


REVIEW

Open Access



# A critical survey of live virtual machine migration techniques

Anita Choudhary<sup>1</sup>, Mahesh Chandra Govil<sup>2</sup>, Girdhari Singh<sup>1</sup>, Lalit K. Awasthi<sup>3</sup>, Emmanuel S. Pilli<sup>1\*</sup>   
and Divya Kapil<sup>4</sup>

## Abstract

Virtualization techniques effectively handle the growing demand for computing, storage, and communication resources in large-scale Cloud Data Centers (CDC). It helps to achieve different resource management objectives like load balancing, online system maintenance, proactive fault tolerance, power management, and resource sharing through Virtual Machine (VM) migration. VM migration is a resource-intensive procedure as VM's continuously demand appropriate CPU cycles, cache memory, memory capacity, and communication bandwidth. Therefore, this process degrades the performance of running applications and adversely affects efficiency of the data centers, particularly when Service Level Agreements (SLA) and critical business objectives are to be met. Live VM migration is frequently used because it allows the availability of application service, while migration is performed. In this paper, we make an exhaustive survey of the literature on live VM migration and analyze the various proposed mechanisms. We first classify the types of Live VM migration (single, multiple and hybrid). Next, we categorize VM migration techniques based on duplication mechanisms (replication, de-duplication, redundancy, and compression) and awareness of context (dependency, soft page, dirty page, and page fault) and evaluate the various Live VM migration techniques. We discuss various performance metrics like application service downtime, total migration time and amount of data transferred. CPU, memory and storage data is transferred during the process of VM migration and we identify the category of data that needs to be transferred in each case. We present a brief discussion on security threats in live VM migration and categories them in three different classes (control plane, data plane, and migration module). We also explain the security requirements and existing solutions to mitigate possible attacks. Specific gaps are identified and the research challenges in improving the performance of live VM migration are highlighted. The significance of this work is that it presents the background of live VM migration techniques and an in depth review which will be helpful for cloud professionals and researchers to further explore the challenges and provide optimal solutions.

**Keywords:** Cloud computing, Virtual machine migration, Virtualization, Pre-copy technique, Post-copy technique, Security

## Introduction

In recent year's IT resources become more powerful, having high processing capabilities and a large amount of storage capacity, which attracts the application developers and service providers to use these resources. Further, the increasing demand for IT resources motivates the researchers and providers to share these resources among end users for efficient utilization and maximize the provider's profit. In cloud computing [1] environment

services are delivered in the form of hardware, software, storage, platform, infrastructure, database and much more using Google's App Engine [2], Microsoft Azure [3], Amazon's EC2 [4], IBM SmartCloud [5], etc. Cloud Computing delivers hardware and software capabilities in the form of services over the internet and allows consumers to be provisioned resources on-demand, on a pay-per-use [6] model. Due to the increased demand for cloud resources, cloud providers handle warehouse size data center, this large scale Cloud Data Centers (CDCs) carry more than thousands of computing servers which are connected by high-speed communication links and consume a large

\*Correspondence: [espilli.cse@mnit.ac.in](mailto:espilli.cse@mnit.ac.in)

<sup>1</sup>Malaviya National Institute of Technology Jaipur, Jaipur, India  
Full list of author information is available at the end of the article

amount of electricity. Further to provide guaranteed services, on an average 30% of servers remains in idle mode and approximately 10–15% of server capacity is used for fulfillment of resource demands [7]. The under utilization or over provisioning of resources result in a phenomenal increase in operational cost and power consumption [8, 9]. In 2013, it was estimated that Google data centers consume approximately 260 million Watts of electricity, which is enough power to give continuous electricity to more than 200,000 houses [10, 11]. In 2014, it has been estimated that IT would contribute only 25% to the overall cost of operating a CDCs whereas about 75% of the total cost would contribute to infrastructure and power consumption [12]. One of the basic solutions of such problem is to switch the idle mode server to either sleep mode or off mode based on resource demands, that leads to great energy saving because idle mode server consumes 70% of their peak power [13].

Virtualization technology was developed by IBM in 1960 to maximize the utilization of hardware resources because powerful and expensive mainframe computers were underutilized. It is a thin software layer running between Operating System (OS) and system hardware, termed as a Virtual Machine Monitor (VMM) or hypervisor, that control, manage, and mapped multifarious VM's (applications running on guest OS) on a single platform [14–16]. Also, it is a complete software and hardware stack to fulfill the incoming request or provide a service to users [17]. Examples of popular virtualization software are VMware ESX and ESXi [18], Kernel-based Virtual Machine (KVM) [19, 20]/Quick Emulator (QEMU), Citrix XenServer [21], Microsoft Virtual PC [22], Microsoft Hyper-V [23], Oracle VM VirtualBox [24], and Parallels Desktop for Mac [25]. The main advantage of virtualization is to provide better resource utilization by running multiple VM's parallels on a single server. Hypervisor supports the legacy OS to combine numerous under-utilized servers load onto a single server and also support fault tolerance and performance isolation to achieve better cloud data centers performance. Due to VM's isolation, failure of one VM does not have an effect on execution/functioning of other VM's and on the entire physical machine [26]. To improve CDC efficiency, different types of resource management strategies like server consolidation, load balancing, server up-gradation, power management etc. are applied through migration of single/multiple VM's. Also to achieve energy efficient environment, it combines numerous servers' loads onto a few physical servers and switch off the idle servers. For improving application performance, hypervisor also helps to migrate the running VM's from a low-performing to another better performing physical server [27]. Consequentially, co-hosting several different types of VM's onto a few servers is a challenging issue for researchers because

resource contention among co-hosted applications that leads to servers over-utilization which results in application performance degradation [15, 28–31]. Also a large number of cloud applications like interactive applications experience frequently changeable workload requests that generate dynamic resource demand which results in Service Level Agreement (SLA's) violation and performance degradation if dynamic server consolidation is used.

To resolve above stated issues, hypervisor selects appropriate VM/VM's and migrate them from over-utilized servers to under-utilized servers for improving the performance. During the process of VM migration, VM's continuously demand additional resources for migration that adversely affect the performance of running application until VM migration completes. So migration process must be finished within minimal time (to release the acquired resources in the short time) by using the optimal targeted server and network bandwidth to get improve migrating application performance, and migration transparency [14, 32, 33]. Hence the role of VM migration is bifold, facilitating improvement in resource utilization and increasing provider's profit.

For VM migration, hypervisor exploits live VM migration [34] for moving VM's between respective servers using shared or dedicated resources. Live VM migration continuously provides the service without interrupting the connectivity of running application during migration time to obtain seamless connectivity, avoiding SLA violation and to get optimal resource utilization. It is also used in adaptive application resource remapping [35]. It is a very useful technique in cluster and cloud environment. It has many benefits like load balancing, energy saving, preserving service availability. It also avoids process level problem such as residual dependencies [33], process dependency on its original (source) node. VM migration controller migrates a single VM [36] or multiple VM's (cluster VM set) [34] on Local-Area Network (LAN) or Wide-Area Networks (WAN) network for efficient management of the resources. If VM migration is performed within LAN [34] servers then it is easy to handle because storage migration is no longer required in Network Attached Storage (NAS) integrated data center architecture. Also, the network management within LAN requires minimal effort because IP address of the corresponding server remains unchanged. VM migration over a WAN network [37] takes a considerable amount of migration time because the transfer of storage migration, limited availability of network bandwidth, IP address management, packet routing, network congestion, and the faulty behavior of WAN links having considerable overheads.

Now a days, most of the hypervisors support live migration but the implementation of live migration with a little or no consideration towards its security. Hence live

migration might be susceptible to range of attacks from Denial-of-Service (DoS) attacks to Man-In-The-Middle (MITM) attacks. During the migration, data can be tampered or sniffed easily as it is not encrypted. Thus compromising confidentiality and integrity of migrating data. These security threats in live VM migration discourages many sectors, such as financial, medical, and government, from taking advantage of VM live migration. Hence, security is the critical challenge that needs examination to provide secure live VM migration.

In the literature, few surveys highlight the importance of VM migration in a cloud environment. Soni and Kalra [38] reviewed different existing techniques which concentrate on minimization of total migration time and downtime to avoid service degradation. Kapil et al. [39] performed a summarized review of existing live migration techniques based on pre-copy and post-copy migration. They considered total migration time, service downtime, and amount of data transferred as a key performance metrics for comparison. They mention some research challenges like the type of network (LAN/WAN), link speed, page dirty rate, type of workload, address wrapping and available resources. Further different aspects of memory migration, process migration, and suspend/resume based VM migration techniques have been surveyed by Medina and Garcia [26]. In this, few VM migration techniques are included and no comparison is performed. The authors have not considered performance parameters of currently running applications under VM migration, network bandwidth optimization, and hybrid VM migration technique for improving migration process. Xu et al. [32] present a survey on performance overheads of VM migration within inter-CDC, intra-CDC, and servers. Their proposed classification does not consider different aspects of VM migration, timing metrics, migration pattern, and granularity of VM migration for highlighting the application performance and resource consumption trade-off. A comprehensive survey has performed by Ahmad et al. [40] covering different VM migration points like VM migration patterns, objective functions, application performance parameters, network links, bandwidth optimization, and migration granularity. They reviewed state-of-the-art live VM migration and non-live VM migration techniques. But the authors did not show any analysis based on performance parameters of VM migration. Moreover, they did not describe the weakness of reviewed techniques. In their extended survey work, Ahmad et al. [41] presented a review on state-of-the-art network bandwidth optimization approaches, server consolidation frameworks, Dynamic Voltage Frequency Scaling (DVFS)-enabled storage and power optimization methods over WAN connectivity. They proposed a thematic taxonomy to categorize the Live VM migration approaches. The critical aspects of VM migration

is also explored by comprehensive analysis of existing approaches. A survey on mechanisms for live VM migration is presented by Yamada [42], covering existing software mechanisms that help and support in live migration. They reveal research issues that not covered by existing works like migration over high speed LAN, migration of nested VMM, and migration of VM attached to pass-through accelerator. The techniques are classified into two categories: performance and applicability. In a long-distance network, how the live migration and disaster recovery are performed with necessary operations is addressed by Kokkinos et al. [43]. They focus on new technologies and protocols used for live migration and disaster recovery in different evolving networks.

In our work, we address the limitations of existing surveys [26, 32, 38–43] and present comprehensive survey on state-of-the-art live VM migration techniques. We consider different important aspects of VM migration while incorporating the trade-off among application performance, total migration time, network bandwidth optimization for meeting the resource management objectives. Our major contributions in this paper can be summarized as follows:

1. Comprehensive literature review of state-of-the-art live VM migration techniques and description of strengths, weaknesses, and critical issues that require further research.
2. Definition of key aspects of migration process like CPU state, memory content and disk storage that affect total migration time and understanding of type of memory and storage content that need to be migrated.
3. Discussion on the various the performance metrics that affect VM migration process.
4. Discussion of various security threats and their categories in live VM migration and explanation of security requirements and existing solutions to mitigate possible attacks.
5. Classification of the existing migration mechanisms into three basic categories: type of live VM migration, duplication based VM migration and context aware migration based on the objectives and techniques used.
6. Identification of specific gaps and research challenges to improve the performance of live VM migration.

The paper is organized as follows: “Background” section presents the background of live VM migration and explain the various components, important features and limitations. In “Types of live virtual machine migration” section, types of live VM migration techniques - pre-copy, post-copy and hybrid techniques are presented. Brief overview of live VM migration models are presented

and a generic model is proposed in “Live virtual machine migration models” section. A comprehensive and an exhaustive survey of the state-of-art live VM migration techniques is done in “Live virtual machine migration frameworks” section. Threats and security requirement in live VM migration is briefly discussed in “Threats in live virtual machine migration” section. Specific research gaps and open challenges in Live VM migration are described in “Research challenges” section. Finally, “Conclusion and future work” section, concludes the paper with future research directions.

## Background

Live VM migration is the technique of migrating the states (CPU, memory, storage etc.) of VM from one server to another server. It is being researched for a decade but still some of the issues require further examination and solutions. The evolution, motivation, and components of live VM migration are given below.

### Evolution and motivation

Live migration of OS is a extremely powerful tool for administrators of CDC's, by allowing a clean separation of hardware and software considerations, and consolidating servers into a single coherent management domain that facilitates load balancing, fault management, resource sharing, and low-level system maintenance. Sapuntzakis et al. [44] pointing the user level mobility and management of the system by migrating the hardware states, called capsule. For reduction of capsule size, authors proposed copy-on-write disks, “ballooning”, demand Paging, and hashing techniques. Authors show using capsule migration, the active applications can be started within 20 min on a 384 kbps network speed. According to them, the VM migration is a better solution instead of installing the application.

At the initial of the cloud for handling the residual dependencies at process level is a difficult task, Clark et al. [33] proposed the idea of live VM migration algorithm, which has the capability to move the entire OS. Authors report that live migration of VM is transferring the memory image from one server to another. Authors also introduced the writable working set concept and focused on data center and cluster environment and implemented migration support for Xen.

Nelson [45] focused on transparent migration system that can migrate unmodified applications on unmodified OS. They have shown that transferring the memory while VM is running, VM experiences less than 1 s of downtime. Huang et al. [46] proposed Random Direct Memory Access (RDMA) based VM migration to avoid the lower transfer rate and high software overhead problem when VM is migrated over TCP/IP (Transmission Control Protocol/Internet Protocol). RDMA access the high speed

interconnections, such as InfiniBand, to enable OS-bypass communication. By RDMA, the memory of one computer can be accessed by another without involving one's operating system. To transfer the VM state traffic socket interface and TCP/IP protocol is used in most VM environment. High speed interconnects and RDMA offers high through-put, as a result, memory pages transfer time can be reduced.

The whole machine migration concept is introduced by Luo et al. [47], in which VM run-time state including memory contents, CPU state, and provided local disk storage is migrated. To minimize the service downtime due to large amount storage, and for maintaining disk storage consistency and integrity, authors proposed a Three-Phase Migration (TPM) (pre-copy, freeze-and-copy, and post-copy) algorithm. To easily carry-out migration process at source server they use the Incremental Migration (IM) algorithm to bringing down migrating data back to the destination server. Block-bitmap is used, for tracking of all the write access of local disk while migration is performed, this also synchronizes the migration of local disk. The experimental results show that TPM algorithm is performed well when used for I/O intensive applications. Also, the downtime of migration is 100 milliseconds equal to shared storage migration and performance overhead for recording write processes is also low.

Furthermore, the growth of cloud computing has led to establishing numerous CDC's around the world that consume a huge amount of electrical energy which results in high operational cost and carbon footprints to the environment. In recent years, the sole concern behind CDC's deployments is to provide high-performance and availability dwindles, without paying much attention to data centers energy consumption. As energy consumption increasing continuously, there is a need to focus on resource management to optimizing them for energy efficiency, while maintaining high-performance. So minimizing the energy usage of data centers is a challenging issue because applications and data size are growing very rapidly which require fast servers and large disk storage to process service request within the defined time period. Hence, eliminating any waste of power in CDC's is very necessary.

Until recent, the aim of resource allocation policies in a CDC's is to provide high performance for the fulfillment of SLA, without considering the energy cost. Based on the performance requirements, the VM's are logically resized and consolidated to the lesser number of servers which leads to reducing energy consumption by switching the idle servers to the either sleep mode or off mode. Further, to explore energy and performance efficiency, three critical issues must be pointed out like: (1) power cycling; excessive power cycling of a server could reduce its reliability; (2) switching among frequencies:

switching resources off in a dynamic environment is a critical from the SLA perspective because the frequently changing nature of workload may not fulfill desired Quality of Services (QoS) due to insufficient number of active servers under peak load; (3) performance management: ensuring SLA's brings issues to performance management in virtualized environment. Hence, all these issues require effective consolidation policies which are more energy-efficient without compromising the defined SLA.

The Power-aware Application Placement problem has investigated by Verma et al. [48]. An application placement controller (pMapper) is used for dynamically placement of applications to minimize power consumption while meeting performance guarantees. Secure energy-aware resource provisioning has proposed by Sammy et al. [49]. For server consolidation, VM migration using Dynamic Round Robin algorithm gives a more feasible solution which reduces energy consumption without compromising on security. Further, Beloglazov et al. [50] divide the VM allocation problem into two parts: one is the admission of new requests for VM provisioning and VM placement on the server and it is treated as bin packing problem, whereas the second is the optimized allocation of VM's and solve it by modification of the Best Fit Decreasing (BFD) algorithm. In the Modified Best Fit Decreasing (MBFD) algorithm sort all VM's in there decreasing order of current CPU utilization, and allocate them to a server that provides the least increment on server power consumption. So it selects the more power-efficient server first.

Live VM migration is required to full fill the running application resource demand. It facilitates the following features:

1. **Load Balancing:** It required when the load is considerably unbalanced and impending downtime often require simultaneous VM (s) migration. It is used for continuing services after fail-over of components which are monitored continuously then load on host distributed to other hosts and no longer sends traffic to that host [51–53].
2. **Proactive fault tolerance:** Fault is an another challenge to guarantee the critical service availability and reliability. Failures should be anticipated and proactively handled, to minimize failure impacts on the application execution and system performance. For this different type of fault tolerance techniques are used [54].
3. **Power management:** switch the idle mode server to either sleep mode or off mode based on resource demands, that leads to great energy saving because idle mode server consumes 70% of their peak power [13], and consolidate the running VM's to fewer active hosts leads to great energy saving. So dynamic allocation of VM's to few active servers as much as possible, VM live migration is a good technique for cloud power efficiency.
4. **Resource sharing:** The sharing of limited hardware resources like memory, cache, and CPU cycles leads to the application performance degradation. This problem can be solved by relocating VM's from over-loaded server to under-loaded server [55, 56]. Although, the sharing of resources leads to cut down operational cost because of switch-off the unnecessary or idle servers [57, 58].
5. **Online system maintenance:** A physical system required to be upgrade and service, so all VM's of that physical server must be moved to an alternate server for maintenance and services are available to users without interruption [59].

#### Components in live virtual machine migration

At the time of live VM migration, it is essential to know about what to migrate or which content must be migrated. In the migration process, it is essential to observe that how migration process handles CPU state, memory contents, and storage contents [60]. CPU state is little bit information and represents the lower bound of service downtime.

#### Memory content

Memory content is a larger amount of information, that incorporate the running processes memory and guest OS memory within the VM. The VM is configured with a large amount of memory, but it may not be fully utilized by VM, so no need to transfer unused memory. Also, the compression technique is used to speed up migration process. Following are the memory module that needs to be moved under the process of migration:

1. **VM Configured Memory:** The amount of actual physical memory that is given to guest VM by the hypervisor. The guest VM used this memory as their own physical memory.
2. **Hypervisor Allocated Memory:** It is part of VM configured memory and actively used by VM but its size is less than the VM configured memory. If a VM access this memory and free it, but the decision of release of memory depends on the hypervisor.
3. **VM Used Memory:** It is currently and frequently accessed through VM OS and all running processes. These memory pages keep track by the guest VM.
4. **Application Requested Memory:** The amount of memory required for running an application and it is allocated by guest VM OS. It is not necessary that the requested memory is within the physical memory, it may be in disk storage when all the VM configured memory is in use.

- 5. Application Actively Dirtyed Memory:** It is the part of the application requested memory which is frequent access and modified by running application so it is commonly resident in memory, so avoid to swap-out on disk storage.

#### **Storage content**

It is a voluntary part of live VM migration. In LAN connections like cluster and CDC uses NAS storage so no need to transfer storage contents. If it is not possible to transfer disk storage or destination cannot access the source disk storage, then a new virtual disk storage needs to be registered on the destination server and finally content needs to be synchronized with source server. The storage contents carry a large amount of information need to be transfer and the full disk image transferred considerable time while transferring through the network. To reduce the transfer time hypervisor can identify the unnecessary storage contents and unused space to avoid them transfer, that leads to reducing the migration time. The different type of storage content that needs to be migrated:

- 1. Virtual Disk Size:** The disk size allocated to VM for its use is called virtual disk size and its size is defined when the VM is created. Generally, hypervisor recommends choice to avail all the disk space when VM is created or to dynamically expand based on storage uses.
- 2. VM Used Blocks:** It is the system and user data blocks, which are stored in VM image. These blocks are accessed and used by guest VM OS. It is the size of data actually contains in VM files and it may not be completely filled by data.
- 3. Hypervisor Allocated Blocks:** It is actually allocated space by hypervisor to VM for data storage and its size may be same as virtual disk storage size if pre-allocation is performed. If the VM free some blocks then hypervisor may not shrink the allocated block size because it is harder or not visible for hypervisor to look at VM level storage, it is only visible to VM level file system that which blocks are in use and which are free. So avoiding unused space and garbage collection block could considerably reduce the migration time, but it is not easy by hypervisor because hypervisor implementation not carries garbage collection blocks information.

#### **Limitation of live virtual machine migration**

VM's can be migrated smoothly among corresponding servers for meeting the performance parameters, load balancing, and energy saving. Live VM migration is a strong management technique in the multi-VM based environment. Cost effective means are offered by cloud

computing service providers and live VM migration is utilized for effective workload movement which has very small service downtime. But this is still a challenge to migrate VM's between private and public cloud as well as different service providers. There is currently no provision for live migration in or out of a public cloud environment [61]. Rackspace could migrate VM's but it is cold migration, not live migration. After 2013, Google Compute Engine [62] uses the live migration for kept customer VM's running, while performing software updates, fixing out some hardware problems, and recovery from unexpected issues that have arisen, as shown in Fig. 1. When compute engine migrates a running VM from one server to another then it migrates the complete instance state in a way that it transparent to the end user and other who access that VM. The process starts with notification that VM's need to be evicted from their current hosting server. Google's cluster management software continuously tracking these events and schedule them based on data center policies. After the VM selection process for migration, Google provides a notification to the guest that a migration is imminent. On the completion of the waiting period, a destination server is selected and asked to set up a new, empty VM (target) to receive the migrating VM (source). Authentication is performed to establish a secure connection between corresponding servers.

#### **Types of live virtual machine migration**

##### **Pre-copy techniques**

The pre-copy technique uses iterative push phase that is followed by stop-and-copy phase as shown in Fig. 2 in the flow chart form. Because of iterative procedure, some memory pages have been updated/modified, called dirty pages are regenerated on the source server during migration iterations. These dirty pages resend to the destination host in a future iteration, hence some of the or frequently access memory pages are sent several times. It causes long migration time. In the first phase, all pages are transferred while VM running continuously on the source host. In a further round, dirty pages will resend. The second phase is termination phase which depends on the defined threshold. The termination is executed if any one out of three conditions meet: (i) the number of iterations exceeds pre-defined iterations, or (ii) the total amount of memory that has been sent or (iii) the number of dirty pages in just previous round fall below the defined threshold. In the last, stops-and-copy phase, migrating VM is suspended at source server, after that move processors state and remaining dirty pages. When VM migration process is completed in the correct way then hypervisor resumes migrant VM on the destination server. KVM, Xen, and VMware hypervisor use the pre-copy technique for live VM migration.

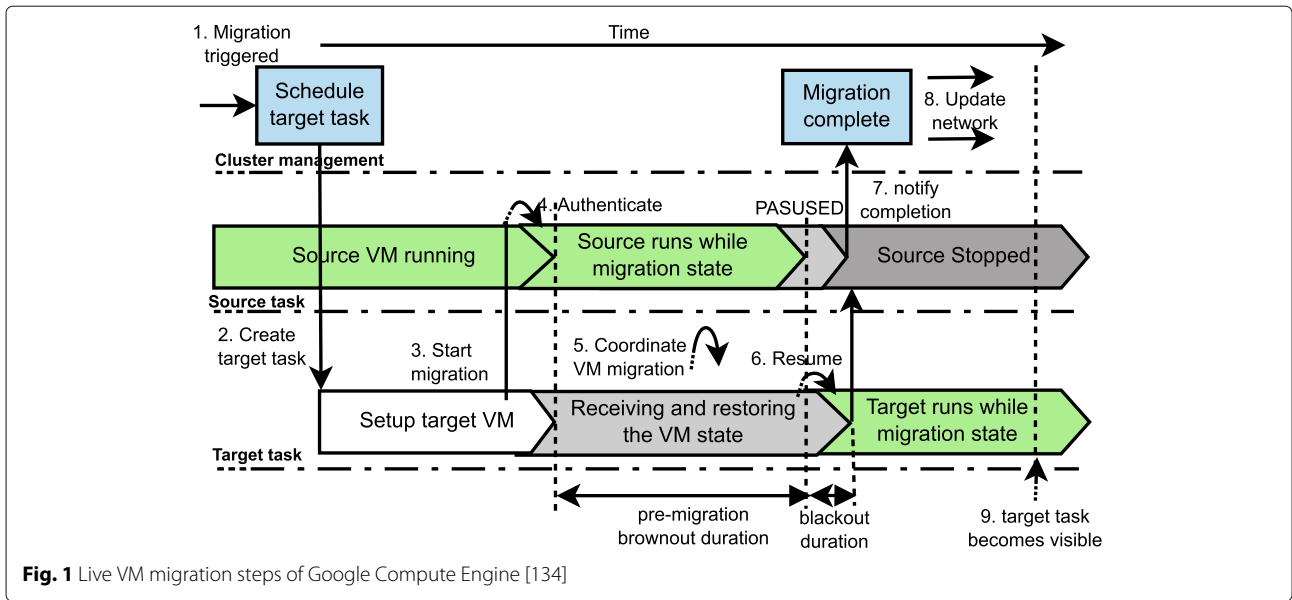


Fig. 1 Live VM migration steps of Google Compute Engine [134]

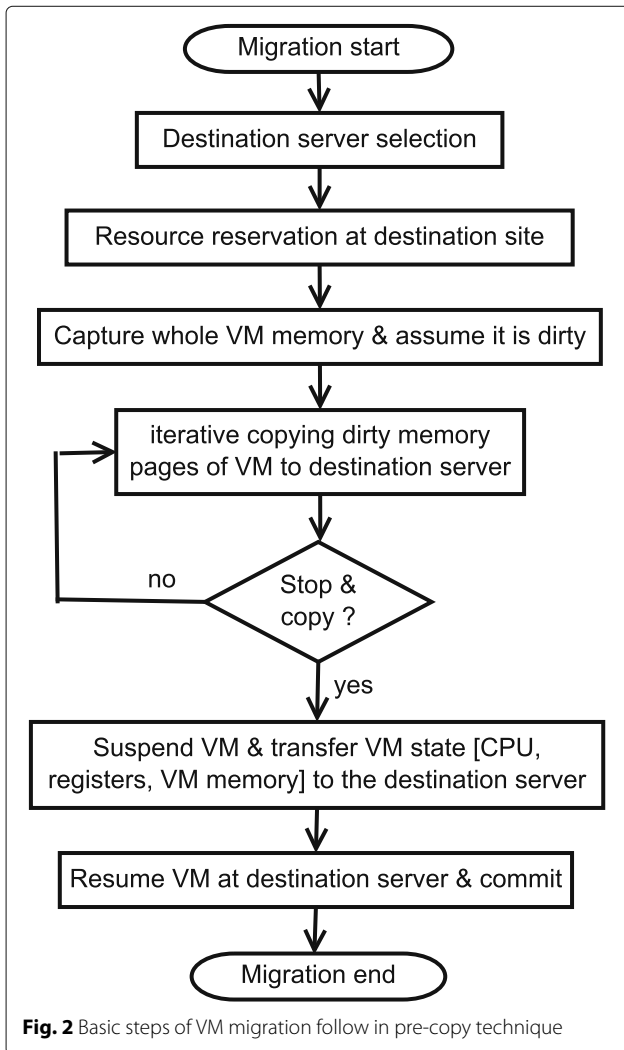


Fig. 2 Basic steps of VM migration follow in pre-copy technique

**Post-copy techniques**

In post-copy migration technique, processor state transfer before memory content and then VM could be started at the destination server. To optimizing live migration of VM's, Hines et al. [63] proposed post-copy technique. Post-copy VM migration technique investigates demand paging, active push, pre-paging, and Dynamic Self-Ballooning (DSB) optimization's approaches for pre-fetching of memory pages at the destination server.

Post-copy technique variations or post-copy optimization approaches:

1. **Demand paging:** It ensures that VM request pages are sent only once over the network. When VM resumes at the target server and requesting memory pages for read/write operation results in page faults, the faulty pages are serviced by retransmission from the source server. Therefore, servicing of faulty pages considerably degrade the application performance. So demand paging provides the simplest and slowest option.
2. **Active push:** It removes residual dependencies from the source server and it pro-actively pushes the VM pages to destination server even when VM is running on the destination server. If the page fault occurs at destination VM then demand paging is used to deal with fault. Therefore, pages are sent only once either via active push or demand paging.
3. **Pre-paging:** It requests to the destination server for future access page, that helps to avoid or mitigate page fault rate. For this, it uses the hint of page access pattern on destination VM. So we can avoid the future page faults in advance and accept the better page pushing sequence to access the patterns.

- 4. **Dynamic self-ballooning (DSB):** DSB is used for avoiding the transfer of free memory pages. This approach speed-up the migration process with negligible performance degradation by periodically releasing free pages of VM back to the hypervisor. Hence sending of the unused page count is increased and total migration time is reduced by avoiding the sending of unused pages to the destination server.

Perhaps all the above approaches the similar memory page re-transmission problem still exists. Consequently, service downtime and total migration time are affected by similar page re-transmission.

Basic steps of post-copy VM migration is presented through flow chart as shown in Fig. 3.

Post-copy has the ability to minimize network page faults, by pushing future requested pages from the source server before they are faulted by running VM. for this active push approach used with adaptive pre-paging. Michael R. et al. [64] compare the performances of the post-copy and pre-copy technique using Xen hypervisor. The results show that different migration metrics like total migration time, pages transferred, and network overhead has improved, VM having a range of workloads. To avoid sending of all duplicate pages, the post-copy technique is used with adaptive pre-paging.

The post-copy technique is effective when the majority of pages are transferred to target server before page faulty occur at destination VM and minor page faults occur due to network faults.

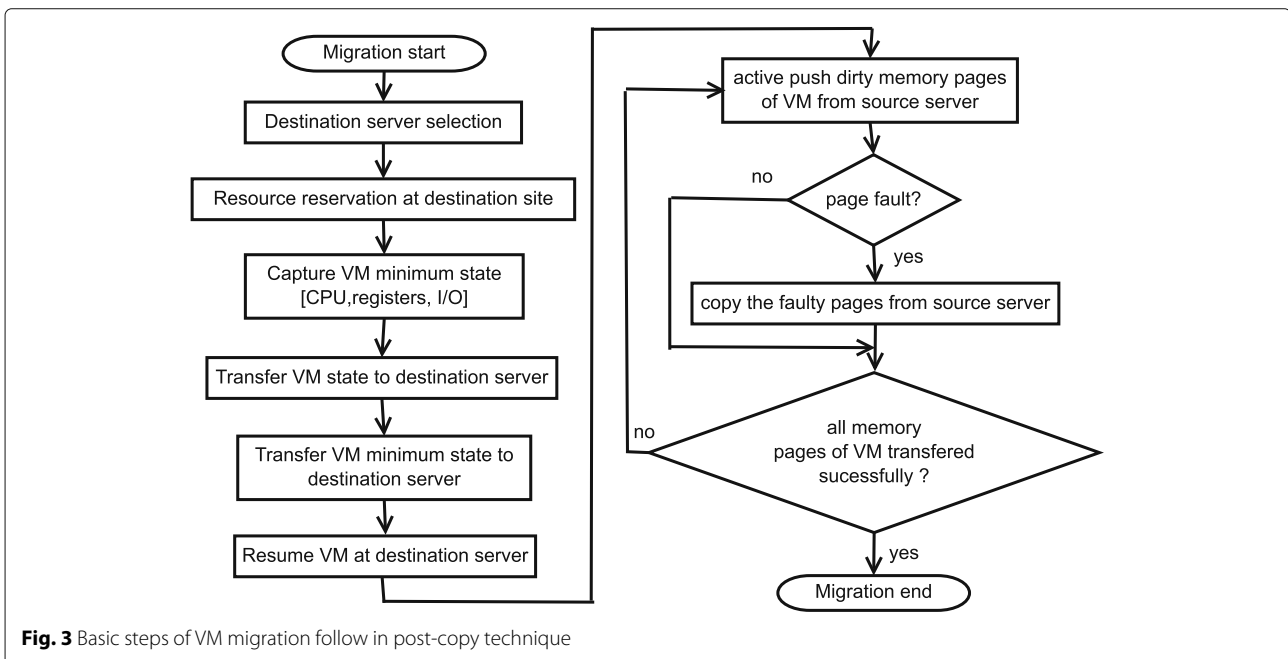
**Hybrid technique**

The hybrid VM migration technique includes both pre-copy and post-copy VM migration phases to improve the total migration time and service downtime. It works in five phases: i) migration preparation phase: under this phase required resources are reserved at the destination server. ii) Bounded pre-copy rounds: in this phase, it identifies and transfers VM working-set to the destination server. iii) VM state transfers phase: VM minimum state is recorded and transfers to the destination server. iv) VM resume phase: transferred VM is resumed at the destination server. The last v) demand paging phase: VM requested faulty pages (due to read/write operations) are bring at destination server from source server for continuing VM execution and synchronization with source VM image.

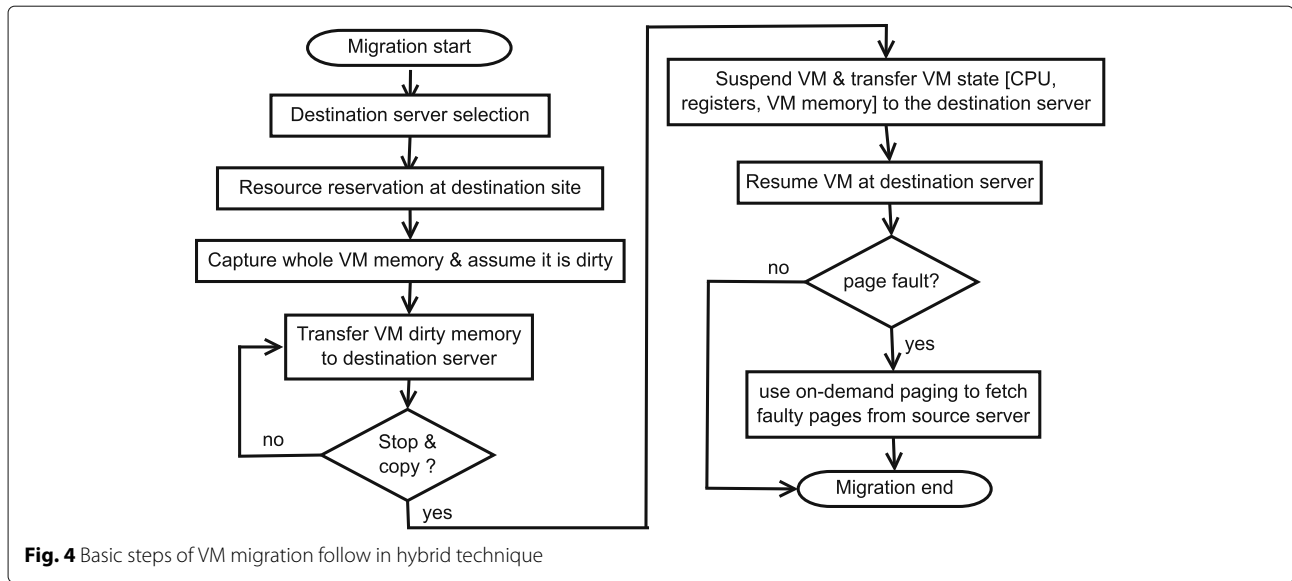
Basic steps of hybrid VM migration is presented through flow chart as shown in Fig. 4.

Effectiveness of Live Virtual Machine Migration technique: Pre-copy technique focus to keep downtime small by minimizing transferred VM’s state, so application service is running without interruption or VM transfer is seamless. But it increases total migration time due to the repeatedly transfer of dirty pages.

In the post-copy technique, all the memory page is transferred at most once and it reduces network page fault by active push memory pages before they are faulted at the target server. During migration free pages are also transmitted in both approaches which increase total migration time. To avoid this problem DSB mechanism is used.







As compared to pre-copy, post-copy technique reduces the number of pages transferred and total migration time. But, the post-copy technique has more downtime than pre-copy technique due to migration latency of page fetching before VM can be resumed on the target server. Another disadvantage is, if any kind of failure occurs during the migration then recovery may not be possible. Table 1 shows a comparison between pre-copy and post-copy technique on the basis of performance metrics [63].

Svärd et al. [65] identify the essential characteristics of live migration. They investigate, categorize, and compare three migration approaches which are pre-copy, post-copy and hybrid techniques. In their work, they migrate VM's with large memory sizes and high dirty rates to find differences and limitations of the migration approaches. They conclude that when robustness is essential then use the pre-copy live migration otherwise use either post-copy or hybrid migration that reduces the service downtime, total migration time and consume fewer resources.

Situations in which, pre-copy or post-copy improve performance: It depends upon workload type and performance goal of migration. Pre-copy would be better approach for read-intensive workload whereas, for write-intensive or large memory workload, post-copy would be better.

**Performance metrics**

Researchers have suggested various performance metrics in live VM migration and these metrics are affected whenever VM migration take-place. Voorsluys et al. [66] show that service levels of running application could be down when it is migrated. So it is very important to migrating OS with minimal zero time when OS is serving live services. They measure the performance of running applications VM inside Xen hypervisor during live VM migration. Kuno et al. [67] assess the performance of live and non-live VM migration. In non live migration, VM stops but there is no performance degradation, whereas in live migration VM, process keep running and performance may degrade. They show that memory writing and host OS

**Table 1** Comparison table of pre-copy and post-copy based on Performance metrics

Performance metrics	Pre-copy technique	Post-copy technique
Preparation time	It includes all the modified pages in iterative memory copying phases	Negligible
Downtime	It includes transferring any remaining dirty pages	It includes transferring another minimal execution state
Resume time	Re-schedule the target VM at the destination server and remove the memory pages at the source server	Majority of post-copy approach transfer the VM state and most of the memory pages in this period
Pages transferred	Transfer number of pages during preparation time period	Transfer number of pages during resume time period
Total migration time	More	Less
Performance degradation	High due to the tracking of dirtied pages for write-intensive workloads	High due to servicing of faulty pages

communication are important reasons for performance degradation. They measured that migration time and memory size of VM is proportional and in both methods of migration, migration time is almost same. Results show that live migration provides better performance when VM is running CPU intensive task and could be better for I/O intensive application if network speed is high. Xen and VMware products have the technology for live migration, called XenMotion and VMotion respectively. Feng et al. [68] compare the performances of them and shows that VMotion generates less amount of transferred data than XenMotion and XenMotion perform better than VMotion in terms of total migration time. Both VMotion and XenMotion performance degrades in the network because of network delay and packet losses. The live migration techniques give better performance in LAN networks. Live VM migration performance can be measured using following metrics and compared in Table 2:

1. **Total Migration Time:** It is the summation of all migrant VM's migration time. Its value can vary due to the amount of data to be moved during migration and migration throughput. It depends on 1) the total amount of memory transferred from source to destination server, and 2) allocated bandwidth or link speed.

$$t_m = \frac{v_m}{b} \tag{1}$$

Where,  $t_m$  = total migration time  
 $v_m$  total amount of memory  
 $b$  = bandwidth

2. **Downtime:** It is the time when service is not running or available due to migration of processor states. Downtime extends because current algorithms are not able to keep a record of dirty pages of migrating VM. The downtime  $t_d$  is depends on page dirty rate  $d$ , page size  $l$ , duration  $t_n$  of the last pre-copy round  $n$ , and link speed  $b$ , Lui et al. [69] define the downtime as:

$$t_d = \frac{d * l * t_n}{b} \tag{2}$$

**Table 2** Factors impacting the metrics

Performance metrics	Factors
Downtime	Synchronization mechanism, low-bandwidth network
Amount of transferred data	It may be larger than the actual run time data size because there must be some redundancy for Synchronization and protocol
Total migration time	Large amount of data (some pages need to be transferred multiple times because of modification)

3. **Pages Transferred:** The amount of memory contain by VM or number of pages transferred during VM migration, it also includes duplicate pages. Liu et al. [69] calculate the page transferred at round  $i$ ,

$$v_i = \begin{cases} v_{mem} & \text{if } i = 0 \\ d * t_{i-1} & \text{otherwise} \end{cases} \tag{3}$$

where,  $v_{mem}$ : the amount of VM memory  
 $t_{i-1}$ : time taken to migrate dirty memory pages, generated during just previous rounds  
 The elapsed time of VM migration  $t_i$  at each round can be calculated as:

$$t_i = \frac{v_{mem} * d^i}{r^{i+1}} \tag{4}$$

network traffic  $v_{mig}$  during VM migration:

$$v_{mig} = \sum_{i=0}^n v_{mem} \left(\frac{d}{r}\right)^i \tag{5}$$

where,  $r$ : memory transmission rate during VM migration.  
 migration latency  $t_{mig}$  is calculated as:

$$t_{mig} = \sum_{i=0}^n t_i \tag{6}$$

4. **Preparation Time:** The time difference between initiation of migration and transferring the VM's state to the target server, while continuing its execution and dirtying memory pages.
5. **Resume Time:** The time when VM migration is done and resume its VM execution at the targeted server.
6. **Application Degradation:** Due to migration the performance of application is interrupted or slow down services during migration
7. **Migration Overhead:** There is need of some extra machine resources to perform a migration.
8. **Performance Overhead:** Degradation of service performance during migration or interrupting the service while executing smoothly The migration process introduce delay, extra logs, and network overheads during applications execution on VM.
9. **Link speed:** It is the most crucial parameter with respect to the performance of VM. The allocated bandwidth or capacity of the link is inversely proportional to service downtime and total migration time. The faster transfer requires more bandwidth, hence it takes less total migration.
10. **Page dirty rate:** It is also the major factor impacting migration behavior. The rate at which VM memory pages are updated by VM applications, it depends on the number of transferred pages in every pre-copy iteration [70]. If the dirty rate is higher than it increases data sent per iteration, leads in increasing

total migration time and service downtime. Dirty page rate and migrating VM performance are not in a linear relationship. If the rate of dirty page generation is lower than link capacity results in lower total migration time and downtime because modified pages are sent frequently. Otherwise, migration performance degrades significantly.

Migration of a VM, running specific application such a memory-intensive, read-intensive or write-intensive.

- If a VM is running memory-intensive applications than VM migration leads to performance degradation due to network traffic, downtime, and latency.
- The pre-copy technique reduces VM downtime and adverse effects on application performance if VM is executing the read-intensive application.
- The pre-copy technique is not performed well if running application is Write-intensive. Because write-intensive application frequently modifies a large number of pages that result in dirty pages transferred multiple time.

### Live virtual machine migration models

In this section, we present a brief review on existing models of live VM migration. The term “model” is used for the theoretical representation of Phases involved in live VM migration. The models may or may not have been implemented. We further propose a generic model of live VM migration, which considers the required phases of live VM migration, based on existing models.

For efficient utilization of CDC resources, frequent live migration is used, but live migration performance is an issue. For this, reliant evaluation method is required to select the best optimum software and hardware combination environments that obtain the best live VM migration performance. For this, Huang et al. [71] proposed a live migration benchmark – Virt-LM solution. Virt-LM benchmark is used to compare live migration performance on different CDC environments among different software and hardware environments. Different types of performance metrics, application, stability, compatibility, usability, and impartial scoring methodology are the main objectives for designing of Virt-LM. To validate effectiveness of Virt-LM, it is run on two hypervisors - Xen 3.3 and KVM-84 on Linux kernel 2.6.27. For this DELL OPTIPLEX 755 physical machines (Intel Core Quad Q6600 CPU 2.4GHz, 2GB RAM, SATA disk) used under test hardware and connected by single 100 Mbit communication links.

Using the live VM migration function, cloud service providers can consolidate many VMs with small workloads into a few servers to achieve high resource utilization. Also, VM's with heavy workloads on a server

migrated to other servers (having a low load) for load balancing. VMware reported that the frequency of running live VM migration invoked by automated load balancing functions in the range of 0 to 80 per hour in their data center, that leads performance degradation of live migration. Furthermore, it is difficult for cloud providers to provide requested resources to the end users in a timely fashion. Kikuchi et al. [51] design a PRISM performance model for parallel live migrations. In their work, data collection and migration processes are performed simultaneously. Their model represents performance characteristics of live migration and that is based on data. The experimental setup for performance measurements consists of one network storage for storing VM images and four physical machines for VM's deployment. Fujitsu PRIMERGY RX200 S5 physical machines (16-core Intel Xeon X5570 CPU 2.93GHz, 32-GB RAM, 3-136-GB hard disk drives) are used as network storage and physical servers and connected by 1GB Ethernet switch. To enable virtualization, XenServer 5.6 is installed on these physical servers.

Cloud support the unprecedented elasticity to dynamically grow and shrink resource availability for hosted applications based on on-demand function. VM migration enables the elasticity function and resource usage can be dynamically balanced over the physical server that allows applications to be dynamically relocated to improve reliability and performance. When a VM should be migrated and how the necessary resources should be allocated, resource availability can help to take this sort of decision. Using regression statistical method Wu and Zhao [72] proposed a performance model. This model is used to predict migration latency and able to generate appropriate resource management decisions. They migrate a Xen-based VM. The method represents that the availability of resources has an impact on migration latency by profiling the migration of different types of VM's that are highly resource-intensive. Performance model can be used to predict the migration time or at least upper bounds for VM's. For experiment two servers source and destination servers configured with 6-core 2.4GHz Opteron CPU's and 32GB of memory. For virtualization Xen 3.2.1 is installed on Linux version 2.6.24.

The existing approaches focused on the VM placement techniques for performance improvement with defined constraints. Some literature works concern about performance and energy cost while handling the VM consolidation. CDC's consume an excessive amount of energy. It is accountable for global increase in energy consumption, and energy cost additionally as a proportion of IT costs. Further, the migration cost may vary considerably because it depends on the type of workload, workload characteristics, and required VM configurations. While considering the migration overhead during migration decision making, Liu et al. [69], investigate

design methodologies to quantitatively predict the energy cost and migration performance. It is based on empirical studies and theoretical analysis on Xen 3.4 platform. This model represents both energy and performance in term of VM migration cost. They validate their model by conducting some set of experiments. The migration performance metrics handle several factors like workload characteristic, VM memory size, memory dirty rate, and network transmission rate. Experiments are conducted on Dell PowerEdge1950 servers (2 Intel quad-core Xeon E5450 CPU 3GHz, 8GB RAM, 250GB SATA hard disk) with 1Gbit Ethernet interface. The host machines running on Red Hat 4.1.2 platform, Linux 2.6.18.8-Xen kernel and Xen 3.4.1 hypervisor are installed. For power consumption measurement WattsUp Pro [73] is used. The results show that the proposed model is about more than 90% prediction accuracy with respect to measured cost and Model-guided decisions considerably reduce the migration cost by more than 72.9% at an energy saving of 73.6%.

In the cloud-based edge networks number of cooperating VM's can implement the tasks traditionally performed by disrupting and expensive network middle boxes. Cerroni and Callegati [74] proposed a model for cloud-based edge network. Their proposed model evaluate the performance of live VM migration for co-operating VM's that implemented a user's profile. They derived the function for service downtime and total migration time as an expression of network profiling and system design parameters. Service downtime is an end-user quality indicator and total migration time is purely related to the network bandwidth availability of an operating environment. Authors considered two types of migration scheduling alternatives, namely sequential migration strategy and parallel migration strategy, and results shows that trade-off exist between them. Furthermore, we find the situation in which parallel migration reduces the migration downtime with the occupancy of lot of resources. Extension of work could involve a general scenario in their model, that consider a more accurate memory profiling for edge network and test whole functionalities on a real system.

In live VM migration eviction time (time to evict the VM state from the source server) metrics is proposed by Deshpande et al. [75]. Eviction time metrics determines how fast the source server goes into the offline mode or the freed resources for further availability to other VM's. The traditional live VM migration techniques like pre-copy and post-copy, treated the eviction time as the total migration time because both the source server and the destination servers are bounded by migration duration. Eviction time value is continuously increase if the destination server not carrying sufficient memory or network bandwidth because it affects the receiving speed of incoming VM traffic, in such situation source server is also tying up. For such problem, they proposed a

Scatter-Gather live migration approach. Scatter-Gather approach reduces the eviction time by decoupling the source and destination server when the destination server is resource constrained. Source server scatters the VM's memory state to multiple middle boxes in the cluster, at the same time destination server starts gathering VM's memory from the middle boxes using a post-copy variant. The experiments are performed on physical machines (dual quadcore CPU 1.7GHz, 16GB DRAM, 1Gbps Ethernet card) connected via Nortel 4526-GTX layer-2 Ethernet switches. The machines are run Linux kernel 2.6.32 and KVM/QEMU 1.6.50 hypervisor with Linux kernel 3.2 as guest OS. Results show that Scatter-Gather reduces the eviction time by up to a factor of 6. It is important for data center administrator's toolbox when low VM eviction time is required.

In the literature, a little work focuses on the cost and performance interference while handling VM migration on corresponding server sides, that leads to SLA violation. For such problems, Xu et al. [52] proposed a lightweight Interference-Aware (iAware) live VM migration solution. It empirically captures the important relationship between performance interference of VM and major factors which are easily accessible in the real environment with defined benchmark workloads on a Xen hypervisor cluster platform. It minimizes both the migration and co-location interference among VM's, using a demand-supply model with multi-resource handling. VM's are hosted SPECCPU2006 [76], Hadoop v0.20.203.0 [77], netperf v2.5.0 [78], SPECweb2005 [79], and NASA Parallel Benchmark (NPB) v3.3.1 [80] respectively to examine run time overheads for mixed workloads. Fifty VM's in Xen virtualized cluster and 10 VM's are assigned to each workload. Large-scale experimental simulations are conducted for evaluating the performance gain and run time overheads in terms of CPU consumption, network throughput, and scalability. Further, the evaluation results are compared with traditional interference-unaware algorithms. Also, they observed that iAware is more flexible and able to co-operate with traditional VM consolidation or scheduling policies in a complementary way. So, the load balancing and power saving can achieve without affecting application performance.

A fractional hybrid pre-copy migration technique for storage and memory migration over WAN is proposed by Zhang et al. [81], it is a kind of adaptive live migration approach. As the name suggests, a fraction of memory and storage is transferred in pre-copy phase. The remaining memory and storage contents are transferred through the variant of post-copy migration (demand paging). The fraction is adjusted that helps to restore the migrating VM's performance to its original level. Proposed approach highlights the VM's migration over WAN, where the storage content migration is a critical research issue. Whereas,

the storage migration over LAN is often required because it is shared between the corresponding servers. They develop a probabilistic prediction model and profiling framework to adaptively find storage and memory fractions to migrate. Two physical machines (Intel Core2 Duo E6750 CPU 2.66GHz, 2GB RAM) with Linux 3.3.4 OS on both host and guest OSes. The Xen 4.1.2 hypervisor is used as memory management and QEMU is modified at the backend. Experiment is emulated on WAN network and results show that the fractional hybrid pre-copy migration solution achieves significantly improved adaptiveness than others while maintaining the responsiveness of post-copy algorithms.

The network contention between the VM application traffic and the migration traffic is a critical issue while dealing with the live VM migration. When VM migration is processed with pre-copy, then the VM continues running at source server whereas in post-copy VM is continuously running at destination server. VM migration applications with a predominantly outbound traffic deal with outgoing migration traffic at the source server whereas in predominantly inbound traffic deal with the incoming traffic at the destination server. Therefore network contention increases the total migration time and degrades the application performance. For the same issue, traffic-sensitive live VM migration model is proposed by Deshpande and Keahey [82], for a reduction in network contention of migration traffic and VM traffic. They used a hybrid technique for the migration of the co-located VM's (VM's that are located on the same server), besides work with any one of pre-determined VM migration technique for migrating all the VM's. Authors use the network traffic profiles by which complements the direction of most VM application traffic, that provides the base of migration technique selection. They implemented it on KVM/QEMU platform. Authors compared their traffic-sensitive migration technique with pre-copy and post-copy VM migration. Two host (16 GB RAM and 8 CPU's) are deployed with one VM (5 GB RAM and 2 vCPU's) each. First VM executes a Netperf [78] client and sends a TCP stream to other VM that running a Netperf server. The results show that their approach reduces network contention for migration, that leads to reduces the total migration time and adverse effects of migration on application VM's performance. For further extent of traffic-sensitive migration, VM's from single source migrated towards different destination servers. During consolidation, VM's from several source hosts are migrated to fewer destination hosts. Similarly, VM's are scattered to more hosts to meet their increasing resource requirement.

Comparison of above-mentioned models is illustrated in Table 3.

For efficient utilization of CDC resources, frequent live migration is performed, but VM performance is an issue

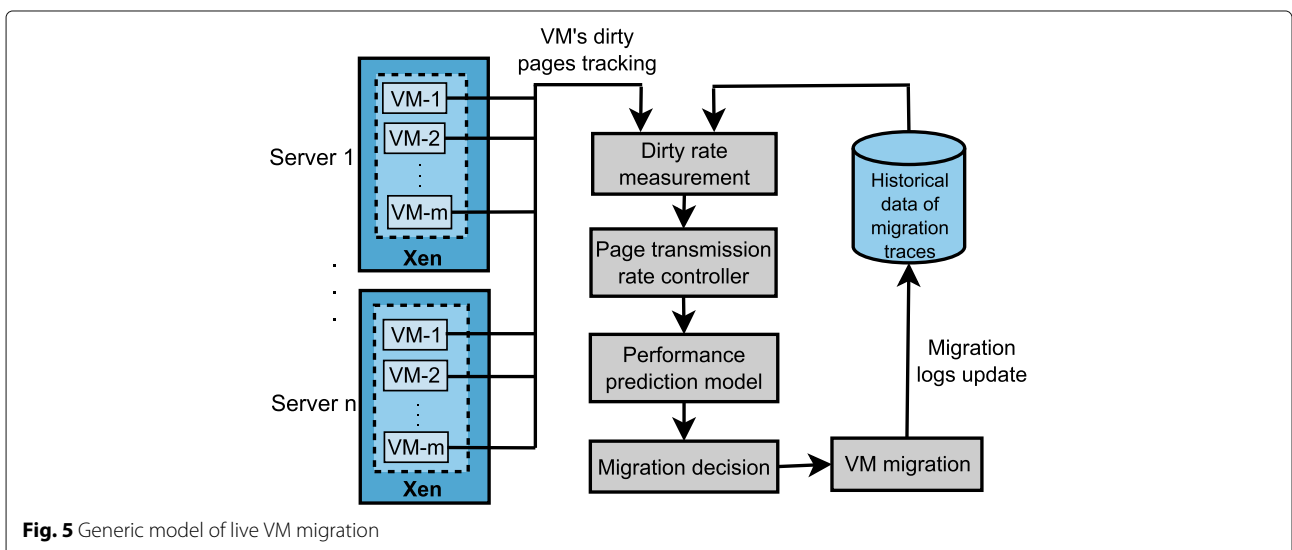
due to resource unavailability during VM state transfer duration. Huang et al. [71], Kikuchi et al. [51], Wu and Zhao [72], Liu et al. [69], Cerroni, and Callegati [74], Xu et al. [52], Deshpande and Keahey [82] proposed a performance aware models whereas Deshpande et al. [75] highlight the eviction time issue, Zhang et al. [81] highlight memory and storage transfer over WAN network issue. Huang et al. [71] proposed a live migration benchmark – Virt-LM solution. Virt-LM benchmark is used to compare live migration performance on different CDC environments among different software and hardware environments. Kikuchi et al. [51] design a PRISM performance model for parallel live migrations. Wu and Zhao [72] proposed performance model is used to predict migration latency and able to generate appropriate resource management decisions. Liu et al. [69] investigate design methodologies to quantitatively predict the energy cost and migration performance. The migration performance metrics handle several factors like workload characteristic, VM memory size, memory dirty rate, and network transmission rate. Cerroni, and Callegati [74] proposed a model for cloud-based edge network to evaluate the performance of live VM migration for co-operating VM's that implemented a user's profile. Xu et al. [52] proposed a lightweight Interference-Aware (iAware) live VM migration solution. It empirically captures the important relationship between performance interference of VM and major factors which are easily accessible in the real environment. Deshpande and Keahey [82] proposed a traffic-sensitive live VM migration model to reduce network contention of migration traffic and VM traffic. Deshpande et al. [75] proposed eviction time metrics determines how fast the source server goes into the offline mode or the freed resources for further availability to other VM's. Zhang et al. [81] develop a probabilistic prediction model and profiling framework to adaptively find storage and memory fractions to migrate over WAN network.

#### Generic model of live virtual machine migration

The generic model of live VM migration is shown in Fig. 5. It includes different steps that are required while taking migration decision. Due to the need of load balancing and server consolidation, migration of some or all VM's from servers is required to migrate. For this, we should select the most appropriate VM or set of VM's, which meet the migration objective or selection criteria. For this, we first measure the each VM memory dirty rate from current and historical page access pattern. Then, controller adjusts the memory page transmission rate to adapt the dirty rate. After this, performance prediction model estimates the performance of VM based on performance metrics like migration time, migration cost, down time, amount of data transfer, etc. Finally, migration decision is taken based on performance metrics to decide which VM/VM's

**Table 3** Comparison of Live VM migration models

Approach	Objective	Technique	Performance metrics	Hypervisor /Simulator
Huang et al. [71]-2011	Select the software and hardware environments that gives the best live migration performance	Virt-LM, compare live migration performance on different software and hardware stacks	Compare the migration performance on different CDC environments	Xen 3.3 and KVM-84
Kikuchi et al. [51]-2011	Construct a performance model of concurrent live migrations	The performance characteristics of live migration represent using PRISM model	Verify quantitative properties regarding live migration performance (Operation orchestration and Resource provisioning)	XenServer 5.6
Wu and Zhao [72]-2011	Predict migration latency	Build the performance model using statistical methods such as regression	Availability of resources have an impact on migration latency	Xen 3.2.1
Liu et al. [69]-2011	Quantitatively predict the energy cost and migration performance	Design a high-level linear model to estimate the migration energy	More than 90% prediction accuracy in measured cost, reduce the migration cost by more than 72.9% at an energy saving of 73.6%	Xen 3.4.1
Cerroni, and Callegati [74]-2014	Proposed a model for cloud-based edge network	Considered sequential and parallel migration strategies scheduling alternatives	Optimize service downtime, total migration time and network bandwidth	Mathematical modeling
Deshpande et al. [75]-2014	Determines how fast the source server goes into offline mode	Find eviction time by Scatter-Gather method	Reduces the VM eviction time and maintain total migration time against pre-copy and post-copy	KVM/ QEMU 1.6.50
Xu et al. [52]-2014	Focus on the cost and performance interference while handling VM migration	Design and implement of iAware, to avoid violations of performance SLAs	Load balancing and power saving can achieved without affecting application performance	Xen
Zhang et al. [81]-2014	Memory and storage migration over WAN	Propose a flexible and adaptive migration framework	Achieves better adaptiveness for various applications over a WAN	Xen 4.1.2
Deshpande and Keahey [82]-2017	Reduction in network contention of migration traffic and VM traffic	Presented a traffic-sensitive approach for migration of co-located VM's	Reduces application degradation and total migration time	KVM/ QEMU



**Fig. 5** Generic model of live VM migration

need to migrated. The historical data of VM's is updated for further migration (if require).

**Live virtual machine migration frameworks**

The existing live VM migration frameworks are discussed and compared in this section. The term "framework" is used for practical implemented techniques. The frameworks involved the proposed techniques and their implementation.

**Classification of migration mechanism**

Classification of VM migration mechanism is shown in Fig. 6 and it is based on the similarity of the migration strategy, used by the authors. We broadly divide it into three categories based on the objective and techniques used by researchers. They are classified based on type of migration, duplication based migration and context aware based migration as described in following sub-sections. We also illustrate the same using generic model for each of three categories.

**Type of migration**

Whenever the migration is performed, either single or multiple VM's are migrated based on the workload and environment conditions.

1. Single VM migration: Only one VM is migrated at a time.

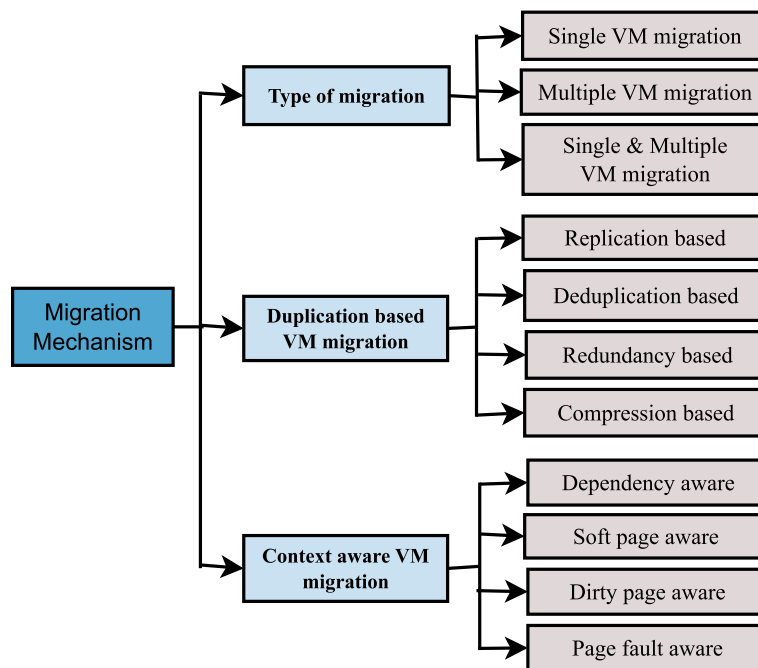
2. Multiple VM migration: Two or more VM's are migrated simultaneously.
3. Single & Multiple VM migration: One or more VM's are migrated simultaneously.

In the following sub-sections, we categorize the existing works in two major types multiple VM migrations and single & multiple VM migrations. We also describe the generic situation from the type of VM migration.

**Multiple VM migration:**

A group of co-located VM's are migrated simultaneously. It allows flexible movement of the bulk of workloads across different CDCs with minimal service disruption. Comparison of Multiple VM migration shown in Table 4.

In modern CDC, live migration technique is widely used. Multiple VM Migration becomes very complex due to many reasons like insufficient resources at destination server and concurrent migration of VM. Ye et al. [83] present a framework based on resource reservation. The reserved resource at source server includes CPU cycles and memory capacity while at destination server it includes all type of resource that is required for hosting a VM. Evaluation is performed on two Dell OPTIPLEX 755 physical machines (Intel Core2 Quad CPU 2.4GHz, 2GB RAM), Ubuntu 8.10 OS (kernel version 2.6.27) in Dom0 and Xen 3.3.1 hypervisor is installed. The VM (1-vCPU and 512GB RAM) images are stored in the Network File



**Fig. 6** Classification of migration mechanism

**Table 4** Comparison of Multiple VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor / Simulator
Ye et al. [83]-2011	Focus on different resource reservation method	Multiple VM migration with resource reservation & parallel migration strategy and workload-aware migration strategy	Downtime, total migration time, and workload performance overheads	Xen 3.3.1
Deshpande et al. [84]-2011	Live gang migration	De-duplication based approach to perform concurrent live migration of co-located VM's	Reduce the total migration time and network traffic overhead	KVM/ QEMU 0.12.3
Lu et al. [85]-2014	Reduce the latency over low network bandwidth and WAN network	Investigate the usefulness of two classic algorithms, min-cut and k-means clustering, in determining which VM's should be co-migrated	Improve both total migration time and network traffic	QEMU/ KVM 2.6.32
Lu et al. [86]-2015	Optimal scheduling of multi-tier VM	vHaul control multi-VM migrations to figure out the optimal scheduling	Minimize service downtime by 70% and improve application throughput by 52%	Xen 4.1.2
Forsman et al. [88]-2015	Balance the load	Present two strategies (push and pull) to balance the load in a system with multiple VM's through automated live migration	Achieve a load-balanced system in 4-15 minute	OMNeT ++ v4.3
Sun et al. [90]-2016	Focus on multiple VM migration problem	Proposed serial migration strategy, mixed migration strategy and develop queuing models	Analysis performance metrics like average waiting time, blocking ratio, average waiting queue length, and average queue length of each migration request	Xen and KVM

System (NFS). Three performance metrics namely total migration time, service downtime, workload performance overheads, are considered to measure the efficiency.

The problem of live gang migration (a group of co-located VM's are migrated simultaneously) is addressed by Deshpande et al. [84]. Authors present the design, implementation, and evaluation of a de-duplication approach (at the page level and sub-page level) for concurrent VM's migration. For detail, proof-of-concept prototype de-duplication strategies and a differential compression technique are implemented to exploit content similarity across VM's. The identical memory pages of VM's are transferred only once during the migration process. They implemented it by modifying an existing single-VM pre-copy migration in the QEMU/KVM environment. Offline implementation of de-duplication based gang migration is processed using Linux 2.6.32 OS and QEMU/KVM-0.12.3 hypervisor at both source and destination machines. Their approach achieves a considerable improvement in both network traffic and total migration time.

If VM's collaborating on a module of application are segregated in geographically distributed clouds, then the inter-cloud communication latency and low network bandwidth over WAN network will considerably degrade the system performance. The solution of such problems is to migrate all of the VM's of a module in a concurrent manner, that eliminates WAN communication latency. But, if the module is large, then it is not easier to simultaneously migrate all the VM's due to limited

bandwidth and higher dirty rate. Lu et al. [85] proposed a migration optimization approach called "Cliques migration". This approach divides a large group of VM's into sub-groups based on the VM's inter-dependency and shuffling mechanism is used to decide the order in which VM's sub-group should be migrated. They proposed R-Min-Cut and Kmeans-SF algorithm as a solution of their research problem. R-Min-Cut algorithm is based on a greedy strategy that implies a chronological order. Whereas in k-means, the clustering is a static process, so the shuffling process is required. The optimized k-means algorithm with shuffling is called as Kmeans-SF algorithm. The experiments are performed on physical machines (Intel(R) Xeon(R) CPU E5-2630 2.30GHz, 64GB RAM, 500 GB hard drives), CentOS Linux kernel 2.6.32, QEMU/KVM hypervisor is installed and machines are connected via 1Gbps bandwidth. Results show that proposed algorithms can reduce inter-cloud data traffic (traffic trace of 68 VM's at IBM production cluster) by 25 to 60%, if the parallel migration is varied from 2 to 32 and also considerably reduce the period in which applications undergo performance degradation. Furthermore, the migration traffic and application traffic have considerable interference, and it is higher when memory dirty rate of migrating VM is higher and also running application is I/O-intensive.

The multi-tier application holds a set of inter-dependent VM's, the live migration of these VM's needs a careful scheduling, so they require multi-VM migrations instead of single VM migration. By observing different types of



multi-tier applications, Lu et al. [86] suggested that dedicated link at the data center, uses different migration approaches that diversely impact the application performance. This happens due to the inter-dependence among functional modules of a multi-tier application. They take observations on vHaul, which controls multi-VM migrations to figure out the optimal scheduling. For evaluating the migration scenario by choosing simple applications (client-server architecture applications) running on 2-VM's and complex multi-tier applications (Apache Olio, a web 2.0 benchmark [87]) running on 4-VM's. Evaluation is performed on physical machines (quad-core Intel Xeon CPU's 3.2GHz, 16GB RAM) Linux 3.2 OS in both Dom0 and VM's, Xen 4.1.2 hypervisor is installed. All the machines are connected via two separate Gigabit Ethernet links. Their results of the vHaul system indicate that it can suggest the optimal multi-VM live migration schedules. Also, their evaluation results show that migration schedule generated by vHaul system performed well than worst-case schedule by 52% in terms of application throughput. Furthermore, the optimal schedule considerably minimize downtime up to 70% during migration. Though the prototype of vHaul is built using pre-copy live migration technique on Xen hypervisor and it is portable to other hypervisors.

During the server overload conditions, there may be a chance of SLA violation, to resolve this issue VM migration is performed for balancing the active servers load. For this Forsman et al. [88] proposed push and pull algorithms to perform necessary VM migrations. The push phase is active when the host gets overloaded and migrates the optimum number of VM's. The pull phase is active when server gets under-loaded and hosting more VM's to achieve efficient utilization of resources. The authors discovered that both the strategies are a complement to each other, so each strategy is come out as "best" under different conditions. Evaluation of proposed algorithm is performed on OMNeT++v4.3 [89] simulator using a simulation testbed. The results show that adding or removing the number of VM's, "best" strategy can able to re-balance the system in 4-15 min.

VM migration can help in improving the resource utilization, QoS parameters whereas reducing the power consumption from providers perspective. In the literature, most researchers focus only on single VM migration using either post-copy or pre-copy migration, only a few focus on multiple VM migration problem. Sun et al. [90] proposed improved serial migration strategy and introduced post-copy migration into it. They proposed the M mixed migration strategy which uses the improved serial migration strategy with parallel migration. Also, authors have developed M/M/C/C and the M/M/C queuing models there are C service channels, and the system can service up to C customers. The proposed approaches also handle

the failure rate of the transmission network. Mainly the memory-intensive live migration technique either use pre-copy or post-copy which are already implemented in Xen and KVM hypervisors. The proposed improved serial migration strategy and m mixed migration strategy for multiple VM migration can be implemented using Xen or KVM. The queuing models are used for analyzing performance metrics like average waiting time, blocking ratio, average waiting queue length, and average queue length of each migration request.

Multiple VM Migration becomes very complex due to many reasons like insufficient resources at destination server and concurrent migration of VM Ye et al. [83] present a framework based on resource reservation. Further, the problem of live gang migration (a group of co-located VM's are migrated simultaneously) is addressed by Deshpande et al. [84]. Lu et al. [85] proposed a migration optimization approach called "Cliques migration" to address the Large module migration problem, it is not easier to simultaneously migrate all the VM's due to limited bandwidth and higher dirty rate. By observing different types of multi-tier applications, Lu et al. [86] suggested that dedicated link at data center, uses different migration approaches that diversely impact the application performance. This happens due to the inter-dependence among functional modules of a multi-tier application. During the server overload conditions, there may be a chance of SLA violation, to resolve this issue VM migration is performed for balancing the active servers load, for this Forsman et al. [88] proposed push and pull algorithms to perform necessary VM migrations. Sun et al. [90] proposed improved serial migration strategy and introduced post-copy migration into it. They proposed the M mixed migration strategy which uses the improved serial migration strategy with parallel migration. Hence all these approaches work for multiple VM migration issue and present sub-optimal solutions with different characteristics of VM and migration environment. Hence we conclude that all these works are based on multiple VM migration but use dissimilar approaches and techniques to propose optimal solutions.

#### **Single & multiple VM migrations:**

In some specific circumstances like bandwidth aware VM migration, both single and multiple VM migration approaches are needed. Comparison of Single & Multiple VM migration shown in Table 5.

Memory-Intensive applications performance is highly affected when migration is performed by pre-copy because the memory dirty rate is higher than the memory transfer rate. For such applications, post-copy VM migration pattern performs better. Shribman et al. [91] proposed an approach that considered VM migration over LAN links. Authors present a XOR Binary Zero Run Length Encoding (XBZRLE) and Least Recently Used

**Table 5** Comparison of Single & Multiple VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor /Simulator
Shribman et al. [91]-2013	Improve Memory-intensive applications performance	Use RDMA stack to reduce the latency of faulty memory pages and pre-paging approach to reduce VM memory size and fasten the VM migration process	Improved total migration time, optimized downtime, and application degradation time	KVM and QEMU
Cerroni [92]-2014	Evaluate the performance for inter cloud data center for cloud federation	Characterizing the VM group migration time for the sequential and parallel migration strategies and use Markovian model of Inter DC network	Parallel migration results in reduced service downtime compared to sequential migration	Mathematical modeling

(LRU) page recording that supports high dirty rate relative to available network bandwidth. This approach used the Remote Direct Memory Access (RDMA) stack to reduce the latency of faulty memory pages. It also uses the pre-paging approach to reducing VM memory size and fasten the VM migration process. Furthermore, to increase the application performance, Memory Management Unit (MMU) is linked to post-copy so the only threads waiting for faulty pages are paused while others can continue their execution. MMU can enable Linux strength of directly handles the faulty page at kernel space by swapping disk pages without any context switching at user-mode. For the implementation of pre-copy approach and post-copy approach; VM's (4-vCPU; 1 GB RAM) that are hosted on physical machine (2 cores, 8 GB RAM) connected with 1 Gbps Ethernet network. Limit the network bandwidth to only 30 MB/s or 240 Mbps for maintaining a high application-dirty-rate / network-transfer-rate ratio. In hybrid post-copy evaluation guest VM (2-vCPU; 4 GB of memory; 1 GB Google SAT working set; 1 Gbps Ethernet network between hosts). Introduce the modification on QEMU and on KVM hypervisor. The proposed approach considerably improved application performance like total migration time, optimized downtime, and application degradation time using optimization strategies.

To evaluate the performance for inter CDC for cloud federation, Cerroni [92] presented a model. They assumed that the network load is increased by a group of co-operating VM's live migration that continuously provides services to end-users. After characterizing the VM's into groups, calculate the migration time for both sequential and parallel migrations. An analytical model is proposed and it is a useful designing tool to dimension the inter-DC network capacity for achieving given performance level by assuming some simple multi-VM live migration strategies for implementation. Represent that sequential VM migration strategy has less detrimental effect on network performance, whereas parallel VM migration strategy has lower service downtime. This model can be used to represent the trade-off between service availability and inter-cloud data center network capacity. The obtained results give an interesting insight to the macroscopic

performance of a federated cloud network but some of the hypotheses used to derive a model may not be fully realistic.

Memory-Intensive applications performance is highly affected when migration is performed by pre-copy because the memory dirty rate is higher than the memory transfer rate. For such applications, post-copy VM migration pattern performs better. Shribman et al. [91] proposed an approach that considered VM migration over LAN links. Whereas to evaluate the performance for inter cloud federation, Cerroni [92] presented a model and assume that the network load is increased by a group of co-operating VM's live migration that continuously provides services to end-users.

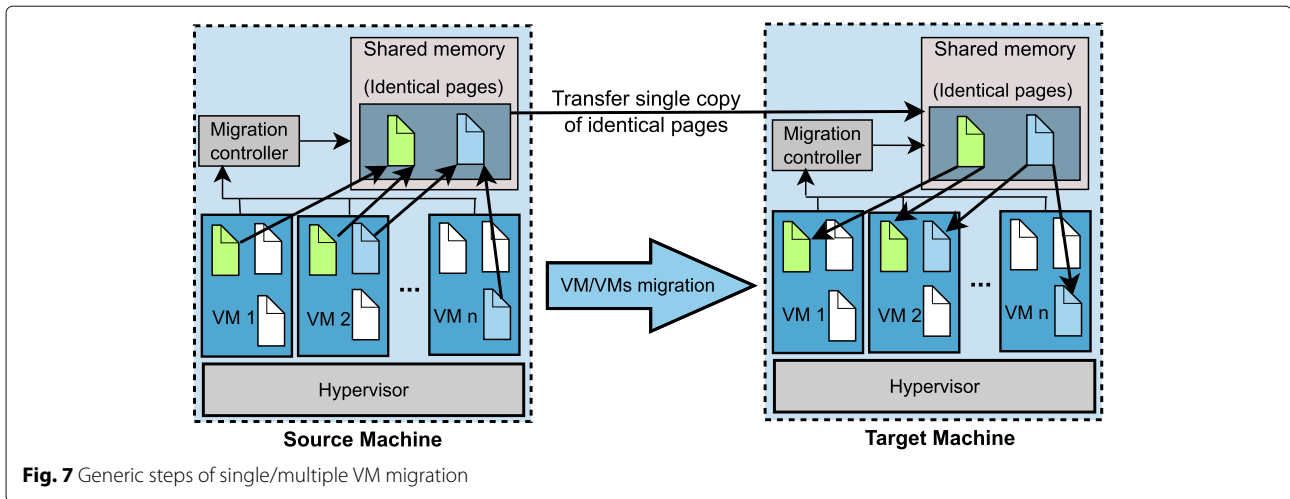
#### **Generic steps of Single/Multiple VM migration**

The key observation behind multiple VM migration is that VM's having the same OS, applications, or libraries can contain a considerable number of identical pages. So during the multiple VM migration track, all the identical pages across co-located VM's and transfer the single copy of these identical pages, at source server side this is done by migration controller. Hence, migration controller initiates the parallel migration of all co-located VM's to the target machine. At the target server, migration controller prepares the server for reception of incoming migrant VM's. Multiple VM migration steps are illustrated in Fig. 7.

#### **Duplication based VM migration**

During the process of migration, VMM detect multiple copies of the same page on single VM or Multiple VM's or on a number of different servers, that leads unnecessary memory pages migration. For handling a large number of pages during migration requiring more network bandwidth or increase network traffic. Different type of memory compression techniques is used.

1. Replication based
2. De-duplication based
3. Redundancy based
4. Compression based



**Fig. 7** Generic steps of single/multiple VM migration

In the following sub-section, we categorize the existing work into four categories like replication based, de-duplication based, redundancy based and compression based VM migration based on the memory pages replica, similarity, and volume.

**Replication based VM migration:**

The same memory page is spread on multiple servers for simultaneous computing and fault (storage and network) recovery. Comparison of existing techniques based on memory page replication is illustrated in Table 6.

During the migration process, either hot or cold migration technique is used that imply the movement of VM between corresponding servers consume server resources and network bandwidth consequently increases the cost. So to reduce such costs Celesti et al. [93] proposed a Composed Image Cloning (CIC) methodology and focuses on the dynamic VM allocation. Instead of considering VM as a single monolithic disk-block, they treated it as a “composable” blocks and “user data” blocks. They setup two different distributed (federated) clouds one is located at the University of Messina having Dual-Core AMD Opteron Processor 2218 HE with 8 GB of RAM servers and second is located in the same metropolitan area having same hardware configuration. On each

cloud, the cluster is composed of a number of servers having Linux OS Debian 5.0lenny on one cloud servers, Ubuntu 8.10 Intrepid Ibex on another cloud servers and KVM hypervisor is installed for virtualization. The CIC methodology able to improve the relocation cost of VM disk-image because data transferred significantly reduces when the number of live migrations increases over a large scale federated cloud.

When a large size VM is migrated over WAN networks with low bandwidth causes a complex live migration. Current techniques do not efficiently deal with such migrations where servers are part of different networks. There are various challenges such as migrating network and storage connections, migrating storage content and persistent state kept at the source server side. To minimize the migration latency’s Bose et al. [94] proposed a technique which combines VM replication with VM scheduling. They use the de-duplication method for finding the VM replicas to compensate the additional storage requirement generated by the increasing number of replicas of different VM’s images. Develop a CloudSpider architecture based on efficient replication strategies using VM replication and scheduling. The replica placement strategies evaluated on CloudSim simulator with physical machines (Intel Core2Duo CPU 2.53 GHz, 2 GB RAM). The proposed

**Table 6** Comparison of Replication based VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor /Simulator
Celesti et al. [93]-2010	Focus on the dynamic VM allocation	VM disk-image relocation, described using CIC methodology	Data transferred amount considerably reduces if the number of migrations increases	KVM
Kumar Bose et al. [94]-2011	Large size VM is migrated over WAN with low bandwidth	Combine VM replication with VM scheduling and use deduplication techniques	Minimize the migration latencies	CloudSim
Kumar Bose et al. [95]-2011	Migration of virtual disk file at run time within acceptable time over WAN	Combining VM scheduling with replication strategies	Minimize the migration time across WAN networks	CloudSim

architecture is capable to minimize migration latency's associated with the live migration of VM images over WAN network.

Live VM migration across high latency low bandwidth WAN within "reasonable" time is nearly impossible due to the large size of the VM image. So, migration of virtual disk file at run time within acceptable time over WAN is a critical challenge. Bose et al. [95] proposed a combined VM replication and scheduling architecture called CloudSpider. The VM image is replicated across different cloud sides is chosen a VM image replica based on dynamically changing cost parameter and treated as a primary copy. Further, the incremental changes in VM replica is propagated towards remaining replicas for synchronization. Authors proposed de-duplication techniques to compensate (by exploring commonalities) the additional storage cost due to additional storage requirements for replica storage. They mainly focus on the VM image replica placement when disparate VM images carried varying degrees of commonality and latency requirements. Modify open-source cloud simulator called CloudSim and incorporating modules on storage de-duplication, storage blocks, and file allocation table. Implementation shows that the success of CloudSpider to minimize storage requirements is highly depends on the working of replica placement algorithm and it can judiciously place the VM image replicas at different sites that minimize the storage requirement.

The movement of VM between corresponding servers consume server resources and network bandwidth consequently increases the cost. So to reduce such costs, Celesti et al. [93] proposed a Composed Image Cloning (CIC) methodology and focuses on the dynamic VM allocation. Bose et al. [94] proposed a technique to minimize the migration latencies by combining VM replication with VM scheduling. Migration of virtual disk file at run time within acceptable time over WAN is a critical challenge. Bose et al. [95] proposed a combined VM replication and scheduling architecture called CloudSpider. Hence all the

mentioned work is work on different challenges while using the replication technique.

#### **De-duplication based VM migration:**

Identify the similar memory pages on single VM and avoid transfer of such pages for improving the efficiency of bandwidth utilization. Comparison of existing work that used the de-duplication approach during migration is presented in Table 7.

Live migration is expensive to use because the large amount of data transfer when migrating the Virtual Clusters (VC). As VM's running on similar OS that indicates the large portion of storage carries identical data. To migrate the VC over WAN network Riteau et al. [37] proposed a VM migration approach called "Shrinker" that based on the de-duplication optimization model. It calls a service that record memory pages identified at source cluster before transferring them to the destination server. Hypervisor uses the service for fetching the status of memory pages before transferring them to the destination server and it transfers memory page identifier only when any of the VM transferred memory page. If the memory page has not been sent, then the hypervisor registers the memory identifier and transfers the page to the destination server. At destination side, distributed content addressing approach is used for transferring pages to the corresponding destination server(s). Also at the destination side, the index server keeps the record of IP address of legitimate source server(s) against memory page hash values, prior to transferring the memory pages to the destination server. The destination server registers the source server with respect to the page hash value at the index server when the required memory pages are received. Due to all the above process, total migration time and amount of data transferred of the proposed approach are reduced. Also, the process is managed by a centralized server that may be a single point of failure of the entire environment. The results are analyzed and performed on the Grid'5000 [96] testbed and implementation is performed

**Table 7** Comparison of De-duplication based VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor / Simulator
Riteau et al. [37]-2011	Migrate the VC over WAN network	Shrinker detects memory pages and disk blocks duplicated in a VC to avoid re-sending over WAN network	Reduce the network traffic	KVM 0.14.0
Deshpande et al. [34]-2012	Optimizing the performance of multiple VM's	Present a IRLM for distributed system	Reduce the data transfer amount and total migration time by up to 44% and 26% in online compression and by up to 17% and 7% in the case of gang migration	QEMU /KVM
Jo et al. [99]-2013	Reduce the total time	Detect duplication between memory pages and storage blocks, maintain an up-to-date map of duplicated pages	30% reduction of the total migration time and 60% reduction for certain benchmark workloads	Xen 4.1

on the KVM 0.14.0-rc0 hypervisor. Redis version 2.2.0-rc4 [97] is used to store key-value for indexing and coordinating the services. The results show that proposed work reduces both total data transferred and total migration time. Another similar technique is proposed by Zhang et al. [98], exploit VM self-similarity ratio and hashing-based finger print to identify and track identical memory pages.

Most of the existing techniques focus on either optimizing the migration performance of single VM or multiple VM's running on the same server to lessen the amount of data transferred among corresponding servers. Deshpande et al. [34] present an Inter-Rack Live Migration (IRLM), for optimizing the performance of multiple VM's migrations, i.e., concurrently migrating multiple VM's from one rack of the server to another rack. They employ de-duplication for improving the efficiency of bandwidth utilization through migration of multiple VM's. Simultaneous de-duplication identifies the similar memory pages using QEMU/KVM thread and transfer them only once by any one of the VM. During mass VM migration, it reduces the traffic load by a distributed replica of VM's memory. The implementation is performed on QEMU/KVM virtualization platform and evaluate it on a cluster testbed (13 physical servers (two Quad core 2GHz CPU's, 16GB RAM, and 1Gbps Network card) connected by Gigabit Ethernet connection. The primary work is performed on 6 servers per rack and 4 VM's per server, IRLM can reduce the amount of data transferred over the core links during migration by up to 44% (and total migration time is reduced by up to 26%) with respect to online compression and by up to 17% (and total migration time is increased by 7%) compared to gang migration. But the proposed framework is computationally expensive and complex because of huge calculations Incorporated like calculation of 160 bit hash value. So the acceptance of proposed framework is limited to servers, that hosting identical VM's or workloads. In contrast, another work Deshpande and Keahey [82] used both pre-copy and post-copy VM migration for lessening the mutual adverse effects of migration traffic and VM application traffic.

Live migration of VM's at distributed servers is important for maintenance, load-balancing, and energy

reduction from the providers and CDC operator perspective. Jo et al. [99] present a technique to reduce the total migration time while keeping the minimum downtime by tracking the VM's I/O operations with NAS device and maintaining an updated memory pages mapping. Under the process of iterative pre-copy migration the memory-to-disk mapping is sent to the destination server, then directly fetch the required pages from NAS device and consistency is maintained by keeping a version number of each transferred page. By running the number of benchmark workloads on a Linux HVM guest (contain Xen 4.1 virtualization environment), the 30% reduction of the total migration time and 60% reduction for certain benchmark workloads.

As VM's running on similar OS that indicates the large portion of storage carries identical data. To migrate the VC over WAN network Riteau et al. [37] proposed a VM migration approach called "Shrinker" that based on the de-duplication optimization model. Deshpande et al. [34] present an IRLM, for optimizing the performance of multiple VM's migration, i.e., concurrently migrating multiple VM's from one rack of the server to another rack. Jo et al. [99] present a technique to reduce the total migration time while keeping the minimum downtime by tracking the VM's I/O operations with NAS device and maintaining an updated memory pages mapping.

**Redundancy based VM migration:**

Having identical memory blocks belonging to different VM's on the same host or large blocks consisting of zero bytes entries. The avoidance of transferring redundant pages leads to reducing power consumption, load and cost of live VM migration. Comparison of existing Redundancy based VM migration approaches is presented in Table 8.

At the WAN level, VM migration transforms the scope of resource provisioning from single to multiple data centers. Wood et al. [100] proposed a CloudNet framework to achieve the live migration and flexible placement of VM's and data over a seamlessly connected resource pool (provided by different CDC). It provides an optimized support for live migration over WAN and beneficial over low bandwidth network links. Authors try to reduce the volume of data transfer over the WAN by avoiding redundant

**Table 8** Comparison of Redundancy based VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor / Simulator
Wood et al. [100]-2015	Flexible placement and live migration of applications	CloudNet framework	Memory migration time reduced by 65%, memory transfer saved 20GB of bandwidth for storage, it leads an overheads reduction by less than 20%	Xen
Jaswal and Kaur [101]-2016	Reduce the downtime and cost	Concept of distance and redundancy elimination mechanism	Reducing power consumption, load and cost of live VM migration	CloudSim

memory pages. If the redundant pages are encountered only a hash is transferred to destination server then it performs lookup operation of the redundant page on the previously received memory page. The advantage of using hash in place of compression is lower overheads. Also, the cost of transferring VM memory and storage contents during migrations can be minimized. For implementing a prototype of CloudNet, Xen virtualized environment, Distributed Replicated Block Device (DRBD) protocol, and commercial router based VPLS/layer-2 VPN. There results show that memory migration time reduced by 65%, memory transfer saved 20GB of bandwidth for storage, this improvements leads an overheads reduction by less than 20%.

In existing system distance based load, consideration is absent where as the proposed system is based upon it. Jaswal and Kaur [101] proposed a technique for offloading the data of a VM to multiple data centers. They used the concept of distance and redundancy elimination mechanism has been used. It is an enhanced hybrid approach which is used for reducing power consumption, load and cost of live VM migration. This proposed system combines the reliability of both pre-copy approach and post-copy approach. The Proposed scheme shows efficient results when compared with the existing techniques. In the proposed technique, migration will be performed by the use of live VM migration which means that the migration does not require the switching off the devices, which is the case in offline migration. The implementation is performed using Cloudsim simulator with 2 VM's of 4 physical machines ( Intel(R)Core(TM) i3 CPU M330 @2.13GHz, 3.00 GB, 64-bit OS). The comparison is based on load among Pre-copy (500 Hz per 1 GB of Data), Post-copy (550 Hz per 1 GB of Data), Hybrid (425 Hz per 1 GB of Data) and Proposed algorithm (201 Hz per 1 GB of Data). Power consumption is also reduced in the proposed system from 180w to 100w.

Wood et al. [100] proposed a CloudNet framework to achieve the live migration and flexible placement of VM's and data over a seamlessly connected resource pool (provided by different CDC). Jaswal and Kaur [101] proposed a technique for offloading the data of a VM to multiple data centers. They used the concept of distance and redundancy elimination mechanism has been used.

#### **Compression based VM migration:**

Memory compression leads to reducing the data transfer amount during migration process. Using the compression, cost of transferring VM memory, storage contents during migration, and service downtime get reduced. Comparison of existing Compression based VM migration approaches is presented in Table 9.

Number of research work is carried out for improving the live VM migration with respect to the data transfer

amount among corresponding servers. Jin et al. [102] provide a Memory Compression (MECOM) based solution to reduce the migration time. The memory is compressed and sent over the network using Xen's pre-copy and stop-and-copy phases. MECOM approach provides fast, stable VM migration, while slightly affecting the VM performance. An adaptive zero-aware compression algorithm (use the memory page characteristics) is designed for balancing the cost and performance of VM migration. The memory pages are compressed in batches on the source side and recovered at the destination in same order. The results show that inherent redundancy in memory areas (like identical memory blocks belonging to different VM's on the same host or large blocks consisting of zero byte entries) to get high compression ratios. The experiment is conducted on several identical servers (2-way quad-core Xeon E5405 CPU's 2GHz, 8GB DDR RAM) and Redhat Enterprise Linux 5.0 at the host OS and guest OS. Compared with Xen 3.1.0, expanded Xen 3.1.0 can reduce downtime by 27.1%, total transferred data by 68.8% and total migration time by 32% on average. Therefore, a VM that carry large memory size may contain more identical pages than a VM with smaller memory size. They further expand their work in Jin et al. [103] and present a VM migration approach based on MECOM approach. To provide live migration for para-virtualized VM's, they used MECOM approach. In this approach, VM services may be a little bit affected based on the characteristics of memory pages. For balancing the performance and cost of VM migration, authors proposed an adaptive zero-aware compression algorithm, in which pages are more quickly compressed in batches on the source server and recover at the destination server. Hence the intent of this approach to implement live migration of VM's including the local persistent state. The experiment is conducted on Xen 3.1.0 virtualized environment that is deployed on several identical servers (2-way quad-core Xeon E5405 CPU's 2GHz, 8GB DDR RAM) and Redhat Enterprise Linux 5.0 at the host and guest OS. Authors compared their proposed approach and algorithm expanded Xen 3.1.0 hypervisor, and comes out with total migration time, downtime, transferred data have reduced by 32%, 27.1%, and 68.8% respectively. But due to low bandwidth availability at WAN, it is still a challenge to fast migrate VM's using MECOM approach because VM's having huge amount of memory and disk data.

The large scale application systems like Systems, Applications and Products (SAP) in Data Processing for Enterprise Resource Planning (ERP) consume a large amount of memory which results in limiting the VM migration. For this Hacking and Hudzia [104] present a system that supports transparent migration of large scale applications without severely affecting their live services. They used

**Table 9** Comparison of Compression based VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor /Simulator
Jin et al. [102]-2009	MECOM	Adaptive zero-aware compression algorithm	Reduce total migration time, downtime and total transferred data on average by 32%, 27.1%, 68.8% respectively	Xen3.1.0
Hacking and Hudzia [104]-2009	Transparent migration of large scale applications	VM to use using delta compression algorithm for memory transfer as well as the introduction of an adaptive warm up phase	Increased data transfer performance and reduced service downtime	KVM
Svärd et al. [36]-2011	Optimizing the network bandwidth utilization	Delta compression techniques	Performance is improved by reducing the page dirty rate or through increased network throughput	QEMU/ KVM 0.11.5
Svärd et al. [105]-2011	Optimize total migration time and service downtime through improved network performance	Transfer the delta compressed pages with weight based priority	Reduces total downtime, migration time and increase migration throughput	KVM 0.13.0
Sahni and Varma [106]-2012	Reduce the number of network page faults	Present a hybrid approach	Minimize transferred data and total migration time	KVM/QEMU
Hu et al. [28]-2013	Improving the VM memory transfer rate	Hybrid memory copy and delta compression mechanism	Increase downtime from 3s to 3.8s w.r.t. pre-copy and decrease total migration time from 235s to 95s w.r.t. pre-copy for ERP system	Xen 3.4.4
Jin et al. [103]-2014	MECOM: Live migration of VM by adaptively compressing memory pages	Adaptive zero-aware compression algorithm	Reduce transfer data, total migration time, and downtime reduce total migration time, downtime and total transferred data on average by 32%, 27.1%, 68.8% respectively	Xen 3.1.0

the delta compression approach for data compression that leads to reducing the data transfer amount during migration and also added an adaptive warm-up data transfer phase. The experiment performed using two identical servers (HP ProLiant DL 580, 4x 3Hz Dual core Xeon, 32GB RAM, Debian 4.0 64 bit) and use the KVM hypervisor. Results show that data transfer is increased and service downtime is reduced without introducing additional service disruption and performance overhead compared to current live migration.

Another work that attempts to improve overall network performance is done by Svärd et al. [36]. Authors used the delta-based compression method in order to increase migration throughput and reduced service downtime. They proposed a binary XOR-Based Run Length Encoding (XBRLE) delta compression method for improving migration performance. The Run Length Encoding (RLE) compression approach combines compressed delta pages for optimizing the network bandwidth utilization. The reverse process is applied at destination server and VM memory pages are fetched using decompression method. The modification is done to the KVM hypervisor. They show that whenever the VM's migrated with high workloads or low-speed connectivity then there is a high risk of service dis-connectivity. The data is recorded in the order

of changes with versions. So, performance is improved by reducing the page dirty rate or through increased network throughput. Also, compression/decompression of VM memory pages consumes extra resources. The tests performed on two physical machines (Intel 2.66 GHz core2quad, 16 GB RAM, Ubuntu 9 OS, QEMU/KVM 0.11.5 Virtualization environment) is used in the evaluation. The evaluation shows that XBRLE compression is beneficial with a highly compressible working set or over slow networks (i.e., WANs) or running heavy workloads with large working sets on the VM's.

By migrating CPU and/or memory intensive VM's two problems occur: one is extended migration downtime that may result in VM failure or service interruption, and second is prolonged total migration time that is harmful to the overall system performance because considerable network resources allocated to complete the VM migration. In long distance migration, these problems become more severe if the available network capacity is low. Another work based on RLE compression and dynamic reordering proposed by Svärd et al. [105], where the authors optimize total migration time and service downtime through improved network performance. The proposed VM migration techniques dynamically reorder the memory pages when migration is under a process that

reduces the re-transmission of likelihood page. Authors assign a page weight to memory page based on the frequency of page has been modified during migration process and transfer these page according to page weight. Consequently, under the migration process, lower weight memory pages get higher priority or sent earlier than higher weight. As a result, instead of transferring the memory pages at equal priority, this approach transfer the delta compressed pages with weight based priority approach that reduces total downtime, migration time and increase migration throughput. The implementation is performed on two machines (3.06 GHz HP G6950, 8 GB RAM, KVM 0.13.0). The critical issue with this approach is that requires large cache memory and more CPU cycles for compressing memory pages.

Hybrid VM migration technique is proposed by Sahni and Varma [106], exploits a hybrid technique which is a combined method of pre-copy and post-copy migration over Ethernet links. In this technique VM migration technique, is worked in three phases such as preparation phase, downtime phase, and resume phase. In the preparation phase, “access bit scanning” method is used to identify the working set of VM (frequently access memory) and introduce flags in the page table that indicates the frequently accessed pages. In the next phase, to resume VM at destination server the CPU register’s status along with working set is migrated. After that, to reduce the network page faults, hypervisor actively pushes the VM memory pages from the source server. Also, the adaptive pre-paging approach is optimized by increasing the search space against faulted pages. The implementation of the prototype in KVM/QEMU. Moreover, Lempel–Ziv–Oberhumer (LZO) compression technique [107] is used for compression of memory pages before memory page transfer. Their proposed technique significantly improved application total migration time, service downtime, and the amount of total data transfer. Therefore, applying compression/decompression method consumes considerable system resources [102].

The Hybrid Memory Data Copy (HMDC) approach proposed by Hu et al. [28], based on delta compression. HMDC approach has used active-push and on-demand paging approaches for improving the VM memory transfer rate. For reducing the network-bounded page faults, optimization methods is used that leads to improving the application performance. In the first phase of VM migration (pre-copy migration phase) process, HMDC pushes the VM memory pages parallel to the dirty pages in an iterative manner. In the next phase, the bitmap list of dirty pages is transferred to the destination server for synchronization of VM’s. In the last phase, the resumed VM access the dirty pages based on the bitmap list. To utilize the network resources at source server, RLE-based delta compression used by HMDC. Nevertheless, migration is

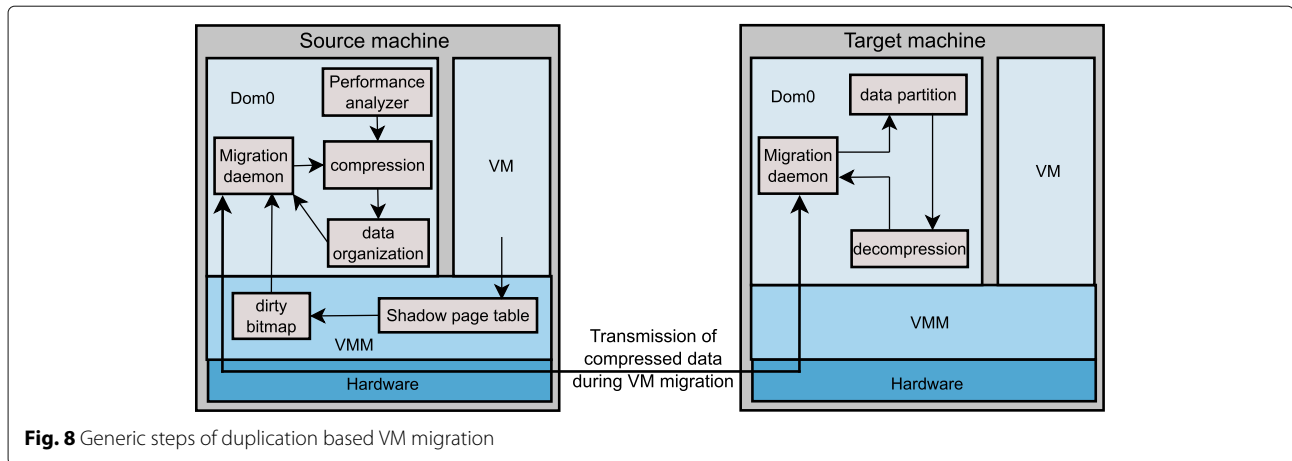
a resource-intensive process, delta compression approach affects the performance of co-hosted application because of high resource sharing. Also, HMDC may face the system crash due to a power outage during migration phase, so it is not robust migration scheme. ERP test case is performed on two machines (Intel 3GHz 4xDual Core Xeon, 32 GB RAM, Ubuntu 10.4 OS, Linux kernel 2.6.32-24, Xen 3.4.4) and 2-way set-associative cache size is 1 GB. Other test cases are performed on two machines (Intel 2.66 GHz core2quad, 16 GB RAM, Ubuntu 9 OS, Linux kernel 2.6.32-24, Xen 3.4.4) and 2-way set-associative cache size is 512 MB. Implementation results show that HMDC evidently reduces VM downtime, total migration time, and total migration data compared to XBRLE and Pre-copy approach.

Jin et al. [102] provide a MECOM based solution to reduce the migration time. MECOM approach provides fast, stable VM migration, while slightly affecting the VM performance. Whereas the large scale application systems like SAP in Data Processing for ERP consume a large amount of memory which results in limiting the VM migration. For this Hacking and Hudzia [104] present a system that supports transparent migration of large scale applications without severely affecting their live services. Another work that attempts to improve overall network performance is done by Svård et al. [36]. Authors used the delta-based compression method in order to increase migration throughput and reduced service downtime. Another work based on RLE compression and dynamic reordering proposed by Svård et al. [105], where the authors optimize total migration time and service downtime through improved network performance. Hybrid VM migration technique is proposed by Sahni and Varma [106], exploits a hybrid technique which is a combined method of pre-copy and post-copy migration over Ethernet links. The HMDC approach proposed by Hu et al. [28], based on delta compression. HMDC approach has used active-push and on-demand paging approaches for improving the VM memory transfer rate. Jin et al. [103] present a VM migration approach based on MECOM approach. To provide live migration for paravirtualized VM’s, they used MECOM approach. Hence all the above-mentioned work uses the compression technique to achieve different performance metrics.

#### **Generic steps of duplication based VM migration**

The steps of duplication based VM migration is illustrated in Fig. 8. Migration daemons or controller running in the Domain0 is responsible for performing migration of running VM’s. In pre-copy phase, migration controller accesses the Shadow page tables existing in the hypervisor layer to trace modified pages in migrated VM’s during the pre-copy phase. shadow paging can also be used to trap access to non-existent pages at the target VM. The





**Fig. 8** Generic steps of duplication based VM migration

shadow page table entries reflect the changes in the dirty bitmap. At the initial of each pre-copy round, migration daemon sends the bitmap first. After that, the is cleared and destroyed. Later, the bitmap and shadow page tables are created for next round. Migration daemon selects the pages for migration based on dirty bitmap entries and compression is performed to reduce network overheads. At the target machine migration, daemon accepts the compressed data and decompression technique is applied to access actual pages. Further, the VM memory page mapping is performed for migrated VM by the migration daemon.

**Context aware VM migration**

The migration decision of some of the memory pages depends on the content of pages. In the following subsection, we categorize the existing work into four categories like -

1. Dependency aware VM migration
2. Soft page aware VM migration
3. Dirty page aware VM migration
4. Page fault aware VM migration

The categories are based on the inter-dependency among single or multi-VM pages, zero content memory pages, the frequency of page dirty and network/page fault aware.

**Dependency aware VM migration:**

The inter-dependency information is used to find out direct or indirect external-dependencies among processes during live VM migration. By transferring the dependent VM in groups will reduce the network traffic. Comparison of existing Dependency aware VM migration approaches are presented in Table 10.

The demand for live migration increases when resources are most scarce. So it is important that live migration process be as fast and efficient; and provide dynamic load balancing, automatic failover, and zero-downtime scheduled maintenance during unscheduled downtime. A dependency-aware live migration approach is proposed by Nocentino et al. [108] and investigates its ability to reduce migration latency and overheads. It can lessen live migration overhead over non-live migration. The proposed approach used a tainting mechanism that was basically developed for an intrusion detection mechanism. The inter-dependency information is used to find out direct or indirect external-dependencies among processes. The development and test environment consists of two Dell PowerEdge 1900 servers, each with two quad core Intel Xeon series 5355 2.66 GHz processors, 4GB of primary memory and a system bus speed of 1333 MHz. Both servers are configured with Xen 3.3.0 and use 32 bit Ubuntu 8.0.4 LTS running an SMP kernel (2.6.18.8) for the server OS. The guest OS is para-virtualized 32 bit Ubuntu

**Table 10** Comparison of Dependency aware VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor /Simulator
Nocentino et al. [108]-2009	Dependency-aware live migration	Presented a novel model for quick live migration of full virtual machines using dependency-aware selective migration of processes	Lessen live migration overhead over non-live migration	Xen 3.3.0
Babu and Savithramma [109]-2016	Optimize pre-copy migration	Novel idea for pre-copy live VM migration process by sending independent instructions to the destination server and start execution	Attain minimal downtime and low application performance impact	KVM 0.14.0

8.0.4 LTS with Linux kernel 2.6.18.8, the VM has 2GB of main memory and 10GB hard disk. The outcomes show that migrating VM process can be considerably streamlined by selectively applying efficient protocol which does not contains external-dependencies.

Babu and Savithamma [109] proposed an idea for pre-copy migration of VM processes and also analyzed process-level performance during migration. They find independent instruction sets at source server and transfer them to destination server to resume VM without waiting for source server to transmit all the instructions. Authors present a novel algorithm that tracks memory update pattern and stop migration process when improvements in downtime are unlikely to occur. The implementation results show that it is beneficial for both Ethernet and RDMA/(InfiniBand) migration. Their work is performed on KVM 0.14.0 hypervisor and was able to minimize downtime and low impact on application performance.

A dependency-aware live migration approach is proposed by Nocentino et al. [108] and investigates its ability to reduce migration latency and overheads. Whereas Babu and Savithamma [109] proposed an idea for pre-copy migration of VM processes and also analyzed process-level performance during migration.

#### **Soft page aware VM migration:**

Soft pages include free pages and kernel status objects, which are already available on the destination server. By avoiding the transfer of such pages leads to decreasing the total migration time without influencing the hosted applications. Comparison of existing Soft page aware VM migration approaches is presented in Table 11.

Post-copy approach provides a “win-win” by reducing total migration time while maintaining the liveness of VM during the migration process. During the post-copy VM migration, application performance considerably degrades by network page faults, Hines et al. [63],

proposed an optimized post-copy VM migration approaches such as demand paging, active push, pre-paging, and DSB to avoid network fault problem. Demand paging approach transfers the memory page over a network, only when the destination server request for that page. Active push approach pro-actively transfers pages to the destination server based on temporal locality heuristic. Pre-paging pre-fetches the pages at destination server based on VM page request pattern or working-set of pages. This approach significantly reduces the page fault through pre-fetching of VM pages to be accessed in the future. DSB reduces the network load by reducing the page transfer rate. Using Ballooning approach, VM periodically releasing the free memory pages and hand over these memory pages back to the hypervisor. Implementation of post-copy along with all of the optimization's on para-virtualized Linux 2.6.18.8 and Xen 3.2.1 virtualized environment. Post-copy improves several metrics including total migration time, pages transferred, and network overheads. Nevertheless, the performance of the proposed approaches is based on the accuracy of prediction heuristic like spatial locality. The extensive comparison of pre-copy and post-copy migration is performed by Hines and Gopalan [64] on the Xen hypervisor using different workloads. They used post-copy with an adaptive pre-paging approach to avoid re-transmission of duplicate pages. Implementation of post-copy along with all of the optimization's on para-virtualized Linux 2.6.18.8 and Xen 3.2.1 virtualized environment. Results show that network-bound page faults reduced up to 21% of the VM's for large scale workloads and the transmission of free memory pages using DSB mechanism are also avoided.

Migration noise (resource consumption due to migration overhead at both source and destination servers) makes the live migration process difficult to handle unpredictable increases in workload due to flash crowds and also lower the total throughput of data centers. Sonic

**Table 11** Comparison of Soft page aware VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor / Simulator
Hines et al. [63]-2009	Mitigate the network faults	Propose different pre-paging strategies and eliminate the transfer of free memory pages in both pre-copy and post-copy	Reducing the number of pages transferred and total migration time	Xen 3.2.1
Hines and Gopalan [64]-2009	Compare post-copy and pre-copy	Adaptive pre-paging and DSB mechanism	Network-bound page faults reduced up to 21% and transmission of free memory pages using DSB mechanism was also avoided	Xen 3.2.1
Koto et al. [110]-2012	Tracks and avoids the transfer of soft pages	Sonic Migration examines memory pages when live migration is triggered, and marks soft pages to avoid transferring them	Migration time is up to 68.3% shorter than that for Xen-based live migration and reduces network traffic by up to 83.9%	Xen 4.1.0

migration approach proposed by Koto et al. [110], track and avoid the transfer of soft pages (free pages and kernel status objects) during the VM migration process. Before triggering a VM migration, guest kernel informs the address of the soft pages to the hypervisor. Sonic migration creates a shared memory between hypervisor and guest kernel so that they can communicate with little intervention of CPU resource. Hypervisor generates a signal for VM to update the shared memory before initiating the stop-and-copy phase. After that, VM sends a hyper call to the hypervisor that triggers the stop-and-copy phase. The proposed approach decreasing the total migration time without influence hosted applications. Implementation is performed on Xen 4.1.0 hypervisor and Linux 2.6.38 OS show the migration time with the proposed prototype is up to 68.3% shorter than that the Xen based live migration and network traffic is reduced by up to 83.9%. However, it generates extra overhead on memory and CPU resources, that affect of the applications.

During the post-copy VM migration, application performance considerably degrades by network page faults, Hines et al. [63], proposed an optimized post-copy VM migration approaches such as demand paging, active push, pre-paging, and DSB to avoid network fault problem. On the other hand, Sonic migration approach is proposed by Koto et al. [110], that track and avoid the transfer of soft pages (free pages and kernel status objects) during the VM migration process.

#### **Dirty page aware VM migration:**

During the migration process, some of the memory pages are continuously updated by running VM. These dirty pages are resent to the destination host in a future iteration, hence some of the frequent access memory pages are sent several times. It causes long migration time. So avoiding the retransmission of frequently accessed pages leads to reduced total migration time and amount of memory transfer. Comparison of existing dirty page aware VM migration approaches is presented in Table 12.

VM memory pages are dirtied at a specific rate, called the dirtying rate, while a VM is running. If the dirty rate is higher than page transferring rate, then the count of dirty pages re-transfer is increased in further iterations. The current algorithm cannot complete the dirty page transfer phase then the only solution involves VM to be prematurely suspended. This is not an appropriate solution. If the memory pages that are left to re-transfer then it causes a long downtime. If pre-copy migration pattern migrates the write-intensive application then application performance degrades significantly during the migration process. Clark et al. [33] proposed dynamic rate limiting method that reduces the application dirty rate for prioritizing the migration process. Consequently, the performance of running applications is badly impacted.

The test migrations between an identical pair of server-class machines (Dell PE-2650 dual Xeon CPU's 2GHz, 2 GB RAM, Broadcom TG3 network interfaces) and are connected through switched Gigabit Ethernet network. XenLinux 2.4.27 as the OS is used in all cases. A theoretical study shows that one VM, requesting and modify some memory pages more frequently than other. Their dynamic network-bandwidth adaptation reducing service downtime to below discernible thresholds and with minimal impact on running services during migration.

Pre-copy approach cap the number of copying iterations to its maximum number of iterations since the writable working set is not guaranteed to converge across successive iterations, especially when VM is executing read-intensive workload. Fei Ma et al. [111] attempt to improve pre-copy approach on Xen hypervisor. They used the bitmap approach in which they mark the frequently updated pages and focus on cluster environment for migration. There is CPU and memory status is needed to be transferred from source to destination server and no need to transfer storage blocks because in cluster environment network-accessible storage system like Storage Area Network (SAN) or NAS are used. The frequently dirty pages are recorded in the bitmap in every iteration process and are transmitted in the last round. It ensures that frequently updated pages are transmitted only once. Therefore it solves the problem of re-transmitting memory pages multiple times that leads to a reduction in total migration time and transferred data. The test environment consists of two physical machines (2.66 GHz dual core Intel, 4 GB RAM, Ubuntu 8.04 as host OS and guest OS, Xen 3.3.0 hypervisor) connected via a Fast Ethernet switch. Results show that improved pre-copy approach compared to pre-copy reduce the total transferred data by 34% and total migration time by 32.5% on average.

For accurate prediction of migration performance, a model is proposed by Akoush et al. [70], which examines the service interruptions for a particular workload. Authors show network link capacity and memory dirty rate are the major factors that highly affect the migration behavior. The predicted value of migration time must be accurate to handle dynamic and intelligent VM placement without affecting application performance. Live VM migration behavior in pre-copy migration technique is investigated in Xen hypervisor platform. The link capacity and page dirty rate highly impact migration performance in a non-linear manner due to hard-stop conditions force migration in last stop-and-copy phase. Authors also implement Average page dirty rate (AVG) and History based page dirty rate (HIST) simulation models, used to predict the performance of pre-copy migration. Experiment is performed on 3 servers (2 Intel(R) Xeon(TM) E5506 CPU's 2.13 GHZ , 6 GB DDR3 RAM, dual Gigabit Ethernet,

**Table 12** Comparison of Dirty page aware VM migration approaches

Approach	Objective	Technique	Performance metrics	Hypervisor / Simulator
Clark et al. [33]-2005	Reduces the application dirty rate for prioritize the migration process	Bound the number of pre-copying rounds, based on analysis of the writable working set	Minimal effect on running services, while reducing total downtime to below discernible thresholds	XenLinux 2.4.27
Fei Ma et al. [111]-2010	Avoid re-transmission of memory pages multiple times	Bitmap page in their improved pre-copy approach	Improved pre-copy approach compared to pre-copy reduce the total transferred data by 34% and total migration time by 32.5%	Xen 3.3.0
Akoush et al. [70]-2010	Accurate prediction of migration performance	Implemented 2 simulation models, termed AVG and HIST, used to predict pre-copy migration performance	Predict migration times to within 90% accuracy for both synthetic and real-world benchmarks	Xen 3.3.1
Ibrahim et al. [112]-2011	Study pre-copy live migration of MPI and OpenMP scientific applications	Novel KVM pre-copy algorithm with different termination criteria	Provide minimal downtime and minimal impact on end-to-end application performance	KVM 0.14.0
Jin et al. [113]-2011	Designing an optimization scheme for live migration	Tries to limit the speed of changing memory through controlling the CPU scheduler of the VM monitor	Can reduce up to 88% of application downtime	Xen 3.1.0
Zaw and Thein [114]-2012	Reducing the amount of transferred data	Framework that includes pre-processing phase in traditional pre-copy live migration	Reduce total data transferred up to 23.67% and total migration time on average by 11.45%, with respect to traditional pre-copy migration	Xen 5.6.0
Yong et al. [115]-2013	Predict the dirty pages for performance improvement	Novel Context Based Prediction algorithm	Shorten total migration time, downtime and total pages transferred significantly	KVM 0.14.0
Mohan and Shine [116]-2013	Reduce the migration time	Avoid dirty page retransmission by sending the log records of modifications	Downtime and the migration time overheads are reduced	Real cloud setup
Nathan et al. [29]-2013	Avoid aggressive pre-copy termination	Adaptive live migration and non-adaptive	Reduced network traffic for migration reduced by 14-93% and migration time reduced by 34-87%	Xen 4.0.1
Yin et al. [53]-2014	Limiting the page transfer attempts	Three-Stage memory transfer approach	Reduce total migration time, total pages transferred and useful for automatic load balancing	Xen 4.1.4
Zang et al. [117]-2014	Find the appropriate bandwidth that guarantee the total migration time and downtime	Theoretically determine the proper bandwidth to guarantee the total migration time and downtime	Guarantee the expected total migration time and downtime	Xen 3.4.3
Desai and Patel [118]-2016	Improve the total migration time and total downtime	Modifying existing pre-copy algorithm handle both low and high dirty page rate and use CBC algorithm	Reduces migration time and total downtime	CloudSim

Citrix Xenserver 5.5.0 (Xen 3.3.1), Ubuntu 2.6.27-7 kernel). The results show that for high speed (10 Gbps) network links, Xen migration architecture does work well. Several optimization's approaches increase the migration throughput by 125.5% (from 3.2 Gbps to 7.12 Gbps). Both AVG and HIST models are more than 90% accurate with respect to actual results.

For commercial applications, KVM and Xen work well but High Performance Computing (HPC) workloads require more CPU cycles during migration process that may not be fulfilled with current KVM rate control and target downtime heuristics, which leads to service degradation drastically. In case of HPC applications statically choose rate limits and downtime is infeasible.

Ibrahim et al. [112] show the behavior of iterative pre-copy live migration for memory-intensive applications (HPC workloads). Without going in detailed knowledge of the application behavior, memory-intensive applications are difficult to migrate. The scientific application memory dirty rate is likely to be higher than the migration draining rate. Authors presented a novel online algorithm, which able to provide minimal impact on application performance by controlling the migration based on the speed of memory updating. The experiment is performed on two (quad-core quad-socket UMA Intel Xeon E7310 (Tigerton) 1.6 GHz, Linux kernel 2.6.32.8, KVM 0.14.0 hypervisor) machines that are connected by dual InfiniBand and Ethernet networks. The results show that the

algorithm achieves reduced downtime and low impact on performance.

When memory dirty rate is higher than pre-copy migration rate than live migration will fail. During the process of pre-copy migration, application performance degrades if the memory dirty rate is higher than the network transfer capacity. To handle this issue, Jin et al. [113] proposed an optimized pre-copy VM migration technique, for this vCPU frequency is changed to control memory dirty rate. The proposed technique adjust memory dirty rate and control the vCPU frequency to the required service downtime limit when memory dirty rate is high, for avoiding application QoS degradation. Memory dirty rate becomes lower. The authors also analyzed that downtime varies with different bandwidth levels while varying memory dirty rate. So, this technique adversely affects the application performance and can be used for some set of applications like gaming applications. Reducing the vCPU frequency, the game is not stopped but affect only visual objects. Experiments performed servers (4 Intel Xeon CPU's 1.6 GHz, 4 GB DDR RAM, Linux 2.6.18, Xen3.1.0 hypervisor) connected by 1000 Mbps Ethernet network. The migration barrier has been loosened up to 4 times using the optimized algorithm. Also the migration of the same workload with and without optimization, VM's downtime (up to 88%) lower dramatically, with the acceptable overhead.

For live migration, mainly pre-copy approach is used in which the performance of VM is affected by total migration time and considerable amount data is transferred during the migration process. Zaw and Thein [114] presented a framework that extends the pre-copy migration phase by including the pre-processing phase to reduce the data transfer amount. They proposed the prediction working set algorithm for pre-processing, which combines Least Recent Used (LRU) cache and splay tree algorithm, which reduce a number of transferred memory pages. Evaluation is performed on a cluster composed of 6 similar servers (two Intel Xeon E5520 quad core CPU 2.2GHz, 8GB DDR RAM, Linux-2.6.18.8 OS, Citrix Xen-5.6.0 hypervisor). The implementation is performed on XEN platform, the proposed framework can reduce total data transferred up to 23.67% and total migration time on average by 11.45%, with respect to traditional pre-copy migration.

The pre-copy migration performs well for the lightweight memory VM's but it cannot guarantee a desirable performance if memory dirty rate is high or if there is low network bandwidth. Re-sending of dirty pages multiple times leads to a performance degradation issue. Yong et al. [115] presents Context Based Prediction algorithm (CBP) and makes use of Prediction by Partial Match (PPM) model to predict the dirty pages in the later iteration based on the historical statistics of dirty page

bitmap. It helps to avoid re-transmission of frequently updating pages by transferring them in the last iteration. The experiment is performed on three identical physical machines (Intel Core 2.93GHz dual processor, 4G RAM), connected via a Gigabit Ethernet and NFS is installed in one of them as the shared storage. Both of host OS and guest OS are Ubuntu Server 11.10, and standard KVM 0.14.0 and the modified KVM 0.14.0 with adding CBP codes for contrast experiment. Implementation results show that CBP algorithm achieves considerable improvement in total migration time, service downtime, and total pages transferred compared with KVM's default migration algorithm.

The pre-copy approach is highly used for minimizing both the total migration time and the service downtime. But it is inefficient in the case of the high dirty rate of memory pages and this will increase the total migration time. The high dirty rate problem is also pointed by Mohan and Shine [116]. In their method, they reduce the total migration time by sending the log records of modifications instead of re-sending the dirty pages or postpone the transmission of frequently dirty pages. It transfers least recently used memory pages till more than half iterations. The VM's are hosted over a cluster of machines (Intel i5 3.10 GHz, 1.88 GB RAM, Ubuntu 10.04 OS) connected via Ethernet network links. The model is designed in such a way that the migration time and the service downtime are reduced.

Live VM migration can be of two types - adaptive method and non-adaptive method. These methods, requires a considerable amount of CPU and network resources during migration, that critically affect the VM performance. This issue requires a building of effective approach that considers both the performance of VM and the resource needs during migration which can help to select the appropriate VM(s) for migration and also allocate the appropriate amount of resource for migration. Nathan et al. [29], investigates the cost-profit analysis for adaptive and non-adaptive VM migration, to avoid aggressive pre-copy termination. An adaptive approach pro-actively adjusts the memory page transfer rate based on VM behavior, whereas non-adaptive approach transfers VM memory pages at maximum possible transfer speed. They combine both the approaches and name it as Improved Live Migration technique (ILM), that reflects applications higher dirty rate and limited resources of the server that concerns during VM migration process. ILM optimizes the performance of un-managed VM migrations by triggering stop-and-copy phase in certain conditions like (1) if number of iterative rounds reached up to pre-defined threshold, (2) if VM memory is copied three times, (3) if dirty rate becomes lower than a pre-defined threshold, or (4) if the dirty rate in the previous round was higher than predefined threshold. To optimize

the communication bandwidth, ILM approach eliminates the free memory pages when migration is under process and also improves the VM migration time and service downtime. Migration workloads performed using the five physical machines (4 cores Intel i5 760 CPU 2.8 GHz, 4GB RAM, Ubuntu Server 10.04 64-bit at both host OS and Guest OS) and all the machines connected by 1 Gbps D-Link DGS-1008D switch. Three machines acted as a controller whereas other two machines installed with Xen 4.0.1 hypervisor. The ILM technique reduces the network traffic by 14-93% and migration time by 34-87% compared to the vanilla live migration techniques.

In pre-copy approach memory pages are transfer number of time that increases total migration time and network traffic whereas in post-copy approach leads to a lot of page fault and high service downtime. A Three-Phase Memory (TPM) transfer approach proposed by Yin et al. [53], determine that the memory pages are transferred at most twice during the whole migration process. This approach ensures that memory page fault occurs only for fraction of memory that leads to lessening total migration time. The TPM transfer having full memory copy, dirty bitmap, and dirty page moving phases for entire VM memory migration. In the full memory copy phase, transfer all the VM memory pages from source to destination server are transferred without interrupting the running applications even pages are continuously modified. In the next dirty bitmap copy phase, the VM at source server is suspended and then all the recorded dirty memory pages are transferred to the destination server. In the last dirty pages copy phase, the VM at the destination server is resumed. Active push and on-demand paging approaches are used to fetch faulty pages from the source server. Implementation is performed on Xen 4.1.4 hypervisor and evaluation is performed under various memory-intensive workloads. Obtain results show that TPM approach can considerably reduce total pages transferred and total migration time. This work is effective for automatic load balancing.

Few of existing live migration techniques can be applied to the delay-sensitive web services applications or a VM backup process that needs to be done in a specific time. Pre-copy migration technique requires frequently varied transfer bandwidth, which is a critical problem for network operators. The accurate prediction of migration time is also not possible. Zang et al. [117] theoretically analyze appropriate bandwidth that guarantees the total migration time and service downtime. Authors first assume the dirty distribution of VM memory pages that follow the deterministic distribution. Then the bandwidth is determined under the condition that dirty distribution obeys the bernoulli distribution. Authors assumed that the dirty frequency of each page is varied and the Cumulative Distribution Function (CDF) of the dirty page frequency is a

reciprocal function. The experiment is conducted on two Servers (Dell, Linux kernel 2.6.18.8-xen, Xen 3.4.3 hypervisor) connected by HP ProCurve 2910a Ethernet. The observed results show that the reciprocal-based model characterize the dirty page rate well and also provides bounded delay guarantee.

In live migration, VM(s) are kept continuously powered-up but it is not the case in offline migration. Desai and Patel [118], proposed an approach by further modifying existing pre-copy algorithm which reduces migration time and downtime in both low-dirty page rate and high-dirty page rate. Further, by compressing whole data with Characteristic Based Compression (CBC) algorithm reduces both the downtime and migration time. Experiments are performed on CloudSim simulator. Proposed algorithm reduces migration time in both high-dirty page rate and low-dirty page rate.

Clark et al. [33] proposed dynamic rate limiting method that reduces the application dirty rate for prioritizing the migration process. Fei Ma et al. [111] attempt to improve pre-copy approach on Xen hypervisor by avoiding re-transmission of memory pages multiple times. For accurate prediction of migration performance, a model is proposed by Akoush et al. [70], which examines the service interruptions for a particular workload. Ibrahim et al. [112] show the behavior of iterative pre-copy live migration for memory-intensive applications (HPC workloads) because HPC applications statically choose rate limits and downtime which is infeasible. During the process of pre-copy migration, application performance degrades if the memory dirty rate is higher than the network transfer capacity. To handle this issue, Jin et al. [113] proposed an optimized pre-copy VM migration technique, for this vCPU frequency is changed to control memory dirty rate. Zaw and Thein [114] presented a framework that extend the pre-copy migration phase by including the pre-processing phase to reduce the data transfer amount. Further re-sending of dirty pages multiple times leads to a performance degradation issue. For this Yong et al. [115] presents CBP algorithm and makes use of PPM model to predict the dirty pages in the later iteration based on the historical statistics of dirty page bitmap. The high dirty rate problem is also pointed by Mohan and Shine [116]. In their method they reduce the total migration time by sending the log records of modifications instead of re-sending the dirty pages or postpone the transmission of frequently dirty pages. Nathan et al. [29], investigates the cost-profit analysis for adaptive and non-adaptive VM migration, to avoid aggressive pre-copy termination. A Three-Phase Memory (TPM) transfer approach proposed by Yin et al. [53], determine that the memory pages are transferred at most twice during the whole migration process. The accurate prediction of migration time is also not possible. Zang et al. [117] theoretically analyze

appropriate bandwidth that guarantees the total migration time and service downtime. Desai and Patel [118], proposed an approach by further modifying existing pre-copy algorithm which reduces migration time and downtime in both low-dirty page rate and high-dirty page rate. Hence all the above works use different approaches to solve the migration problem.

#### **Page fault aware VM migration:**

In the post-copy migration, if any kind of failure occurs during the migration then recovery may not be possible. By using the check pointing, recovery & trace, and replay techniques to enable fast and transparent VM migration. Comparison of existing Page fault aware VM migration approaches is illustrated in Table 13.

In the literature, existing migration approaches mainly focus on transferring the VM run-time state by using the pre-copy approach. The pre-copy approach synchronizes the VM states at both source and destination sides that increase the network traffic, application downtime, and migration cost especially for memory-intensive workloads. Storing the state of data movement traces of non-deterministic events in the log file is called Check pointing. It is used at a later time for re-execution of the past process or failed process. So it is helpful for proactive failure and debugging. Liu et al. [119] implemented CR/TR-Motion, this approach reduces total migration time, service downtime and network bandwidth consumption. They used the check pointing, recovery & trace, and replay techniques to enable fast and transparent VM migration. The discrete event occurring in the system is monitored by trace daemon and generate the log files. In the first step, the checkpoint is transferred to the destination server. At source side, it iteratively generates the logs and transfers them to the destination server. This process is continued up to the complete transfer of logs,

and then the process is suspended at the source side and resumed at the destination side. The main advantage of their work is that the amount of trace data is smaller than the amount of data transferred in traditional pre-copy migration. But, this approach does not perform well in multi-core environments. Experiments are performed on similar physical machines (AMD Athlon 3500+ processor, 1GB DDR RAM, a modified version of Linux 2.4.20 as host OS, RHEL AS3 Linux kernel 2.4.18 as guest OS) and to transfer the VM images Intel Pro/1000 Gbit/s NIC is used. Results show that CR/TR-Motion can drastically reduce migration overheads compared with the pre-copy algorithm: service downtime, total migration time, data to synchronize the VM state, application performance overhead up to 72.4%, 31.5%, 95.9%, 8.54% respectively.

Further, Liu and Fan [120] proposed a hybrid technique for recovering the system using check pointing, recovery & trace, and replay with CPU scheduling. The execution log files of source VM are copied but dirty pages are not copied in this approach. It reduces the amount of transferred data. This algorithm also reduces downtime by CPU scheduling. During the migration process, check pointing logs are transferred in the first round, and log files are transferred in the iterative round, so the log of previous round will be sent in next round. Experiments are performed on two identical physical machines (Intel Atom CPU D410 processor, 2 GB DDR RAM, CentOS 5.5 kernel 2.6.18 as host OS and guest OS, Xen 3.0.1 as a hypervisor) and connected by 1000 Mb/s Ethernet network. Results show that proposed hybrid technique compared with a pre-copy algorithm can reduce total migration time and service downtime: up to 43.84%, 62.12% respectively.

It is still a challenging issue to dynamically optimize VM packing on the host, due to frequently changing

**Table 13** Comparison of Page fault aware VM migration approaches

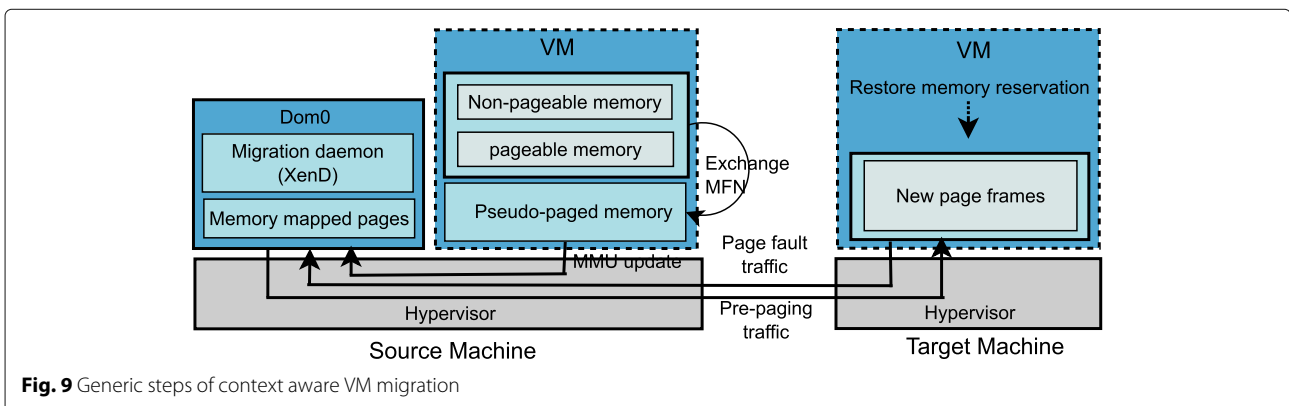
Approach	Objective	Technique	Performance metrics	Hypervisor / Simulator
Liu et al. [119]-2009	Achieve fast, transparent VM migration	Design and implementation of a novel approach CR/TR-Motion	Reduce migration overheads compared with pre-copy algorithm: service downtime, total migration time, data to synchronize the VM state, application performance overhead up to 72.4%, 31.5%, 95.9%, 8.54% respectively	Real cloud setup
Liu and Fan [120]-2011	Fast and transparent live migration	Novel approach based on recovering system (log files) and CPU scheduling	Proposed hybrid technique compared with pre-copy algorithm can reduce total migration time and service downtime : up to 43.84%, 62.12%	Xen 3.0.1
Hirofuchi et al. [121]-2010	VM migrations within LAN network	Advanced live migration mechanism that enables instantaneous VM relocation on physical machines	Reduced the total number of transferred memory pages, and efficiently copied them by utilizing available network bandwidth	KVM-84/88 and qemu-kvm-0.11

resource demands. A light weight KVM hypervisor extension proposed by Hirofuchi et al. [121] for migrations of VM's within LAN network. The KVM extension has an additional driver in guest-OS that seamlessly processes on-demand memory access query generated by VM. The CPU state and device states are transferred to the destination server using QEMU before moving the VM memory content for continuous progress of VM. Based on the migrated VM CPU states and device states, the destination server uses a QEMU hypervisor to resume migrant VM, based on received states. For servicing the application's I/O request, a kernel triggers the page fault handler of Virtual Memory (VMEM) driver (which works as a page fault handler) send a request for transferring the faulty pages to the destination server, only if pages are not available. Also, QEMU initiates background threads for actively pushing pages to the destination server. As a result, virtual memory based migration provides the freedom to work independently without demanding another driver for handling migrant VM. The implementation is performed on two (source and destination) physical machines (2-core processor, 4 GB RAM, KVM (KVM-84/88 and qemu-kvm-0.11) hypervisor) connected by two GbE network segments. SPECweb2005 benchmark [122] is used for performance measurements for web servers. The experimental results show that a heavily-loaded VM is successfully migrated to destination server within 1 s. The proposed mechanism fastly move VM state (including memory pages) to the destination server compared to the pre-copy approach. It also reduces the number of pages transferred by effectively using available network bandwidth. Another work focus that after migration, a guest OS fails to boot-up, while loading device drivers or device configuration adjustment. To solve these problems Ashino et al. [123] presented Estimation of Directions of Arrival by Matching Pursuit (EDAMP) method for VM migration between heterogeneous hypervisor implementation. It only modifies the files and does not destroy the device driver.

Check pointing is used at a later time for re-execution of the past process or failed process. So it is helpful for proactive failure and debugging. for such scenario, Liu et al. [119] implement CR/TR-Motion, reduce total migration time, service downtime and network bandwidth consumption. They used the check pointing, recovery & trace, and replay techniques to enable fast and transparent VM migration. Further, Liu and Fan [120] proposed a hybrid technique for recovering the system using check pointing, recovery & trace, and replay with CPU scheduling. A light weight KVM hypervisor extension proposed by Hirofuchi et al. [121] for migrations of VM's within LAN network. Therefore, live VM migration problem is solved using different approaches.

**Generic steps of context aware VM migration**

Basic steps of context aware VM migration is illustrated in Fig. 9. It uses the pseudo-paging migration approach, swap out all pageable memory in the VM to an in-memory pseudo-paging memory within the guest kernel. At source machine, page fault detection and servicing are implemented through the use of two loadable kernel modules, one is inside Dom0 and other is inside migrating VM. These modules use MemX [124] system that provides transparent remote memory access at the kernel level for both Xen VM's and native Linux systems. When the migration is started, the migrating VM memory pages are swapped out to a pseudo-paging memory which is exposed by the MemX module in the VM. The swapping is done using either one of Machine Frame Number (MFN) exchange mechanism, which transfers the ownership of the pages to a co-located VM or remapping the pseudo-physical address of the VM pages with zero copying overhead. The latter one is more effective because it generates fewer calls into the hypervisor which lead to a lower overhead. After the swapping process at source machine, the pages are mapped to Dom0. During the service downtime, CPU state and non-pageable memory are transferred to the target machine. The non-pageable



**Fig. 9** Generic steps of context aware VM migration



memory transfer overhead can be considerably reduced via the any of the hybrid approach.

**Threats in live virtual machine migration**

Live migration is quite a new idea and its security aspects are not fully discovered.

The popularity of cloud computing caught the attention of many hackers, allowing them to find new ways to attack either cloud services or customer’s data. These attacks may range from Denial-of-Service (DoS) attacks to Man-In-The-Middle (MITM) attacks. These kind of threats in live VM migration discourages many sectors, such as financial, medical, and government, from taking advantage of VM live migration. Hence, this is one of the critical factors that needs examination, while VM migration is being considered. There are various active and passive attacks possible while migration is under process and some of them are discussed below:

1. **Bandwidth stealing:** Attacker may steal the network bandwidth by taking the control over source VM and migrate it to the destination.
2. **Falsely advertising:** The attacker may advertise false information of resources over network and try to attract others for migrating their VM’s towards attacker side.
3. **Passive snooping:** Attacker tries to get unauthorized access of data.
4. **Active manipulation:** Attacker tries to modify the data which is travelling from one server to another [125].

To detect and prevent such attacks, there are several cryptographic algorithms available that are used for

encryption and decryption of data. The following steps must be considered when migration is initiated, both at the source and destination server:

1. The person who initiates migration should be authenticated.
2. Security among various entities must be preserved at every step.
3. Entire migration information should be kept confidential.

**Security concern in VM migration**

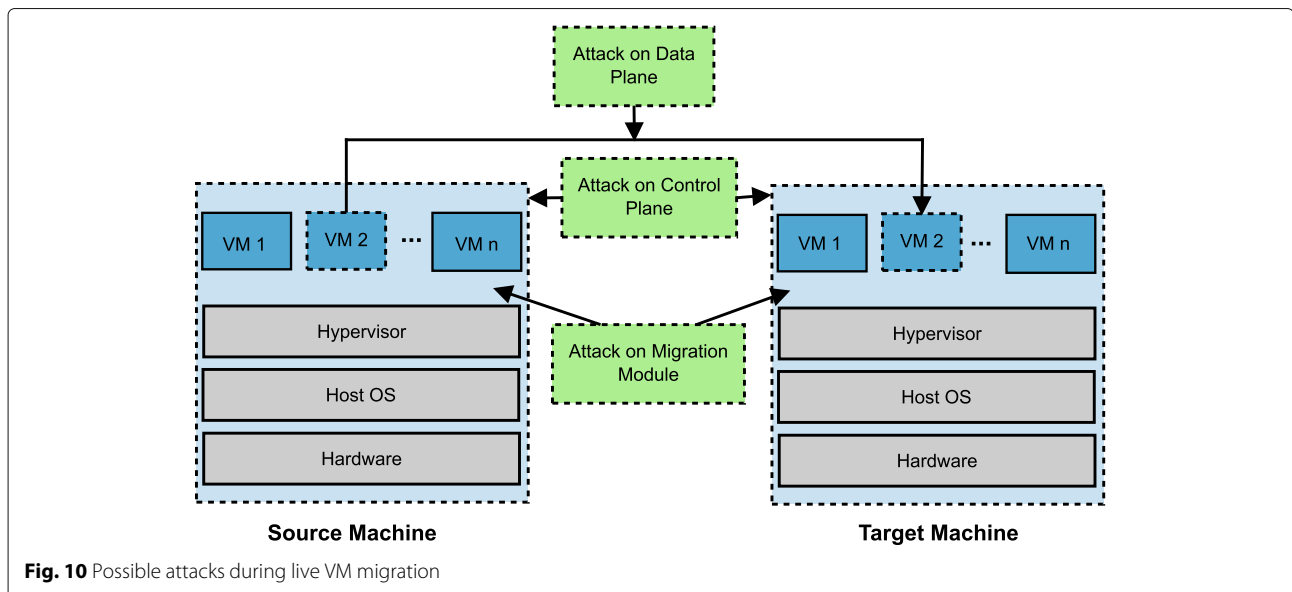
Live VM migration leads to number of security threats in CDC’s that maybe directed at hypervisors like KVM, Xen and VMware. Hypervisors are not able to secure sensitive information during migration and are vulnerable for attack. Attacker gets complete control of hosted VM and on VMM.

For the secure VM migration, much research has been accomplished with a focus on offline migration. However live VM migration still needs to be actively investigated.

Live VM migration suffers with many vulnerabilities and threats which can be easily explored by the attackers. Anala et al. [125], Jon et al. [126], Sulaiman and Masuda [127] demonstrated live migration threats. Basedon their demonstrations, live migration attacks can be targeting one of these three different classes: (1) control plane (2) data plane and (3) migrationmodule. This is illustrated in Fig. 10.

**Control plane**

Migration process at both source and destination side are handled by system administrator who is having all the controls and authority to perform secure VM migration



**Fig. 10** Possible attacks during live VM migration

operations (e.g. creating new VM, migrating VM, terminate a running VM, defining the VM's setting, etc). This will prevent spoofing and replay attacks. No other user can do the migration process if the access control of the administrator's interface is secure. The mechanism of communication used by the hypervisor should also be authenticated and must be resistant against any tampering [126]. A lack of security in the control plane may allow an attacker to exploit live migration operation in different ways:

1. **Denial-of-Service (DoS) attack:** Attacker will create many VM's on the host OS just to overload the host OS, which will not be able to accept any more migrated VM's.
2. **Unnecessary migration of VM:** Attacker will overload the host OS by unneeded VM's. This will force execution of the dynamic load balancing feature, which will ensure migration of some VMs to balance the load.
3. **Incoming Migration Control:** The attacker can initiate an unauthorized migration request, so VM can be migrated from secure source physical machine to a compromised attacker machine. This may result in attacker getting full control on the legitimate VM.
4. **Outgoing Migration Control:** The attacker can initiate the VM migration and can make the overuse of the cloud resources which can lead to failure of the VM.
5. **Disrupt the regular operations of the VM:** An attacker may migrate a VM from one host to another host without any goal except to interrupt the operations of the VM.
6. **Attack on VMM and VM:** Attacker will migrate a VM that has a malicious code to a host server that has the target VM. This code will exchange information with the VMM and the target VM through a covert-channel. This channel will compromise the confidentiality of the host server by leaking target VMs' information.
7. **Advertising for false resource:** Attacker advertises false resource availability for the target VM. For example, advertising that there is a large number of unused CPU cycles. This results in migration of the VM's to a compromised hypervisor.

#### **Data plane**

Several contents (e.g., kernel states and application data) of memory are transferred from source to destination server in the data plane. It is possible that the attacker can passively snoop and steal or actively modify confidential information. Thus, the transmission channel must be secured and protected against various active and passive attacks. In the VM migration protocol, all migrated

data are transferred as clear data without any encryption. Hence, an attacker may place himself in the transmission channel to perform a man-in-the-middle attack using any of the techniques: Address Resolution Protocol (ARP) spoofing, Domain Name System (DNS) poisoning, or route hijacking [126]. Man-in-the-middle attack can be one of the two types of attacks - passive and active:

1. **Passive attack:** Attacker observes the transmission channel and other network streams used to get the information of migrating VM. The attacker gains information from the VM's migrating memory (e.g., passwords, keys, application data, capturing packets that are already authenticated, messages that have sensitive data will be overheard, etc.) [125].
2. **Active attack:** This attack is the most serious attack in which the attacker manipulates the memory contents (e.g., authentication service and pluggable authentication module in live migration) of migrating VM's [128].

#### **Migration module**

Migration module is a software component in the VMM that allows live migration of VM's. A guest OS can communicate with the host system and vice versa. Moreover, the host system has full control over all VM's running over its VMM. If the attacker is able to compromise the VMM via its migration module, then the integrity of all guest VM's that are running above this VMM will be affected. Any VM in the future that will migrate to the affected VMM will also be compromised. VM with a low security level is exploited using the attack techniques in the migration module. When an attacker discovers a VM with a low security level during the migration process, they will attempt to compromise it and can do it easily. They can use it as a gate to compromise other VM's on the same host with higher levels of security [129]. Moreover, the attacker will be able to attack the VMM itself, after identifying a way to enter the system.

#### **Security requirement in VM migration**

There are security requirements that must be implemented in the live VM migration, which will enhance the security level in the previous classes to protect both VMs and host servers from any attack - before, during, and after the live migration process. Aias et al [130] and John et al [126] discussed security requirements in live VM migration. Following are the security requirements that should be implemented in VM live migration: (1) defining access control policies, (2) authentication between sender source server and the destination server, (3) non-repudiation by source and destination server, (4) data confidentiality while migrating a VM, (5) data confidentiality before and after migration, and (6) data integrity and availability.

### **Security requirements to mitigate attacks in the Control Plane and the Data Plane**

1. **Defining access control policies:** By defining control policies on the control plane, VM's and the host server will be protected from unauthorized users. If attackers can compromise the interface console, they might perform unauthorized activities such as migrating a VM from one host to a legitimate target VMM [130].
2. **Authentication between source and destination server:** Implement strong procedures of authentication and identification in order to prevent unauthorized users from entering administrators' interface.
3. **Data integrity and availability:** This requirement will stop some attacks, such as a denial-of-service attack, which causes unavailability of either the source host or the receiver host. This can be done by applying strict policies for accessing control.
4. **Data confidentiality during migrating the VM:** In order to prevent a man-in-the-middle attack from getting any sensitive information, all data during migration must be encrypted

### **Security requirements to mitigate attacks on the Migration Module**

1. **Authentication between source and destination server:** A strong authentication mechanism must be used between source and destination server. Firewall can also be used for more security options [130].
2. **Non-repudiation by source or destination server:** The source and destination server must observe the system's activities and record all the migration activities [130].
3. **Data confidentiality before and after migration:** Data should be encrypted at both source and destination servers. Whenever the attack happens at either guests VM's data or the host's data, then the original information not be affected.
4. **Data integrity and availability:** The virtualization software must be updated so that it can be protect from vulnerabilities like heap overflow and stack overflow [130].

### **Existing solutions for providing security in VM migration**

1. **Isolating Migration Network:** The Virtual LAN (VLAN) that contains source and destination servers is isolated from other migration traffic over the network. This reduces the risk of exposure of migration information to the whole network.
2. **Network Security Engine Hypervisor:** It extends the firewall and IDS/IPS functionality at hypervisor level, which secures the migration from external attack and raise an alarm when intrusion is detected over network.

3. **Secure VM-vTPM (Virtual Trusted Platform Module) Migration protocol:** The protocol includes various steps like authentication, attestation and then different stages of data transfer. At the first step both the parties authenticate for further communication. The source VM start transferring to the destination only after verification of integrity. The migrating VM files are stopped by the vTPM where the files are encrypted and then transferred to the destination. After transferring all the files of a VM, vTPM is deleted.
4. **Improved vTPM Migration protocol:** The protocol is an improved version of vTPM that consists of trust component also. It first performs authentication, integrity verification as performed in vTPM. After that the source and the destination server negotiate keys using Diffie-Hellman key exchange algorithm. The migrating VM files are protected with keys and encryption methods that enable the secure transfer of VM files.
5. **SSH (Secure Shell) Tunnel:** It is established between source and destination proxy servers for secure migration that hide the details of the source and destination VM's [131].

### **Research challenges**

Migration must be seamless to provide the continuous services. Live migration moves the VM without disconnecting with the client. Performance of live VM migration must be very high for continuous services. Current techniques face many challenges while migrating memory and data intensive applications, like - network faults, consumption of bandwidth and cloud resources, overloaded VM's. Common challenges that hamper live migration are: transfer rate problem, page re-send problem, missing page problem, migration over WAN network, migration of VM with the larger application, resources availability problem, and address-wrapping problem.

### **Transfer rate**

During the iterative phase of pre-copy live VM migration, the VM's pages are sent over the network between corresponding servers. As the source VM is running during this process, its memory contents are constantly updated. Because memory bandwidth is higher than network bandwidth, there is a high risk of memory pages being dirtied at a faster rate than they can be transferred over the network. As a result, these dirty pages are transferred repeatedly while the amount of remaining dirty pages transfer does not decrease. This means that the migration process gets stuck in the iterative phase and as a result, the migration may have to be forced into the stop-and-copy phase with a large number of dirty pages remaining to transfer. As the VM is suspended during the stop-and-copy phase, this

leads to extended migration downtime and a prolonged total migration time. Even in less severe cases, where the algorithm does not need to be forced to proceed stop-and-copy phase, total migration time and service downtime are still extended to some degree.

The transfer rate problem poses a high risk to the continuous service operation, as an extended migration downtime can lead to interruption of services and possibly disconnection of clients, lost database connections, or other issues. Even if the migration downtime is short enough for network connections not to drop (typically a few seconds for TCP connections over LANs or the internet), timing errors, missed triggers, etc, might occur and decrease the application's stability and performance. Svård et al. [105] shown experimental results that live migration of enterprise applications, downtimes as low as one second caused unrecoverable application problems.

#### Page Re-send

Live migration of a VM requires significant CPU and memory resources, although the heaviest load is put on the network. As a VM can easily have several gigabytes of RAM, a large amount of data is transferred during the live migration process. This problem is amplified in pre-copy migration as the source VM is running during the iterative phase and pages that have already been transferred are often being dirtied again. Since the state of the destination VM once resumed must be an exact copy of the source VM's state, these pages must be re-sent.

The page re-send problem was first discussed by Clark et al. [33] and can lead to excessive resource consumption, as only the final version of a page is used and re-sending pages during migration consume both network and CPU resources. Furthermore, the page re-send problem is a challenge to the predictability criteria as it is not known beforehand the total number of pages that are to be re-transferred, making it difficult to estimate how long migration takes to complete.

Svård et al. [105] present that pre-copy migration is affected by both the page-resend and transfer rate problems. These problems are related as the transfer rate is a cause of page-resend. However, factors like memory size, page dirtying rate, and memory write patterns also affect the number of pages resends.

#### Missing Pages

Post copy live migration algorithms resume the destination VM prior to completely transferring memory contents to the destination server. Then, the execution is switched to the destination side, the missing pages or faulty pages are transmitted from the source over the network. Due to the low bandwidth availability and higher migration latency, there is a performance penalty associated with accessing faulty memory pages. The residual dependency

problem also imposes a high risk of performance degradation for the hosted applications after the VM execution has switched to the destination server. If the performance degradation is severe, the transparency and continuous service objectives may not be met. The missing page problem also imposes a loss of robustness as it is not possible to fall-back to the source VM if the live migration fails, e.g. due to network disconnects that occur before the entire RAM content has been transferred. As the destination VM is not restarted until all memory pages are present in pre-copy methods, such algorithms are not affected by the missing page problem.

#### Migration over WAN network

The existing VM migration techniques cannot deal efficiently with VM migration over a WAN where the source and the destination servers are part of different networks [95]. Live VM migration across WAN network is big challenge as:

1. **Migrating network and storage connections:** TCP connection survives VM migration and its application without disruption in network connections if the source and destination servers are on the same sub-net. Otherwise, migration process also deals with breaks when migration occurs across sub-nets.
2. **Migrating storage content:** migration of large size virtual disk over WAN takes a long time. Hence the volume of data transferred over the WAN is also critical.
3. **Persistent state remains at the source side:** The re-located VM accesses the earlier centralized storage repository, over the WAN. Nevertheless, network latencies and considerable bandwidth usage result in poor I/O performance.

#### Migration of VM running larger applications

There are many challenges in current migration technology; the big issue appears when they are applied on large size workload application systems such as SAP ERP. These applications consume a huge amount of memory and storage capacity, that cannot be transferred seamlessly because they generate service interruption. So the limitations with larger applications are dis-connectivity of service, interruption of service, difficulty to maintain consistency & transparency, and unpredictability & rigidity in VM loads [104].

#### Resources availability

Resource availability is most important when a VM is migrated. Live VM migration consumes CPU cycles and I/O bandwidth between corresponding servers. If there is need of some CPU operation and it is not available then migration time would increase. Hence, if there is no

necessary resource available then migration couldn't be completed. Availability of resource affect the performance of the migration and total migration time. It can also help to make a better decision, such as when to migrate VM and how to deal with server resource allocation servers [69].

### Address warping

The address-warping problem is also one of the critical issues while dealing migration process at WAN level. The address of the VM warps from the source to the destination server which complicates the status of the connected LANs and the WAN networks. Therefore it is difficult to move real time application running on VM such as online games or conferences. It may cause long downtime, so downtime and complexity can be avoided [132].

### Live migration for high-speed LAN

The existing migration techniques assumed the network bandwidth is 1 Gbps. But in the large CDCs, servers are connected with high speed links like - 10 Gbps and 40 Gbps. So the transfer rate is higher and transmitted more data during the migration period, which implies that the trade-off between CPU utilization and network utilization is different from 1 Gbps speed [42]. Therefore, exploring the migration techniques for high speed LAN can further optimize the CDC performance, as well as the downtime is reduced.

The other research challenges in Live VM migration are Network fault [133], memory intensive application [133], memory state between clusters [133], Live migration of nested VMM [42], Live migration of VM attached to pass-through accelerators [42] pointed by authors.

### Conclusion and future work

Live VM migration is the process of moving a running VM or multiple VM's from one server to another. The services running on VM's must be available all the time to the users, hence they must be migrated while they are continuously running. This is possible only if VM's are migrated with zero downtime. The motivation behind live VM migration is - load balancing, proactive fault tolerance, power management, resource sharing, and online system maintenance. We identify the types of contents that need to be migrated during migration which are CPU state, memory content, and storage content. We discuss pre-copy, post-copy and hybrid techniques of VM migration and present basic steps used in the migration process. We mention the important performance metrics which affects the migration overheads.

The comprehensive survey of state-of-the-art Live VM migration approaches are divided into two broad categories. We first discuss the models - which are theoretical phases. Then we discuss the frameworks - which

are practical implementation. The live VM migration frameworks are further divided into three sub-categories like the type of migration, duplication based VM migration, and context aware VM migration. These categories are based on the (i) single or multiple VM migration, (ii) replication, duplication, redundancy and compression VM/VM's memory pages, and (iii) dependency among VM's, soft page, dirty page (dirty page rate) and page fault due to network of VM pages. The existing approaches of all the above sub-categories are compared based on performance metrics. Threats in live VM migration are discussed and categorize the possible attacks in three categories (control plane, data plane and migration module) based on the type of attack. Finally we mention some of the critical research challenges which require further research for improving the migration process and efficiency of CDC's.

In our future work, we will propose a novel approach which would be able to reduce service downtime and total migration time. We will also optimize the migration technique in the hypervisor to improve the performance of the live VM migration.

### Abbreviations

ARP: Address resolution protocol; AVG: Average page dirty rate; BFD: Best fit decreasing; CBP: Context based prediction; CDC: Cloud data centers; CIC: Composed image cloning; DoS: Denial-of-service; DRBD: Distributed replicated block device; DNS: Domain name system; DSB: Dynamic self-ballooning; DVFS: Dynamic voltage frequency scaling; ERP: Enterprise resource planning; EDAMP: Estimation of directions of arrival by matching pursuit; HPC: High performance computing; HIST: History based page dirty rate; HMDC: Hybrid memory data copy; IP: Internet protocol; IRLM: Inter-rack live migration; KVM: Kernel-based virtual machine; LRU: Least recently used; LZ0: Lempel-ziv-oberhumer; LAN: Local-area network; MFN: Machine frame number; MITM: Man-in-the-middle; MMU: Memory management unit; MECOM: Memory compression; MBFD: Modified best fit decreasing; NAS: Network attached storage; NFS: Network file system; OS: Operating system; PPM: Prediction by partial match; QoS: Quality of service; QEMU: Quick emulator; RDMA: Random direct memory access; RLE: Run length encoding; SSH: Secure shell; SLA: Service level agreements; SAN: Storage area network; SAP: Systems, applications and products; TPM: Three-phase migration; TCP: Transmission control protocol; VM: Virtual machine; VMEM: Virtual memory; VMM: Virtual machine monitor; VLAN: Virtual local-area network; TPM: Virtual trusted platform module; WAN: Wide-area networks; XBZRL: Xor binary zero run length encoding

### Acknowledgements

We are thankful to anonymous reviewers for their valuable feedback and comments for improving the quality of the manuscript.

### Funding

Not applicable.

### Availability of data and materials

Not applicable.

### Authors' contributions

AC, DK and ESP surveyed the literature exhaustively. They prepared the logistic of each paper - brief write up on the contribution of each paper, tools used, techniques used, and summary of results. MCG and LKA conceived the need for the study and designed the outline of sections. They also guided choosing of parameters for evaluating various techniques. They have validated the tables giving the comparison. GS and ESP participated in design and coordination of all sections and helped to draft the manuscript. AC and DK

wrote most of the content and did the analysis. MCG and LKA approved the final version to be submitted.

#### Competing interests

Anita Choudhary, Mahesh Chandra Govil, Girdhari Singh, Lalit K. Awasthi, Emmanuel S. Pilli, and Divya Kapil declare that they have no competing interest.

#### Publisher's Note

Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

#### Author details

<sup>1</sup>Malaviya National Institute of Technology Jaipur, Jaipur, India. <sup>2</sup>National Institute of Technology Sikkim, Ravangla, India. <sup>3</sup>Dr. B. R. Ambedkar National Institute of Technology, Jalandhar, India. <sup>4</sup>Graphic Era Hill University, Dehradun, India.

Received: 28 June 2017 Accepted: 25 September 2017

Published online: 07 November 2017

#### References

- Mell P, Grance T (2011) The NIST Definition of Cloud Computing Recommendations of the National Institute of Standards and Technology. Technical report. doi:10.1136/emj.2010.096966 arxiv: 2305-0543
- Choosing an App Engine Environment | App Engine Documentation | Google Cloud Platform. <https://cloud.google.com/appengine/docs/the-appengine-environments>. Accessed 04 Nov 2016
- Intro to Microsoft Azure | Microsoft Azure. <https://azure.microsoft.com/en-in/documentation/articles/fundamentals-introduction-to-azure/>. Accessed 04 Nov 2016
- Elastic Compute Cloud (EC2) Cloud Server & Hosting – AWS. <https://aws.amazon.com/ec2/>. Accessed 04 Nov 2016
- IBM - Cloud Computing for Builders & Innovators. <http://www.ibm.com/cloud-computing/>. Accessed 04 Nov 2016
- Buyya R, Buyya R, Yeo CS, Yeo CS, Venugopal S, Broberg J, Broberg J, Brandic I, Brandic I (2009) Cloud computing and emerging IT platforms: Vision, hype, and reality for delivering computing as the 5th utility. *Futur Gener Comput Syst* 25(6):599–616
- Uddin M, Shah A, Alsaqour R, Memon J, Saqour RAHASRAHA, Memon J (2013) Measuring efficiency of tier level data centers to implement green energy efficient data centers. *Middle East J Sci Res* 15(2):200–207
- Beloglazov A, Buyya R (2010) Energy Efficient Resource Management in Virtualized Cloud Data Centers. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. IEEE, United States. pp 826–831
- Zhou M, Zhang R, Zeng D, Qian W (2010) Services in the Cloud Computing era: A survey. In: 4th International Universal Communication Symposium. IEEE, Beijing. pp 40–46
- Storage Servers. <https://storageservers.wordpress.com/>. Accessed 07 Sept 2017
- Koomey JG (2011) Growth in Data Center Electricity use 2005 to 2010. PhD thesis
- Belady CL (2012) In the data center, power and cooling costs more than the it equipment it supports. <http://www.electronics-cooling.com/2007/02/in-the-data-center-power-and-cooling-costs-more-than-the-it-equipment-it-supports/>. Accessed 18 May 2016
- Fan X, Weber WD, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In: Proceedings of the 34th Annual International Symposium on Computer Architecture. ACM, California Vol. 35. pp 13–23
- Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A (2003) Xen and the art of virtualization. In: Proceedings of the Nineteenth ACM Symposium on Operating Systems Principles. ACM, NY Vol. 37. p 164
- Younge AJ, Henschel R, Brown JT, Laszewski GV, Qiu J, Fox GC (2011) Analysis of Virtualization Technologies for High Performance Computing Environments. In: IEEE 4th International Conference on Cloud Computing. IEEE Computer Society, Washington. pp 1–8
- Bugnion E, Devine S, Rosenblum M, Sugerman J, Wang EY (2012) Bringing Virtualization to the x86 Architecture with the Original VMware Workstation. *ACM Trans Comput Syst ACM Ref Format Bugnion* 30(4):1–51
- Desai A (2012) Managing Virtualization with System Center Virtual Machine Manager. <http://anildesai.net/index.php/2007/12/managing-virtualization-with-system-center-virtual-machine-manager/>. Accessed 07 Sept 2017
- vSphere ESXi Bare-Metal Hypervisor. <http://www.vmware.com/products/esxi-and-esx.html>. Accessed 04 Nov 2016
- KVM. [http://www.linux-kvm.org/page/Main\\_Page](http://www.linux-kvm.org/page/Main_Page). Accessed 04 Nov 2016
- Kivity Qumranet A, Qumranet YK, Qumranet DL, Qumranet UL, Liguori A (2007) Kvm: the Linux Virtual Machine Monitor. In: Proceedings of the Ottawa Linux Symposium, Ontario, Canada. pp 225–230
- Hypervisor x86 & ARM. <https://www.xenproject.org/developers/teams/hypervisor.html>. Accessed 04 Nov 2016
- Microsoft Virtual PC. [http://microsoft\\_virtual\\_pc.en.downloadastro.com/](http://microsoft_virtual_pc.en.downloadastro.com/). Accessed 04 Nov 2016
- Microsoft Hyper-V Server 2016. <https://technet.microsoft.com/en-us/hyper-v-server-docs/hyper-v-server-2016>. Accessed 04 Nov 2016
- Oracle VM VirtualBox. <https://www.virtualbox.org/>. Accessed 04 Nov 2016
- Parallels Desktop (for Mac) - Parallels Desktop 11 for Mac. <http://in.pcmag.com/parallels-desktop-10/46064/review/parallels-desktop-for-mac>. Accessed 17 Jan 2017
- Medina V, García JM (2014) A survey of migration mechanisms of virtual machines. *ACM Comput Surv* 46(3):1–33
- Ferretto TC, Netto MAS, Calheiros RN, De Rose CAF (2011) Server consolidation with migration control for virtualized data centers. *Future Gen Comput Syst* 27(8):1027–1034
- Hu L, Zhao J, Xu G, Ding Y, Chu J (2013) HMDC: Live virtual machine migration based on hybrid memory copy and delta compression. *Appl Math Inf Sci* 7(2 L):639–646
- Nathan S, Kulkarni P, Bellur U (2013) Resource Availability Based Performance Benchmarking of Virtual Machine Migrations. In: Proceedings of the ACM/SPEC International Conference on Performance Engineering. ACM, Prague, Czech Republic. pp 387–398
- Åsberg M, Forsberg N, Nolte T, Kato S (2011) Towards real-time scheduling of virtual machines without kernel modifications. In: IEEE International Conference on Emerging Technologies and Factory Automation, ETFA. IEEE, Toulouse
- Habib I (2008) Virtualization with KVM. *Linux J* 2008(166)
- Xu F, Liu F, Jin H, Vasilakos AV (2014) Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions. *Proc IEEE* 102(1):11–31
- Clark C, Fraser K, Hand S, Hansen JG, Jul E, Limpach C, Pratt I, Warfield A (2005) Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation - Volume 2. USENIX Association, Berkeley. pp 273–286
- Deshpande U, Kulkarni U, Gopalan K (2012) Inter-rack live migration of multiple virtual machines. In: Proceedings of the 6th International Workshop on Virtualization Technologies in Distributed Computing Date. ACM, Delft, The Netherlands. pp 19–26
- Atif M, Strazdins P (2014) Adaptive parallel application resource remapping through the live migration of virtual machines. *Futur Gener Comput Syst* 37:148–161
- Svärd P, Hudzia B, Tordsson J, Elmroth E, Svärd P, Hudzia B, Tordsson J, Elmroth E (2011) Evaluation of delta compression techniques for efficient live migration of large virtual machines. In: Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. ACM, California Vol. 46. pp 111–120
- Riteau P, Morin C, Priol T (2011) Shrinker: Improving Live Migration of Virtual Clusters over WANs with Distributed Data Deduplication and Content-Based Addressing. In: Proceedings of the 17th International Conference on Parallel Processing and Distributed Computing - Volume Part I. Springer, Bordeaux. pp 431–442
- Soni G, Kalra M (2013) Comparative Study of Live Virtual Machine Migration Techniques in Cloud. *Int J Comput Appl* 84(14):19–25
- Kapil D, Pilli ES, Joshi RC (2013) Live virtual machine migration techniques: Survey and research challenges. In: Proceedings of the 3rd

- IEEE International Advance Computing Conference. IEEE, Ghaziabad. pp 963–969
40. Ahmad RW, Gani A, Siti SH, Shiraz M, Xia F, Madani SA (2015) Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues. *J Supercomputing*, 71(7):2473–2515
  41. Ahmad RW, Gani A, Hamid SHA, Shiraz M, Yousafzai A, Xia F (2015) A survey on virtual machine migration and server consolidation frameworks for cloud data centers. *J Netw Comput Appl* 52:11–25
  42. Yamada H (2016) Survey on Mechanisms for Live Virtual Machine Migration and its Improvements. *Inf Media Tech* 11:101–115
  43. Kokkinos P, Kalogeras D, Levin A, Varvarigos E (2016) Survey: Live Migration and Disaster Recovery over Long-Distance Networks. *ACM Comput Surveys* 49(2):1–36
  44. Sapuntzakis CP, Chandra R, Pfaff B, Chow J, Lam MS, Rosenblum M (2002) Optimizing the migration of virtual computers. *ACM SIGOPS Oper Syst Rev* 36(SI):377–390
  45. Nelson M, Lim BH, Hutchins G (2005) Fast transparent migration for virtual machines. In: Proceedings of the annual conference on USENIX Annual Technical Conference. ACM, Berkeley. pp 25–25
  46. Huang W, Gao Q, Liu J, Panda DK (2007) High performance virtual machine migration with RDMA over modern interconnects. In: 2007 IEEE International Conference on Cluster Computing. IEEE, Washington. pp 11–20
  47. Luo Y, Zhang B, Wang X, Wang Z, Sun Y, Chen H (2008) Live and incremental whole-system migration of virtual machines using block-bitmap. In: IEEE International Conference on Cluster Computing. IEEE, Tsukuba. pp 99–106
  48. Verma A, Ahuja P, Neogi A (2008) pMapper: Power and migration cost aware application placement in virtualized systems. In: IFIP International Federation for Information Processing, vol. 5346 LNCS. pp 243–264. doi:10.1007/978-3-540-89856-6\_13
  49. Sammy K, Shengbing R, Wilson C (2012) Energy Efficient Security Preserving VM Live Migration In Data Centers For Cloud Computing. *J Comput Sci* 9(2):33–39
  50. Beloglazov A, Abawajy J, Buyya R (2012) Energy-aware resource allocation heuristics for efficient management of data centers for Cloud computing. *Futur Gener Comput Syst* 28(5):755–768. doi:10.1016/j.future.2011.04.017
  51. Kikuchi S, Matsumoto Y (2011) Performance modeling of concurrent live migration operations in cloud computing systems using prism probabilistic model checker. In: IEEE 4th International Conference on Cloud Computing. IEEE, DC. pp 49–56
  52. Xu F, Liu F, Liu L, Jin H, Li B, Li B (2014) iAware: Making live migration of virtual machines interference-aware in the cloud. *IEEE Trans Comput* 63(12):3012–3025
  53. Yin F, Liu W, Song J (2014) Live Virtual Machine Migration with Optimized Three-Stage Memory Copy. In: Future Information Technology. Springer, Berlin. pp 69–75
  54. Bala A, Chana I (2012) Fault Tolerance - Challenges, Techniques and Implementation in Cloud Computing. *Int J Comput Sci Issues* 9(1):288–293
  55. Shrivastava V, Zerfos P, Lee KW, Jamjoom H, Liu YH, Banerjee S (2011) Application-aware virtual machine migration in data centers. In: Proceedings - IEEE INFOCOM. IEEE, Shanghai. pp 66–70
  56. Mishra M, Das A, Kulkarni P, Sahoo A (2012) Dynamic resource management using virtual machine migrations. *IEEE Commun Mag* 50(9):34–40
  57. Dong J, Jin X, Wang H, Li Y, Zhang P, Cheng S (2013) Energy-Saving virtual machine placement in cloud data centers. In: Proceedings - 13th IEEE/ACM International Symposium on Cluster, Cloud, and Grid Computing, CCGrid 2013. IEEE/ACM, Delft, the Netherlands. pp 618–624
  58. Zheng J, Ng TSE, Sripanidkulchai K (2011) Workload-aware live storage migration for clouds. In: Proceedings of the 7th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. ACM, California Vol. 46. pp 133–144
  59. Bai W, Geng W (2014) Operation and Maintenance Management Strategy of Cloud Computing Data Center. *Adv Sci Technol Lett* 78(MulGrab):5–9
  60. Hu W, Hicks A, Zhang L, Dow EM, Soni V, Jiang H, Bull R, Matthews JN (2013) A quantitative study of virtual machine live migration. In: Proceedings of the 2013 ACM Cloud and Autonomic Computing Conference on - CAC '13, Florida
  61. Suen CH, Kirchberg M, Lee BS (2011) Efficient migration of virtual machines between public and private cloud. In: 3rd IEEE International Conference on Cloud Computing Technology and Science, CloudCom 2011. IEEE, United States. pp 549–553
  62. Compute Engine - IaaS | Google Cloud Platform. <https://cloud.google.com/compute/>. Accessed 07 Sept 2017
  63. Hines MR, Deshpande U, Gopalan K (2009) Post-copy live migration of virtual machines. *ACM SIGOPS Oper Syst Rev* 43(3):14–26
  64. Hines MR, Gopalan K (2009) Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. ACM Press, Washington. pp 51–60
  65. Ard PS, Walsh S, Hudzia B, Tordsson J, Elmroth E (2013) The Noble Art of Live VM Migration -Principles and Performance of precopy, postcopy and hybrid migration of demanding workloads. Technical report, Tech Report UMINF
  66. Voorsluys W, Broberg J, Venugopal S, Buyya R (2009) Cost of virtual machine live migration in clouds: A performance evaluation. *Lect Notes Comput Sci* 5931 LNCS:254–265
  67. Kuno Y, Nii K, Yamaguchi S (2011) A study on performance of processes in migrating virtual machines. In: 10th International Symposium on Autonomous Decentralized Systems. IEEE, Kobe, Japan. pp 567–572
  68. Feng X, Tang J, Luo X, Jin Y (2011) A performance study of live VM migration technologies: VMotion vs XenMotion. In: Proc. of SPIE-OSA-IEEE Asia Communications and Photonics. IEEE, Shanghai. pp 83101B-1-6
  69. Liu H, Xu CZ, Jin H, Gong J, Liao X, Xu CZ, Liao X, Jin H, Gong J, Liao X (2011) Performance and energy modeling for live migration of virtual machines. In: Proceedings of the 20th International Symposium on High Performance Distributed Computing. ACM, California. pp 171–182
  70. Akoush S, Sohan R, Rice A, Moore AW, Hopper A (2010) Predicting the Performance of Virtual Machine Migration. In: IEEE International Symposium on Modeling, Analysis and Simulation of Computer and Telecommunication Systems. IEEE, Miami Beach. pp 37–46
  71. Huang D, Ye D, He Q, Chen J, Ye K, Huang D, Ye D, He Q, Chen J, Ye K (2011) Virt-LM: a benchmark for live migration of virtual machine. In: Proceeding of the Second Joint WOSP/SIPEW International Conference on Performance Engineering. ACM, Karlsruhe Vol. 36. pp 307–316
  72. Wu Y, Zhao M (2011) Performance Modeling of Virtual Machine Live Migration. In: IEEE 4th International Conference on Cloud Computing. IEEE, DC. pp 492–499
  73. Watts up? <https://www.wattsupmeters.com/secure/index.php>. Accessed 07 Sept 2017
  74. Ceroni W, Callegati F (2014) Live migration of virtual network functions in cloud-based edge networks. In: IEEE International Conference on Communications. IEEE, Sydney. pp 2963–2968
  75. Deshpande U, You Y, Chan D, Bila N, Gopalan K (2014) Fast server deprovisioning through scatter-gather live migration of virtual machines. In: IEEE 7th International Conference on Cloud Computing, CLOUD. IEEE, AK. pp 376–383
  76. SPEC CPU® 2006. <https://www.spec.org/cpu2006/>. Accessed 07 Sept 2017
  77. Welcome to Apache™ Hadoop®! <http://hadoop.apache.org/>. Accessed 07 Sept 2017
  78. The Netperf Homepage. <https://hewlettpackard.github.io/netperf/>. Accessed 07 Sept 2017
  79. SPECweb2005. <https://www.spec.org/web2005/>. Accessed 07 Sept 2017
  80. NAS Parallel Benchmarks. <http://www.nas.nasa.gov/Software/NPB>. Accessed 08 Nov 2016
  81. Zhang W, Lam KT, Wang CL (2014) Adaptive live VM migration over a WAN: Modeling and implementation. In: IEEE International Conference on Cloud Computing, CLOUD, AK. pp 368–375
  82. Deshpande U, Keahey K (2017) Traffic-sensitive Live Migration of Virtual Machines. *Future Gener Comput Syst* 72:118–128. doi:10.1016/j.future.2016.05.003
  83. Ye K, Jiang X, Huang D, Chen J, Wang B, Kejiang Y, Xiaohong J, Dawei H, Jianhai C, Bei W, Ye K, Jiang X, Huang D, Chen J, Wang B (2011) Live Migration of Multiple Virtual Machines with Resource Reservation in Cloud Computing Environments. In: IEEE 4th International Conference on Cloud Computing. IEEE, DC. pp 267–274

84. Deshpande U, Wang X, Gopalan K (2011) Live gang migration of virtual machines. In: Proceedings of the 20th International Symposium on High Performance Distributed Computing. ACM Press, California. pp 135–146
85. Lu T, Stuart M, Tang K, He X (2014) Clique migration: Affinity grouping of virtual machines for inter-cloud live migration. In: Proceedings - 9th IEEE International Conference on Networking, Architecture, and Storage. IEEE, Tianjin, China. pp 216–225
86. Lu H, Xu C, Cheng C, Kompella R, Xu D (2015) vHaul : Towards Optimal Scheduling of Live Multi-VM Migration for Multi-tier Applications. In: IEEE 8th International Conference on Cloud Computing. IEEE, New York. pp 453–460
87. Olio Incubation Status - Apache Incubator. <http://incubator.apache.org/projects/olio.html>. Accessed 07 Sept 2017
88. Forsman M, Glad A, Lundberg L, Ilie D (2015) Igorithms for automated live migration of virtual machines. *J Syst Softw* 101:110–126
89. Varga A, Hornig R (2008) An overview of the OMNeT++ simulation environment. In: Proceedings of the 1st international conference on Simulation tools and techniques for communications, networks and systems & workshops. ACM, Marseille. p 60
90. Sun G, Liao D, Anand V, Zhao D, Yu H (2016) A new technique for efficient live migration of multiple virtual machines. *Future Gener Comput Syst* 55:74–86
91. Shribman A, Hudzia B (2012) Pre-Copy and Post-Copy VM Live Migration for Memory Intensive Applications. In: Proceedings of the 18th International Conference on Parallel Processing Workshops. Springer, Rhodes Island. pp 539–547
92. Cerroni W (2014) Multiple virtual machine live migration in federated cloud systems. In: Proceedings - INFOCOM IEEE. IEEE Computer Society, ON. pp 25–30
93. Celesti A, Tusa F, Villari M, Pulaifto A (2010) Improving virtual machine migration in federated cloud environments. In: 2nd International Conference on Evolving Internet. IEEE, Rhode Island. pp 61–67
94. Kumar Bose S, Brock S, Skeoch R, Shaikh N, Rao S (2011) Optimizing live migration of virtual machines across wide area networks using integrated replication and scheduling. In: IEEE International Systems Conference. IEEE, QC. pp 97–102
95. Bose SK, Brock S, Skeoch R, Rao S, Kumar Bose S, Brock S, Skeoch R, Shaikh N, Rao S (2011) CloudSpider: Combining Replication with Scheduling for Optimizing Live Migration of Virtual Machines across Wide Area Networks. In: 11th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing. IEEE, CA. pp 13–22
96. Grid5000. <https://www.grid5000.fr/mediawiki/index.php/Grid5000:Home>. Accessed 08 Sept 2017
97. Redis. <https://redis.io/>. Accessed 08 Sept 2017
98. Zhang X, Huo Z, Ma J, Meng D (2010) Exploiting Data Deduplication to Accelerate Live Virtual Machine Migration. In: IEEE International Conference on Cluster Computing. IEEE, Crete. pp 88–96
99. Jo C, Gustafsson E, Son J, Egger B, Jo C, Gustafsson E, Son J, Egger B (2013) Efficient live migration of virtual machines using shared storage. In: Proceedings of the 9th ACM SIGPLAN/SIGOPS International Conference on Virtual Execution Environments. ACM Press, New York Vol. 48. pp 41–50
100. Wood T, Ramakrishnan KK, Shenoy P, Van Der Merwe J, Hwang J, Liu G, Chaufourrier L (2015) CloudNet: Dynamic pooling of cloud resources by live WAN migration of virtual machines. *IEEE/ACM Transactions on Networking* 23(5):1568–1583
101. Jaswal T, Kaur K (2016) An Enhanced Hybrid Approach for Reducing Downtime, Cost and Power Consumption of Live VM Migration. In: Proceedings of the International Conference on Advances in Information Communication Technology & Computing. ACM, Bikaner
102. Jin H, Li D, Wu S, Shi X, Pan X (2009) Live virtual machine migration with adaptive memory compression. In: Proceedings - IEEE International Conference on Cluster Computing. IEEE, LA. pp 1–10
103. Jin H, Deng L, Wu S, Shi X, Chen H, Pan X (2014) MECOM : Live migration of virtual machines by adaptively compressing memory pages. *Futur Gener Comput Syst* 38:23–35
104. Hacking S, Hudzia B (2009) Improving the live migration process of large enterprise applications. In: Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing. ACM, Barcelona. pp 51–58
105. Svard P, Tordsson J, Hudzia B, Elmroth E (2011) High Performance Live Migration through Dynamic Page Transfer Reordering and Compression. In: IEEE Third International Conference on Cloud Computing Technology and Science. IEEE, Divani Caravel Athens. pp 542–548
106. Sahni S, Varma V (2012) A Hybrid Approach to Live Migration of Virtual Machines. In: IEEE International Conference on Cloud Computing in Emerging Markets (CCEM). IEEE, KA. pp 1–5
107. Cedric JL, Bockhaven V Cryptanalysis of, and practical attacks against E-Safenet encryption. Technical report, University of Amsterdam, Netherlands
108. Nocentino A, Ruth PM (2009) Toward dependency-aware live virtual machine migration. In: Proceedings of the 3rd International Workshop on Virtualization Technologies in Distributed Computing. ACM, Barcelona. pp 59–66
109. Babu BS, Savithramma RM (2016) Optimised pre-copy live VM migration approach for evaluating mathematical expression by dependency identification. *Int J Cloud Comput* 5(4):247
110. Koto A, Yamada H, Ohmura K, Kono K (2012) Towards Unobtrusive VM Live Migration for Cloud Computing Platforms. In: Proceedings of the Asia-Pacific Workshop on Systems. ACM, Seoul. pp 1–6
111. Ma F, Liu F, Liu Z (2010) Live virtual machine migration based on improved pre-copy approach. In: IEEE International Conference on Software Engineering and Service Sciences. IEEE, Beijing. pp 230–233
112. Ibrahim KZ, Hofmeyr S, Iancu C, Roman E (2011) Optimized pre-copy live migration for memory intensive applications. In: International Conference for High Performance Computing, Networking, Storage and Analysis. ACM/IEEE, Seattle. pp 1–11
113. Jin H, Gao W, Wu S, Shi X, Wu X, Zhou F (2011) Optimizing the live migration of virtual machine by CPU scheduling. *J Netw Comput Appl* 34(4):1088–1096
114. Zaw EP, Thein NL (2012) Improved Live VM Migration using LRU and Splay Tree Algorithm. *Int J Comput Sci Telecommun J* 3(3):1–7
115. Yong C, Yusong L, Yi G, Runzhi L, Zongmin W (2013) Optimizing Live Migration of Virtual Machines with Context Based Prediction Algorithm. In: International Workshop on Cloud Computing and Information Security. Atlantis Press, Shanghai. pp 441–444
116. Mohan A, Shine S (2013) An optimized approach for live VM migration using log records. In: 4th International Conference on Computing, Communications and Networking Technologies. IEEE, Tiruchengode. pp 4–7
117. Zhang J, Ren F, Lin C (2014) Delay guaranteed live migration of Virtual Machines. In: Proceedings - IEEE INFOCOM. IEEE, ON. pp 574–582
118. Desai MR, Patel HB (2016) Performance Measurement of Virtual Machine Migration Using Pre-copy Approach in cloud computing. In: Proceedings of the Second International Conference on Information and Communication Technology for Competitive Strategies. ACM Press, Udaipur. pp 1–4
119. Liu H, Jin H, Liao X, Hu L, Yu C (2009) Live migration of virtual machine based on full system trace and replay. In: Proceedings of the 18th ACM International Symposium on High Performance Distributed Computing - HPDC '09. ACM Press, Garching. pp 101–110
120. Liu W, Fan T (2011) Live migration of virtual machine based on recovering system and CPU scheduling. In: 6th IEEE Joint International Information Technology and Artificial Intelligence Conference. IEEE, Chongqing. pp 303–307
121. Hirofuchi T, Nakada H, Itoh S, Sekiguchi S (2010) Enabling Instantaneous Relocation of Virtual Machines with a Lightweight VMM Extension. In: 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing. IEEE, Melbourne. pp 73–83
122. SPECweb2005. <https://www.spec.org/web2005/>. Accessed 09 Sept 2017
123. Ashino Y, Nakae M (2012) Virtual Machine Migration Method between Different Hypervisor Implementations and Its Evaluation. In: Proceedings - 26th IEEE International Conference on Advanced Information Networking and Applications Workshops, WAINA 2012. IEEE, Fukuoka. pp 1089–1094
124. Hines MR, Gopalan K (2007) MemX: supporting large memory workloads in Xen virtual machines. In: Proceedings of the 2nd International Workshop on Virtualization Technology in Distributed Computing. ACM Press, Reno. pp 1–8
125. Anala MR, Shetty J, Shobha G (2013) A framework for secure live migration of virtual machines. In: 2013 International Conference on



- Advances in Computing, Communications and Informatics (ICACCI). IEEE. pp 243–248. doi:10.1109/ICACCI.2013.6637178
126. Oberheide J, Cooke E, Jahanian F (2008) Exploiting live virtual machine migration. BlackHat DC Briefings, Washington DC
  127. Sulaiman NAB, Masuda H (2014) Evaluation of a Secure Live Migration of Virtual Machines Using Ipv6 Implementation. In: 2014 IIAI 3rd International Conference on Advanced Applied Informatics. IEEE. pp 687–693. doi:10.1109/IIAI-AAI.2014.142
  128. Ahmad N, Kanwal A, Shibli MA (2013) Survey on secure live virtual machine (VM) migration in cloud. In: Conference Proceedings - 2013 2nd National Conference on Information Assurance, NCIA 2013. pp 101–106. doi:10.1109/NCIA.2013.6725332
  129. Chen Y, Shen Q, Sun P, Li Y, Chen Z, Qing S (2012) Reliable migration module in trusted cloud based on security level - Design and implementation. In: Proceedings of the 2012 IEEE 26th International Parallel and Distributed Processing Symposium Workshops, IPDPSW. IEEE, Shanghai. pp 2230–2236. doi:10.1109/IPDPSW.2012.275
  130. Aiash M, Mapp G, Gemikonakli O (2014) Secure live virtual machines migration: Issues and solutions. In: Proceedings - 2014 IEEE 28th International Conference on Advanced Information Networking and Applications Workshops, IEEE WAINA 2014. pp 160–165. doi:10.1109/WAINA.2014.35
  131. Wang W, Wang W, Wu X, Lin B, Miao K, Dang X (2010) Secured VM Live Migration in Personal Cloud. In: Proceedings of ICCET, China. doi:10.1.1.232.9449
  132. Kanada Y, Tarui T (2011) A “network-paging” based method for wide-area live-migration of VMs. In: The International Conference on Information Networking. IEEE, Kuala Lumpur. pp 268–272
  133. Singh G, Gupta P (2005) A Review on Migration Techniques and Challenges in Live Virtual Machine Migration. In: 5th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO). IEEE, Noida. pp 542–546
  134. Google Cloud Platform Blog: Google Compute Engine uses Live Migration technology to service infrastructure without application downtime. <https://cloudplatform.googleblog.com/2015/03/Google-Compute-Engine-uses-Live-Migration-technology-to-service-infrastructure-without-application-downtime.html>. Accessed 07 Sept 2017

**Submit your manuscript to a SpringerOpen<sup>®</sup> journal and benefit from:**

- Convenient online submission
- Rigorous peer review
- Open access: articles freely available online
- High visibility within the field
- Retaining the copyright to your article

---

Submit your next manuscript at ► [springeropen.com](http://springeropen.com)

---