Azerbaijan Journal of High Performance Computing, Vol 5, Issue 1, 2022, pp. 3-32
https://doi.org/10.32010/26166127.2022.5.1.3.32

A Mechanism for Using the Flushing Process Migration Mechanism in Distributed Exascale Systems

Faezeh Gholamrezaie[1], Azar Feyziyev[2]

[1]*Shahed University, Tehran, Iran, faeze.gholamrezaie@shahed.ac.ir*
[2]*Azerbaijan State Oil and Industry University, Baku, Azerbaijan, Azar.Feyziyev@asoiu.edu.az*

*Correspondence: Faezeh Gholamrezaie, Shahed University, Tehran, Iran, faeze.gholamrezaie@shahed.ac.ir

Abstract
The effect of dynamic and interactive events on the function of the elements that make up the computing system manager causes the time required to run the user program to increase or the operation of these elements to change. These changes either increase the execution time of the scientific program or make the system incapable of executing the program. Computational processes on the migration process and vector algebras try to analyze and enable the Flushing process migration mechanism in support of distributed Exascale systems despite dynamic and interactive events. This paper investigates the Flushing process migration management mechanism in distributed Exascale systems, the effects of dynamic and interactive occurrences in the computational system, and the impact of dynamic and interactive events on the system.

Keyword: Distributed Exascale Computing System, Process Migration, Flushing, Dynamic and Interactive Events, Mathematical Model.

1. Introduction
Dynamic and interactive events can occur at any moment of the implementation process of the scientific program in the computing system (Sharifi, M., Mirtaheri, S. L., & Khaneghah, E. M., 2010; Khaneghah, E. M., 2017; Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N., 2018, February). This occurrence of a dynamic and interactive event, while affecting the computing processes in the system, also affects the functioning of the elements that make up the computing system's manager. The impact of dynamic and interactive events increases the working time of the elements that make up the computing system's manager or changes the functioning of these elements. These changes either increase the implementation time of the scientific program or cause the inability to implement the program (Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismayilova, N., & Bakhishoff, U., 2018; Khaneghah, E. M., ShowkatAbad, A. R., et al., 2018). The dynamic and interactive events cause new situations in the computing system, and the mechanisms of dealing with them were not defined and considered during the design of the computing system. This issue becomes a more complicated situation, especially regarding the function of the immigration manager. Other elements that make up the system manager are activated due to the formation of specific situations. In contrast, the distributed manager activates this manager. The set of activities performed by the process migration manager is based on the information received from the load balancer (Adibi, E., &

Khaneghah, E. M., 2018; Khaneghah, E. M., Khoshrooynemati, T., & Feyziyev, A., 2022; Mousavi Khaneghah, E., Noorabad Ghahroodi, R., & Reyhani ShowkatAbad, A., 2018).

Unlike the load balancer and resource discovery, the process migration manager does not obtain information from the computing system when a dynamic and interactive event occurs (Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N., 2018, February; Khaneghah, E. M., ShowkatAbad, A. R., et al., 2018; Mousavi Khaneghah, E., Noorabad Ghahroodi, R., & Reyhani ShowkatAbad, A., 2018). It operates based on the information the load balancer sends (Noshy, M., Ibrahim, A., & Ali, H. A., 2018). This event may cause the state of the existing computing system to be different from the state of the system on which the process migration manager operates. The more the number of exciting elements increases in the activities related to process migration, in this case, the change in the system status due to the occurrence of a dynamic and interactive event has a higher impact on the process migration process. After calling the process migration manager by the load balancer, the process migration manager collects information about the source computing element, the destination computing element, and the status of the migrant process in each transfer state. It does not and assumes that the information received from the load balancer is fixed about the migration system (Asadi, A. N., Azgomi, M. A., & Entezari-Maleki, R., 2020; Takagawa, Y., & Matsubara, K., 2019).

The occurrence of dynamic and interactive events and their effects on the system, especially on the operation of the process migration manager, may either cause the cause of the migration to be violated or cause the state of the source and destination computing element to change, or cause the time Process migration increase (Kaur, T., & Kumar, A., 2022). This issue is that in distributed Exascale systems, dynamic and interactive events and their effects on process migration factors may cause the process of implementing process migration activities (Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N., 2018, February; Khaneghah, E. M., ShowkatAbad, A. R., et al., 2018). Has failed, or the source computing element allocates more time than is acceptable for the execution of migration-related activities to execute the migration. In distributed Exascale systems, dynamic and interactive events can affect the status of each of the influential factors in the process migration process and change it in a way that is compatible with the information of the process migration manager (Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N., 2018, February; Khaneghah, E. M., ShowkatAbad, A. R., et al., 2018; Mousavi Khaneghah, E., Noorabad Ghahroodi, R., & Reyhani ShowkatAbad, A., 2018; Kaur, T., & Kumar, A., 2022). The one obtained from the load balancer should be different. This issue is particularly relevant in mechanisms based on the third computing element, such as flushing (Thulasidasan, S., 2000; Weinhold, C., Lackorzynski, A., et al., 2016; Richmond, M., & Hitchens, M., 1997).

In this type of process, migration manager mechanisms, in addition to the conventional elements in the migration system, a third element known as the file server is also defined. Although the definition of the file server reduces the dependence of the process migration process on the source computing element, to take advantage of the process migration mechanism, it is necessary to consider the effects of the occurrence of the event. There is also a dynamic and interactive element on the file server (De Paoli, D., & Goscinski, A., 1998; Zarrabi, A., 2012).

The file server in the migration manager of the flushing process causes the source element to be kept away from the process of data transfer activities. After transferring information to the file server, the source element to the action, Others can pay (Douglis, F., & Ousterhout, J., 1991). The existence of the mentioned conditions makes the source computing element, by transferring the program code and data to the service element, be kept away from the process of implementing process migration activities. In distributed Exascale systems, the occurrence of a dynamic and interactive event may change the state of each computing element, making it impossible to continue the execution of the process. In distributed Exascale systems, the computing element is present in more than one global activity, and the global activity that is affected by the dynamic and interactive event must be migrated so that it does not affect other global activities (Sharifi, M., Mirtaheri, S. L., & Khaneghah, E. M., 2010; Khaneghah, E. M., 2017; Mousavi Khaneghah, E., & Sharifi, M., 2014). On the other hand, in distributed Exascale systems, because the central concept is changed from process to global action, a mechanism such as complete copying cannot be used. In the complete copying mechanism, because the global activity member process has a suspension time equal to the migration time, there is a possibility of failure of the global activity execution process due to interaction and communication requests of other processes with the migrant process.

Changing the system state due to a dynamic and interactive event means changing the system state to an unknown state for the process migration manager (Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N., 2018, February; Khaneghah, E. M., ShowkatAbad, A. R., et al., 2018). From the point of view of the process migration manager, the unknown situation is a situation in which the execution mechanism is not defined based on the system descriptive indicators from the point of view of the flushing process migration manager. In another state, it is a situation where the process migration activity is not meaningful, or it is a situation where the use of the feature of the service element for process migration does not make sense. The flushing process migration manager should be able to continue the process migration process by considering the impact of dynamic and interactive events. For this purpose, this article investigates the effects of emotional and interactive events on this function while analyzing the function of the migration manager of the flushing processes. This analysis provides the possibility of developing the concept of the available function of the migration manager of flushing processes to support dynamic and interactive events based on the idea of vector algebra.

2. *Flushing Migration Mechanism Functionality*

Several mechanisms have been defined for managing process migration according to the requirements of scientific and applied programs and with a focus on reducing the time of process migration. In computing systems, the function of the process migration manager is such that the process cannot respond to the requests of other processes during the suspension period (Pickartz, S., Gad, R., et al., 2014, August; Maeda, S., Sato, K., Sakiyama, N., Yano, H., & Hayashi, T., 2007). This issue increases the time of implementation of the scientific and practical program of which the process is a part. In most of the mechanisms used for process migration, the activities and functions of the process migration manager are based on the information received from the distribution manager (Thakkar, N., & Pandya, A., 2013). System load distribution is based on the load balancer mechanism. In traditional computing systems, the process that has caused the load balance situation to collide, based on the information of the load balancer, from the source computing element to the destination computing element, or the element to which the transfer of the process causes rebalancing. The process migration manager, in traditional computing systems, during the implementation of the activities related to the transfer of the migrating process from the source computing feature to the destination, about the process status, the origin, and destination computing element status, as well as the factors affecting the selection of the process It does not collect information for process migration (Nimbalkar, M. V., Pathak, G. R., & Nagargoje, H., 2015, February; Czarnul, P., & Krawczyk, H., 2000, August) [23, 24]. Conventionally, in traditional computing systems, the process migration manager works based on three (origin, destination, process) and, in the case of the events that occurred in the computing system, especially the influential events. It does not collect information on the migration process. This event is due to the functional nature of traditional computing systems. In conventional computing systems, after the distributed load manager is activated and the process migration manager is called, the system status does not change. The information sent to the process migration manager in executing activities related to migration is a good process (Shribman, A., & Hudzia, B., 2012, August).

Among the traditional process migration mechanisms, due to the use of the file serving element, the flushing process migration mechanism provides the possibility that a) the process must be able to transfer to the file serving element. b) The source computing element is kept away from the process migration process. C) After transferring the migrating process to the file serving element, the source computing element does not have a role in the activities related to the migration process. If any event leads to the failure of the process migration activities, the migrating process does not return to the source computing element. This state makes it possible to define two sets of independent activities, peer-to-peer and service-provider-based, in the migration manager of flushing processes (Zarrabi, A., Samsudin, K., & Ziaei, A., 2013, March). Peer-to-peer activities refer to activities in which the migration manager of the flushing process transfers any information directly between the source and the destination. Server-based activity is where the flushing process migration manager uses the server to perform process migration activities. With this approach, the function of the migration manager of the flushing process is in the form shown in formula number 1.

$$F(lushing)\left[: (Source)^{P2P\,(t)}_{Server-Oriented\,(Accept)} \overset{\left[\begin{smallmatrix}(Limitatio_{ntime}, Acceptance,\\ Limitation_{server}, IPC_{limitation})\end{smallmatrix}\right]}{\leftrightarrows} (Destination)^{Data_{transfer}\,(t)}_{Kernel_{operaion}} \right]^{Stable_{State}}_{local-outsource} \quad (1)$$

As can be seen in formula number 1, the function of the migration manager of processes is based on flushing, a revolution in the source computing element, taking into account the constraints and limitations of the process, the serving element, the destination computing element It also transfers interactive and communication mechanisms to the destination computing element. In the valuable function of the flushing process migration manager, the source element is defined based on two functional parts: peer-to-peer and server-server-receiver, a peer-to-peer function as a time-independent variable. In the available function of the flushing process migration manager, the destination computing element is based on the volume of data transferred between the server computing element and the destination computing element, which is a function of the independent variable of time, as well as the transfer of the process, which is in the form of server-server It takes place between two calculation elements, origin, and destination. The separation of the source and destination computing elements is based on the activity these two elements perform directly and indirectly (Castain, R. H., Solt, D., Hursey, J., & Bouteiller, A., 2017, September; Kruglick, E., 2015). The source computing element works with the destination computing element in the field of process transfer as a server and receiver and the process data transfer as a peer-to-peer with the server. In the flushing process migration manager, the source computing element transfers the process from the source to the destination (Binder, J., 2014). Also, process data is transferred from the start to the serving computing element so that the source computing element is not present in the migration process. Figure 1 shows the function of the flushing process migration manager.

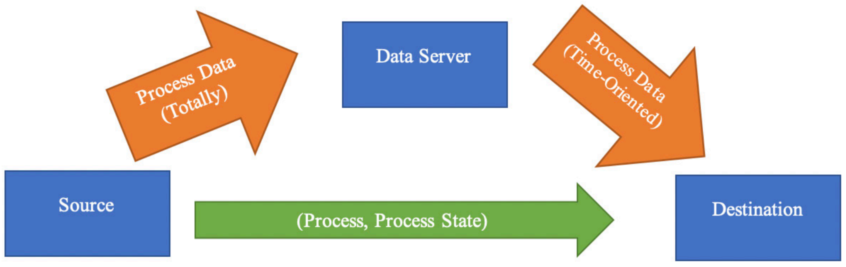

Fig. 1: The function of the migration element of the flushing process

As seen in Figure 1, the information related to the process and the status of the process is done in the form of server-receiver and the state of acceptance or non-acceptance of the process in the destination computing element. The transfer of process data between the source computing element and the serving element and the element is done at once. In the case of the serving computing element and the destination computing element, it is done peer-to-peer and as a function of time (Matsuzawa, K., Hayasaka, M., & Shinagawa, T., 2020; Matsuzawa, K., Hayasaka, M., & Shinagawa, T., 2018, June).

As can be seen in figure 1 and function 1, in the process of implementing the migration manager's activities based on flushing, the mapping of the state of the migrating process in the source computing element to the state of the process in the destination computing element is executed twice. In the initial mapping implementation, the stored data structures related to the process are mapped from the source computing element to the destination computing element based on the constraints and limitations of the destination computing element. In this situation, from the point of view of the process migration manager, the migration process is defined in the source computing element based on the Server-Oriented (Accept) function (Shah, V., & Donga, J., 2020).

The Server-Oriented (Accept) function expresses ExaFlushing manager, the structural data part of the operating system of the migrating process, based on the stipulation of the destination computing element, should be able to set the data structures of the operating system should be mapped in the destination computing element to establish the necessary condition for the migration of flushing processes. The mapping of data structures is done only based on the condition and concept of acceptance. The load balancer calls the process migration manager in traditional computing systems such as the cluster computing system. The status of the source and destination computing elements does not change in the process of implementing the process migration activities (Singh, G., & Singh, P., 2021; Chou, C. C., Chen, Y., Milojicic, D., Reddy, N., & Gratz, P., 2019, November). Therefore, the condition of matching the data structure of the migrant process to the corresponding data structure in the destination computing element be established. In traditional expandable computing systems, if the status of the computing element changes due to the implementation of local activities, if the data structure of the migrating process is not matched with the state of the corresponding data structure in the destination computing element, the migration condition is not established (Stoyanov, R., & Kollingbaum, M. J., 2018, June). In the process migration manager of flushing, the meaning of the correspondence between the data structure of the migrant process with the destination computing element is the ability of the destination computing element to create descriptive data structures of the migrant process in the source computing element, as well as the possibility. The initialization refers to the value in the origin computing element at the beginning of the suspension time (Ma, F., Liu, F., & Liu, Z., 2010, July).

In the process migration element based on flushing, the data structure related to the migrant process may be created in the destination computing element. Still, there is no possibility of setting values that lead to the option of continuing the execution of the process in the destination computing element. In traditional expandable computing systems, it is impossible to create a corresponding data structure for the migrant process in the destination computing element, or it is impossible to set the elements of the data structure to continue the execution of the migrant process. The Server-Oriented (Accept) function cannot run (Su, K., Chen, W., Li, G., & Wang, Z., 2015, December; Shan, Z., Qiao, J., & Lin, S., 2018, October).

The local and migrant process types are different from each other. In the first mapping, because for the migrant process, the creation of the corresponding structure data set defined in the source computing element is done in the destination computing element, so from the point of view of the local operating system of the destination computing element, the migrant process It is considered as a local process. The process migration manager is based on flushing to prevent this issue, and the possibility of defining specific restrictions and limitations on the migration process performs the mapping process based on the local-outsource concept and defines fields in the process descriptor data structure that separates the two (Cui, Y., Chen, H., & Zhu, L., 2020; Jalaei, N., & Safi-Esfahani, F., 2021).

As seen in Figure 1 and Function 1, the second implementation of the mapping refers to the transfer of data from the migrant process to the serving element in the computing system. The limitations and restrictions governing the second mapping include the time limitations for the implementation of the process migration activity, the limitations governing the manager of the process migration, and the limitations governing the interaction and communication mechanisms between the processes. In the second mapping, unlike the first mapping, the concept of the source space does not refer to the computing element of the source (Zhao, J., Hu, L., Xu, G., Chang, D., Ding, Y., & Fu, X., 2013; Hao, J., Ye, K., & Xu, C. Z., 2019, June)In the second mapping, the source computing element refers to the serving element. In the flushing mechanism, the source computing element transfers the data space of the immigrant process to the serving element at once, so from the moment of the data space transfer, the source computing element is not present in the functional equations of the mentioned mechanism. From the point of view of the migration manager of flush-based processors, the migration process in the service element is described based on the transfer function from the source to the destination. The independent variable of this function is the time variable. The equivalent of this function in the destination computing element is the transferred data function, which itself has an independent variable of time. In mapping the condition of the sufficiency of the flushing process migration activity, the balance is between increasing the overhead of the migration operation and reducing the data dependence on the service element (Pecholt, J., Huber, M., &

Wessel, S., 2021, November).

3. Related Work

It is first necessary to explore this mechanism's features, advantages, and disadvantages in traditional systems, to present and examine the flushing mechanism in distributed Exascale systems. Flushing is the first process migration mechanism that uses a third element for process transfer. In (Douglis, F., 1987; Rungta, M., 2006), The flushing mechanism has been used for migration processes in the sprite operating system. Sprite OS is a file-based operating system. In this operating system, in addition to the source machine and the destination machine, a third element called the file serving element is used to perform migration operations. In the sprite operating system, the purpose of using the file serving element is to overcome many remaining dependencies in the algorithms introduced to implement migration operations in traditional systems (Hartman, J. H., & Ousterhout, J. K., 1990, June).

In the operating system, to carry out the migration process of processors, at first, the process is suspended, and the status of execution and control and brief information about the process, such as buffered messages, and file descriptors, are sent to the destination machine (Ganguly, D., Zhang, Z., Yang, J., & Melhem, R., 2019, June). During migration, code, heap, or stack sections are not moved; the memory pages, including stack and heap information and cache memory blocks, are dispatched from the source machine to the file serving element. In this operating system, to perform the process migration, the process selected for migration is suspended, and all the memory pages are sent from the source machine to the file serving element. Then, during the migration, the destination element receives its required memory pages from the file serving element (Priyanka, H., & Cherian, M., 2020). If a memory page error occurs, the immigrant process quickly receives the information it needs from the file serving element. When the message containing the permission to continue the execution of the process is sent to the destination machine, the execution of the process continues in the destination machine.

In the article (Matsuzawa, K., Hayasaka, M., & Shinagawa, T., 2020), Optimizing the file server is considered a suitable method to speed up access to memory pages when a page error occurs in the process of migration processors when using the flushing mechanism. Since the sprite operating system uses file-based interprocess communication, optimizing the file serving element in the system increases the possibility of faster access to memory pages when a page error occurs. As a result, it can be said that the migration mechanism of flushing processors supports the fault tolerance feature to a great extent (Bradford, R., Kotsovinos, E., Feldmann, A., & Schiöberg, H., 2007, June).

In the migration mechanism of flushing processors (Rodrigues, M., Roma, N., & Tomás, P., 2015, October), in addition to the source element and the destination element, a third element called the file serving element is also used to implement the migration process of the flushing processors. The computing load of the source element is reduced, and it can be concluded that this mechanism is executed independently of the size and volume of the migration processor and the modified memory pages, which can be considered one of the advantages of the migration mechanism of the flushing processors(Forbes, E., & Rotenberg, E., 2016, October).

According to the article (Milojičić, D. S., Douglis, F., Paindaveine, Y., Wheeler, R., & Zhou, S., 2000), in HP laboratories, which investigates the process migration mechanism in traditional systems, the flushing strategy is described as a process migration strategy without residual dependencies. It does. In this article, the migration processing suspension time in the flushing mechanism is the same as the time in the complete copy mechanism. Still, due to the use of the file serving element, it is considered lazier than the precopy and sluggish means. The flushing mechanism is designed for systems based on files, and communication between processors is done by sending and receiving files.

The flushing strategy in systems that are message-oriented and inter-process communication in them is established by sending and receiving messages; as a result of this, the process suspension time during migration is short in them, and it does not perform optimally; Because there is a possibility of overhead appearing during the migration process. Also, in the flushing mechanism, the time of suspending the processors is relatively high due to the existence of the file serving element and the necessity of sending status processors to this element. The most significant number of bytes of the status processors refers to the virtual memory that the processor has access to, which is the problem. It can be considered as one of the disadvantages of the flushing mechanism; But on the other hand, the presence of processors at this level, as well as the use of a large amount of virtual memory, helps to establish better transparency in the migration process of processors and the implementation of traditional systems.

According to the (Van Steen, M., & Tanenbaum, A., 2002), the result of the University of California research on the process migration mechanism of flushing in traditional systems due to the use of a third element during the execution of the process migration operation, the migration operation remains hidden from the system users. As a result, This is the only sign of proof that the process migration occurred is a sudden decrease in the load on the source machine and its simultaneous increase in the destination machine. On the other hand, unlike the message-oriented process migration mechanisms, the flushing mechanism uses the call of procedures in the kernel for communication between processors. These two features increase the level of complexity of systems that use the migration mechanism of flushing processors to perform migration operations; Because, in the worst case, each kernel call must be coded in a way that can separate remote processes from local processes and execute them away from the processor migration process.

In an article (Khorandi, S. M., Mirtaheri, S. L., Khaneghah, E. M., Sharifi, M., & Ghiasvand, S., 2011, December), the concept of workload on the source element during the operation of migration processors in traditional systems is considered a determining feature. Since the migration mechanism of the flushing processors is designed to have the most negligible dependence on the source, the source machine is not required to provide services to the migration processor during the execution of the migration process and after that. For this reason, in this mechanism, the load of The work done on the source computing element is minimal. Reducing the computing load on the source computing element is the most important reason for switching to the migration mechanism of flushing processors in traditional computing systems.

According to an article (Jahanjou, H., Miles, E., & Viola, E., 2015, July), one of the advantages of the migration mechanism of flushing processors is to establish a balance between increasing the overhead of the flash operation and reducing the dependence on the source. In this article, two modes are considered for candidate processes for migration; In this way, the process is in a normal state at first, and as soon as the direction of migration to the destination machine is selected, the process mode changes to the migration mode. Finally, after the migration operation, the process returns to its normal state, and the process execution in the destination machine starts from It is resumed.

Normal →Migration→Normal

In another development of the migration mechanism of flushing processors, which is called the generic or hybrid algorithm and is mentioned in (Jahanjou, H., Miles, E., & Viola, E., 2015, July), for the migration process, four cycles before and after migration and again before and after the migration is considered. This algorithm, a combination of complete copying, pre-copying, and flushing algorithms, has been developed in such a way that the execution of the migrant process is initially suspended in the source computing element. Then the new process is created in the destination computing element and resumes its execution in the destination machine. In the final stage, whatever is left of the process in the source computing element is destroyed.

In the pre-migration cycle, as the process constantly changes the pages in the virtual memory, they are transferred from the process's address space to the file serving element to remove the computing load from the source computing element.

In the migration cycle, there are two processes in the system at the same time, which show both migration processes, one of which is creating in the destination computing element and the other is transferring all the information to the file serving element to reduce the computing load of the source element.

In the post-migration cycle, processing states, except memory pages, are transferred from the source to the destination computing element. When all the memory pages are transferred to the destination machine, the cycle after migration returns to the pre-migration cycle so that the process can migrate to the source computing element or any other computing element ready to accept the immigrant process in the system if necessary.

Pre-migration→Migration→Post-migration→Pre-migration.

Because in the migration mechanism of hybrid processors, when the process is migrated, the process execution resumes in the destination machine, unlike other algorithms of migration processors in traditional systems that need to transfer a large amount of process information to perform the migration operation, it depends on the transfer of very little information. This algorithm's delay can be considered as short as the delay of the lazy algorithm.

Migrated processing may be delayed in the post-migration cycle due to requests for pages of memory that have not yet been moved in their natural order. Therefore, it can be said that compared to the pre-copying algorithm, the combined algorithm has a higher delay time. On the other hand, some memory pages may have been transferred to the destination computing element before the migrated process requests them, and the process not be delayed to access them. Therefore, the hybrid algorithm has a shorter delay than the lazy algorithm.

The combined algorithm sends the entire process status to the destination machine in parallel. This problem reduces the process execution time in this algorithm compared to complete copying and flushing algorithms. In addition, the migration mechanism of hybrid processors, similar to the mechanisms of simple flushing and complete copying, does not depend on the source. The temporary dependence on the source computing element is completely lost after the complete transfer of the processing state to the destination computing element. Since even if the process does not intend to migrate, some pages in the memory are transferred to the file serving element in the cycle before migration, the algorithm of hybrid migration processors has high network traffic. The network traffic generated in the hybrid mechanism is almost simple with full copy, Pre-copying, and flushing algorithms.

In other versions of the flushing migration algorithm, which is referred to as the evolutionary migration algorithm in [3], the existence of residual dependencies on the file serving element, overhead, and the high processing time is among the disadvantages of the flushing migration algorithm; But unlike the migration algorithm of the pre-copying processors, the algorithm of the generative migration processors has no preparation time.

Idle time is the sum of the process suspension time in the resource computing element and the time of any disconnection of the process with the environment and other processes during the migration process of processors.

4. Influence of Dynamic and Interactive on Pre-Copy Migration

In distributed Exascale systems, there is a possibility that dynamic and interactive events occur at any moment of the program execution process and activities related to the management of program processes. A dynamic and interactive event may cause the state of the system description to change so that the mechanism (or mechanisms) used to manage the system cannot manage and control the existing situation. It is also possible to have dynamic and interactive events like other system management parts during the activities related to the migration manager. To analyze and understand the occurrence of dynamic and interactive events on the function of the immigration manager, the immigration management system can be considered for each execution of the immigration manager. In the system of migration processors created based on the flushing mechanism, in addition to the computing element, the source, destination, and migration process, there is also the computing element serving the file. In the large-scale distributed system, each of the constituent elements of the immigration system can be a member of one (or more than one) nationwide activity.

The membership of the constituent elements of the migration system in national activities defines each system element and, accordingly, the entire migration system of flushing processors in distributed Exascale systems; based on the concept of national activity., the concept of non-abstraction of the process is brought up, Considering the idea of a global activity to define the element. In traditional computing systems and, consequently, in the migration management of conventional flushing element processors, the concept of processing is considered abstract. Considering the idea of process in an abstract form in the migration process processes makes the process of migration an independent concept without dependence on other processes. The migration mechanism of flushing processors, considering the idea of processing in an abstract form in traditional computing systems, only the processes related to the transfer of the process between the source computing element with the destination and serving computing elements, as well as the process data transfer from the serving computing element to the destination computing element considers and manages. The process migration manager does not collect information about the source computing, destination, serving, and migrating process. Therefore, the changes made to each of the components of the migration system are not defined for the process migration mechanism of flushing in traditional computing systems. The failure of the activities related to the process migration manager is considered only in the case of the mentioned transitions. The load balancer reviews any changes related to the migration system, and in case of changes resulting in no need for the process migration activity or failure of the process migration activity, the load balancer fails the process migration manager. The activities related to process migration are inherently time-consuming, and their implementation increases the scientific and applied program time. Possible change is minimal in traditional systems.

Even though in distributed Exascale systems, every moment of the implementation process of scientific and applied programs such as Human Exposure to Electromagnetic Fields, High-Temperature Superconductivity, and Seismic Imaging, there is a possibility of dynamic and interactive events[]. The concept of the system is designed based on an abstract process model. Based on this, the process migration activity fails model, and the completion of the activity due to failure and re-execution of the related activity Process migration due to the change of flushing has justification for use.

A dynamic and interactive event in the system may affect any of the components of the flushing process migration system and increase the possibility of failure of flushing process migration activities. On the other hand, considering global movement and defining the function of the flush process migration system increases the frequency of element changes. The definition of the serving element, while having advantages such as reducing the process migration overhead on the source computing element, increases the reliability of the process migration process. These three features can be used in distributed Exascale systems. In distributed Exascale systems, the process migration manager, in addition to the task of transferring the process from the source computing element to the destination computing element, has the task of moving the process (or processes) that are directly or indirectly affected by dynamic and interactive events and require implementation in the new accountability structure is in charge.

In distributed Exascale systems, considering the definition of migration based on the concept of global activity makes it necessary to analyze and evaluate the diffusion effects of global activity. Global activity diffusion effects mean being affected by other elements related to the process in which the dynamic and interactive event occurred. When a process, either as a candidate process for process migration or as a process defined in the destination or origin computing element, undergoes a dynamic and interactive event, in this case, other processes may be under its influence. The concept of diffusion effects makes that in distributed Exascale systems, the element of process migration management both needs to redefine the fundamental element (and change the essential element from the process to another concept) and also needs to consider the idea of process dependency, and Not considering the immigrant process as an abstract concept. Changing the central element of the process migration manager from process to global activity and redefining the concept of flushing process migration based on global activity necessitates considering the communication and interactions between the migrant process and other processes. This event makes it impossible to determine the destination computing element only by the distribution element. In this situation, the process migration manager, if the status of the migration system changes, either based on the change of the national activities defining the process migration system or due to the occurrence of a dynamic and interactive event, about The computing element of the destination should make a decision.

In distributed Exascale systems, the occurrence of dynamic and interactive events causes one (or more than one) element of the process migration system or elements that are affected and influential on the process migration system to change in such a way that the pattern governing The process migration system is not defined for the flushing process migration manager. Moreover, the flushing migration mechanism does not have an existing status management mechanism. From the point of view of the flushing process migration manager, the process migration system is defined in the form <, <PS, GA, <CS, CF, CD>, TF, TO<SS, DS, FS>>. The flushing process migration system's definition is based on the three spaces of the source, destination, and serving element computing elements. Because the mechanism of the migration manager is the flushing process, the definition of all three spaces is based on the function of the service element. In each of the three mentioned spaces, the function of the serving element is considered as a dependent variable of space and time as an independent variable. Formula 2 defines each of the three mentioned spaces in the flushing mechanism.

$$\forall SS, DS \in GA\ \exists FS \in GA :: f(Operation) = \begin{Bmatrix} Data_{Transfer}(t_{running})\ if\ System = DS \\ System_{state}(t_{running})\ System = SS \end{Bmatrix} \quad (2)$$

As can be seen in formula number 2, the computing space of the serving element can be considered for both the length of the source and destination computing element. When the system under review is the destination computing element, this element transfers data during the execution time from the serving element to the destination computing element. When the system under investigation is the source computing element, this element considers the system's state as input.

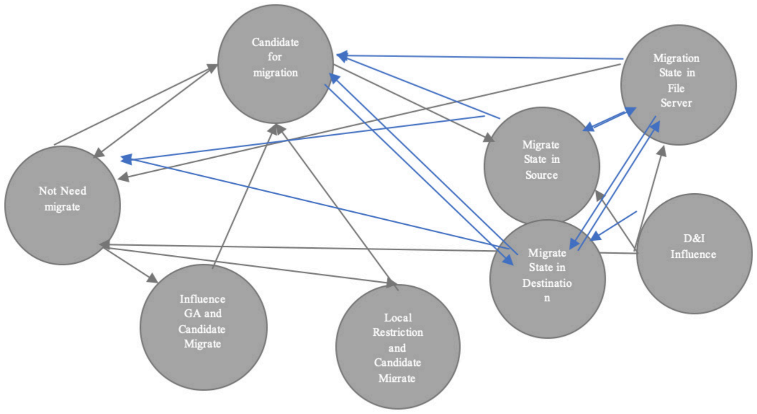
Candidate for migration
Migration State in File Server
Migrate State in Source
Not Need migrate
D&I Influence
Migrate State in Destination
Influence GA and Candidate Migrate
Local Restriction and Candidate Migrate

Fig. 2: Definable states for the PS vector

As seen in formula number 2, SS (Source Space) and DS (Destination Space) spaces, which are members of global activity, are defined in terms of FS space. The FS space includes operations performed between the source computing element and the server to transfer the data of the migrating process and between the serving element and the receiving element to continue the process migration. The element under review is the element in which the program code related to the immigrant process was placed. The FS space is defined based on data and system status concepts. If the examined element is the destination element, then all the activities of the FS space and the SS and DS areas are defined based on the concept of data. Suppose the investigated element is the source computing element. In that case, all the activities of the FS space and, consequently, the SS and DS space are defined based on the concept of the state of the computing system. Both computing system state and data elements are determined based on time independent variable or execution of system activities. In formula number 2, the concept of time implicitly refers to process changes. Time is a variable that shows that the state of the process has changed due to execution in the central processor. In the migration system, changing the value of the time independent variable means changing the system's status, the FS space's pattern, and the system's data and state.

The PS, GA, and CX variables, where X can be the source, the computing element, the destination, or the service element, are of matrix type. One vector can be

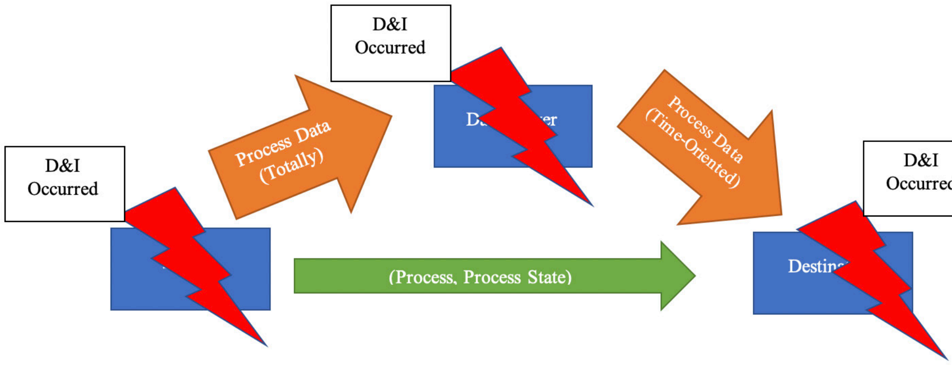

Fig. 3: Dynamic and interactive occurrence situations in the flushing mechanism

considered for each matrix. Therefore, PS, GA, and CX variables are each a vector that is a function of the independent time variable. The PS vector indicates and describes the state of the process from the point of view of the flushing process migration manager. The definition of the PS vector is based on the load balancer's information and the flushing processes' migration manager. To maintain the data of the migration process at every moment of implementing the activities related to the migration manager of the flushing process. The PS vector information is in the form of a binary pair (State, Data), where state indicates the state of the vector at any moment of implementing activities related to the migration manager of flushing processes, and data also represents the related data structure. In the implementation process related to the PS vector, the data structure associated with the maintenance of the migration process data is designed in the form of a pointer data structure that always points to the current location of the migration process data. Considering that the migration process is moving from the source computing element to the server or the server to the destination, the value of Data can refer to more than one space. Suppose this vector is in line with the positively defined vector by the load balancer. In that case, the processing requirements corresponding to the vector can be answered in the computing element.

In distributed Exascale systems, from the process migration element, in addition to transferring the process in a situation where the source computing element cannot execute the process, for cases where the process is a part of the global activity and migrates according to the function of the global movement, is also used. The relative advantage is considered a way to continue implementing national activities in nationwide activities.

The dynamic and interactive events of the large-scale distributed system are such that at any moment, the computing element may not have a relative advantage in responding to global activity requests, and based on the resource discovery mechanism, the process (or processes) may be transferred from the national activity by the process migration manager. In this situation, in addition to the two elements of load distribution management and process migration, the PS data structure information is also completed by the resource discovery manager.

In distributed Exascale systems, the PS vector is considered in the form (State, Data, Machine), where the machine describes the state of the computing element executing the process about the requirements of the process. The development of the concept of the process as a part of the national activity, as well as considering the element of the national acts as a central element, suggests the need to think about the state of the process about the influential elements in the computing element of origin, destination and service provider. Make The PS vector represents the state of the process from the point of view of the process migration manager in an abstract form. The imaginable conditions for the PS vector, from the point of view of the immigration manager, are limited, and your processes; these states are shown in Figure 2.

As can be seen in Figure 2, flushing in distributed Exascale systems, Migrate State in File Server, Migrate State in Source, Migrate State in Destination, and states Related to the migration management and its impact on the dynamic and interactive event Influence GA and Candidate Migrate, Local Restriction and Candidate Migrate, and D&I Influence is considered. As seen in Figure 2, the imaginable modes for the process status in the flushing process migration mechanism in the large-scale distributed system, compared to other agencies, and even the development of different mechanisms in Distributed Exascale systems are more. The reason for this issue is the concept of the service element and the influence of the process status on the service element from the idea of dynamic and interactive events. In figure number 3, the possible situations of dynamic and interactive events occurring in the flushing mechanism and the need to have the ability to manage dynamic and interactive events based on figure 1 are shown.

As can be seen in Figure 3, in addition to the conventional situations of dynamic and interactive events occurring in the source and destination computing element or during the process migration process, in three cases of data transfer from the computing element. Originating from the serving computing element, in the transfer of data from the serving computing element to the receiving computing element, and in the operation of the serving computing element, there is also the possibility of dynamic and interactive events.

After the dynamic and interactive event, the destination machine cannot meet the requirement of the migrant process The occurrence of a dynamic and interactive event in the serving computing element, unlike the occurrence of a dynamic and interactive event in the source and destination computing element, is a secondary event. The occurrence of dynamic and interactive events in both source and destination computing elements can be identified both by the developed process migration mechanism and based on the information collected by the distributed load manager. According to its occurrence, the management and control mechanism of the said incident should be established. Conventionally, the impact of the dynamic and interactive event on the function of the source computing element can be seen in changing the status of this element in such a way that either the process that needs to migrate no longer needs to migrate. Another state causes the requirements. It was considered that a new model should be created for the immigrant process, which the current migration model cannot manage.

In the flushing, due to the change of the data flow path from the direct pattern between the source and destination to transfer through the serving unit, It is necessary to consider the situations in which the occurrence of dynamic and interactive events in the serving computing affects the function of the flushing.

A dependency event is a dynamic and interactive event in the serving computing element, which means the event occurs in any element of the computing system, especially the source or destination computing element, which causes the state of the service element to change. The donor is unacceptable in the process of implementing process migration activities. A change in the status of the migrant process or a difference in the requirements of the migrant process in each of the two computing elements of origin and destination, or a change in the global activity in any computing element of the system member of which the migrant process is a part, and this change causes a difference in the function of the migrant process. It can cause the status of the service element to become unacceptable in the process of implementing migration activities.

The change of the status of the service element based on the evolution of the working class of the migrating process causes a) the process of process migration to be challenged due to the lack of data management capabilities of the process. b) The process migration process is challenged due to the inability to manage the data required and allocate data based on the pattern required by the process. In the first case, the serving element cannot manage the data received from the source computing element. In the second case, the service element cannot provide service to the migration process in the destination computing element based on the data receiving pattern of the migration process. The occurrence of any of the two mentioned situations makes implementing the activities of the migration manager of the flushing process impossible. All three mentioned situations are affected by D&I Influence status. The occurrence of the D&I Influence situation means a dynamic and interactive event in the migration process. In Figure No. 2, the Migrate State in File Server, Migrate State in Source, and Migrate State in Destination states describe the effectiveness of the flushing process migration mechanism from the occurrence of dynamic and interactive events.

In the process migration mechanism of flushing, unlike the mechanisms of other process migration mechanisms, only the impact of the dynamic and interactive event on the migrating process or the beneficial elements affecting the migrating process leads to the non-significance of the process migration. It does not come back. In this mechanism, it is possible that changing the status of the serving computing element also causes the concept of process migration of the migrant process to be meaningless. In the flushing, the occurrence of a dynamic and interactive event and its impact on all three concepts of the process, the beneficiary elements, and the service element may cause no need to migrate. In another state, the status of the migrant process has become a method that does not qualify to be selected as a process that should be transferred.

In distributed Exascale systems, to decide on the occurrence of a dynamic and interactive event and its impact on the beneficiary elements of the migration and process, the migration manager of the flushing process, from the concept of description transformations, uses the activity matrix. The FM board represents the generating board of the flushing process migration system. This board is defined in the form (S, D, F, G, P, I) and represents the elements that play a role in implementing the activities related to the flushing process migration. The FM board is defined based on the calculation element of origin, destination, and service provider, the national movement of which the immigration candidate process is a part, the immigrant process, and the processes affecting the immigration candidate process. From the point of view of the migration manager of the flushing process, a migration system is created for each execution of the flushing migration mechanism, and the migration system's generation space is defined based on the FM board. From the point of view of the flushing migration manager, the transformations describing the process migration process activity are determined based on the three spaces of SourceServer, FileServer, and ServerDestination.

The source server space is an n-dimensional vector space defined based on the FM board. In the source server space, n represents the features of the source computing element that contribute to the operation of the migration candidate process. In traditional computing systems, n is conventionally one-dimensional or two-dimensional and includes the time allocated to the process in the source computing element and the time constraints governing the migrating process. For the vector space of SourceServer, the orthogonal base Alpha={α_1,..., α_n} can be considered, where α_i represents the characteristic of the source computing element that affects the candidate processing element. The ServerDestination space is an m-dimensional space defined based on the FM board. In the ServerDestination space, m represents the characteristics of the destination computing element required by the migration candidate process. In traditional computing systems, such as the source server space, the said space is defined as one-dimensional or two-dimensional based on the request's time and constraints. For the ServerDestination vector space, the orthogonal base Beta={β_1,..., β_m} can be considered, where β_i indicates the characteristic requested by the processing from the serving computing element. Similarly, it can be stated that the FileServer vector space is defined based on the FM board and has k dimensions. For the FileServer vector space, the orthogonal base Landa={γ_1,..., γ_k} can be defined, where γ_i represents the features of the computing element serving to manage the process of process migration activities.

The migration manager of the flushing process transfers the SourceServer vector space to the FileServer space under the PreMig linear transformation and transfers the FileServer vector space to the ServerDestination based on the Migrate linear transformation. If the source server vector space is transferred to the FileServer space, the linear operator PreMig is displayed on each vector α_i based on formula 3.

$$PreMig\ \alpha_j = \sum_{i=1}^{m} A_{ij}\,\beta_i \tag{3}$$

The migration manager of the flushing process performs a linear transformation of each vector α_j according to formula No. 3 to the SourceServer vector space. In formula No. 3, α_j represents the vector of features defined in the source computing element that affects migration. If it is possible to satisfy the attribute α_j in the destination computing element, or if the feature αj causes dependence on the source, then the direction of the vector is negative. The size of the vector α_j is equal to the importance of the feature for the process. The process can be with a negative vector with an angle; in this case, the rise of the process indicates the dependence coefficient created by the feature for the migrant process in the destination computing element. The vector β_i represents the process's required feature in the server's computing element. Because the feature of the mentioned vector is given in the service element, the vector β_i is a vector in the direction of the positive vector of one. The size of this vector indicates the importance of the feature required for processing in the serving element. In traditional computing systems, the length of this vector indicates the amount of data transferred between the source computing era and the serving computing element. Suppose the vector β_i has an angle with a positive vector. In that case, this angle indicates the dependence of the required feature of the process, indicating the fulfillment of a part of the capability needed for the process in the serving computing element.

In traditional computing systems, conventionally, both the vector α_j and the vector β_i do not have an angle with negative and positive vectors. This issue indicates that in the case of transferring the start of the migration candidate process from the source computing element to the serving computing element, the process does not have any features related to the source computing element. Also, the serving computing element necessarily has all the required features. They are responsible for the process. In traditional computing systems, calculating the linear transformation function of formula number 3 is not a function of time. It is performed only once during data transfer between the source and the serving computing elements. In traditional computing systems, formula number 3 is based on data. In conventional computing systems, the migration manager of the flushing process transfers data from the source computing factor to the serving computing element based on the linear transformation of formula No. 3. Hence, vector α_j and vector β_i only contain data characteristics. Is. In traditional computing systems, PreMig linear transformation is equal to creating a communication and interaction mechanism between external file processing and data transfer in the said mechanism.

In formula number 3, the scalars A_{1j} to A_{mj} show the coordinate transformation of PreMig α_j in the orthogonal base of Beta. Each value of A_{ij} indicates that in the transfer of the migrant candidate process by the flushing mechanism, the feature α_j of the process is answered by what features β_i available in the server. In an ideal state and traditional computing systems, $|\alpha_j|=|\beta_i|=A_{ij}$. This issue indicates that in conventional computing systems, when the candidate process transfers the vector space α_j to the vector space β_i under the linear transformation of PreMig by the process migration mechanism of flushing, a correspondence of one is a relationship between the characteristics of the source computing element, which is established on the effective candidate processing element with the features requested by the processing from the service-providing computing element. On the other hand, the equal size of the vector α_j with the size of the vector β_i and the equality of these two values with the value A_{ij} indicates that each feature of the source computing element affects the process by one and only one feature requested by the process from the service computing element. It is a giver. Also, the valuable feature of the computing element on the migration candidate process is fully answered by the corresponding feature requested by the process from the service-providing computing element. In formula number 3, the linear transformation of PreMig with mn scalar A_{ij} moves the vector space α_j to the vector space β_i. In traditional computing systems, the mn scalar A_{ij} is transferred based on a one-to-one correspondence. In conventional computing systems, it is not possible to consider the feature of the source computing element affecting the migration process, which is answered by more than one requested feature in the serving computing element. In formula number 3, matrix A, an m*n matrix where A(i,j)= A_{ij}, is called a linear transformation matrix to two ordered bases Alpha and Beta. Matrix A describes the pattern of data transmission between the source computing element and the serving computing element based on the characteristics of vector spaces α_j and β_i.

In case of transfer of SourceServer vector space to FileServer space, the linear operator Migrate has displayed on each β_i the vector according to formula 4.

$$Migrate\beta_j = \sum_{j=1}^{k}\left[\sum_{i=1}^{m} A_{ij}\,\beta_i\right] B_{ij}\,\gamma_i \tag{4}$$

The migration manager of the flushing process performs a linear transformation of each β_i vector based on formula number 4 into the FileServer vector space. In formula number 4, β_j indicates the required feature of the process in the service computing element. If the feature β_j cannot be satisfied in the computing element of the service provider, or the feature β_j causes the immigrant candidate process not to be able to continue to transfer and becomes dependent on the service element, or the service element can manage the data of the process β_j based on the template. If the data management is not governing the service-providing element, then the direction of the vector is negativeβ_j. The vector size β_j indicates the required feature of the process that the service manager must provide. In the traditional computing system, the direction of the $\beta_j$vector cannot be in the order of the negative one vector. In conventional computing systems, the load balancer selects a specific server manager for a migration system and makes this choice based on examining the process requirements, especially the data requirements of the process, so the service element The provider can meet the needs of the β_j process. In traditional computing systems, there is conventionally a service element in the computing system that has the task of meeting the requirements of β_j processes, especially in data management.

The processing vector β_j can have an angle with a negative vector; in this case, this angle indicates the coefficient of meeting the processing requirements by the service manager. This angle means which part of the requirements of the β_j process, especially in data management, can be met by the service element and which parts cannot. In computing systems, the migration manager of flushing processes can define the acceptable coefficient of the service provider. This coefficient is created based on the angle of the vector β_j with the negative vector. If this angle is more significant than a specific limit, the service element can manage the data and process requirements of β_j.

The vector γ_i represents the characteristics of the service element to continue the activities related to process migration. The vector γi can be a positive vector with an angle; in traditional computing systems, this angle indicates the fulfillment of part of the requirements needed to continue the process migration activity. In conventional computing systems, the vector γ_i is in the direction of the positive vector because every feature and capability defined in the service-providing element must be used to implement the activities related to the flushing-based process migration. Based on the rise of the vector γ_i with a positive vector and the angle of the vector β_j with a negative vector, the flushing process migration manager can tell about the ability to use a specific service element to perform related activities to process migration. Suppose the vector γ_i angle with a positive vector is less than a specific limit, and the tip of the vector β_j with a negative vector is more than a particular limit. In that case, the service element can meet part of the requirements of the process and the Provision of a part of the features required to implement the flushing process migration activity. The flush process migration manager can decide based on these capabilities and characteristics regarding using a specific service element.

The vector γ_i's size equals the capability that can be provided to continue the activity of the process migration manager. In the process migration system based on flushing, the space of the serving element can be defined as (D, T, P, P). According to this definition, the space of the service element includes the template used to manage the data of the migration process, the template used to transfer data from the service element to the destination computing element, and the template used to process activities related to migration management. It includes a flushing process as well as a migration process. For each of the constituent elements of the form defining the space of the serving element, a vector (or vectors) γ_i can be determined, indicating the serving element's ability for each of the constituent elements of the form. The vector γ_i can be defined as the result of the mentioned vectors or as the space (γ_{i_D} , γ_{i_T} , γ_{i_P} , γ_{i_p}). In traditional computing systems, the size of the vectors γ_{i_T} and γ_{i_P}, γ_{i_P} is equal to zero. The size of the vector γ_{i_D}, which is similar to the size of the vector γ_i, indicates the amount of data transferred based on the data transfer pattern between the serving and destination computing elements.

In traditional computing systems, the vector γ_i, like the vectors α_j and β_j, does not have an angle with negative and positive vectors. This issue shows the pattern governing the decision-making to start the process transfer by the load balancer. The load balancer considers the computing element as the serving computing element with data management capability and an excellent pattern for data transfer from the serving element to the destination computing element. In traditional computing systems, the absence of an angle between the vector γ_i and the positive and negative vectors indicates that when starting the migration process, the candidate migrates from the source computing element to the serving computing element, the serving computing element. The provider has capabilities and features that can continue implementing activities related to the migration of flushing processes. In traditional computing systems, the service element is necessarily selected to implement the activities related to process migration. This issue is that in conventional computing systems, the calculation of the migrate linear transformation function is not a function of the independent variable of time, and the implementation of the migrate linear transformation function is only for one time and at the beginning of the activities related to process migration. Flushing takes place. In traditional computing systems, formula 4 is based on data. In conventional computing systems, the migration manager of the flushing process selects the serving element to manage data based on the pattern required by the source computing element and manages data based on the way needed for the destination computing element. Transfer from the serving element to the destination computing element. This issue causes that in formula 4, the data transfer process from the source computing element to the destination computing element through the service element is considered. The implementation of each of the sigmas is shown in formula number 4. Data transfer performance is from the source computing element to the server and the server to the destination. In traditional computing systems, in formula 4, the vector γ_i , like the vectors α_j and β_j, only includes the data feature. In traditional computing systems, the linear transformation of Migrate is equal to creating a communication and interaction mechanism between the external processing of the file between the three elements of source, server, and destination and data transfer.

In formula 4, the scalars A_{1j} to A_{mj} follow the definition given in formula 3. In formula 4, the scalars B_{1j} to B_{kj} show the coordinate transformation of $Migrate\beta_i$ in Landa's orthogonal base. Each value of B_{ij} indicates that in the case of transfer of the migrant candidate process by the flushing mechanism, the feature α_i of the process, which was answered by the features β_j available in the server, causes the feature β_j by which feature γ_i Available in the service provider should be answered. In the ideal case and traditional computing systems, $|\beta_j|=|\gamma_i|=B_{ij}$. This issue indicates that in conventional computing systems, when the candidate process transfers the space obtained by moving the vector space αi to the vector space under the linear transformation of PreMig to the area, The vector γ_i migrates under the linear transformation, and there is a one-to-one correspondence between the required features of the destination computing element and the features and capabilities of the serving computing element to continue the transfer process. The equality of the vector size β_j with γ_i and the equality of these two values with the value of B_{ij} It is accountability. Indicate that every feature required by the destination computing element affecting the process can be met by one and only one feature requested by the process from the serving computing element. The valuable feature of the destination computing element affecting the migration candidate process is fully answered by the corresponding feature of the process request in the serving computing element.

Matrix B describes the pattern of data transmission between the destination computing element and the serving computing element based on the characteristics of the vector space resulting from the linear transformation of the vectors α_j and β_j and the vector γ_i. In formula 4, the linear migration with kn scalar B_{ij} moves the vector space β_j to the vector space γ_i. In traditional computing systems, kn scalars B_{ij} are transferred based on one-to-one correspondence. In conventional computing systems, it is impossible to consider the feature of the destination computing element affecting the migration process, which is answered by more than one feature of the serving element. In formula number 4, matrix B, a k*n matrix where B(i,j) = B_{ij} is called the matrix of a linear transformation of two ordered Landa bases and the ordered basis resulting from the linear transformation of Alpha and Beta.

5. ExaFlushing

The possibility of a dynamic and interactive event occurring at any moment of implementing the activities related to the flushing migration manager is created so that the function of this element, shown in formula 4, is influenced by the event. ¬ The mentioned data should be placed. The function's effectiveness of the flushing migration manager causes linear transformations, and the vector spaces described in formula 4 are changed in executing the activities of the process migration manager. This change, as shown in Figure 2, may cause the activities related to the migration of the flushing process to be canceled. Meaningless, or one of the two parts of the implementation of the activities associated with the migration of the process E-flushing is violated, or the source computing element is changed in a way that does not require the implementation of flushing process migration, or the service element is changed in a way that can respond and continue the process migration process. Alternatively, the computing age of the destination is unsuitable for transferring the process. The occurrence of any of the mentioned situations causes the process of implementing the activities related to the migration of the flushing process to be challenged if the skill manager of the flushing process does not work at the function level. Moreover, whether the architectural story cannot manage the mentioned situations leading to the failure of the migration process.

The migration manager of the large-scale flushing process uses the same architecture as shown in Figure 4 to make decisions about the occurrence of dynamic

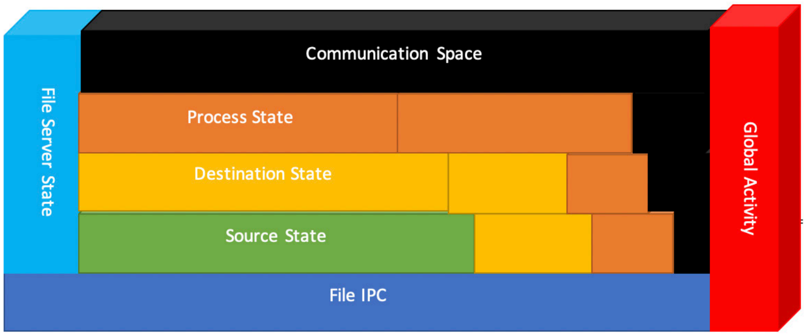

Fig. 4: The architecture of the flushing manager in the large-scale distributed system

and interactive events and their impact on the beneficiary elements of the flushing migration system.

Figure 4 shows two global activity elements, and the service elements are considered the core elements of the flushing manager architecture. The concept of global activity is the development of the process concept. The global activity makes the flushing process migration manager, in addition to considering the process in the local computing element, think of the process in the global activity as well as the interactions and communications between the process and other processes members of the global activity. Taking a picture of global activity makes it possible to consider the idea of process interactions and communications with effective and influential beneficiary processes in the communication space. The communication space allows the flushing process migration manager to monitor the effects and interactions between the process migration candidate process with other global activity member processes and the occurrence of dynamic events and Manage interaction.

In distributed Exascale systems, dynamic and interactive events are possible during the activities related to the transfer of processing data from the source computing to the destination computing. In such a way, it either causes the process required to be solved by the source computing and the activity related to processing migration to be canceled, or there is a need to choose another service element. In Figure 4, global activity refers to a set of processes interacting and communicating with the candidate process migration process. The condition of using the flushing process migration mechanism and its development for managing dynamic and interactive ExaFlushing events is the inability of the source computing manager to continue the process migration activities and grant the proxy of data transfer to the service element. From the point of view of the ExaFlushing process migration manager, if a process (or more than one process) cannot continue its activities in the source computing element for any reason, in this case, among the processes that cannot continue operating in the element According to the source calculation, a process is selected as a candidate process for process migration.

From the point of view of the ExaFlushing manager, selecting the service means changing the implementation process of the national activity. Unlike traditional computing systems where a service element is defined in the computing system, the computing systems Distributed macro-scale may have more than one service element or an element that is considered a service element, due to the occurrence of dynamic and interactive events, lacks data management capabilities. The migration candidate process is a process. In distributed Exascale systems, because the concept of processing is not considered an abstract concept and is always a part of global activity, the service element must have the ability to manage data requirements. The processes interacting with the candidate process should also have a process migration. Because the candidate process migration process is part of global activity, it interacts and communicates with other processes in the Communicate space.

In addition to managing the pattern of data transfers, the service element must be present in the Communicate space of the migration candidate process with other member processes of the national activity and manage data. For this purpose, the ExaFlushing manager should be able to extract, based on the function, the global activity that the migration candidate process is a part of the system. From the point of view of the ExaFlushing process migration manager, the process is an n-dimensional vector space, each dimension of which represents a global activity member process, and the machine is an m-dimensional vector space, each size of which represents a computing element in The section (or sections) of the national activity is currently being implemented. The Process and Machine vector spaces are defined based on the Global Activity board, which represents the global activities running in the large-scale distributed computing system. For Process vector space, the ordered base Process={α_1,..., α_n} can be defined as that α_i represents the requested features of the i-th process member of the global activity. For the Machine vector space, the ordered base Machine={β_1,..., β_m} can be defined, where β_j represents the characteristics of the computing element j-th in responding to global activities. From the point of view of the ExaFlushing process migration manager, it is possible to consider the linear transformation of G from the Process vector space to the Machine vector space based on formula 5.

$$\mathrm{G}\,\alpha_J = \sum_{i=1}^{m} \left[A_{ij_{io}}, A_{ij_{file}}, A_{ij_{memory}}, A_{ij_{process}}\right] \beta_i \tag{5}$$

As can be seen in formula number 5, the global activity function describes each of the n vectors G α_i uniquely as a linear combination of the features of the computing elements implementing the global activity expressed in formula number 5. Formula 5 shows how to linearly combine the computing elements executing the global activity section (or sections). Formula 5 states that each α_J process can be answered by a linear combination of β_i or features of computing elements. Formula No. 5 leads to the creation of an affine plane for each α_J process, which indicates the process of implementing the global activity based on the computing elements that make up the large-scale distributed system. In formula 5, the scalars A_(1j_x) to A_(mj_x), where X can be any value, IO, Memory, File, and Process, represent the services and features that each member process of the global activity of each The computing element of the member of the large-scale distributed system receives. The scalar A_ij represents the features and service that the α_J process receives from the i-th computing element in the context of resource type X. In traditional computing systems, the scalar A_ij is calculated only for the type of processing resource, and it indicates the processing time that the processing α_J takes from the i-th computing element.

The definition of the global activity function causes the ExaFlushing process migration manager to decide on the selection of the serving element and change the flow of the global activity before starting the transfer of the migration candidate process from the source computing element. The ExaFlushing process decides whether after the occurrence of a dynamic and interactive event in the source computing element after the start of the migration activity or the occurrence of a dynamic and interactive event in the serving element and Revision in the determination of the service element uses the concept of the approximation vector to determine the new service element or the acceptability of the status of the current service element to continue the migration process. The ExaFlushing manager considers the Server vector space as a vector subspace of the internal multiplication space of the Machine vector space and θ as a vector of the Machine vector space. The vector θ represents the source computing element's requirements in the data management model field. The ExaFlushing process migration manager describes the needs of the source computing element according to the conditions of dynamic and interactive events, as well as the temporal and spatial constraints governing the request of the source computing element regarding the data management model based on the θ vector. Server vector space is a space with finite dimensions. The θ vector implicitly includes the reason for the absence of the source computing element in the process migration process and the temporal and spatial limitations governing the source computing element in the context of the data management model. It is possible to consider the orthogonal base Alpha={α_1,..., α_n} for the Server vector space, where each α_i member represents the capabilities of a service element in the data management model. Member α_i indicates the pattern the serving element uses to transfer data from the source to the serving computing element. Different patterns can be considered for each α_i member. Patterns such as gradual transfer, one-time transfer, transfer of changed pages, or the last accessed data transfer pattern can be regarded as the most important patterns that can be considered for each α_i member. Each of these patterns creates an orthogonal base for the Server vector space. From the point of view of the ExaFlushing process migration manager, the Selected Server vector can be used to find the best-serving element from the Server vector space based on formula number 6.

$$SelectedServer = \sum_{k} (\theta|\alpha_k)\alpha_k \tag{6}$$

As seen in Formula 6, the Selected Server vector in the Server vector space is the best approximation for θ by the member vectors of the Server vector space, and θ-α is perpendicular to every vector of the Server vector space. Considering that Selected Server is the best approximation for θ by Server vectors, the Selected Server vector and its corresponding computing element are unique and in the system's current state and after a dynamic and interactive event occurs and the state of the source computing element or element changes. The previous server, the computing element described by the Selected Server vector, is the only option selected as a server.

A dynamic and interactive situation can occur in the service element during data transfer between the service element and the destination computing element. In this case, the status of the descriptive parameters of the serving computing component and the destination adding element changes in such a way that either the serving element cannot meet the pattern required by the destination computing element, or the destination computing element cannot accept The process migrates. In such a situation, the ExaFlushing process migration manager needs to redefine the Communicate space shown in Figure 4 so that it can be based on the interactive and communication status between the computing elements with each other regarding the change of the serving computing element or the destination computing element. Communicating space in the most general state can be equivalent to the description space of the global activity system. The Communicate space is described in such a situation based on the four <MAtrib, SAtrib, AAtrib, t>. In the mentioned four, MAtrib represents the governing pattern of data transfer in the computing element of the destination or the serving element. The value of SAtrib indicates the system's characteristic, which is equal to the interactions and communications between the processes that make up the Communicate space. The AAtrib value indicates the pattern required for data transfer between the serving and destination computing elements. The value of t also indicates the time and occurrence of dynamic and interactive events in the system. The ExaFlushing process migration manager decides which servers can fulfill the Communicate space again in case of a dynamic and interactive event. The destination uses what computing element the goal can use to restore the Communicate space. For this purpose, the ExaFlushing process migration manager uses formulas 7 and 8.

$$\forall SelectedServer,\begin{bmatrix}\left(\frac{\delta SelectedServer(E,t)}{\delta SAtrib(E,t)}\right),\left(\frac{\delta SelectedServer(E,t)}{\delta MAtrib(E,t)}\right),\\ \left(\frac{\delta SelectedServer(E,t)}{\delta AAtrib(E,t)}\right),\left(\frac{\delta SelectedServer(E,t)}{\delta t}\right)\end{bmatrix} \tag{7}$$

$$\forall Data_{Source},\begin{bmatrix}\left(\frac{\delta Data_{des}(E,t)}{\delta SAtrib(E,t)}\right),\left(\frac{\delta Data_{des}(E,t)}{\delta MAtrib(E,t)}\right),\\ \left(\frac{\delta Data_{des}(E,t)}{\delta AAtrib(E,t)}\right),\left(\frac{\delta Data_{des}(E,t)}{\delta t}\right)\end{bmatrix} \tag{8}$$

As can be seen in formulas 7 and 8, the ExaFlushing process migration manager, every time a dynamic and interactive event occurs, as well as in the process of starting the process migration from the source computing element to the serving element and the server calculates formulas 7 and 8 to determine the status of the Communicate system to the computing element of the destination. Formulas 7 and 8 show the quality of the Communicate system for each serving element or destination computing element in each of the states being calculated.

The ExaFlushing process migration manager uses function 7 to check the status of the Communicate system after a dynamic and interactive event occurs in the serving element. If the derivative of each of the four describing the state of the Communicate system after the occurrence of the dynamic and interactive event has not changed compared to the state after the occurrence of the dynamic and interactive event concerning the dependent variable of the service element's data transmission pattern. The result of the dynamic and interactive event on the state of the Communicate system is that the design governing the data transfer between the serving element and the destination computing element has not changed. In formula 7, the independent variable related to the SelectedServer variable is equal to the time and occurrence of the dynamic and interactive event.

In formula number 7, the partial derivative $\frac{\delta SelectedServer(E.t)}{\delta MAtrib(E.t)}$ indicates the partial changes of the pattern governing data transfer in the selected server compared to the variable of the pattern governing data transfer in The Communicate system is based on two independent variables: time and the occurrence of a dynamic and interactive event. If the value of the said derivative is equal to the derivative of the previous situation, in this case, the ExaFlushing process migration manager can, regardless of the other results of the derivatives, decide that the selected serving element can transfer data based on The model of data transfer considered by Nasr has the destination of computing, to make a decision. If the value of two partial derivatives is not equal, the ExaFlushing process migration manager decides based on other derivatives. The value of the partial derivative $\left(\frac{\delta SelectedServer(E.t)}{\delta SAtrib(E.t)}\right)$should be equal to 1 in the standard case. The selected service-provider element uses the data transfer model that governs the Communicate system. Suppose the value of the partial derivative $\left(\frac{\delta SelectedServer(E.t)}{\delta SAtrib(E.t)}\right)$ is any value other than one. In that case, the dynamic and interactive event has caused the situation. The pattern governing the data management of the service element should be changed so that it can no longer be a member of the Communicate system.

The value of the partial derivative $\left(\frac{\delta SelectedServer(E.t)}{\delta t}\right)$ indicates the frequency of changes in the governing pattern of data management in the service element due to dynamic and interactive events and time relative to time. Ideally, the serving element should be able to use the same pattern for data management to transfer between the serving computing element and the destination. In distributed Exascale systems, dynamic and interactive events cause the frequency of the data transfer pattern to change between the serving and destination computing elements. This variable indicates the change in the serving element's data transmission pattern. The partial derivative value $\left(\frac{\delta SelectedServer(E.t)}{\delta AAtrib(E.t)}\right)$ indicates the difference between the data transfer pattern of the serving element and the destination computing element. Suppose the value of the partial derivative$\left(\frac{\delta SelectedServer(E.t)}{\delta AAtrib(E.t)}\right)$ is equal to one. In that case, it means that the occurrence of the dynamic and interactive event has not changed the transmission pattern matching. Otherwise, the ExaFlushing process migration manager defines a variable suitable for the acceptable level of changes in the data transfer pattern of the serving element to the destination computing element. If the acceptable limit is established, it is possible to continue implementing process migration activities.

The things stated by the ExaFlushing process migration manager are also true in the case of Formula 8 and based on the results obtained from the partial derivatives of the independent variables describing the state of the Communicate system compared to the dependent variable $Data_{des}$. The computing element decides the possibility of continuing the activities related to Process migration from the serving computing element to the destination. Each of the four derivatives mentioned in Formula 8 about whether the dynamic and interactive event has made it possible to continue process migration activities based on the ExaFlushing mechanism or not. Made a decision

As seen in Figure 4, the operation of the ExaFlushing process migration manager is based on the four spaces of Process State, Destination State, Source State, and File Server State. Each of the four mentioned spaces can be considered as (Data Pattern, Data Execution, Transfer Pattern, Time). Except for the code transfer part of the executive program, which, like the traditional Flushing mechanism, takes place directly between the source and destination computing element and does not change the state of the said computing element, in other cases, the pattern that governs the data is the data-oriented index.

In the form defined for the four statuses of the stakeholders in the ExaFlushing process migration process, considering that the data state and the influential factors describing it may change at any moment of the process migration process due to the occurrence of The dynamic and interactive data or the influence of the dynamic and interactive event to change is considered as an independent variable of time. The concept of time also refers to implementing the activities carried out by each of the four beneficiaries of the ExaFlushing migration manager. In the definition form (Data Pattern, Data Execution, Transfer Pattern, Time), the data pattern refers to the data structure of the process and what structure the data of the process has. The processing data structure indicates the required capabilities of the serving or destination computing element for data storage. Data processing is a set of activities stakeholders must perform in the data migration process. The transmission pattern is how to transfer data between two computing elements from the source to the computing element of the server and from the server to the computing element of the destination. The patterns used for this purpose must be under the process's requirements and the computing element's capabilities.

References

Adibi, E., & Khaneghah, E. M. (2018). Challenges of resource discovery to support distributed exascale computing environment. *Azerbaijan Journal of High Performance Computing, 1*(2), 168-178.

Asadi, A. N., Azgomi, M. A., & Entezari-Maleki, R. (2020). Analytical evaluation of resource allocation algorithms and process migration methods in virtualized systems. *Sustainable Computing: Informatics and Systems, 25,* 100370.

Binder, J. (2014). *Migration of processes from shared to dedicated systems* (Doctoral dissertation, Master's thesis, Vienna University of Technology, Wien, Austria).

Bradford, R., Kotsovinos, E., Feldmann, A., & Schiöberg, H. (2007, June). Live wide-area migration of virtual machines including local persistent state. In *Proceedings of the 3rd international conference on Virtual execution environments* (pp. 169-179).

Castain, R. H., Solt, D., Hursey, J., & Bouteiller, A. (2017, September). Pmix: process management for exascale environments. In *Proceedings of the 24th European MPI Users' Group Meeting* (pp. 1-10).

Chou, C. C., Chen, Y., Milojicic, D., Reddy, N., & Gratz, P. (2019, November). Optimizing post-copy live migration with system-level checkpoint using fabric-attached memory. In *2019 IEEE/ACM Workshop on Memory Centric High Performance Computing (MCHPC)* (pp. 16-24). IEEE.

Cui, Y., Chen, H., & Zhu, L. (2020). Improved Post-Copy Live Migration with Memory Page Prefetching. *International Journal of Performability Engineering, 16*(5).

Czarnul, P., & Krawczyk, H. (2000, August). Parallel program execution with process migration. In *Proceedings International Conference on Parallel Computing in Electrical Engineering. PARELEC 2000* (pp. 50-54). IEEE.

De Paoli, D., & Goscinski, A. (1998). The rhodos migration facility. *Journal of Systems and Software, 40*(1), 51-65.

Douglis, F. (1987). *Process migration in the Sprite operating system.* CALIFORNIA UNIV BERKELEY COMPUTER SCIENCE DIV.

Douglis, F., & Ousterhout, J. (1991). Transparent process migration: Design alternatives and the Sprite implementation. *Software: Practice and Experience, 21*(8), 757-

785.

Forbes, E., & Rotenberg, E. (2016, October). Fast register consolidation and migration for heterogeneous multi-core processors. In *2016 IEEE 34th International Conference on Computer Design (ICCD)* (pp. 1-8). IEEE.

Ganguly, D., Zhang, Z., Yang, J., & Melhem, R. (2019, June). Interplay between hardware prefetcher and page eviction policy in cpu-gpu unified virtual memory. In *Proceedings of the 46th International Symposium on Computer Architecture* (pp. 224-235).

Hao, J., Ye, K., & Xu, C. Z. (2019, June). Live migration of virtual machines in OpenStack: A perspective from reliability evaluation. In *International Conference on Cloud Computing* (pp. 99-113). Springer, Cham.

Hartman, J. H., & Ousterhout, J. K. (1990, June). Performance Measurements of a Multiprocessor Sprite Kernel. In *USENIX Summer* (pp. 279-288).

Jahanjou, H., Miles, E., & Viola, E. (2015, July). Local reductions. In *International Colloquium on Automata, Languages, and Programming* (pp. 749-760). Springer, Berlin, Heidelberg.

Jalaei, N., & Safi-Esfahani, F. (2021). VCSP: virtual CPU scheduling for post-copy live migration of virtual machines. *International Journal of Information Technology, 13*(1), 239-250.

Kaur, T., & Kumar, A. (2022). Virtual migration in cloud computing: A survey. In *International Conference on Innovative Computing and Communications* (pp. 785-796). Springer, Singapore.

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

Khaneghah, E. M., Khoshrooynemati, T., & Feyziyev, A. (2022). ExaLazy: A Model for Lazy-Copy Migration Mechanism to Support Distributed Exascale System. *Azerbaijan Journal of High Performance Computing, 4*(1), 170-187.

Khaneghah, E. M., Mollasalehi, F., Aliev, A. R., Ismayilova, N., & Bakhishoff, U. (2018). Challenges of load balancing to support distributed exascale computing environment. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA)* (pp. 100-106). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N. (2018, February). Challenges of process migration to support distributed exascale computing environment. In *Proceedings of the 2018 7th international conference on software and computer applications* (pp. 20-24).

Khaneghah, E. M., ShowkatAbad, A. R., et al. (2018). ExaMig matrix: Process migration based on matrix definition of selecting destination in distributed exascale environments. *Azerbaijan Journal of High Performance Computing, 1*(1), 20-41.

Khorandi, S. M., Mirtaheri, S. L., Khaneghah, E. M., Sharifi, M., & Ghiasvand, S.

(2011, December). Local robustness: A process migration criterion in HPC clusters. In *International Conference on Innovative Computing Technology* (pp. 374-382). Springer, Berlin, Heidelberg.

Kruglick, E. (2015). *U.S. Patent No. 9,189,271.* Washington, DC: U.S. Patent and Trademark Office.

Ma, F., Liu, F., & Liu, Z. (2010, July). Live virtual machine migration based on improved pre-copy approach. In *2010 IEEE International Conference on Software Engineering and Service Sciences* (pp. 230-233). IEEE.

Maeda, S., Sato, K., Sakiyama, N., Yano, H., & Hayashi, T. (2007). *U.S. Patent No. 7,313,599.* Washington, DC: U.S. Patent and Trademark Office.

Matsuzawa, K., Hayasaka, M., & Shinagawa, T. (2018, June). The quick migration of file servers. In *Proceedings of the 11th ACM International Systems and Storage Conference* (pp. 65-75).

Matsuzawa, K., Hayasaka, M., & Shinagawa, T. (2020). Practical quick file server migration. *ACM Transactions on Storage (TOS), 16*(2), 1-30.

Matsuzawa, K., Hayasaka, M., & Shinagawa, T. (2020). Practical quick file server migration. *ACM Transactions on Storage (TOS), 16*(2), 1-30.

Milojičić, D. S., Douglis, F., Paindaveine, Y., Wheeler, R., & Zhou, S. (2000). Process migration. *ACM Computing Surveys (CSUR), 32*(3), 241-299.

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

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

Nimbalkar, M. V., Pathak, G. R., & Nagargoje, H. (2015, February). Mobile agent: Load balanced process migration in Linux environments. In *2015 International Conference on Computing Communication Control and Automation* (pp. 561-564). IEEE.

Noshy, M., Ibrahim, A., & Ali, H. A. (2018). Optimization of live virtual machine migration in cloud computing: A survey and future directions. *Journal of Network and Computer Applications, 110,* 1-10.

Pecholt, J., Huber, M., & Wessel, S. (2021, November). Live Migration of Operating System Containers in Encrypted Virtual Machines. In *Proceedings of the 2021 on Cloud Computing Security Workshop* (pp. 125-137).

Pickartz, S., Gad, R., et al. (2014, August). Migration techniques in HPC environments. In *European Conference on Parallel Processing* (pp. 486-497). Springer, Cham.

Priyanka, H., & Cherian, M. (2020). The Challenges in Virtual Machine Live Migration and Resource Management. *International Journal of Engineering Research & Technology, 8*(11).

Richmond, M., & Hitchens, M. (1997). A new process migration algorithm. *ACM SIGOPS Operating Systems Review, 31*(1), 31-42.

Rodrigues, M., Roma, N., & Tomás, P. (2015, October). Fast and scalable thread migration for multi-core architectures. In *2015 IEEE 13th International Conference on Embedded and Ubiquitous Computing* (pp. 9-16). IEEE.

Rungta, M. (2006). Transparent Process Migration: Design Alternatives and the Sprite Implementation.

Shah, V., & Donga, J. (2020). *Load balancing by process migration in distributed operating system.* LAP LAMBERT Academic Publishing, 84 pages.

Shan, Z., Qiao, J., & Lin, S. (2018, October). Fix page fault in post-copy live migration with RemotePF page table assistant. In *2018 17th International Symposium on Distributed Computing and Applications for Business Engineering and Science (DCABES)* (pp. 40-43). IEEE.

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

Shribman, A., & Hudzia, B. (2012, August). Pre-copy and post-copy vm live migration for memory intensive applications. In *European Conference on Parallel Processing* (pp. 539-547). Springer, Berlin, Heidelberg.

Singh, G., & Singh, P. (2021). A taxonomy and survey on container migration techniques in cloud computing. *Sustainable Development Through Engineering Innovations,* 419-429.

Stoyanov, R., & Kollingbaum, M. J. (2018, June). Efficient live migration of linux containers. In *International Conference on High Performance Computing* (pp. 184-193). Springer, Cham.

Su, K., Chen, W., Li, G., & Wang, Z. (2015, December). Rpff: A remote page-fault filter for post-copy live migration. In *2015 IEEE International Conference on Smart City/SocialCom/SustainCom (SmartCity)* (pp. 938-943). IEEE.

Takagawa, Y., & Matsubara, K. (2019). Yet another container migration on FreeBSD. *AsiaBSDCon 2019 Proceedings,* 97-102.

Thakkar, N., & Pandya, A. (2013). Process migration in heterogeneous systems. *International Journal for Scientific Research & Development, 1*(7), 2321-0613.

Thulasidasan, S. (2000). Issues in process migration. *University of Southern California, December, 15,* 1-10.

Van Steen, M., & Tanenbaum, A. (2002). Distributed systems principles and paradigms. *Network, 2,* 28.

Weinhold, C., Lackorzynski, A., et al. (2016). FFMK: A fast and fault-tolerant microkernel-based system for exascale computing. In *Software for Exascale Computing-SPPEXA 2013-2015* (pp. 405-426). Springer, Cham.

Zarrabi, A. (2012). A generic process migration algorithm. *International Journal of Distributed and Parallel Systems, 3*(5), 29.

Zarrabi, A., Samsudin, K., & Ziaei, A. (2013, March). Dynamic process migration framework. In *2013 International Conference of Information and Communication Tech-*

nology (ICoICT) (pp. 410-415). IEEE.

Zhao, J., Hu, L., Xu, G., Chang, D., Ding, Y., & Fu, X. (2013). A fast live migration algorithm of virtual machine with CPU scheduling. In *Proceedings of the international conference on grid, cloud, and cluster computing (GCC)* (p. 115). The Steering Committee of The World Congress in Computer Science, Computer Engineering and Applied Computing (WorldComp).

Submitted: 26.12.2021
Accepted 03.05.2022