International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2022/212560 Volume 9, Issue 3, May – June (2022)

RESEARCH ARTICLE

A Container Migration Technique to Minimize the Network Overhead with Reusable Memory State

Gursharan Singh
Department of Computer Science and Engineering, Lovely Professional University, Punjab, India
gursharan.16967@lpu.co.in

Parminder Singh
Department of Computer Science and Engineering, Lovely Professional University, Punjab, India
parminder.16479@lpu.co.in

Received: 05 May 2022 / Revised: 06 June 2022 / Accepted: 17 June 2022 / Published: 28 June 2022

Abstract – Cloud computing is a new computing technique for massive data centers that keeps computational resources online rather than on local machines. As cloud computing grows in popularity, so does the need for cloud resources. Container placements on physical hosts in Infrastructure-as-a-Service data centers are constantly tuned in response to the usage of host resources. When a container is migrated, a huge amount of data is transferred between hosts, and in some cases when it migrates back then the same amount of data is transmitted again. In this paper, the proposed approach for container migration to migrate back to the same host is described. Container migration enables load balancing, system maintenance, and fault tolerance, among other things. In some cases, the container will migrate back to the same host. The original image kept on the source host can be reused in such cases. The memory pages similar to the source image will not be sent back; only the updated pages will be transferred. This approach helps in reducing the amount of data transmission over the network. Furthermore, if the container image is kept on the source host, it will provide demand paging and help recover from failure at the destination host. The result shows the average rate of reduction in the data transfer over the network by 60.68% compared to standard pre-copy and 52.30% compared to advanced pre-copy.

Index Terms – Container Migration, Pre-Copy, Dump Reusing, Page Recovery, Network Overhead, Memory Prediction.

1. INTRODUCTION

The need for cloud computing has eased the dynamic deployment of computing, networking, and storage resources to provide on-demand services [1]. Traditionally, directly executing on the operating systems has been the core piece for hosting the service by employing the resources [2]. With the advancement of virtualization, one of the vital virtualization technologies used to host cloud services, virtual computers (containers) may share processing, networking, and storage resources from real machines. Due to its flexibility and tiny footprint, the container, on the other hand, is the growing virtualization instance to offer a more elastic services architecture [3]. Under various Service Level Agreements (SLAs), application providers can lease virtualized instances (containers) from cloud providers of several types. The instances are then initialized by the container or container administrators. The cloud broker or orchestrator chooses the best possible placement based on the available resources and the allocation rules [4].

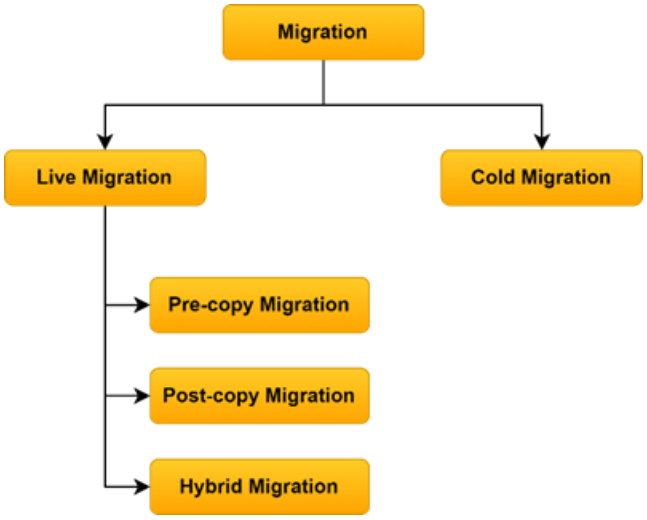

Figure 1 Categories of Container Migration

The live migration of processes, virtual machines, containers, and storage is a critical component in cloud computing for supporting dynamic resource management. It can migrate and synchronize the operating state of an instance of a container from one host to another without affecting services. [5]. Live migration provides a general solution that does not require application-specific configuration or administration.

Many studies have been conducted on resource utilization through live migration, such as fault tolerance, load balancing, system upgrade, hardware and software maintenance, etc. AWS, Azure, Google, IBM, RedHat, and

RESEARCH ARTICLE

other cloud service providers have begun to integrate live container migration.

1.1. Background

It is critical for migration management to reduce migration costs and maximize migration performance while meeting resource utilization goals. Live migration and virtualization are the leading performance factors of cloud computing [6]. Virtualization is achieved with the live migration of container instances, where the type of migration and the amount of data transfer are the two main factors [7]. The types of migration are illustrated in Figure 1 and further discussed in detail in Section 1.3.

The solutions for migration management and problems since dynamic resource management necessitates several instances of migrations to fulfill the objectives is highlighted. In addition, we examine relevant state-of-the-art works and identify future research opportunities.

1.2. Virtualization

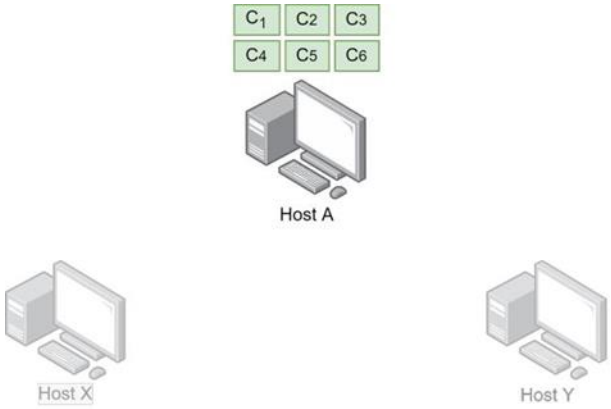

Figure 2 System Configuration and Container Placement when Source Host is Idle

The virtual machine and the container are the two industry-standard solutions for virtualization used in live migration. This part introduces the container runtimes and the memory tracking method that enables isolation and virtualization of resources. The container runtime is software that generates and maintains containers on a computing node. Apart from Docker, many others include containers, Container Runtime Interface (CRI), and low-level container runtime **(**runc). Live container migration standards are Checkpoint and Restore in Userspace (CRIU) [8]. It uses Process Trace (ptrace) to capture processes and inject parasite code that dumps the process's memory pages into the image file using its address space. Additionally, the state of the task assigned, registry entries, linked files, and the credentials are captured and preserved in the container's dump files. While a process tree is being checked, CRIU produces checkpoint files for each linked child process. CRIU uses information from the dump files created during checkpointing to restore processes on the destination host.

If a system is in the ideal situation, then it can handle the containers. When a system is overloaded then it will initiate the migration process of some instances (containers) to transfer them to another available host. This can happen in some other situations as well like fault tolerance, load balancing, system upgrade, etc.

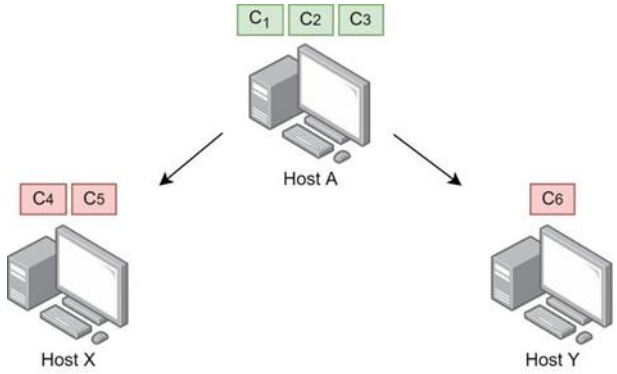

Figure 3 System Configuration and Container Placement when Source Host is Overloaded

As shown in Figure 2, "Host A" is managing all the containers from C1 to C6. But when the Host system is overloaded for may be due to any other reason, then it will start migrating the containers to other available Hosts. The scenario after the migration is shown in Figure 3.

1.3. Migration Types

Migration allows you to access the resources, processes, and containers virtually. The migration of instances and storage may be classified as "cold" and "live" migration. There are three types of live migration: pre-copy, post-copy, and hybrid migration [9]. The system design of live migration and its continual adjustment and refinement aims to reduce total migration time and the amount of data transmission. The time when the container service is not available due to synchronization requirements is counted as downtime. The migration time for a single instance is between the start of the pre-dump phase and the completion of the post-migration phase during which the instance is operating on the destination host [10]. A memory transfer of the container may be classified into three phases to do a performance trade-off analysis:

1) The push phase, during which the instance continues to execute on the source host and the related memory and data is pushed to the destination host across the network.

2) A stop-and-copy phase in which the instance is first stopped and then memory and data is transferred over the

RESEARCH ARTICLE

network to the destination. The instance will continue at the destination after the conclusion of the phase.

3) The pull phase, during which the new instance operates while retrieving faulty memory pages.

Cold Migration: In comparison to live migrations, it includes the transfer of a single copy of memory and disc or a dump file for a single container checkpoint from the source to the destination host. In other words, this category includes stop-and-copy approach. While cold migration is more straightforward than live migration. The amount of data transfer and the migration time directly depends on the nature of the task and the amount of data assigned.

Pre-copy migration: It transfers the container memory in multiple iterations. A single page is migrated many times depending on the number of iterations. Optimized pre-copy on the transfer of the modified pages in these iterations. During this process, the container is running [11]. It is classified into the following phases:

1) Initialization: The first step is to select the target host to expedite subsequent migrations.

2) Reservation: configures the shared file server (optional) and initializes an instance container on the destination host for the reserved resources.

3) Iterative pre-copy migration: It transfers the modified pages to the destination host. In the initial round, the initial memory states are duplicated.

4) Stop-and-Copy: When this process meets the threshold value in terms of the number of iterations or the amount of data transfer, that is the last iteration.

5) Commitment: The sync of the source host will get the destination host's commitment to the successfully cloned instance.

6) Activation: The new instance is allocated reserved resources.

The post-copy container migration technique suspends the instance at the source and restarts the same instance on the destination host by transferring the container execution states and remaining pages. If any page is unavailable at the destination host, the page fault occurs, and the same page will be accessed from the source host. The service will also break if the running instance fails, since the originating host does not have a running instance with its memory set. Compared to pre-copy migration, a post-copy technique can significantly reduce the migration process's time. The main reason for the decrease in performance is the regular demand for memory pages from the source copy.

To strike a balance between the three phases of live migration, hybrid post copy [12] uses a hybrid approach. Pre-copy and post-copy migrations can also be used as an optimization approach. It all starts with the pre-copy approach, which copies dirty pages repeatedly. If the memory copy iteration fails to attain a specific percentage increase over the previous iteration, the post-copy migration will be activated. The migration time will be reduced in some cases, but the downtime will be somewhat higher. There are certain downsides to post-copy migration, such as slower processing speeds and the possibility of container reboots if the network is unstable when extracting faulty pages.

This paper is organized as follows: Section 2 describes the literature review and the research gap. The problem identification, along with the objective of the study, is discussed in Section 3. The tools and techniques used in the proposed system and the memory reusing model are discussed in Section 4. The experimental setup is discussed in Section 5. In Section 6, the evaluation of the system model is elaborated in detail and concluded in Section 7.

2. LITERATURE REVIEW

Virtualization in cloud computing has dramatically improved due to the development of containers. When compared to virtual machines, container migration is much more efficient. Cloud services are now migrating to a container environment, necessitating new research to improve this technique.

Karhula et.al. have examined the notion of function as a service for IoT edge devices. The checkpointing method to conserve resources on resource-constrained devices to halt long-running blocking functions is used [13]. Additionally, the live container migration using CRIU is demonstrated. The assessment demonstrates good results for IoT device checkpoints capability. In IoT systems, these building pieces can be used for the fault-tolerant, offload resources, and boost efficiency and availability at the IoT edge. For the purpose of freezing a running container, Jiaxin Feng et. al.[14] Describe the CRIU. Compared to the re-deployment technique, the restored container takes stateful migration to retain its state after being frozen. The CSS approach selects and migrates the container with a capacity of 512MB. The container migration time is 48 seconds, significantly faster than the time required for standard minute-level wireless reconfiguration.

Load balancing is presented using the migration method based on the two options. Compared to the conventional wireless reconfiguration technique, the migration mechanism may preserve the DU/state CU's while requiring minimal service interruption. It is to handle memory properly. Janaina Schwarzrock et al. highlighted the concept of virtual memory, like page faults and TLB, which increases control switching between hosts and affects the overall performance. The transition overhead establishes a reliance between various local setups and should be considered in online techniques [15].

RESEARCH ARTICLE

Ranjan Sarpangala et al. described the development and operation of VAS CRIU, a novel technique for reducing memory checkpoint and restore time utilizing task address space [16]. After their first launch phase, applications may be snapshotted into a VAS and then immediately restored into fresh container instances. Additionally, the snapshot might contain information about popular pages, which could be used for page pre-copying. A system that monitors containers' performance and the hosts on which they run. Analyzing, describing, and developing forecast models can benefit from this data. Microservices and NFVs are two use cases that are considered. They have fine-tuned resource provisioning techniques by analyzing data from the monitoring system to establish a cloud provisioning platform that enhances container workload utilization and implementation through live migration. All the details of our lightweight resource monitoring tool, which allows for the offline and real-time examination of active migration workloads, along with the impact on their hosts, are described [17].

Florian Hofer et al. propose a migration architecture and demonstrate that containerized apps may run on shared resources without jeopardizing planned execution within specified time restrictions via a custom-built orchestration tool [18]. They investigate the boundaries of three system configurations using latency and computational performance studies and write a summary. Using Layrub, a data placement approach for GPU-accelerated deep learning. DNN models of all shapes and sizes may be trained using Layrub's extraordinary memory optimization. Experimentation has shown that Layrub can reuse a consistent amount of memory space no matter how deep the network is. The authors further highlight the advantages of Layrub by comparing it favorably with GeePS, vDNN, MXNet, and TensorFlow on several DNN models and datasets. Using Layrub might help you keep your memory as efficient as possible [19].

Evangelos Vasilakis et al. provide a novel data migration technique for hybrid memory systems that accounts for the overheads mentioned above and significantly increases migration efficiency and effectiveness [20]. It is based on discovering that migrating memory segments stored in the last level cache reduce migration load. Their solution is based on the current status of the last level cache to forecast reuse and prioritize memory segments for transfer. Thus, when segments are present in the last level cache, they are transferred at a lower cost. The results demonstrate that our technique beats existing state-of-the-art migration designs by 12.1% in terms of system performance and 13.2% in-memory system dynamic energy reduction.

Mathematical modeling, heuristic, machine learning, and meta-heuristic are the four basic types of container scheduling algorithms. Machine Learning is the ideal solution for anticipating workloads and performance indicators because of its great capacity to validate the system to anticipate outputs based on prior data and training. In complex work contexts, such a view helps schedulers with better resource allocation while dealing with shifting user request rates [21].

Gundall et al. provides a unique paradigm is offered that relies on both existing migration methodologies and virtualization technologies, with the primary goal of reducing service downtime. A test set is also used to examine the notion. The results suggest that the proposed strategy can achieve a reduced downtime. Furthermore, the overall migration time for the maximum performance option is in milliseconds [22].

Terneborg et al. expand container migration with a proposed method that supports fail-over and live migration, which means it might be incorporated into existing container tools. Furthermore, evaluation results are supplied, which may be used to compare to an existing migration approach [23]. They have also discussed the current migration methodologies and metrics for assessing different migration approaches. They have accomplished a lower total migration time and downtime similar as of pre-copy migration [24].

Zhi et al. intend to save cost on resources by using as few machine resources as feasible by using a suitable dynamic container migration capability, therefore cluster-scale layout of container has been the topic of this paper. A method is presented to decrease fragmentation, hence improving machine resource efficiency and achieving the cost-cutting aim. Experiments indicate that the method efficiently prevents fragmentation and reduces resource consumption in container layouts on a wide scale [25].

Zheng et al. offer a scheduling technique for a two-level approach for container real-time resources. To decide on container migration, LTSM is utilized to estimate resource use and select the environment. In addition, for simulation trials, they used CloudSim, an open-source program. The results demonstrate that the method may increase the global resource utilization of containers while lowering data center energy usage [26].

Yang et al. An online prediction approach called user trajectories is given to address the challenge in prediction accuracy. A scheduling algorithm is designed to identify servers based on user movement speeds and latency to reduce duplicate network traffic. The results of our tests indicate that the proposed prediction methods outperform the usual technique. It reduces network traffic by 65\% while meeting task delay standards. Furthermore, it adapts to changes in the user's journey speed and surroundings to ensure service stability [27].

Chen et al. In the container migration, the PSO is used for hyper-parameter adjustment in order to enhance the model's prediction performance. The findings of the experiments suggest that autonomous hyper-parameter adjustment can

RESEARCH ARTICLE

increase prediction accuracy. Meanwhile, in MSE, R2, and MAE, the prediction performance is better than the previous system without managing hyper-parameters by 19.3\%, 4\%, and 11.7\%, respectively, compared to the existing system without managing hyper-parameters. Furthermore, the PSL beats other algorithms like as RNN, GRU, and LSTM in terms of prediction performance [28].

Dai et al. \cite{dai2021deep} predicting failure before it occurs is critical for making the cloud service more effective. The ability to forecast defective nodes allows service to be migrated to healthy nodes, increasing service availability. To successfully handle this problem, proactive fault prediction approaches to forecast future failures can be employed. In this research, using time series data to forecast the failure in a cluster using the bidirectional LSTM model [29].

Prediction of memory changes is the core component of pre-copy container migration. It should be chosen wisely according to pre-copy migration's pre-dump, iterative, and final dump phases. A prediction scheme or set of multiple methods should be applied to the memory pages to be migrated to the destination host to improve the prediction mechanism. Some of the popular schemes available are Particle Swarm Optimization (PSO) based prediction schemes, Long Short-Term Memory (LSTM) based schemes, and Artificial Neural Network (ANN) based schemes.

3. PROBLEM FORMULATION

With the increasing popularity of containers in cloud computing the data transmission also increases over the network, which leads to high network traffic. Although the container helps to minimize the migration size compared to VMs, it can still be reduced further. The primary disadvantage of existing live migration approaches is that they need extensive data transfer to relocate a container. Transferring a huge volume of data introduces two complications:

1) The migration process results in memory accesses that decrease the performance of containerized apps.

2) Consolidating several containers onto a single host simultaneously congests the host’s network and slows the consolidation.

A memory reuse approach to minimize the quantity of data exchanged during live migration is suggested. A container may migrate back to the host on which it was previously operated. When the container migrates away from the host, the memory image is retained on the host, and the image is reused when the container migrates back to the host later. The reduced data volume results in a faster migration time and enhanced optimization via container placement algorithms. The container migration technique used in this method is pre-copy, and in the case of the pre-copy migration technique, the process is divided into three phases. These phases are called pre-dump, iterative dump, and final dump. As mentioned earlier, this method comes after all three of these three phases. The memory is reused in the event of container migration to reduce the data transmission over the network.

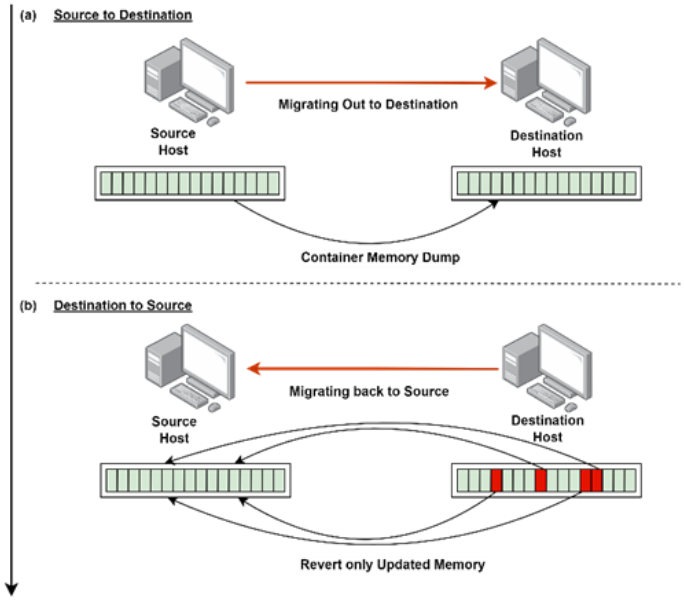

Figure 4: (a) The Process of Transferring the Complete Set of Memory Pages from Source to Destination Host During Container Migration and (b) Shows the Process of Transferring Highlighted Pages Back to the Source Host

As you can see in Figure 4(a), it shows the migration from source to destination, where the first three phases of pre-copy are applicable. In the case of standard pre-copy migration, all the memory pages or the related configuration are migrated to the destination host. Now we are going to extend this migration process further.

Suppose a container is migrated to another host due to any of the following reasons: fault tolerance, system update, load balancing, or any other reason. In that case, there is a chance that the container will come back to the same host once the purpose of migration is achieved. In the case of common techniques, while migrating back to the source host, the same migration process of copying memory pages from the destination host back to the source host is followed.

In the proposed technique, when it is decided to migrate back to the same host, it will not send back all the memory pages that are currently on the destination host. As you can see in Figure 4(b), the pages indicated in red are those that have been modified by the destination host. Only the modified pages will be transmitted to the source host when migrating back. The proposed methodology is explained in Figure 5. If the request to migrate the container is initiated, then it will send back only the modified pages.

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2022/212560 Volume 9, Issue 3, May – June (2022)

RESEARCH ARTICLE

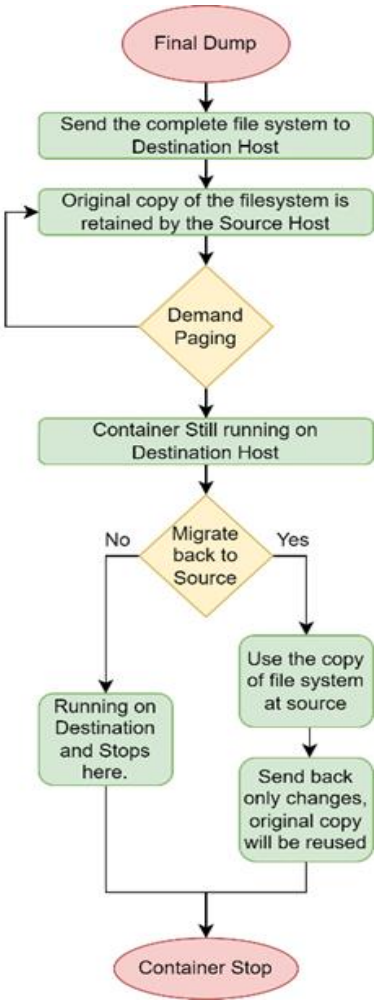

Figure 5 The Detailed Process of Proposed Methodology

4. SYSTEM MODEL

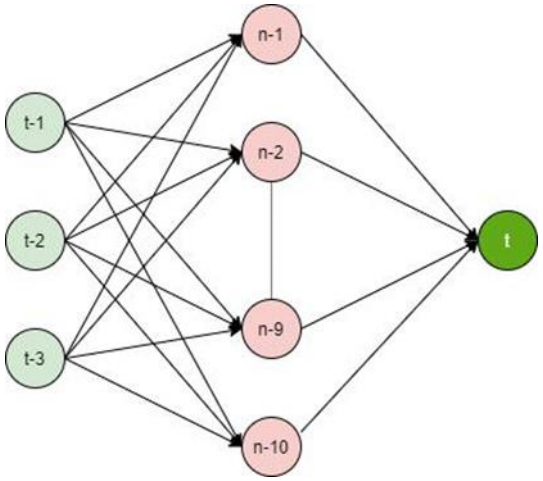

Figure 6 The ANN Prediction Model Architecture used in Proposed System Model

The proposed migration model works in two different phases. Phase 1 is the container migration from source to destination using the predictive pre-copy approach. LSTM is used to predict the set of pages to be migrated in the iterative dump in this approach. Memory migration is predicted using the LSTM in a network model. An ANN is used to create the prediction model's architecture. This model utilizes three input layers that interact with a hidden layer of ten cells (LSTM cells) to produce a single output layer as shown in Figure 6. Where t1, t2, and t3 represent the input layers, n1, n2...n3 denotes the LSTM cells and t in the final single layer output. Each LSTM cell used in Figure 6 as n1, n2...n3 is represented as shown in Figure 7. This is a representation of each cell used in the ANN network.

The LSTM module comprises three gates: the forget gate, input gate, and output gate. The quantity of data that can pass via these gates is limited. A sigmoid function and an operation are used in each gate. The dotted line handles the data generated by these gates. The union of h_{t-1} and x_t determines the value of the Sigmoid function, which is between 0 and 1. The output is obtained by multiplying the sigmoid result by the gate's input. If the sigmoid result is 1, for example, the gate output will be the same as the input since it is multiplied by one. The unit processes the input data for each input vector to the LSTM network as follows:

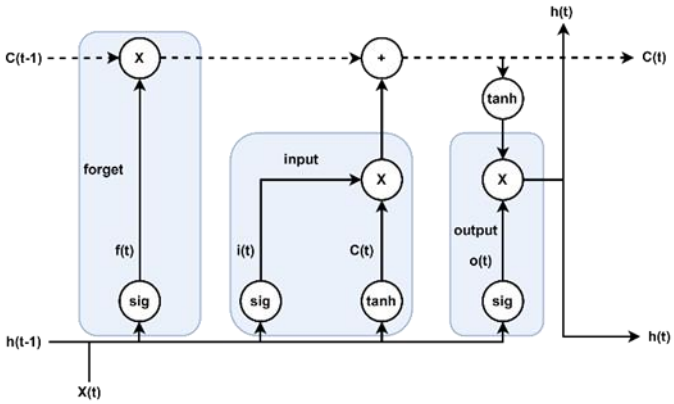

Figure 7 LSTM Cell Architecture

1) A new vector will be created by adding the h_{t-1} (hidden state vector) and x_t (input vector). The newly created vector will used as input to tanh function and to the three gates.

2) The flow of previously stored cell states regulated by the forget gate:

$$f_t = sig(W_f * [h_{t-1}, x_t] + b_f)$$

3) The C_t candidate value for the present cell state is calculated as follows:

$$C_t = tahn(W_c * [h_{t-1}, x_t] + b_c))$$

RESEARCH ARTICLE

4) The amount of C_t to be added to current cell is fixed by input gate and then C_t multiplies with i_t.

$i_t = sig(W_i * [h_{t-1}, x_t] + b_i))$

5) The final calculated C_t is as:

$C_t = f_t * C_{t-1} + i_t * C_t$

6) The output gate determines how much C_t is passed to the next cell. The hidden state h_t is calculated as follows:

$O_t = sig(W_O * [h_{t-1}, x(t)] + b_0))$

$h_t = O_t * tanh(C_t)$

Phase 2 of the proposed technique is to migrate back to the same source host. The migration process is different in this case. Instead of transferring all the memory pages related to a container, this technique will send only the modified pages. This reusing technique reduces the data transfer with a huge difference. If a particular memory page is not modified on the destination host, then that page will not participate in migration back to the same source.

5. EXPERIMENTAL SETUP

Using the time series of the prior three observations, this network design can anticipate the next batch of pages that will be transferred. There are two possible circumstances in which this model can be used. The first approach is called single-step prediction, and it makes a single forecast based on several time-series inputs. Direct and recursive prediction are the other two methods of multi-step prediction.

Table 1 LSTM Model Configuration Parameters

Parameters	Range
Input size	3
Output size	1
Number of LSTM Layers	1
Number of LSTM Units	1
Kernel initializer	Lecun uniform
Loss function	MSE
Optimizer	adam
Batch size	64
Number of epochs	10

The mean square error is the loss function used to train LSTM and ANN prediction models. This prediction model is trained using a multi-step direct technique to forecast the number of active pages moved in the subsequent round. The model was trained using a Tensor Processing Unit (TPU) supplied by colab. The configuration parameters are listed in Table 1.

Result: Set of Modified Pages

Parameters: d_b, P_{id}, r, R_{max}

while $r \leq R_{max}$ do

  while (mem_{pool}) do

    if P_{id}.db TRUE then

    M_{pool}.append(P_{id})

    end

  end

  r+ = 1

end

return (M_{pool} [P_{id}])

Algorithm 1 To Identify Modified Pages in Pre-Copy Container Migration Process

Algorithm 1 will return the set of modified memory pages. The parameters used in this Algorithm 1 are as: P_{id} is representing pages of memory pool with page id, R_{max} is the maximum number of iterations and db is a Boolean data member to store dirty bit status, r is set to 1 and it is used for rounds/iterations. According to the selected memory pool 1 $\leq r \leq R_{max}$ and R_{max} = 10.

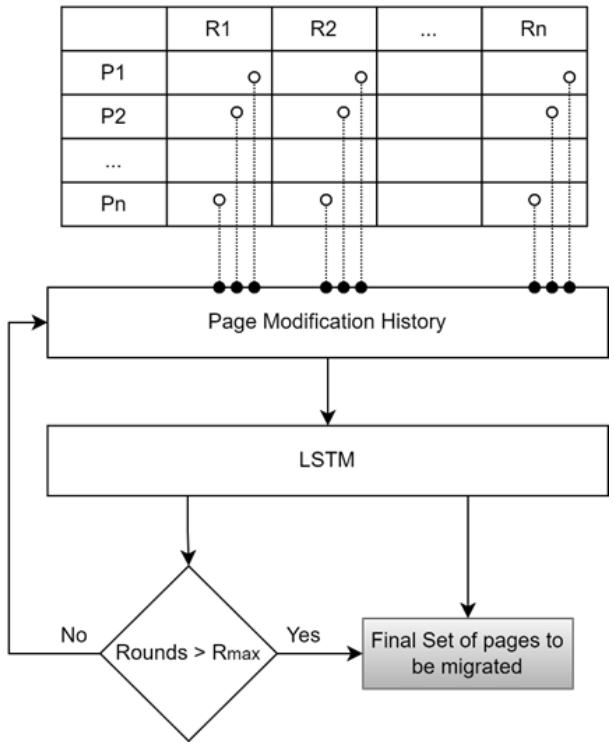

Figure 8 System Architecture to Identify Updated Pages using LSTM

RESEARCH ARTICLE

When a container is migrated from any source host to destination, then all its memory along with its system configuration is sent to the destination host. Once it is acknowledged by the receiving host, then the memory dump is removed from the source host as shown in Figure 9(a). Everything related to that container is available on destination host only. In this standard pre-copy technique, there is no option to recover the pages from source host.

The LSTM is integrated with the proposed prediction system. As shown in Figure 8, the array represents the memory pages in each round. Where P_1,P_2,P_n represents the memory pages and R_1,R_2,R_n denote the number of iterations.

The updated status of each page will be stored in "Page Modification History". This will be provided as an input to the LSTM module. In this module, the cells mentioned in the ANN network are used to predict the updated pages. This process will be repeated up to the maximum number of iterations. This module will produce the final set of memory pages to be migrated to the destination host. We proposed a technique to provide page recovery and reuse the memory while migrating back to the same host. The LSTM is used to migrate from source to destination only. When it is decided to migrate back to the same host, we will migrate back only the modified pages The rest of the pages will be recovered from the copy of dump at host as shown in Figure 9(b). The set of pages identical to the pages at source host, will be discarded. This complete scenario is implemented with container CloudSim 4.0 and LSTM algorithm used to predict memory pages to be migrated.

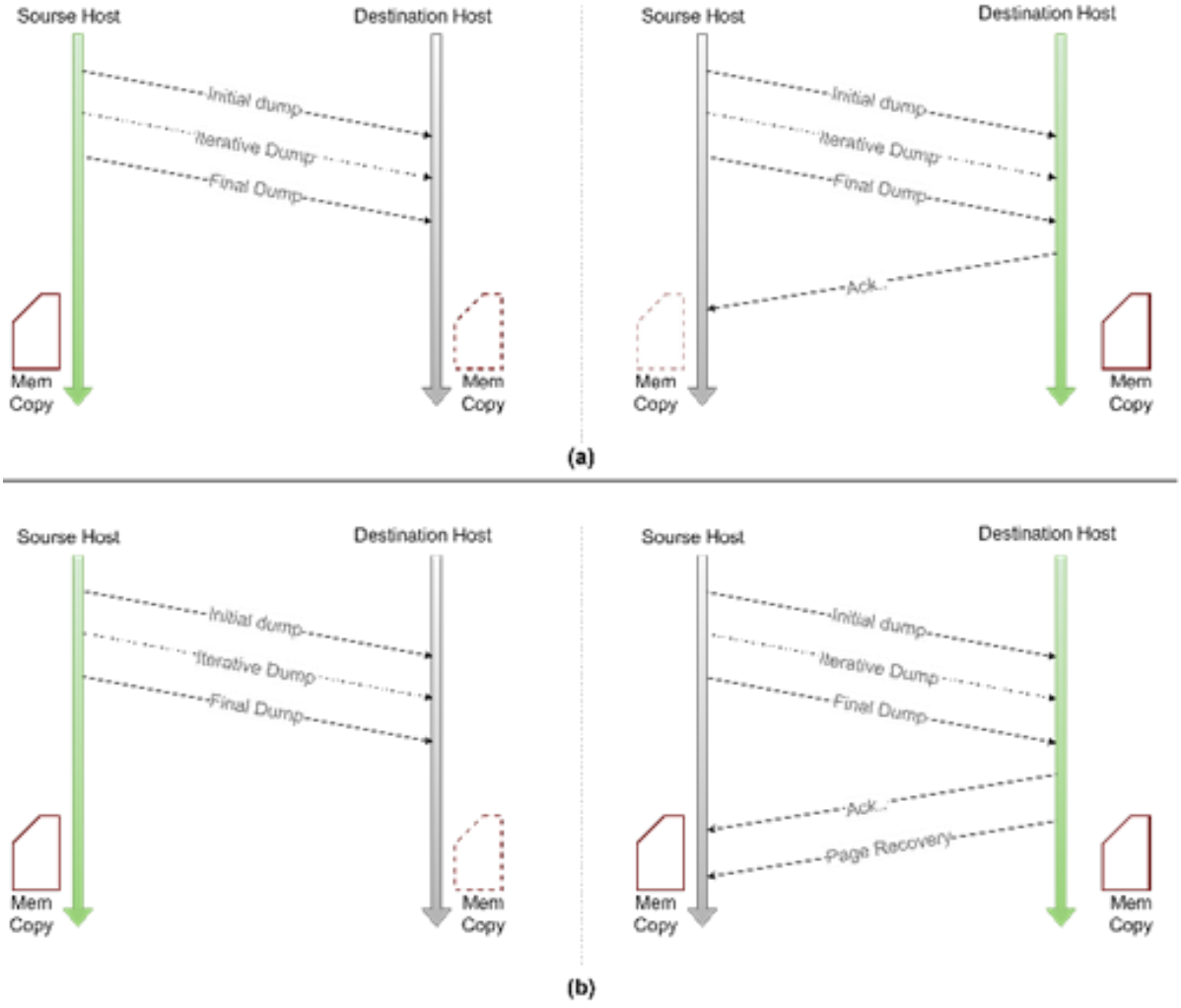

Figure 9 The Process of Transferring the Container Memory where a). Depicts the Existing Approach, in which the Container's Memory is Deleted from the Source Host when it is Migrated to the Destination Host and b). is a Proposed Approach for Providing Page Recovery by Keeping Memory on the Source Host Following a Successful Migration.

International Journal of Computer Networks and Applications (IJCNA)
DOI: 10.22247/ijcna/2022/212560 Volume 9, Issue 3, May – June (2022)

RESEARCH ARTICLE

6. PERFORMANCE AND EVALUATION

Migration performance and cost modelling is an essential component of migration management to assess and forecast migration requests' overall cost and performance. Single migration performance indicators have been the subject of several studies. We categorize these indicators according to time and data quantity. Data transmission size is the critical element for determining the network overhead associated with network migration. It is substantially positively associated with migration time for pre-copy migration. The overall quantity of data transmitted equals the sum of the amounts transmitted on each occasion. It consists of two components: memory data and storage data.

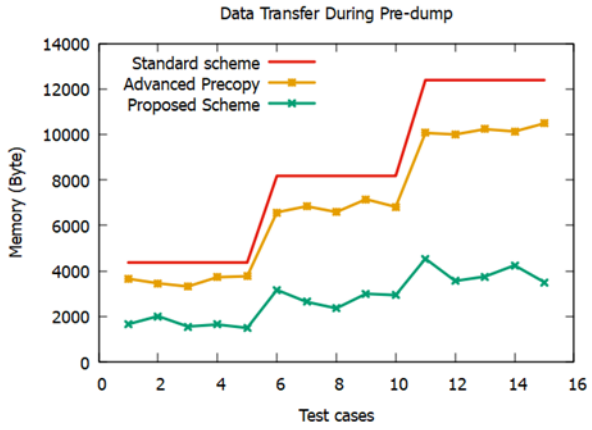

Figure 10 The Amount of Data Transferred when Migrating Back to the Same Host with 15 Test Cases. Where Test Cases 1 to 5 were Implemented with 5 Containers, 6 to 10 with 10 Containers, and 11 to 15 with 15 Containers by using the Standard Pre-Copy, Advanced Pre-Copy, and the Proposed Technique which Revert only Updated Pages

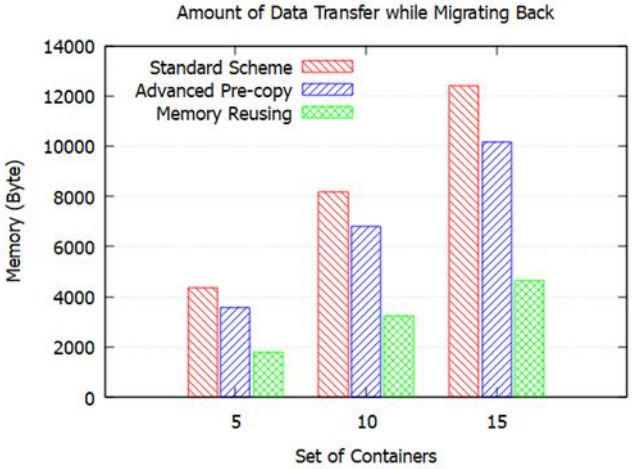

Figure 11 The Amount of Data Transferred with the Batch of 5 Containers, 10 Containers, and 15 Containers by using the Standard Pre-Copy, Advanced Pre-Copy, and the Proposed Technique

For better understanding and outcomes, we have tested the proposed memory reusing approach on different containers. There are 15 test cases, wherein the first 5 test cases, the set of 5 containers, are implemented. As mentioned in Figure 10, the number of bytes transferred during the migration of containers is specified. The memory transfer with the existing approach [30] is represented in red. And according to the proposed approach, when it is migrating back to the same host, we are reverting only the updated pages. You can see the difference in data transfer in the proposed technique as represented in the red-green color. The yellow color represents the migration from source to destination with the proposed scheme of our previous research.

We have stored the memory dump copies of the containers migrated to the destination host. Because in some cases, when containers are migrated due to load balancing, fault tolerance, system upgrade, etc., the container migrates back to the same host. As we mentioned in the process of memory reusing in Section 3, when the container is migrated back to the same host, we send only the modified memory pages. These copies of the memory dump at the source will be used to initiate the containers on the source host. In the proposed technique, there is an additional space overhead on the source host. This additional space overhead will occupy the memory of the host system only. But with this small overhead, we can reduce the costly data transmission over the network. The result shows, the data transfer during migrating back is reduced.

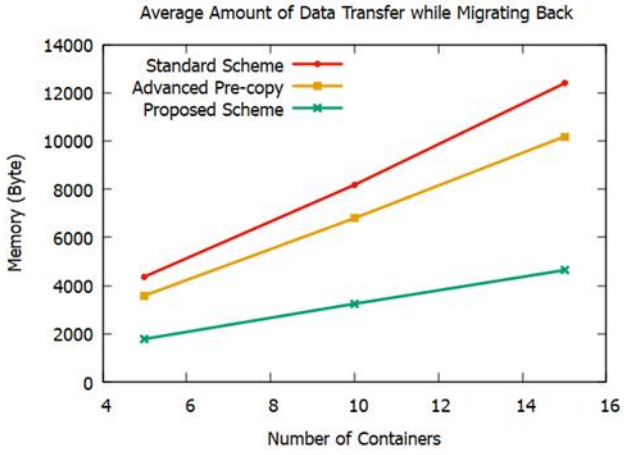

Figure12 The Average Rate of Data Transferred when Migrating Back to the Same Host with the Batch Size of 5 Containers, 10 Containers, and 15 Containers by using the Standard Pre-Copy, Advanced Pre-Copy, and the Proposed Technique

In the same way, we have run containers in three batches of 5, 10 and 15 containers. Results shows that, if the number of container increases in a batch of migration, then the percentage of data transmission over the network is improved.

RESEARCH ARTICLE

Further in Figure 11, the amount of data transferred during existing approach of container migration and the proposed technique are shown. Here we can identify the difference in the transmission.

The rate of data transfer in the standard pre-copy approach, advanced pre-copy, and the proposed pre-copy approach are discussed in detail. The average rate of data transferred when migrating back to the same host with a batch size of 5 containers, 10 containers, and 15 containers is illustrated in Figure 12.

With the proposed reusing mechanism of container migrations, the memory state stored at the source host is utilized while migrating back and helps in reducing the data transmission over the network. The number of bytes transferred in the proposed technique is much lower than the existing techniques. It shows the average rate of reduction in the data transfer over the network by 60.68% compared to standard pre-copy and 52.30% compared to advanced pre-copy.

7. CONCLUSION

The migration process is divided into three phases in the pre-copy container migration technique. The first two phases are pre-dump and iterative dump. The pre-dump was implemented with the PSO algorithm and the iterative dump is implemented with LSTM. After that in the final dump, the memory related to the container will be removed from the source host. We recognized a few cases (fault tolerance, system upgrade, load balancing, etc.), where containers go back to the same host. In such cases, we have implemented the proposed migration technique that helps to reduce the data transfer over the network and it outperforms compared to the existing system. As a future direction, this technique can be implemented in various cloud environments like VMs used for various services, fog, edge computing, etc., where the instances are moving rapidly. The other alternative is to use the centralized instance image to reduce data transfer between source and destination host. In such cases, we have implemented the proposed migration technique that helps to reduce the data transfer over the network. This can be further enhanced by ensuring whether the container will be migrated back or not. Accordingly, the approach to migration will be decided.

REFERENCES

[1] Michael Armbrust, Armando Fox, Rean Griffith, Anthony D Joseph, Randy Katz, Andy Konwinski, Gunho Lee, David Patter-son, Ariel Rabkin, Ion Stoica, et al. A view of cloud computing. Communications of the ACM, 53(4):50–58, 2010.

[2] Sean Marston, Zhi Li, Subhajyoti Bandyopadhyay, Juheng Zhang, and Anand Ghalsasi. Cloud computing—the business perspective. Decision support systems, 51(1):176–189, 2011.

[3] Dirk Merkel et al. Docker: lightweight linux containers for con-sistent development and deployment. Linux journal, 2014(239):2, 2014.

[4] Ann Mary Joy. Performance comparison between linux containers and virtual machines. In 2015 International Conference on Advances in Computer Engineering and Applications, pages 342–346. IEEE, 2015.

[5] Ying Mao, Yuqi Fu, Suwen Gu, Sudip Vhaduri, Long Cheng, and Qingzhi Liu. Resource management schemes for cloud-native platforms with computing containers of docker and kubernetes. arXiv preprint arXiv:2010.10350, 2020.

[6] Gursharan Singh, and Parminder Singh. "A Taxonomy and Survey on Container Migration Techniques in Cloud Computing." In Sustainable Development Through Engineering Innovations, pp. 419-429. Springer, Singapore, 2021.

[7] Keerthana Govindaraj and Alexander Artemenko. Container live migration for latency critical industrial applications on edge computing. In 2018 IEEE 23rd International Conference on Emerging Technologies and Factory Automation (ETFA), volume 1, pages 83–90. IEEE, 2018.

[8] Gursharan Singh, Parminder Singh, Mustapha Hedabou, Mehedi Masud, and Sultan S. Alshamrani. "A Predictive Checkpoint Technique for Iterative Phase of Container Migration." Sustainability 14, no. 11: 6538, 2022.

[9] Alessandro Ferreira Leite, Azzedine Boukerche, Alba Cristina Magalhaes Alves de Melo, Christine Eisenbeis, Claude Tadonki, and Célia Ghedini Ralha. Power-aware server consolidation for federated clouds. Concurrency and Computation: Practice and Expe-rience, 28(12):3427–3444, 2016.

[10] Radostin Stoyanov and Martin J Kollingbaum. Efficient live migration of linux containers. In International Conference on High Performance Computing, pages 184–193. Springer, 2018.

[11] Carlo Puliafito, Carlo Vallati, Enzo Mingozzi, Giovanni Merlino, Francesco Longo, and Antonio Puliafito. Container migration in the fog: A performance evaluation. Sensors, 19(7):1488, 2019.

[12] TianZhang He, Adel N Toosi, and Rajkumar Buyya. Performance evaluation of live virtual machine migration in sdn-enabled cloud data centers. Journal of Parallel and Distributed Computing, 131:55– 68, 2019.

[13] Pekka Karhula, Jan Janak, and Henning Schulzrinne. Checkpoint-ing and migration of iot edge functions. In Proceedings of the 2nd International Workshop on Edge Systems, Analytics and Networking, pages 60–65, 2019.

[14] Jiaxin Feng, Jiawei Zhang, Yuming Xiao, and Yuefeng Ji. Demon-stration of containerized vdu/vcu migration in wdm metro optical networks. In 2020 Optical Fiber Communications Conference and Exhibition (OFC), pages 1–3. IEEE, 2020.

[15] Janaina Schwarzrock, Michael Guilherme Jordan, Guilherme Ko-rol, Charles C de Oliveira, Arthur F Lorenzon, Mateus Beck Rutzig, and Antonio Carlos S Beck. Dynamic concurrency throt-tling on numa systems and data migration impacts. Design Automation for Embedded Systems, 25(2):135–160, 2021.

[16] Ranjan Sarpangala Venkatesh, Till Smejkal, Dejan S Milojicic, and Ada Gavrilovska. Fast in-memory criu for docker containers. In Proceedings of the International Symposium on Memory Systems, pages 53–65, 2019.

[17] Alejandro E González and Emmanuel Arzuaga. Herdmonitor: Monitoring live migrating containers in cloud environments. In 2020 IEEE International Conference on Big Data (Big Data), pages 2180–2189. IEEE, 2020.

[18] Florian Hofer, Martin Sehr, Alberto Sangiovanni-Vincentelli, and Barbara Russo. Industrial control via application containers: Maintaining determinism in iaas. Systems Engineering, 24(5):352– 368, 2021.

[19] Hai Jin, Bo Liu, Wenbin Jiang, Yang Ma, Xuanhua Shi, Bingsheng He, and Shaofeng Zhao. Layer-centric memory reuse and data migration for extreme-scale deep learning on many-core archi-tectures. ACM Transactions on Architecture and Code Optimization (TACO), 15(3):1–26, 2018.

[20] Evangelos Vasilakis, Vassilis Papaefstathiou, Pedro Trancoso, and Ioannis Sourdis. Llc-guided data migration in hybrid memory sys-tems.

RESEARCH ARTICLE

In 2019 IEEE International Parallel and Distributed Processing Symposium (IPDPS), pages 932–942. IEEE, 2019.
[21] Moussa, Walid, Mona Nashaat, Walaa Saber, and Rawya Rizk. "Comprehensive Study on Machine Learning-Based Container Scheduling in Cloud." In International Conference on Advanced Machine Learning Technologies and Applications, pp. 581-592. Springer, Cham, 2022.
[22] Gundall, Michael, Julius Stegmann, Mike Reichardt, and Hans D. Schotten. "Downtime Optimized Live Migration of Industrial Real-Time Control Services." arXiv preprint arXiv:2203.12935 (2022).
[23] Terneborg, Martin. "Enabling container failover by extending current container migration techniques." (2021).
[24] Terneborg, Martin, Johan Karlsson Rönnberg, and Olov Schelén. "Application Agnostic Container Migration and Failover." In 2021 IEEE 46th Conference on Local Computer Networks (LCN), pp. 565-572. IEEE, 2021.
[25] Zhi, Zhang, Zhao Zhuofeng, and Li Han. "Static layout and dynamic migration method of a large-scale container." In 2021 IEEE 5th Advanced Information Technology, Electronic and Automation Control Conference (IAEAC), vol. 5, pp. 1897-1901. IEEE, 2021.
[26] Zheng, Siyuan, Fenfen Huang, Chen Li, and Haobin Wang. "A Cloud Resource Prediction and Migration Method for Container Scheduling." In 2021 IEEE Conference on Telecommunications, Optics and Computer Science (TOCS), pp. 76-80. IEEE, 2021.
[27] Yang, Run, Hui He, and Weizhe Zhang. "Multitier Service Migration Framework Based on Mobility Prediction in Mobile Edge Computing." Wireless Communications and Mobile Computing 2021 (2021).
[28] Chen, Lei, and Weiwen Zhang. "A deep learning-based approach with PSO for workload prediction of containers in the cloud." In 2021 13th International Conference on Advanced Infocomm Technology (ICAIT), pp. 204-208. IEEE, 2021.
[29] Dai Vu, Dinh, Xuan Tuong Vu, and Younghan Kim. "Deep Learning-based fault prediction in cloud system." In 2021 International Conference on Information and Communication Technology Convergence (ICTC), pp. 1826-1829. IEEE, 2021.
[30] Aditya Bhardwaj and C Rama Krishna. A container-based technique to improve virtual machine migration in cloud computing. IETE Journal of Research, pages 1–16, 2019.

Authors

Gursharan Singh is currently pursuing a Ph.D. degree in Computer Science and Engineering at Lovely Professional University, Punjab, India. He is working as an Assistant Professor in the School of Computer Science and Engineering at Lovely Professional University. His research interest includes virtualization, cloud computing, container migration, and network security.

Parminder Singh is a Postdoctoral Researcher at Mohammed VI Polytechnic University (UM6P), Ben Guerir, Morocco. He has been working as an Associate Professor in the school of computer science and Engineering, Lovely Professional University, India. He has completed his B.Tech. from Punjabi University, Patiala, and M.Tech. from Punjab Technical University, Jalandhar. He has done his Ph.D. from Lovely Professional University in 2019. He has published more than 50 papers in reputed journals, conferences, and book chapters including IEEE transactions. His research interests include cyber-security, machine learning, and Cloud/Fog/Edge computing. He has been a session chair and technical program committee member in various national and international conferences. He has won a 'Best Smart City Project' award from the Government of India in 2019. He has been awarded Research Excellence awards in the A+ category from LPU in 2019, 2020, and 2021. He has also been awarded a Teaching Excellence award in 2021 from LPU. Currently, 6 students are pursuing Ph.D. under his supervision. 15 Master's dissertations and 1 Ph.D. thesis have been completed under his supervision. He is an active member of IEEE and ACM.

How to cite this article:

Gursharan Singh, Parminder Singh, "A Container Migration Technique to Minimize the Network Overhead with Reusable Memory State", International Journal of Computer Networks and Applications (IJCNA), 9(3), PP: 350-360, 2022, DOI: 10.22247/ijcna/2022/212560.