT: A Scalable and High-performance Astrophysical Magnetohydrodynamics Code." *The Astrophysical Journal Supplement Series* 228.2 (2017): 23.

Ehsan Mousavi Khaneghah is a faculty member of Computer Engineering Department of Shahed University.
His research interest is design and development of distributed computing systems. He is researching on the developing of a distributed Exascale computing system. He had a patent called “"PMamut: runtime flexible resource management framework in a scalable distributed system based on nature of the request, demand and supply and federalism." U.S. Patent No. 9,613,312. 4 Apr. 2017.”
Reyhaneh Noorabad Ghahroodi is BSc student of computer engineering in Shahed University. She began her research with Beowulf clusters. The main field of her research is process migration and scrutiny its challenges in a variety of traditional and modern computing systems. Amirhosein Reyhani ShowkatAbad, a BS student at the Shahed University. He is interested in natural language processing and High Performance Computing systems and has done some researchers in the mentioned fields.

The process migration plays an undeniable roll in all computing systems, so the performance of this element has direct effects on the performance of the system. In Distributed Exascale computing systems, due to the probability of happing dynamic and interactive nature, using a single mechanism for process migration trend will reduce the performance of the system. For enhancing the performance of the system, a flexible mechanism with the ability of adapting to all kind of processes is required. MDPM mechanism by considering five vector parameters as indicator of each process, tries to choose the most suitable mechanism among four traditional mechanisms.