The XENVMC communication protocol to improve the efficiency of live migration for co-resident virtual machines

Barbara Charchekhandra

Jadavpur University, Department of Mathematics, Kolkata, India

Email: Charchekhandrabar32@yahoo.com

Abstract

One of the most important features offered by virtualization is the live migration of virtual machines, and it is defined as the migration of a virtual machine from one physical host to another physical host while maintaining the continuity of service for applications running on this machine. The PRE-COPY algorithm is one of the most important live migration algorithms. The working principle of this algorithm is based on the frequent transfer of changed memory pages (dirty page) between the source and the target, so that the repetition stops at a certain threshold. This algorithm suffers from a page re-send problem, which is moving the same memory pages repeatedly in each Itern, which increases the amount of data sent over the network, and thus the total live migration time increases and the total of memory pages transferred becomes much larger than The actual size of the memory. In this paper, we build a model to improve the total time of live migration in the event that co-resident Inter-VM communication. We have made a modification to the XENVMC communication protocol, which is a protocol based on the concept of shared memory to handle communication and speed up data transfer between co-resident virtual machines. The modification of the XENVMC protocol has improved the total live migration time and reduced the number of pages transmitted over the network, for two main reasons: the first is to reduce the number of changed pages in memory caused by communication between virtual machines, and the second reason is due to the acceleration of communication and data exchange between that co-resident Inter-VM communication.

Keywords: communication protocol; live migration; virtual; XENVMC

1. Introduction

Cloud computing is developing rapidly due to its flexibility, scalability, energy saving and money saving features, which has made it a research target and a constant quest by researchers to develop its components in order to obtain the best possible performance.

Virtualization is one of the most important components of cloud computing, due to the features it provides by running multiple systems on the same physical device while providing complete isolation between these systems, so that each system works independently on a separate virtual machine (VM).

Live migration is one of the most important features offered by virtual environments, defined as the transfer of a VM virtual machine from one physical device to another on it without interrupting the application service that works in order to save energy, money, distribute the load and perform maintenance operations.

The Virtual Machine Manager (VMM) is the main responsible for the operation of more than one virtual machine on one physical machine, it is also responsible for the process of live migration of the virtual machine and for the mechanism of communication between virtual machines . The principle of operation of live migration algorithms is based on the repeated transfer of variable memory pages(dirty page) between the source and the target, so that the repetition stops at a certain threshold.

The significant change in memory pages will increase the number of pages sent over the network, which increases the total amount of memory transferred, and therefore the network consumption rate, the total time of live migration, downtime and response time of applications running on the virtual machine being migrated will increase.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

The pre-copy algorithm suffers from the problem of repeatedly resending the same pages on each turn, which increases the amount of data sent over the network, and the sum of the transferred memory pages becomes much larger than the actual memory size, which significantly increases the load on the network.
Applying live migration on a vm1 virtual machine located on a host1 host that has a connection and data transfer with a VM2 virtual machine on the same host, or on another host will lead to additional loads and significant memory changes during the migration process caused by requests coming from the client, and by data exchanged between virtual machines connected to each other. Therefore, in our research, we were directed to present a model based on the shared memory approach between virtual machines connected to each other to improve the efficiency of live migration in terms of time and productivity and reduce the load on the network, by bypassing the traditional TCP/IP connection, calls and unnecessary switching between shared virtual machines in hosting.

2-the importance of research and its objectives:

Most of the previous studies were based on the study of live migration and the improvement of its algorithms from only one aspect, which is the application of the migration order on a virtual machine that does not have communication or data transmission with any other machine, whether it is located on the same physical host or on another physical host.
Therefore, the aim of this research was to improve the efficiency of live migration in terms of time, productivity and optimal consumption of network resources, by applying a model based on a shared memory approach to address the connection between connected and shared hosting virtual machines.

3-research methods and materials:

We have modified the XENVMC protocol to support live migration of virtual machines, so that the mechanism of the protocol and the Pre-cop algorithm are taken advantage of to obtain the proposed model (Advance-XENVMC).
Figure (1) shows the practical environment of the experiment, where in our experiment we used three computers, each equipped with a core i3 processor and memories .The 4 GB version of CentOS 7 was installed on both devices and the Xen virtual environment was installed to work as a hypervisor of the partial default mode on both devices, and we used the third device to work as a shared storage device for the virtual disk of the virtual machines, and we also put both servers in a cluster. We have equipped a vm1 virtual machine with a memory size of 1 Mb on one of the servers providing the e-mail service, and we wrote a shell-language program on another VM2 virtual machine that sends and receives e-mails, so that data is exchanged between VM1 and VM2 during the migration of the virtual machine .VM1
We applied different loads during the migration of the vm1 virtual machine by changing the number of clients and the size of the file sent, in order to make a change to the memory pages, and occupy the network during the execution of the live migration order, as changing the memory pages during the transfer of the virtual machine is one of the most important factors in increasing the total time of live migration. Changing the number of requests and the size of the mutual file is done through the following program, which we called Mclient:

```
For I in 1 to n do
For j in 1 to n do
echo . | mail -r test$i.user@XENlab.local -a file -s "This is test"
test$j.user@XENlab.local
```

Since: n the number of users and the file file are variable in size, these are values that are changed for multiple experiences.
For example, for n=20, the number of e-mails on the network will be 400 messages according to the nested for ring, and each message contains a 3 MB file.
We performed different scenarios, calculated the total time of live migration in each scenario, compared the results during the migration of the virtual machine in the case of applying the proposed model, and in the case of applying the pre-copy algorithm without the model.
We ran the NMON Performance Analyzer to get and analyze the results.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

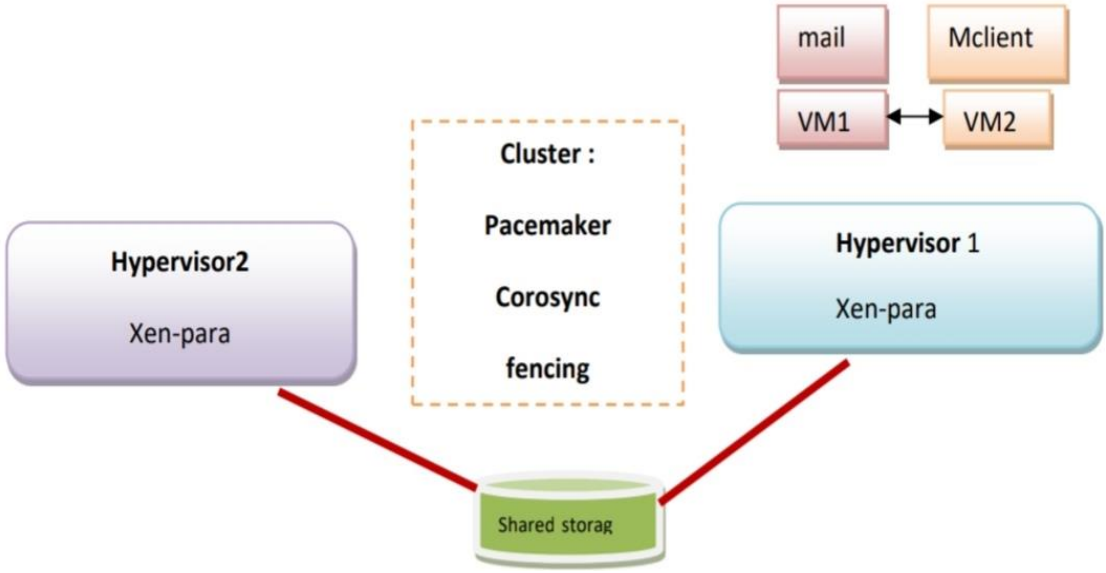

Figure 1: the working environment of the experiment

4-virtualization virtual environment [1]:

It is a technology that mainly aims to divide the computer's physical resources from processor and memory, so that more than one operating system can be run on the same physical hardware, which provides an optimal consumption of physical resources, especially after the great development in the computational capacity and physical hardware of computer hardware, which is not often used and fully invested by the various operating systems and applications running on this hardware. Applications on virtual machine operating systems deal with real physical resources through a Virtual Machine Manager (Hypervisor) called VMM, which is software installed on the host machine of the virtual machine, responsible for forming a virtual abstract layer between applications and physical resources.

The virtual machine manager allocates physical resources by default from the CPU, memory, network card and storage devices to each operating system running on a virtual machine.

The architecture of virtual environments varies depending on the level of isolation between virtual machines and based on the location of the Virtual Machine Manager relative to the physical hardware. Figure (2) shows the position of virtual machines on the virtual environment.

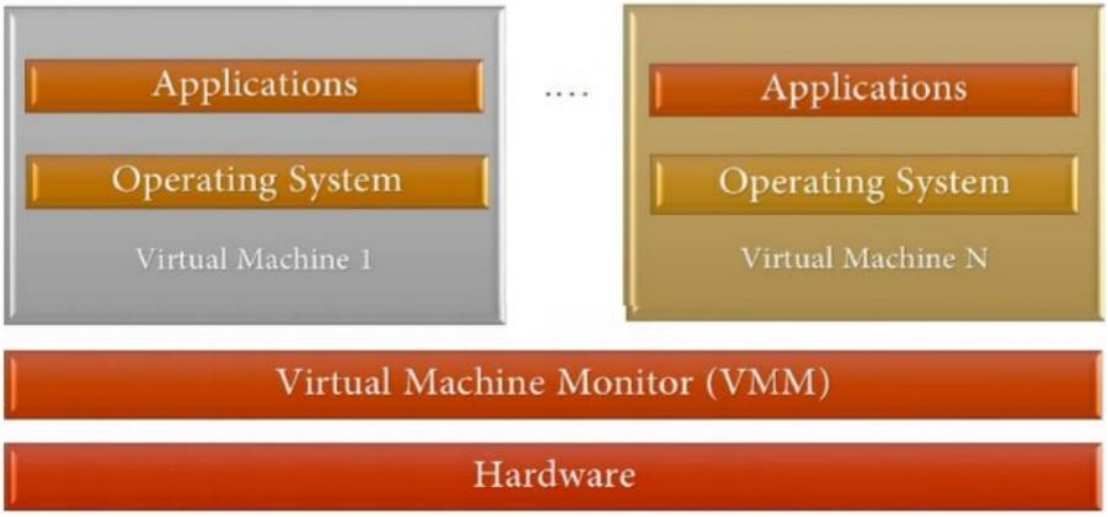

Figure 2: virtual environment

5-live migration of virtual machines Live migration:

Live migration of virtual machines is one of the most important features provided by the virtual environment, defined as migrating the virtual machine from one physical server to another physical server, while ensuring the continuity of the service provided by it with a very small downtime approaching zero, in order to balance the load between physical servers, save energy, or in order to perform maintenance operations for the source physical server [2].

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

The migration of a virtual machine involves the transfer of both memory pages, processor state, input and output processes from the source server to the target. The manager of the virtual machine (hypervisor) is considered the main one responsible for the live migration process, so he must fulfill the following conditions during the live migration application.

1-service continuity: the application of live migration should not cause a deterioration in the performance of applications running on the virtual machine.

2-optimal resource consumption: you should not consume resources significantly during the application of the migration process.

3-prediction: it should be possible to predict the total migration time, downtime and resources that will be consumed on the target server from memory and processing unit and packet display on the network.

4-transparency: the migration process should be transparent for both applications running on the virtual machine and users of these applications.

The pre-copy algorithm is considered the first and most important algorithm that implemented live migration of a virtual machine [4].

6-the pre-copy copying algorithm [5]:

This algorithm is implemented according to several stages as shown in Figure (3), namely:

A. Preparation stage: the resources necessary for the functioning of the virtual machine are reserved after the target host is selected.

b. Repeated copy stage: the manager of the virtual machine copies all the memory pages from the source to the target server, while the virtual machine is still working on the source, if the dirty pages change during this process they will be re-sent and repeat the process from 2 to n-1 times, where the condition to stop sending and move to the next phase is the number of 29 iterations, or the size of the pages changed in the previous transmission is 256 KB, which are the default values of the algorithm.

c. Stop and copy phase: the VM virtual machine on the source is stopped, all the remaining pages and processor registers are transferred to the target, and then the VM resumes working on the target.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

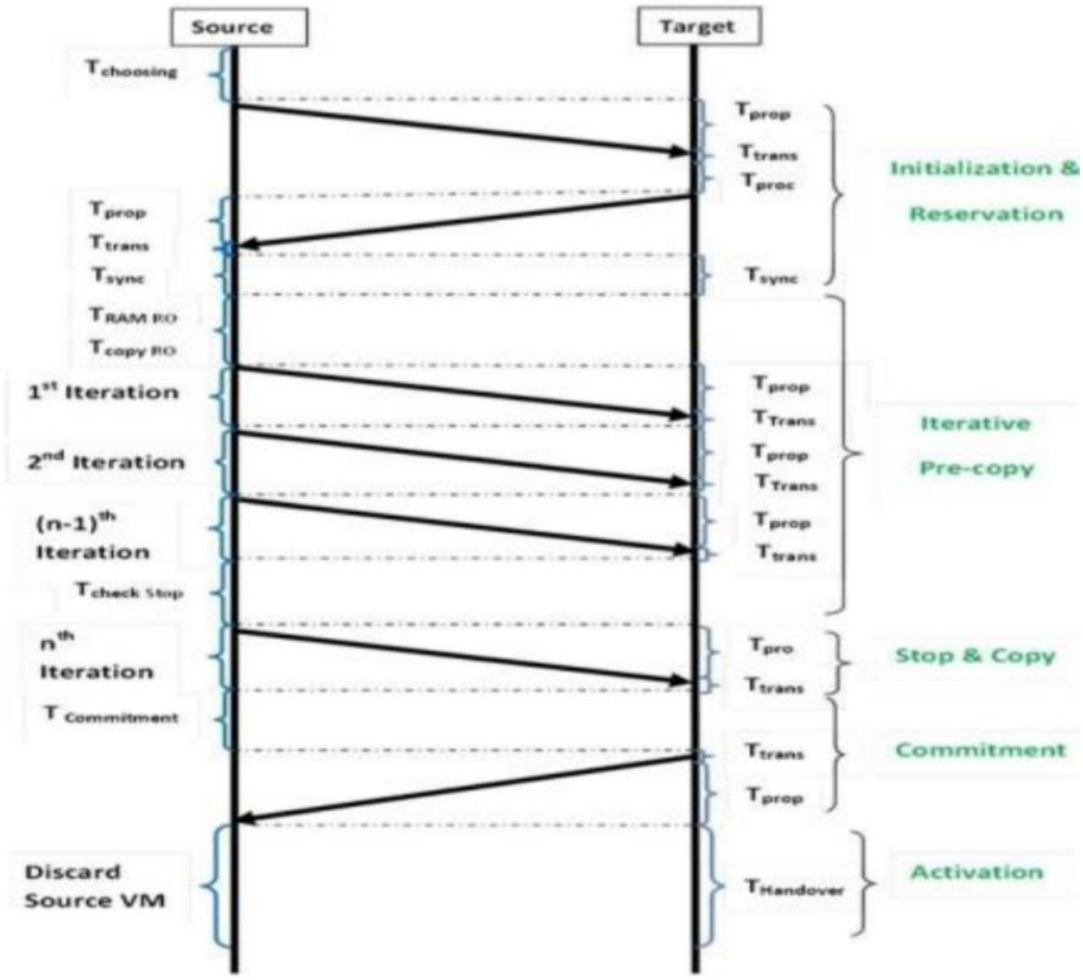

Figure 3: stages of implementation of the pre-pre-copy algorithm

The performance of live migration algorithms is measured by four criteria [6]:

1 - the number of pages transferred: it is the total number of pages transferred during the migration. To get

For best performance this value should be as low as possible, it should also be equal to the total number of pages of the virtual machine being migrated, but in the pre-copy algorithm pre-copy is always more, due to the frequent transfer of changing memory pages during multiple roles.

The total pages transferred Vmig is defined as the total number of pages in all n iterations, and is given by the following relation:

$$\text{Vmig} = \sum_{i=0}^{n} Vi \qquad (1)$$

Where Vi is the number of pages transferred per iteration, and N is the total number of iterations.

2-the total migration time: it is the time taken to transfer the entire virtual machine from the source to the target, and this time should be as little as possible, and is given by the following relationship:

$$\text{Tmig} = \sum_{i=0}^{n} Ti \qquad (2)$$

Where Ti is the time it takes to complete iteration i.

3-downtime :it is the time taken in the migration process to stop the virtual machine at the source and resume work on the target host. This time directly affects the availability of the service, since the values of this time depend on the pages remaining in the last iteration. Downtime is measured as the time it takes to repeat the last migration, or it can be calculated by calculating the service downtime. Figure 4 shows the timeline of live migration.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

4-additional load: it is the additional data that is transferred during migration, which is defined as the size of the memory pages sent during iterations on the real page size of the virtual machine, and is given by the relationship:

$$\text{Rd} = \frac{\text{Vmig}}{\text{Vmem}} \qquad (3)$$

Vmig is the total size of the memory pages moved during the migration and Vmem is as small as possible for Best Performance.

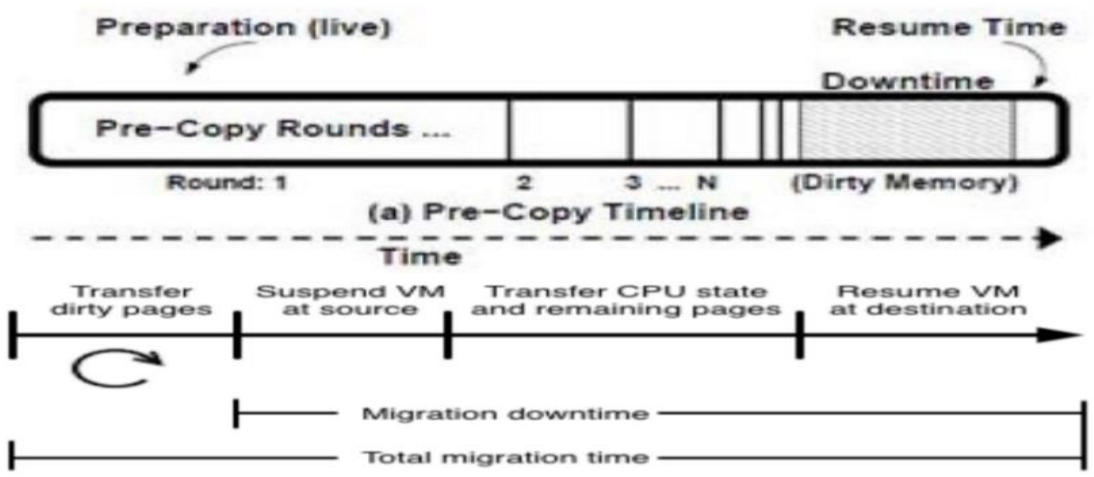

Figure 4: timeline of live migration, total time and downtime during deportation

The pre-copying algorithm suffers from the following problems [7].

- Transfer rate problem:

The changed pages (dirty page) increase at a faster rate than the rate of their transfer over the network, which consumes the network significantly, which affects the continuity of the service and increases the response time of applications running on the virtual machine, and may cause the client to disconnect.

- Page re-send problem:

Sending pages changing in turns from 2 to n - 1 may lead to the transfer of the same Pages repeatedly, which increases the amount of data sent over the network, and therefore the total migration time of the live increases, and the total amount of transferred memory pages becomes much larger than the actual memory size, which significantly increases the load on the network.

The implementation of a live migration command on connected and shared virtual machines while data is being transmitted between them, will significantly increase the two previous problems.

7-the architecture of the Xen virtual environment:

XEN acts as a basic abstract layer between virtual machines and hardware, controls the performance of virtual machines, controls the scheduling processes of the CPU, as well as the division of memory between different virtual machines running on the same hardware. XEN is widely adopted as an open-source virtual machine manager, as it works in full virtual mode, can be modified to work as a partial virtual mode, and also supports live migration of the virtual machine.

The Xen structure when it works in the partial default mode consists of two Domains: Domain (0) and domain(U). Domain (0) is considered a special, privileged user who has access to physical I / O resources and interaction with other virtual machines [8]. All other virtual machines running on Xen are called domain (U).

Modifying Xen to work as a partial default mode requires turning on domain (0) first and then domain (U), as well as installing both Xendo XM and LibXENctrl packages

Domain U is a restricted user that does not have direct access to physical hardware and is managed via domain 0.

Figure (5) shows the Xen structure with the pre-copy algorithm, since to implement the pre-copy algorithm XENBUILT uses special data to transfer memory pages to the virtual machine. XEN uses a shadow page table to record the changed memory pages on the virtual machine during migration, and also uses another binary map

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

called Dirty log bitmap to record the changed pages or dirty pages. Both the shadow page table and the binary map are used to manage the transfers of memory pages to the virtual machine during the execution of the migration order, and for each iteration the binary map is checked to determine the position of the changed pages to be migrated.

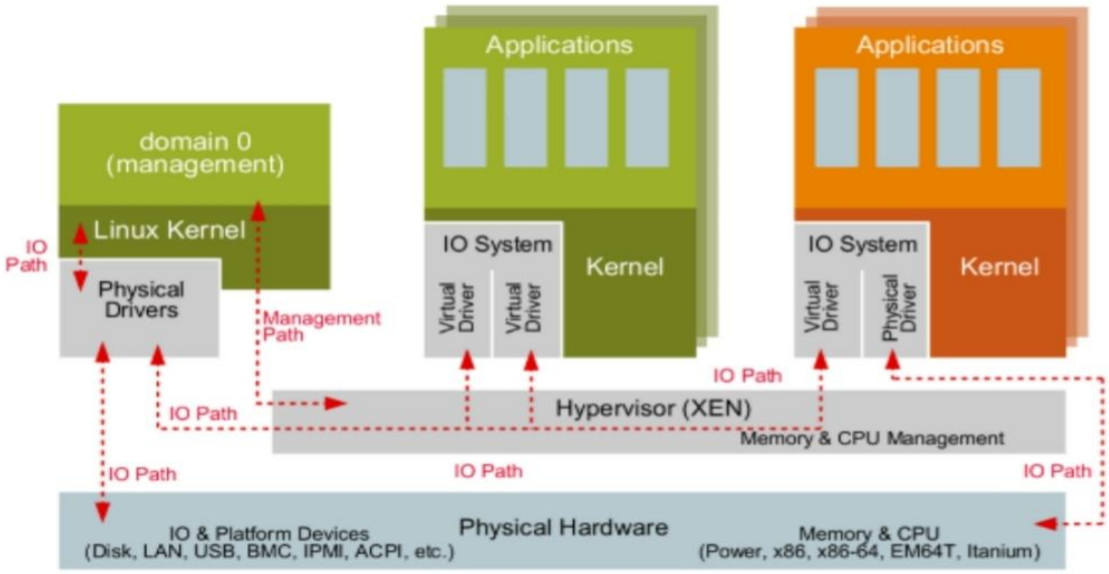

Figure 5: Xen architecture with a pre-copy algorithm

8-network architecture of the Xen virtual environment [9]:

XEN configures a virtual network interface for each virtual machine running on DOMU, so that the real physical network hardware is accessed by domain DOM0, the so-called high-permittivity and isolated domain (IDD).

IDD contains a backend network driver for each virtual machine so that this machine communicates through its frontend network interface with the backend assigned to it within IDD, in order to be able to access the physical network hardware and send and receive data through it. Figure (6) shows the network structure and VEXEN input and output streams.

XEN provides front-end and back-end shared buffers for exchanging input and output operations between connected virtual machines (shared I/O Ring buffers).

Memory page sharing operations for exchanged packets are managed through the granted table and the event Channel. The transfer of data between the sender and the receiver according to this model will lead to significant changes in the main memory pages of both the sending virtual machine and the receiver for each data exchange, which in turn will increase the load during the execution of the live migration order on one of the two virtual machines.

For example, if VM1 sends a data packet TOVM2, the frontend network driver in VM1 first forms grant table references(GR), which point to the memory page containing the packet data in the shared memory cache, and then notifies the backend network driver in IDD through the event channel to fetch the grant table reference .After obtaining GR, VMM uses the interface (GNTTABOP_map) to set the memory page indicated by GR on its own memory, placing the data in the structure sk_buf, so that this data can be processed by the network stack.

After detecting that the destination of this packet is VM2 and it is a co-resident, the back-end driver in the VMM gets GR from the VM2 shared storage memory, copies sk_buf into the memory indicated by VM2'S GR using the GNTTABOP_copy interface, and finally alerts the VM2 virtual machine through the event channel that there is a data packet to be received [10].

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

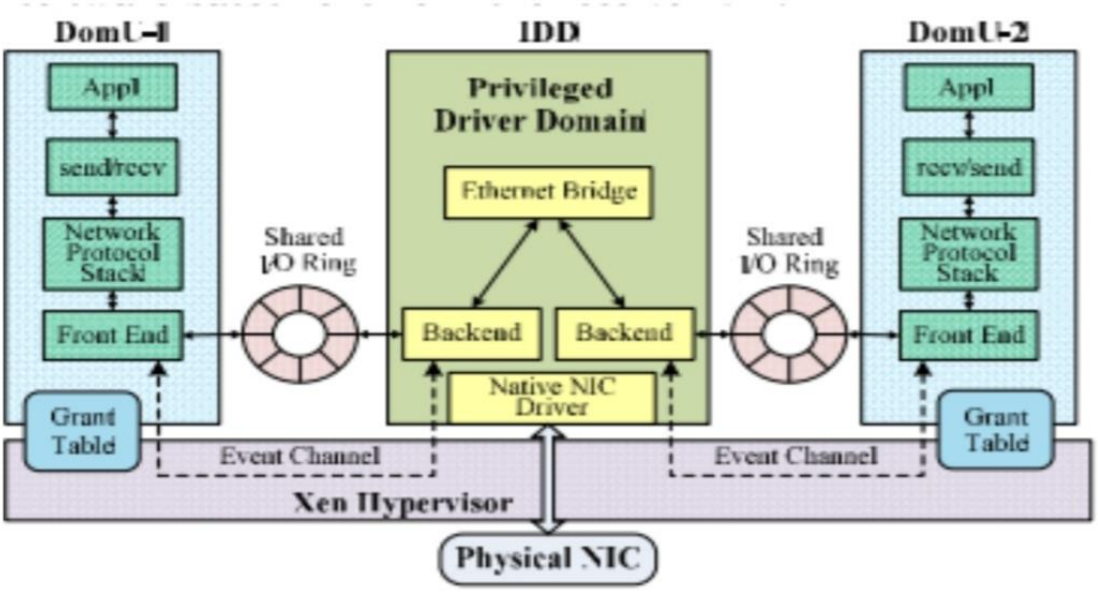

Figure 6: XEN network structure

In short, the transmission of data packets between the sender and the receiver will go through a long path passing by VMM on the sender's side, then the TCP/IP stack, and then VMM on the receiver's side, even though the sender and receiver are located on the same physical environment.

All transmitted packets are addressed by the sender and receiver memory, which causes significant changes in memory, which causes additional burdens during the application of live migration, since the memory used to transmit and send data is a virtual memory formed by the Virtual Machine Manager, so a new protocol or communication form had to be found that bypasses the traditional TCP/IP connection, and achieves communication between virtual machines located on the same host (co-resident VM)without the intervention of the Virtual Machine Manager VMM, via the IDD Connection bridge.

One of the most important of these was the model based on shared memory to improve communication between virtual machines located on the same host, which we will talk about in the following paragraphs.

9-XENVMC communication protocol:

A network communication protocol that handles communication between virtual machines, characterized by transparency and the ability to recognize the location of virtual machines connected to each other, whether they are on the same host (Hypervisor) or on different hosts.

The XENVMC protocol accelerates communication between virtual machines by bypassing the traditional transfer path between linuxdomus, and using the shared memory approach to establish communication between connected virtual machines located on the same physical host, where XENVMC is designed according to the following principles [11].

- High Performance:

XENVMC redirects network requests at the Virtual Machine Manager level to shared memory channels, which leads to shorter connection paths and fewer switches between Vmsovmm, and the evaluation shows that XENVMC improves network throughput up to about 8 times compared to the netfront-netback mode of the XEN network infrastructure, when connecting between virtual machines located on the same host.

- Multi-level transparency :

XENVMC can be used without making any modification to the XEN or Linux kernel, or even to applications running on the virtual machine.

9-1 XENVMC core module [12]:

The XENVMC kernel module contains six subunits as shown in Figure (7), namely:

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

- Connection manager (Connection Manager): responsible for creating or canceling memory-based connections shared between two virtual machines,and encodes each connection through the pair (ip/domID, port) to be accessed later.

- Data Transfer Manage): responsible for sending and receiving data, supports both blocking mode and unblocking via bufferifo

- Event and message Manager: deals with alert notifications related to the transfer of data between the sender and the receiver, so that the mechanism of notifications and messages between virtual devices is implemented across the boundaries of physical devices.

- System Call Analyzer: intercept system calls transparently. Intercepts and analyzes relevant system calls. If a connection is specified between shared co-resident VMs virtual resident machines it bypasses traditional TCP/IP paths.

- Virtual machine State publisher: is responsible for announcing the modification of the shared residence membership of virtual machines located on the same host.

- Live Migration Assistant: supports transparent switching between local and remote mode connection for virtual machines.

To improve the efficiency of live migration of virtual machines connected to XENVMC, modify the communication protocol and shared hosting

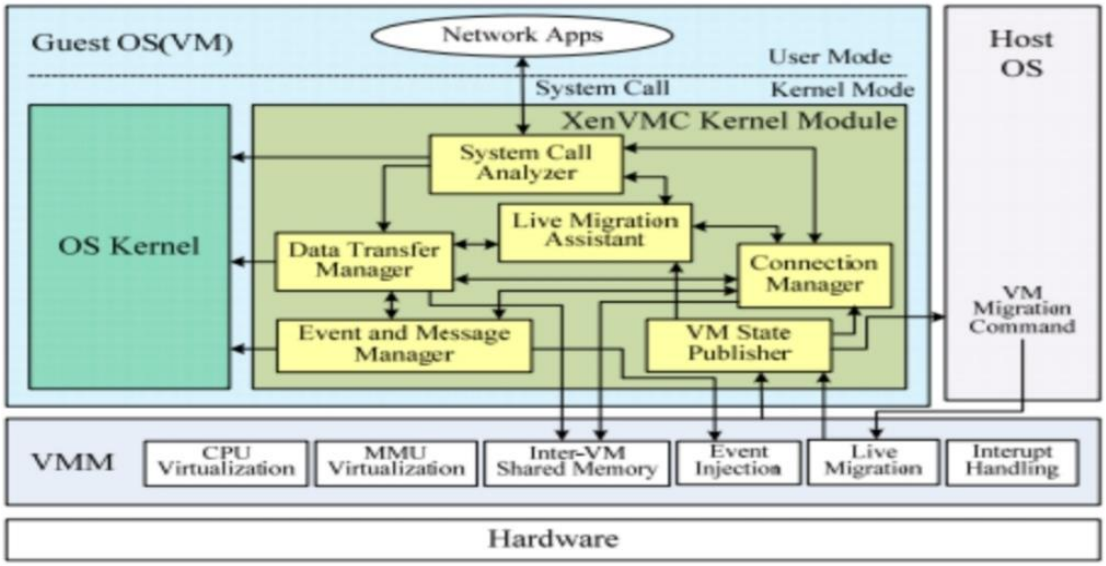

Figure 7: XENVMC kernel module

9.2 mechanism of action of the XENVMC protocol:

1-update the shared residence information of the connected virtual machines:

The protocol XENVMC uses the Matrix VMs [] for updating and automatic detection of shared virtual machines in hosting, so that each virtual machine maintains its own copy of the Matrix .VMs[] Figure (8) shows that each element in the VMS [] array represents a virtual machine that is considered a common resident on the same host uniquely identified by the pair (ip, domID), and refers to a table (Hash table)containing all connections of shared virtual machines in the hosting, which have a connection with the current virtual machine.

The connection in the XENVMC system is represented by the Register (conn-t), which consists of a set of fields, namely the local port (Iport) and the remote port (report), the two registers (sender-t) and (receiver-t). The sender-t register contains a pointer pointing to the sending virtual machine, a list of processes waiting to write data, and also contains a pointer pointing to the shared virtual memory allocated for data transfer. The structure of the record (recever_t) is similar to the structure of the previous record (sender_t) all these processes are set through the (Atomic_t) architecture to allow parallel access of a number of users [13].

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

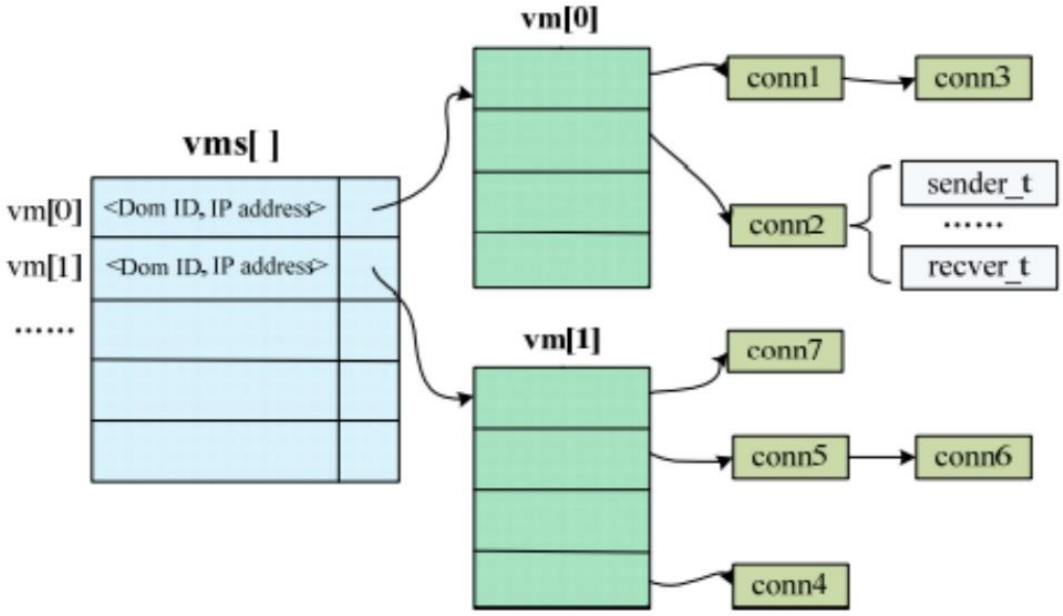

Figure 8: the connection matrix of the XENVMC protocol

The VMS [] array is updated directly when a new virtual machine is created on the same host, or when a virtual machine is migrated, to the current host, or from the current host to another host, and these updates are exchanged between all virtual machines located on the same host.

2-setting and disconnecting the connection based on the shared memory approach[14]:

When one of the guest's virtual machines detects the first network traffic to a shared virtual machine in hosting with it, the connection of the sent virtual machine is configured and set up according to the following steps:

1-the connection structure conn_t is initialized, memory pages are set as a data buffer and data is written into them.

2-the recipient initializes itself by initializing the structure recver_t and asks the sender to establish the connection, attaching a domID.

3-the sender initializes sender_t in a similar way, and then the dependent gnttab_grant_ foreign_acc provided by the Xen grant table is activated to share the allocated memory pages with the receiver. And the sender calls HYPERVISOR_event_channel_op to create the event Channel.

4-the sender's domid, evtchn_port, and a pointer are all sent to shared memory pages through the message and alert manager.

5-after receiving this information from the receiver, the receiver connects to evtchn_port, which is the sender's event port, sets shared memory pages on its address space, reads data from the shared memory, and then alerts the sender that the connection setup procedure is over. Figure (9) shows the steps for configuring the connection between the sender and the receiver.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

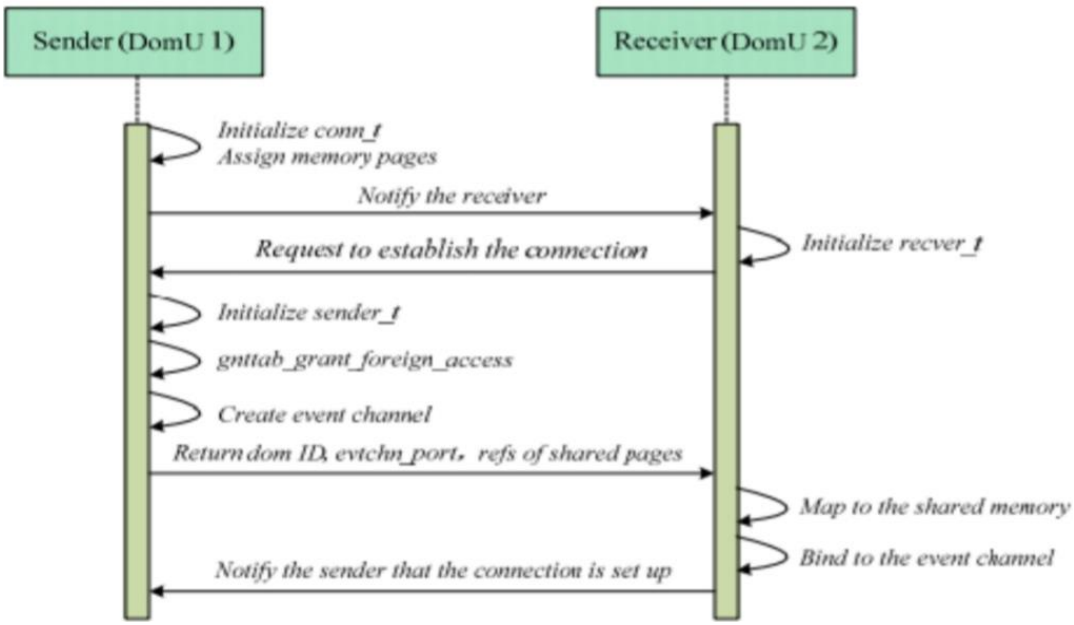

Figure 9: establishing the connection in the XENVMC protocol

When the live migration command is executed on one of the guest virtual machines, turned off, or uninstalled the XENVMC kernel copy, the connection is released according to the following steps:

1-the shared memory of all guest connections is released and demolished, and the event publisher updates the shared accommodation information by updating the VMS array.

2-stop sending data and notify the recipient to start the cancellation process.

3-stop all future actions from receiving data by locking the structure recerver_t after that, the shared memory pages are unassigned and the event channel is closed, then the recerver_t structure is released.

4-The Event channel is closed by the dispatcher, since the pages of the shared memory are released, freeing the struct sender_t structure after the connection is demolished.

5-the conn_t architecture is edited, and the VMs[] is updated on all shared hosting virtual machines.

3-data transmission and reception algorithm [15]:

Sending and receiving data between shared hosting virtual machines is organized according to a FIFO circular queue, this list is built according to the grant schedule of.Xen shared memory communication channels have been developed so that XENVMC is able to receive data synchronously with multiple cores [15].

Sending and receiving data according to the XENVMC algorithm is carried out according to the following stages:

- Sending data:

1. it is ensured that the sender and receiver are on the same host.

2-the VMS array is searched [] in the absence of a local communication channel for the sender, the data transfer manager calls the connection manager to configure the conn_t structure, and inserts it into the VMS array[].

3-if the unused space in the FIFO buffer is sufficient to send data, the sender copies the data to the BUFFERFIFO, and informs the receiver via the event channel that the data is ready to be received, otherwise the sender enters a waiting phase according to the I/O Mode lock mode, and waits until the receiver copies the data from the buffer and alerts the sender about it.

4-the structure sender_t is updated to highlight the available space in the BUFFERFIFO.

5-if the live migration command is executed on the sent virtual machine or stopped according to any of the previous stages, the connection will be immediately canceled and the virtual machine will be deleted from the hosting, the VMs array will be updated[] and the data will be sent according to the TCP/IP pattern.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

- Data reception:

1-the array is searched VMs [], in the absence of a local communication channel, conn_t is initialized and inserted into the array VMs[].

2-the receiver determines whether there is enough data ready in the buffer (FIFO)for reading. If yes, it copies the data from the buffer and notifies the sender of the receipt of the data, otherwise, if the sender enters the lock/O mode, the receiver waits for the sender's file to copy the data to the buffer before reading, and when the sender alerts the receiver that the sending process is over, the receiver copies the data from the buffer.

3-the recver_t structure to record the received data is updated.

4-if the live migration command is executed on the receiving virtual machine or stopped according to any of the previous stages, the connection will be immediately canceled and the virtual machine will be deleted from the hosting, the VMs array[]will be updated and the data will be received according to the TCP/IP pattern.

10-modification of the algorithm for sending and receiving data in the XENVMC communication protocol:

We have used the shared memory approach to transfer and exchange data between virtual machines connected and shared with the hosting, in order to improve the efficiency of live migration, by limiting the significant change in memory resulting from data exchanges between virtual machines during the migration process, as all these changes are on the shared connection memory, which is memory that is not sent to the target host during the implementation of live migration, which significantly reduces the sending of changing pages in each role of the algorithm .Pre-copy we built the proposed model, which we called Advance-XENVMC by modifying the sending and receiving algorithm in the communication protocol XENVMC where it was Modify the algorithm according to the following:

• To send data

Data is transmitted according to three phases:

- First phase: if the sender and the receiver are on the same host, the transition is made to the second phase, otherwise the data is transmitted via the TCP/IP form.

- The second phase: we have written the **Keep-con-with-migration** affiliate, which performs the following tasks:

1-at any stage of the second phase, if a callback is receivedcallback from the manager of virtual machines to migrate the sent virtual machine, the migration command is executed without disconnecting and the execution of the phases of the second phase continues.

2-the array is searchedvms [] in the absence of a local communication channel for the sender, the data transfer manager calls the connection manager to configure the conn_t structure, and inserts it into the VMS [] array.

3-if the unused space in the FIFO buffer is sufficient to send data, the sender copies the data to the BUFFERFIFO, notifies the receiver via the event channel that the data is ready for reception, otherwise the sender enters a waiting phase according to the lock/O mode mode, and waits until the receiver copies the data from the buffer and alerts the sender about it.

4-the structure sender_t is updated to highlight the available space in the FIFO buffer.

5-if an alert is received that the migration process has ended at any step of the second phase, the finish_mig child is called, Moving to the third phase.

- The third phase: we have implemented the phases of this phase by writing the dependentfinish_mig , which performs the following test:

o if the data transmission process ends, the virtual machine's connections and data are deleted from all VMs arrays [], canceling their shared membership on the same host.

o if the data transmission process is not completed, the data transfer is completed through the traditional TCP/IP communication form, and the connections and data of the virtual machine are deleted from all matrixsvms [], canceling their joint membership on the same host.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

• Second: to receive data

Data is also received in three phases:

First phase: if the sender and receiver are on the same host, the transition is made to the second phase, otherwise the data is received via the model .TCP/IP

The second phase: the subordinate is called **keep-con-with-migration,** which performs the following tasks:

1-at any stage of the second phase if a callback call is received from the Virtual Machine Manager for the migration of the future virtual machine, the migration command is executed without disconnecting and the execution of the phases of the second phase continues.

2-the VMS [] array is searched, in the absence of a local communication channel, conn_t is initialized and inserted into the VMS [] array.

3-the receiver determines whether there is enough data ready in the buffer (FIFO) for reading. If the answer is yes, it copies the data from the buffer, notifies the sender of the receipt of the data, otherwise, if the sender enters the lock/O mode, the receiver waits for the sender's file to copy the data to the buffer before reading, and when the sender

By alerting the receiver to the end of the transmission process, the receiver copies the data from the buffer.

4-the recver_t structure to register the received data is updated.

5-if an alert is received that the migration process has ended at any step of the second phase, the finish_mig child is called, Moving to the third phase.

- The third phase: the finish_mig follower performs the following test:

o if the data reception process ends, the virtual machine's connections and data are deleted from all VMs arrays [], canceling their shared membership on the same host.

o if the data reception process does not end, the data transfer is completed through the traditional TCP/IP connection form, the connections and data of the virtual machine are deleted from all matrixes VMs [], canceling their joint membership on the same host.

11. results and discussion:

We performed various scenarios so that the load is increased by increasing the number of requests and increasing the size of the sent file, and the total live migration time was calculated in each scenario if our proposed model was applied and the results were compared if the pre-copy algorithm was applied only to execute the live migration order.

1-the first test we changed the number of requests by changing the N value of the program client and installing the file size of 1 MB according to the Table (1):

Table 1: number of clients and the size of the file sent

20	16	12	8	4	**n**
400	256	144	64	16	**User**
1	1	1	1	1	**File size/mb**

- Figure (10) shows the total time of live migration, where the horizontal axis represents the number of users and the vertical axis represents the time per second. The total live migration time was calculated during the live migration application on the vm1 virtual machine from the moment the command was executed until the virtual environment was running on the target server, where these values were obtained through the Xenmotion tool.

- The results show that our proposed model has significantly improved the total live migration time, and this becomes apparent as the number of requests increases, as the optimization rate may reach 50%, due to the fact that the proposed model reduces additional changes in memory as a result of communication between the vm1

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

virtual machine and the VM2 machine during the execution of the live migration order, as these changes are limited to shared memory located on the same host, and their transfer to the target is not repeated during the migration of the virtual machine as happens if the pre-copy algorithm is used only.

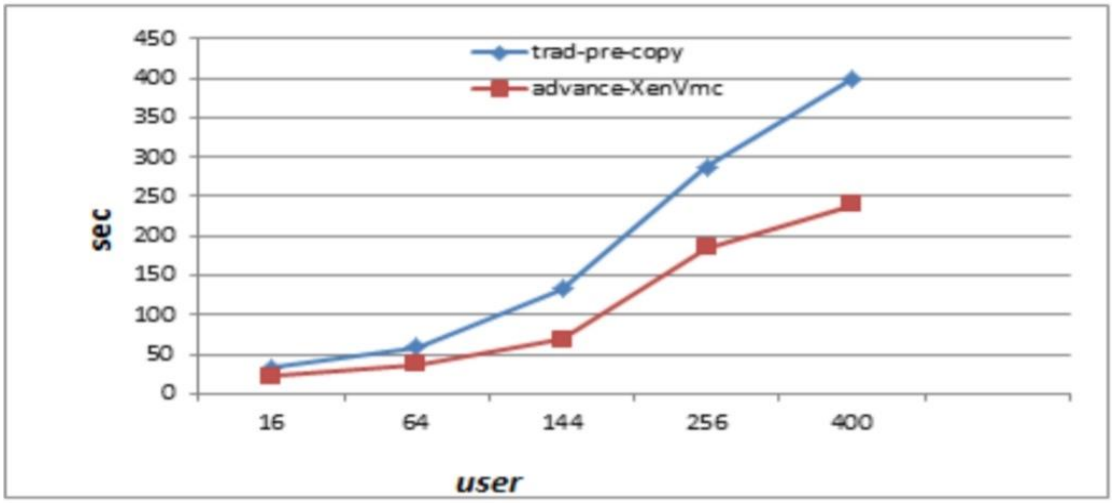

Figure 12: total time of live migration during sending a file of 3 MB size

5. Conclusion:

In our research, we presented a model based on the shared memory approach to improve and speed up the connection between connected and co-hosted virtual machines, which positively affects the improvement of the total live migration time during the migration of connected and co-hosted virtual machines for two reasons:

- First: the speed of data exchange between the sender and the receiver will lead to a speed in the execution of applications running on the virtual machine being migrated.

- Secondly: reducing large changes in memory caused by communication between the virtual machine being migrated and other machines located on the same host, which significantly reduces the sending of memory pages in each role of the pre-copying algorithm.

- The results show that the proposed model (Advance-XENVMC) has significantly improved the total live migration time, and this is evident as the number of requests increases, with the percentage of improvement reaching 50% if the load caused by the connection increases significantly.

- Improve the proposed model of network consumption during the application of live migration.

It is expected that we will get better results if the proposed model is combined with live migration algorithms that send pages that change little in the current role and pages that change significantly are postponed to the last role, and this is what we will implement in the future.

References

[1] VIRENDRA TIWARI, DR.AKHILESH A.WAOO, BALENDRA GARG, ASSISTANT PROFESSOR, ASSOCIATE PROFESSOR, ASSISTANT PROFESSOR, 2020- **Study On Virtualization Technology And Its Importance In Cloud Computing Environment**. International Journal of Creative Research Thoughts ; Volume 8, ISSN: 2320-2882.

[2] PRATEEK JAIN, 2021- **Optimized Pre-Copy Live Virtual Machine Migration for Memory-Intensive Workloads.** National College of Ireland.

[3] JANNARM AND T. KAEWKIRIYA, 2016- **Framework of Dynamic Resource Allocation System for Virtual Machine in Virtualization System**. International Journal of Computer Theory and Engineering, Vol. 8, No. 4.

[4] EZHILARASIE, RAJAPACKIYAM, ARVIND VENKATESA SUBRAMANIAN, UMAMAKESWARI ARUMUGAM, 2020- **Live Migration of Virtual Machines using Mirroring Technique**.
Journal of Computer Science; DOI**:**
https://doi.org/10.3844/jcssp.2020.543.550,Volume 16, 543- 550.

[5] MOHAMED ESAM ELSAID, HAZEM M. ABBAS ,CHRISTOPH MEINEL,2021- **Virtual machines Pre-copy live migration cost modeling and prediction: a survey.** Distributed and Parallel Databases;
https://doi.org/10.1007/s10619-021-07387-2.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023

[6] SANGEETA SHARMA, AND MEENU CHAWLA, 2016- **A three phase optimization method for pre-copy based VM live migration**. Sharma and Chawla Springer Plus 5:1022 DOI 10.1186/s40064-016-2642 2.
[7] AHMAD SAKER AHMAD, HAIDER KHALIL, 2018- **Improve the total migration time of live migration in virtualization data centers.** Tishreen University Journal, k,ISSN:2079-3081,vol.39.
[8] G. SUNITHA REKHA, 2018- **A Study On Virtualization And Virtual Machines**. International Journal of Engineering Science Invention (IJESI), Volume 7 Issue 5 Ver. III.
[9] Dr. KASSEM KABALAN ,HAIDER KHALIL,2017- **Performance evaluation of virtual machine manager by using DRBD as a shared storage of virtual disk**. ISSN: 2079- 3081,vol.39.
[10] YI REN, QI ZHANG,JIANBO GUAN, JINZHU KONG, HUADONG DAI, and LISONG SHAO, 2016 - **Shared-Memory Optimizations for Inter Virtual Machine Communication**. ACM Computing Surveys, Volume 48, No: 49.
[11] QI ZHANG, LING LIU, YI REN, KISUNG LEE, YUZHE TANG, XU ZHAO, YANG ZHOU, 2013- **Residency Aware Inter- VM Communication in Virtualized Cloud: Performance Measurement and Analysis.** IEE, DOI:
10.1109/CLOUD.2013.116.
[12] YI REN,LIU RENSHI,YOU ZIQI(ZIVE*), 2016*- **XENVMC - A Residency Aware Transparent Inter-VM Network Communication Accelerator.** XENVMC Group of NUDT.
[13] YI REN, LING LIU, XIAOJIAN LIU, JINZHU KONG, HUADONG DAI, QINGBO WU1, YUA, 2013- **A Fast and Transparent Communication Protocol for Co-Resident Virtual Machines.** IEE, ISBN:978-1-4673-2740-4.
[14] YI REN, RENSHI LIU, QI ZHANG, JIANBO GUAN, ZIQI YOU, YUSONG TAN, QINGBO WU, 2020- **An Efficient and Transparent Approach for Adaptive Intra-and Inter-Node Virtual Machine Communication in Virtualized Clouds**. IEEE,
DOI 10.1109/ICPADS47876.
[15] YOU ZI-QI, REN YI, LIU REN-SHI, GUAN JIAN-BO AND LIU LI -PENG, 2018- **Optimization of Co-resident Inter-VM Communication Accelerator XENVMC Based on Multi-core.** Computer Science, Vol. 45, Issue (3): 102 107.doi: 10.11896/j.issn.1002-137X.2018.03.017.

Doi: https://doi.org/10.54216/PMTCS.010103
Received: September 24, 2022 Revised: November 09, 2022 Accepted: January 15, 2023