Azerbaijan Journal of High Performance Computing, Vol 3, Issue 2, 2020, pp. 151-163
https://doi.org/10.32010/26166127.2020.3.2.151.163

Challenges of Using Live Process Migration in Distributed Exascale Systems

Zeinab Sohrabi, Ehsan Mousavi Khaneghah

Department of Computer Engineering, Faculty Engineering, Shahed University, Tehran, Iran, Zeinab.Sohrabi@shahed.ac.ir, EMousavi@Shahed.ac.ir

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

Abstract

Virtual machine-based process migrator mechanisms have the potential to be used in distributed exascale systems due to their ability to execute process execution and support environments with the heterogenous of the computational unit. The ability to reduce process suspension time and use the concept of live process migrator makes it possible to use this mechanism to transfer processes in distributed exascale systems to prevent related process activity failure. The performance function of a virtual machine-based process migrator mechanism cannot manage dynamic and interactive events and the effects of this event on the mechanism operation and the change in the basic concept of system activity from the concept of the process to the concept of global activity. This paper examines the challenges of dynamic and interactive event occurrence on virtual machine-based process migrators by analyzing VM-based migrator's performance function.

Keyword: Virtual Machine-Based Process Migration, Distributed Exascale Systems, Dynamic and Interactive Events, Process State.

1. Introduction

In traditional high performance computing systems, if the computational unit is not capable of executing the process or cannot meet the time constraints of processing the process, and there is a computational unit in the system in which it is possible to continue the execution process, the load balancer uses the process migrator to change the process execution unit (Khaneghah, E. M., ShowkatAbad, A. R., & Ghahroodi, R. N., 2018, February; Duolikun, D., Watanabe, R., Enokido, T., & Takizawa, M., 2017, March; Khaneghah, E. M., ShowkatAbad, A. R., et al., 2018). In traditional high performance computing systems, the process migrator based on (source, destination, selected process) transfers the process from the source to the destination. The migrant process should be able to continue in the destination. In the traditional system, the ability to continue the process of process execution in the computational unit as well as the concept of suspension time are two essential factors in determining the mechanisms and functional patterns of the process migrator (Jain, N., Menache, I., Shepherd, F. B., & Naor, J. S., 2017; Reghenzani, F., Pozzi, G., et al., 2016, September).

If the mechanism used by the process migrator can transfer and maintain the status of the process during the implementation of activities related to the process migrator and also the destination can provide the required process resources based on status to have the current execution, it is possible to continue the execution of the process in the destination (Takagawa, Y., & Matsubara, K., 2019; Pickartz, S., Breitbart, J., & Lankes, S., 2016; Gholami, M. F., Daneshgar, F., Low, G., & Beydoun, G., 2016).

During its suspension, the migrant process is unable to respond to requests from other processes, especially interactions with these processes. The inability of the process to respond to other processes increases the execution time of the process and, consequently, the execution time of the scientific application of which the process is a member of it. Numerous mechanisms have been proposed for the process migration, although they have been proposed to adapt to the various scientific application requirements in the field of migration. However, in most of them, the mechanical design's focus is on reducing the suspension time and the possibility of interacting with other processes during the suspension time. The inability of the process to interact with other processes and increase the scientific program's execution time may cause challenges to other processes that are members of the computing system in the continuation of their implementation process. The mentioned lack of interaction and communication may cause executing other processes to fail (Jain, N., Menache, I., Shepherd, F. B., & Naor, J. S., 2017; Reghenzani, F., Pozzi, G., et al., 2016, September).

Multiple process migration mechanisms attempt to transmit the program code section of the migrant process at once. During the code's migration, because the program code is not running in either source and destination, it has no execution capability, and it cannot be said that the process is running. This causes the process to lack the ability to communicate with other processes and not being able to run. Numerous process migration mechanisms try to reduce the process suspension time by using the concept of instantaneous transfer (Jain, N., Menache, I., Shepherd, F. B., & Naor, J. S., 2017; Reghenzani, F., Pozzi, G., et al., 2016, September; Noshy, M., Ibrahim, A., & Ali, H. A., 2018).

In addition to the program code section, the process has a data section related to the process. Process data has a higher volume compared to the program code section. The full-copy process migration mechanism transfers all parts of the program data to the destination, and the flushing migration mechanism transfers the data to the server in one go. Process migration mechanisms gradually transfer data over some time or in exchange for observing a specific event. The gradual transfer of data and reducing the non-running time of the migratory process allow the transfer of data according to the requirements of the migratory process from the source to destination. Because to execute the process, there is a need for the program code along with the process execution data for a specific execution program code; the process suspension includes the start time of the process execution suspension until the data transfer required to start the process execution or continuation of the execution process. During this

period, the process is suspended (Varadarajan, S., & Ruscio, J., 2009; Junior, P. S., Miorandi, D., & Pierre, G., 2020, December).

In traditional systems, the concept of virtual machine-based process migration is used to manage this situation and enable the execution of the process during the transfer period. In the virtual machine-based process migration mechanism, the migratory process migrates between the source and destination in a way that does not stop providing process services or interactions and connections between other processes and the migratory process. In this mechanism, memory resources, files and inputs, and outputs, especially network communications and processing resources related to the migratory process, are transferred from the destination. This transfer is so that the process of communication with other processes or process service does not stop. In traditional systems, if the migration time of the virtual machine containing the process from the source to the destination is negligible compared to the transfer time of the process without the virtual machine, then the migration of virtual machine-based processes in the sentence will be a live migration (Noshy, M., Ibrahim, A., & Ali, H. A., 2018; Singh, G., & Gupta, P., 2016, September).

In distributed exascale systems, dynamic and interactive events can occur at any time in the program execution. The occurrence of a dynamic and interactive event during the migration of a migratory process may alter the elements influencing the transfer of the process that causes the process migrator to fail. If traditional mechanisms are used for process migrators, the failure of the process migrator activity, unlike the failure of the other elements that make up the system manager, cannot be detected until the end of the activity.

In traditional process migrator, such as virtual machine-based process migration mechanisms, the process migrator does not collect information from the status of the elements of the computing system beneficiaries of the process migration, and if the status changes, the elements affecting the process migration are unaware of the new situation and operate based on information submitted by the load balancer. This is especially true of the virtual machine-based process migration mechanism. In a virtual machine-based process migration mechanism, the migratory process is transferred in the form of a virtual machine. This allows more information sets to be transmitted than traditional migration mechanisms in this way. Any change in the beneficiary status in process migration may violate the function of the virtual machine-based process migration (Singh, G., & Gupta, P, 2016, September; Gharb, H., Khaneghah, E. M., et al., 2019).

In this paper, while analyzing the function of the virtual machine-based process migration mechanism and examining its unique features, the effects of dynamic and interactive events on the mechanism's function will be investigated. Analyzing the effects of the dynamic and interactive events on the process migration mechanism's functioning makes it possible to consider what features will cause the migration mechanism of virtual machine-based processes while maintaining compatibility is

used in distributed exascale systems.

2. Related Work

In (Tanenbaum, A. S., Van Renesse, R., et al., 1990), the amoeba operating system is proposed to reduce full migration suspension. This operating system uses a complete copy mechanism to transfer the process. This distributed operating system is used for two reasons. The first reason is its independent addressing and kernel design; the second reason is its availability feature, which can be easily implemented on basic equipment configuration at no high cost. The distributed amoeba operating system is used to implement parallel algorithms and a software platform for a distributed programming language. For several reasons, Amoeba is a good platform for migration. Amoeba communication mechanisms have spatial transparency. Its micro-core design maintains the process's status, especially in files and machines that access servers through transactions (Steketee, E., Zhu, W. P., & Moseley, P., 1994, June).

In (Smith, J. M., 1988), the system V operating system is introduced based on process migration to reduce the suspension time. In this article, the system V system is a distributed operating system that runs on a cluster. Each host runs on an identical small kernel with some runtime modules. System V provides transparency, minimal connections, and residual dependencies from the source machine. The System V operating system gathers load information from other workstations, and when the migration request is issued, it selects the first available workstation and starts the migration process. In this operating system, users can control process migration, and by executing special commands, they can transfer local processes to idle machines.

In (Milojičić, D. S., Douglis, F., et al., 2000), the Sprite operating system is described to reduce residual interdependencies between processes. The operating system provides ease and transparency for both users and applications by creating processes during a host's execution. Processes can re-access remote resources at any time, including files, machines, and network connections from multiple locations. A user returns to workstations when their processes are off, this process quickly returns to its source and runs there, or it may be suspended elsewhere. The main purpose of Sprite's operating system is to minimize residual dependencies.

In (Zheng, Y., & Nicol, D. M., 2011, June), the OpenVZ mechanism is a type of carrier that allows multiple sets of processes to run separately. This mechanism is based on a hierarchical processing structure. It is known as a single bed as a single core sample. Each separate sample can be inspected to save a carrier's full status and then restart. Inspection and restart can be loaded as kernel modules. Process carriers are stored in a continuous state. Dependencies such as processing hierarchies, identifiers, and shared resources are stored and retrieved during reboot. One of the advantages of this method is the execution of several sets of separate processes and also, this algorithm does not require any special hardware (Kovari, A., & Dukan, P., 2012, September).

In (Barham, P., Dragovic, B., et al., 2003) addresses the Xen operating system. Xen

was the first hypervisor to be implemented at the system level. Xen uses two patterns to convey status: either process-based or automated. Virtual machine migration in Xen is such that managers can migrate live with physical hosts across the local network with Xen virtual machines. During this local procedure, it repeatedly copies the virtual machine memory to the destination without stopping. This process requires a pause of about 60-300 milliseconds to perform the final coordination before starting the virtual machine at the final destination, which provides integrated migration (Gupta, D., Cherkasova, L., Gardner, R., & Vahdat, A., 2006, November).

Another virtual machine-based process migration mechanism mentioned in (Chirammal, H. D., Mukhedkar, P., & Vettathu, A., 2016) is the KVM mechanism. Live migration in this type of virtual machine uses the preset mechanism, which means that if a guest page changes after copying, that page must be copied again. KVM executes the infected page implementation, which uses it as a bitmap of the last modified pages. KVM either reads guest pages or maps them and maps them to write after the first write access.

As mentioned in (Zheng, Y., & Nicol, D. M., 2011, June), MOSIX is a virtual machine-based process migration mechanism. It is the first high-performance computing operating system based on process migration. Process migrator in MOSIX means that source and destination nodes work together to make an immigration decision. During migration, only infected pages and process migration environments are transferred, while clean pages enter the page whenever an error occurs in the destination node. This operating system's advantages include scalability and dynamic configuration structure and error tolerance (Barak, A., Guday, S., & Wheeler, R. G., 1993).

In (Barak, A., & La'adan, O., 1998), the single-system image mechanism, or SSI, of virtual machine-based process migration is introduced. This mechanism's main purpose is to provide a single machine for all processes that can manage inhomogeneities by keeping users away from the underlying layers. This mechanism tries to consider all resources as local processing resources. One of the benefits of SSI is transparent access to the process and its resources, which facilitates process migration.

3. Virtual Machine Live Migration Mechanism Function

In traditional high performance computing systems, the result of calling the process migrator by the load balancer is either success and transfer of the process or failure and non-transfer of the process (Mousavi Khaneghah, E., Noorabad Ghahroodi, R., & Reyhani ShowkatAbad, A., 2018). From the perspective of the process migrator, the inability to transfer the process means the occurrence of a situation that makes it impossible to transfer the code or data, based on the mechanism used to transfer the process. With this approach, if the process transfer fails, the mechanism used to transfer code or data lacks transfer capability. This lack of capability can be due to a) the impossibility of transferring the code from the source to the destination, b) the inability to execute the code transferred to the destination, c) the inability to correctly

transfer the processing status from the source to destination, d) Lack of sufficient information transfer to start the execution of the process in the destination, e) Inability of the mechanism to transfer data, and f) Inability to transfer data commensurate with the needs of the migratory process in the destination.

By classifying the mentioned situations regarding the inability of the process migrator's mechanism to transfer the process, it can be stated that the failure of the activities related to the process migrator. It can be considered as a) transfer challenges b) non-execution challenges c) incorrect destination challenges d) disproportionate mechanism challenges, e) process status disruption challenges. In the traditional high performance computing system, a virtual machine-based process migration mechanism is used to manage transfer, non-execution, and process status challenges.

The virtual machine concept makes it possible for the process migrator to transfer the virtual machine containing the process that requires the migration of a process from the source to the destination to transfer the process. The transfer of the virtual machine from the source to the destination focuses on the heterogeneous and the possibility of executing the process. The virtual machine concept makes it possible for the process migrator to transfer the virtual machine containing the process that requires the migration of a process from the source to the destination to transfer the process. The transfer of the virtual machine from the source-to-destination enables the migration mechanism to continue processing in the destination. A virtual machine migration mechanism is a set of activities performed to transfer the status of a virtual machine, including memory pages and CPU state, from one physical machine to another. Based on this definition, the virtual machine-based process migration function can be considered in the form expressed in Eq.1.

$$f(Migrtion_{live}) = \begin{bmatrix} process_{state}(t,e) & process_{memory}(t,e) \\ process_{dependency(p,t)} & VM_{state}(t) \end{bmatrix} (state_{vm\&process}) \quad Eq.1$$

As can be seen in Eq.1, the process migrator's function is a 2 * 2 matrix, which is a function of the independent variable of the status of the migratory process. Eq.1 implicitly states that the application or process is running on the destination in the virtual machine.

In Eq.1, the process state (t, e) variable indicates the process's status. The process state variable is described based on two independent time variables and events that lead to a process state change. Given that the process runs in a virtual machine, the conceivable situations for the process can be considered according to the pattern shown in Figure 1.

As shown in figure 1, the combinations of the two elements of the virtual machine and the processor in the ready and running modes can be considered acceptable states for the processor and the virtual machine. In figure 1, the virtual machine is considered a process, making it possible to consider four states related to the ready and running of the two vice-virtual machine processes and the migratory process.

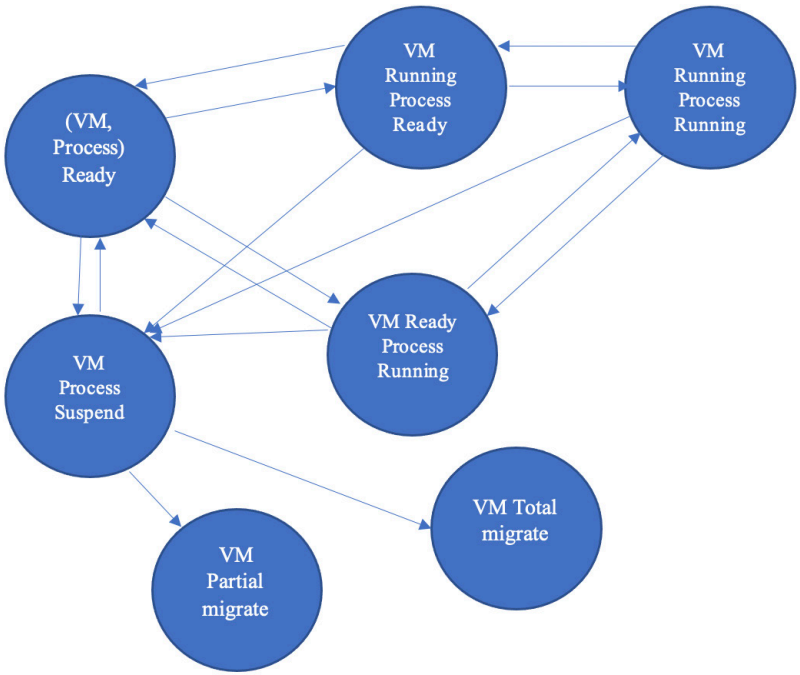

Fig. 1 Processes and virtual machine states

Figure 1 defines the suspension mode as well as the migration mode for the virtual machine. Figure 1 shows that the virtual machine's suspension and migration status are also considered the suspension and migration of the process. In high performance computing systems, the concept of virtual machine-based process migration is based on two general patterns of complete virtual machine migration or partial virtual machine migration.

In the complete virtual machine migration pattern, the process of executing the virtual machine in the source stops, and the virtual machine is transferred from the source to the destination. In the partial virtual migration pattern, the data-related part of the process is transferred from the virtual machine running in the source to the destination's corresponding virtual machine. As in figure 1, in traditional high performance computing systems, after changing from a suspended state to a partial or complete migration state, the virtual machine cannot return and stop working. The function of the virtual machine-based process migration in traditional high performance computing systems is based on the existence of high trust in the migration implementation process, heterogeneous management, and the management of the process implementation process.

The axial of the virtual machine-based process migration mechanism is based on the fact that either the virtual machine transferred from the source to the destination, implements the processing phase and the concept of virtual machine necessarily

manages any heterogenous and challenges in virtual machine transfer or the corresponding virtual machine in the destination, allows the process to continue its execution process in the corresponding virtual machine by transferring status and data. In the mechanism of migration of processors based on a virtual machine, the concept of migration activity failure is not considered. The nature of the virtual machine-based process migration mechanism is the feasibility of the process execution process in the destination based on the virtual machine concept.

As can be seen in Eq.1, the virtual machine-based process migration element's function is defined on a variable axis (state (vm & process)). This is due to the possibility of considering dissimilar states for both the processing element and the virtual machine. The variable (state (vm & process)) indicates the function description of the process migrator. In a virtual machine-based process migration mechanism, unlike other mechanisms defined for the operation of data-driven migration management based on data and time, focuses are defined based on the status of the main elements of migration.

In other mechanisms used by the process migrator, the activity's focus is on reducing the process suspension time by changing the data transfer process. In other mechanisms, the main assumption of the process migration mechanism design is that the time required to transfer the code is negligible compared to the time required to transfer the data. This means that these mechanisms define data-drivenness and how data is transmitted in the shortest time or only the required data is transmitted. In a virtual machine-based process migration mechanism, the virtual machine transfer time or the virtual machine stop time in the source, and the virtual machine start operating in the destination are negligible in many cases. Based on the virtual machine, time is not a concept. There is no data-driven focus in a virtual machine-based process migration mechanism either because data is transmitted as part of the virtual machine.

In the virtual machine mechanism, the function description is based on the status of the process, the memory of the process, the dependence of the process on other processes, and the virtual machine's status with the focus on the status of the process and the virtual machine. The process state variable, the memory state of the process in the virtual machine, and the IPC between the process and other processes are defined based on two variables, time and event. During the execution process or as a result of the event, the process's status, the process memory, or the interactions and connections of the process with other processes may change.

4. What Means Dynamic and Interactive in Live Process Migration Terminology

In distributed exascale systems, dynamic and interactive events can occur at any time in the execution time. A dynamic and interactive event causes the state of the system descriptor parameters to change so that the mechanism (or mechanisms) used by the constituent elements of the system manager cannot manage and execute activities. The occurrence of a dynamic and interactive event and the state of the system

is beneficial in terms of the migration mechanism's functioning. The process migration mechanism does not collect information about its activities after being called by the load balancer. This causes that after changing the system status and the parameters describing the system status from the perspective of the process migrator, the system status with the status considered by the process migrator that corresponds to the system status at the time the call is by the load balancer, there is a difference.

The difference between the system's status from the point of view of the process migrator and the current status of the system may change the cause of migration or the status of the beneficiary in migration. This is especially true of the virtual machine-based migration mechanism, which focuses on the status of the system and the beneficiary in migration activities. In a virtual machine-based process migration mechanism, the function is defined based on the process's status and the virtual machine. The virtual machine-based process migration element did not collect information about the system's status and the beneficiary in the migration activity, especially the status related to the process and the virtual machine, and based on the information received from the process migrator, the process starts working.

The load balancer's information is related to the moment t=Alpha, which has called the process migrator. The process migrator does not change the data structure of the process information and the virtual machine during the execution period of the process migrator activities. This issue in traditional high performance computing systems, because the status of the process and the virtual machine does not change, and the status of the description of the source and destination, does not cause problems in the implementation of virtual machine-based process migrator activities.

However, in distributed exascale systems, changing each of the beneficiaries' status may make it impossible for the process migrator to continue to operate. Due to this issue, the functional space of the process migrator can be considered as (<DState, SState, VMState, PState>, Process, VM, <Transfer>). In defining the functional space of the virtual machine-based process migrator, the focus of the definition is based on the concept of the status of the process and the virtual machine and the source and destination. The system's activity is defined on the two elements of the processor and the virtual machine, and the activity that can be executed in the system is also transferred.

According to Eq.1, as well as considering the migration system of virtual machine-based processes and beneficiary in it, the effects of dynamic and interactive events on the function of the migration of virtual machine-based processes can be based on Figure 2.

As can be seen in figure 2, the occurrence of a dynamic and interactive event can affect any of the factors that define the functional space of the manager and the concept of the capabilities of the virtual machine in the process migrator process. When a dynamic and interactive event occurs in the migration system, it may depend on the status of the process, the virtual machine, the source, destination, and the capabilities

of the virtual machine used to process the migration. The impact of the event on the status of each of these factors may cause the virtual machine-based process migrator to lack execution capability.

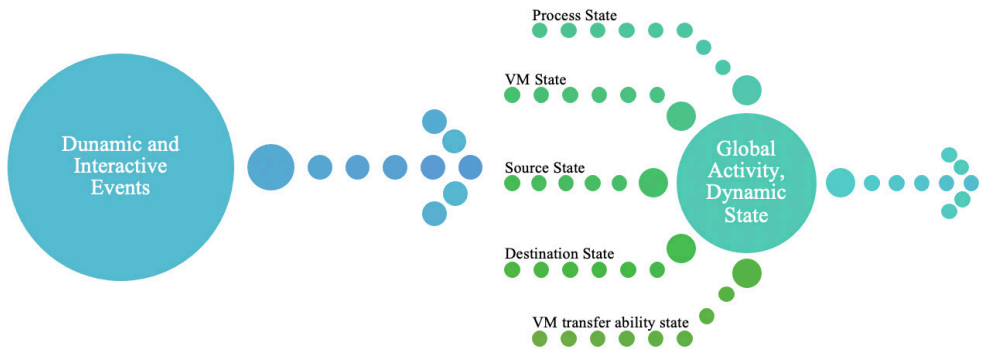

Fig. 2 The effects of dynamic and interactive events on the performance of the virtual machine-based process migrator

5. Challenges

Defining the functionality of a virtual machine-based process migrator is axial to describing the situation. The occurrence of a dynamic and interactive event may also change any of the descriptions used in the process migration system at any point in the process of implementing the activities of the virtual machine-based process migrator. This eliminates the need to change the function and descriptor function of the process migrator. In distributed exascale systems, unlike traditional high performance computing systems, the axial element of system activities is based on the concept of global activity. Using the concept of global activity causes the space of influence and influential of the process to change. In traditional high performance computing systems, the process state definition space is one-dimensional because the process is an abstract concept, while in a distributed exascale system changes to three-dimensional (Process State, Communication State, Global Activity State).

In distributed exascale systems, the migration of virtual machine-based processes due to a) the possibility of continuing to execute the process b) the possibility of executing the process in heterogeneous environments the load balancer is also responsible for transferring the process to create a responsive structure. In distributed exascale systems, the migration of virtual machine-based processes must be capable of transferring processes while maintaining communications and process interactions with other global activity related to processing or other global activities. This causes

that in addition to the interaction space in the migration's useful function, processes need to consider interactive pages and communication between processes.

In distributed exascale systems, if the process migrator's mechanism is a mechanism other than a virtual machine-based process migrator, then the dynamic and interactive nature of the computational element occurs. The source causes the status of the computational element to change for the process. This change may cause the constraints on the process to change so that either the cause of the process migrator is violated or the process migrator needs to be created. This is also true for the destination. The occurrence of a dynamic and interactive event may cause the destination's conditions to change for the migratory process so that the cause of the transfer to the computational element is violated, or the computational element can accept the migratory process. This is while if the migration mechanism of the process is based on a virtual machine, a dynamic and interactive event causes the status of source or destination for the virtual machine as a process. This change may lead to the cause of the process migrator being violated or the need for process migration, the occurrence of a dynamic and interactive event may also affect the virtual machine as a machine containing migratory processes. Thus, in distributed exascale systems, dynamic and interactive events in each computational element and each migratory process create a two-dimensional space.

6. Conclusion

Due to its ability to execute processes and use them in heterogeneous environments, the virtual machine-based process migrator mechanism can be used in distributed exascale systems and used in these environments as a process migrator mechanism. The possibility of executing the process in the shortest possible time in the destination and considering the process interactions as efficient features of this mechanism is distributed. The focus of the virtual machine-based process migrator mechanism is on the process's focus and the virtual machine. This causes the mechanism to be affected by a dynamic and interactive event. The occurrence of a dynamic and interactive event in the virtual machine-based process migrator mechanism, unlike other process migrator mechanisms, is effective in two dimensions. Also, the need to redefine axial activity from the processing element to the global activity element is another challenge of applying this mechanism in distributed exascale systems.

Reference

Barak, A., & La'adan, O. (1998). The MOSIX multicomputer operating system for high performance cluster computing. *Future Generation Computer Systems, 13*(4-5), 361-372.

Barak, A., Guday, S., & Wheeler, R. G. (1993). *The MOSIX distributed operating system: load balancing for UNIX (Vol. 13)*. Berlin: Springer-Verlag.

Barham, P., Dragovic, B., et al. (2003). Xen and the art of virtualization. *ACM*

SIGOPS operating systems review, 37(5), 164-177.

Chirammal, H. D., Mukhedkar, P., & Vettathu, A. (2016). Mastering KVM virtualization. Packt Publishing Ltd.

Duolikun, D., Watanabe, R., Enokido, T., & Takizawa, M. (2017, March). An eco migration of virtual machines in a server cluster. In *2017 IEEE 31st International Conference on Advanced Information Networking and Applications (AINA)* (pp. 1098-1105). IEEE.

Gharb, H., Khaneghah, E. M., et al. (2019) Challenges of execution trend in distributed Exascale system. *Journal of Distributed Computing and Systems, 2*(1), 140-151.

Gholami, M. F., Daneshgar, F., Low, G., & Beydoun, G. (2016). Cloud migration process - A survey, evaluation framework, and open challenges. *Journal of Systems and Software,* 120, 31-69.

Gupta, D., Cherkasova, L., Gardner, R., & Vahdat, A. (2006, November). Enforcing performance isolation across virtual machines in Xen. In ACM/IFIP/USENIX International *Conference on Distributed Systems Platforms and Open Distributed Processing* (pp. 342-362). Springer, Berlin, Heidelberg.

Jain, N., Menache, I., Shepherd, F. B., & Naor, J. S. (2017). U.S. Patent No. 9,619,297. Washington, DC: U.S. Patent and Trademark Office.

Junior, P. S., Miorandi, D., & Pierre, G. (2020, December). Stateful Container Migration in Geo-Distributed Environments. In *CloudCom 2020 12th IEEE International Conference on Cloud Computing Technology and Science*.

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.

Kovari, A., & Dukan, P. (2012, September). KVM & OpenVZ virtualization based IaaS open source cloud virtualization platforms: OpenNode, Proxmox VE. In *2012 IEEE 10th Jubilee International Symposium on Intelligent Systems and Informatics* (pp. 335-339). IEEE.

Milojičić, D. S., Douglis, F., et al. (2000). Process migration. *ACM Computing Surveys (CSUR), 32*(3), 241-299.

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.

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.

Pickartz, S., Breitbart, J., & Lankes, S. (2016). Implications of process-migration in virtualized environments. In *Proceedings of the 1st COSH Workshop on Co-Scheduling of HPC Applications* (p. 31).

Pickartz, S., Lankes, S., Monti, A., Clauss, C., & Breitbart, J. (2016, July). Application migration in HPC - A driver of the exascale era?. In *2016 International Conference on High Performance Computing & Simulation (HPCS)* (pp. 318-325). IEEE.

Reghenzani, F., Pozzi, G., et al. (2016, September). The MIG framework: enabling transparent process migration in Open MPI. In *Proceedings of the 23rd European MPI Users' Group Meeting* (pp. 64-73).

Singh, G., & Gupta, P. (2016, September). A review on migration techniques and challenges in live virtual machine migration. In *2016 5th International Conference on Reliability, Infocom Technologies and Optimization* (Trends and Future Directions) (ICRITO) (pp. 542-546). IEEE.

Smith, J. M. (1988). A survey of process migration mechanisms. *ACM SIGOPS Operating Systems Review, 22*(3), 28-40.

Steketee, E., Zhu, W. P., & Moseley, P. (1994, June). Implementation of process migration in amoeba. In 14th International Conference on Distributed Computing Systems (pp. 194-201). IEEE. J. M. Smith, *"A survey of process migration mechanisms," ACM SIGOPS Operating Systems Review, vol. 22, no. 3,* pp. 28-40, 1988.

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

Tanenbaum, A. S., Van Renesse, R., et al. (1990). Experiences with the Amoeba distributed operating system. *Communications of the ACM, 33*(12), 46-63.

Varadarajan, S., & Ruscio, J. (2009). U.S. Patent No. 7,536,591. Washington, DC: U.S. Patent and Trademark Office.

Zheng, Y., & Nicol, D. M. (2011, June). A virtual time system for openvz-based network emulations. In *2011 IEEE Workshop on Principles of Advanced and Distributed Simulation* (pp. 1-10). IEEE.

Submitted: 20.07.2020
Accepted: 12.11.2020