J Supercomput (2015) 71:2473–2515
DOI 10.1007/s11227-015-1400-5

Virtual machine migration in cloud data centers: a review, taxonomy, and open research issues

Raja Wasim Ahmad · Abdullah Gani · Siti Hafizah Ab. Hamid · Muhammad Shiraz · Feng Xia · Sajjad A. Madani

Published online: 13 March 2015

Abstract Virtualization efficiently manages the ever-increasing demand for storage, computing, and networking resources in large-scale Cloud Data Centers. Virtualization attains multifarious resource management objectives including proactive server maintenance, load balancing, pervasive service availability, power management, and fault tolerance by virtual machine (VM) migration. VM migration is a resource-intensive operation as it constantly requires adequate CPU cycles, memory capacity, system cache, and network bandwidth. Consequently, it adversely affects the performance of running applications and cannot be entirely overlooked in contemporary data centers, particularly when user SLA and critical business goals are to be met. The unavailability

R. W. Ahmad (✉) · A. Gani · S. H. Ab. Hamid
Center for Mobile Cloud Computing (C4MCC), FSKTM,
University of Malaya, 50603 Kuala Lumpur, Malaysia
e-mail: wasimraja@siswa.um.edu.my

A. Gani
e-mail: abdullah@um.edu.my

S. H. Ab. Hamid
e-mail: sitihafizah@um.edu.my

M. Shiraz
Department of Computer Science, Federal Urdu University of Arts,
Science and Technology, Islamabad, Pakistan
e-mail: muh_shiraz@yahoo.com

F. Xia
School of Software, Dalian University of Technology, Dalian, China
e-mail: f.xia@acm.org

S. A. Madani
COMSATS Institute of Information Technology, Islamabad, Pakistan
e-mail: madani@ciit.net.pk

of a comprehensive survey on VM migration schemes that covers various VM migration aspects such as migration patterns, sequence, application performance, bandwidth optimization, and migration granularity has motivated this review of existing schemes. This paper reviews state-of-the-art live and non-live VM migration schemes. Through an extensive literature review, a detailed thematic taxonomy is proposed for the categorization of VM migration schemes. Critical aspects and related features of current VM migration schemes are inspected through detailed qualitative investigation. We extract significant parameters from existing literature to discuss the commonalities and variances among VM migration schemes. Finally, open research issues and challenges with VM migration that require further consideration to develop optimal VM migration schemes in Cloud Data Centers are briefly addressed.

Keywords Virtualization · VM migration · Post-copy · Pre-copy · Data Centers

1 Introduction

The rapidly increasing demands of modern resource rigorous enterprise and scientific applications have stimulated the conception of large-scale Cloud Data Centers (CDC). Usually, CDCs are over-provisioned to guarantee absolute service reliability and availability [1]. However, on average, 30 % of cloud servers persistently remain idle the majority of time and often employ 10–15 % of their resource capacity [2]. Resource underutilization results in truly phenomenal growth of cloud operational cost and energy consumption [3,4]. According to Environmental Protection Agency (EPA) report [3], it was tentatively estimated that by 2011 CDCs would consume 61 TWh of electricity. In 2013, it was roughly estimated that Google data centers devoured 260 million Watts of electricity, which is 0.01 % of the global energy—an ample amount to consistently power 200,000 homes [5,6]. As today's clouds have adequate processing power, virtualization technology is chosen to consolidate several workloads onto a single physical server to significantly improve cloud resource utilization and power efficiency. Motivated by multitasking and time-sharing systems, virtualization has significantly improved the Return on Investment (ROI) due to optimal resource usage. Furthermore, virtualization technology scales CDC by migrating virtual servers across physical servers to achieve various resource management objectives such as server maintenance [7], load balancing [8], power management, and fault tolerance [9–11].

The CDC embodies a relatively large group of networked servers and storage farms offering highly reliable services to organizations for remote storage, processing, or distribution of fairly large-scale data [12]. CDC provides a foundation for the cloud computing paradigm [13–15] to offer elastic cloud services to its subscribers [15,16]. More recently, scientists, developers, and business personnel have begun to efficiently utilize cloud service models encompassing Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) to adeptly meet business goals [12]. Virtualization technology proficiently manages CDC through highly dynamic resource provisioning, effective resource monitoring, automated security, and systematic change request handling [17] to efficiently utilize underlying resources [18].

Virtualization runs a software layer between the hardware and operating system, known as a virtual machine monitor (VMM) or hypervisor, to control and map numerous VMs (applications running inside OS) on a single platform [19–21]. The hypervisor dexterously manages shared hardware resources and entertains OS requests to access hardware resources. Among characteristics that set the hypervisor apart include support for the legacy operating system, fault tolerance, and performance isolation to augment CDC performance. VMM supports the legacy operating system to unify several underutilized server workloads onto a single server. It also guarantees that the failure of one VM does not impact the proper functioning of the entire physical machine [22]. However, co-hosting multiple VMs onto a single physical server is very challenging due to resource contention among co-hosted applications and system performance degradation attributed to server overutilization [21,23–26].

To resolve the issues stated above, the hypervisor carefully selects and migrates VMs from over-utilized to underutilized servers. The role of VM migration is twofold. To achieve energy consumption, it unifies various servers' workloads onto a few physical servers. Then, to improve an application's performance, the hypervisor migrates the workload from a low-performance server to a high-performance server [27]. As additional resources are consumed during the VM migration process, the performance of applications running within migrant VMs is severely affected until VM migration completes. Consequently, the VM migration process must be successfully completed within nominal migration time (to free system resources as soon as possible) while utilizing optimal server and network resources to enhance application performance, bandwidth utilization efficiency, and migration transparency [19,28,29].

The hypervisor effectively exploits live [30] or non-live [31] VM migration patterns to move VMs between corresponding servers either using dedicated or shared system resources. A live VM migration pattern continues servicing the application during elapsed VM migration time to achieve seamless connectivity, avoid service level agreement (SLA) violation, and attain optimal resource utilization. Non-live VM migration halts application services during VM migration and offers predictable VM migration and service downtime. For efficient resource management, the VM migration controller either migrates a single VM [32] or a complete VM cluster (multiple VMs) [30] across LAN or WAN links. However, VM migration within LAN [30] is easy to manage since storage migration is not required owing to the network-attached storage (NAS) integrated CDC architectures. Furthermore, network management within LAN only requires limited management effort because the IP address remains unchanged for the outside world. Alternatively, VM migration over a WAN link [33] considerably prolongs VM migration duration due to storage migration, IP management, limited available bandwidth, network congestion, and the error-prone nature of WAN links.

To the best of our knowledge, there exist only a few surveys that underline the significance of VM migration within CDC. In [28], the authors surveyed performance overhead of virtualization within a physical server, intra-CDC and inter-CDC VM migration. Nevertheless, this survey is lacking in briefly discussing the non-live VM migration methods. Furthermore, the proposed taxonomy does not cover the various vital aspects of VM migration technology, including VM migration pattern, execution resource constraints, and VM migration granularity to highlight the trade-off between resource consumption and application performance during VM migration process.

Similarly, a survey covering diverse aspects of process migration, memory migration, and suspend/resume based VM migration schemes have been reported in [22]. However, the authors considered only a very few VM migration approaches with inadequate analysis of VM migration approaches. Further, the authors did not consider performance aspects of running applications during VM migration, network optimization, and hybrid VM migration pattern to improve migration processes. In comparison to the above-mentioned studies [22,28], our review is more detailed and comprehensive. Our proposed classification and analysis are more diverse and cover various aspects of VM migration, which are missing in reported literature such as optimization applied by pre-copy, post-copy, and hybrid methods to optimize the network, running, and co-hosted application's performance. Moreover, the offline VM migration schemes are also thoroughly investigated to highlight their strengths and weaknesses.

The main contribution of this article is to comprehensively review state-of-the-art VM migration schemes while considering several essential VM migration aspects, including highlighting the trade-off between application performance and total migration time, network performance optimization, offline VM migration, pre-copy and post-copy VM migration pattern, and hybrid methods to meet resource management objectives within CDC. We critically reviewed state-of-the-art VM migration schemes and highlighted their strengths, weaknesses, and issues that need further research. A novel taxonomy for VM migration schemes is proposed to classify existing literature. Furthermore, we briefly discussed several applications of VM migration process and parameters that affect VM migration process. The highly critical aspects and significant features of the existing VM migration schemes are investigated through qualitative analysis. We derive key parameters from the literature for comparisons of VM migration schemes. The comparison parameters include: (1) Migration Pattern, (2) Bandwidth Utilization Efficiency, (3) App QoS Degradation Duration, (4) Network Link, (5) Migration Granularity, (6) Migration Duration, (7) Downtime, (8) Execution Resource Constraints, (9) Hypervisor Type, and (10) Network Management Policy. Finally, open research issues and challenges in VM migration schemes are discussed that need further investigation to develop optimal VM migration schemes to increase body of knowledge in this domain of study.

The rest of the paper is structured as follows. Section 2 presents discussion on CDC, existing virtualization technologies, and VM migration process. Section 3 discusses the taxonomy for the classification of current VM migration schemes, the performance parameters, and VM migration technologies. Section 4 presents detailed review on state-of-the-art VM migration schemes, analysis of existing VM migration schemes, and comparisons of recent VM migration schemes based on selected parameters. Conclusions and future directions are followed by a brief discussion on open research issues and challenges in Sects. 5 and 4, respectively.

2 Background

This section defines and discusses the notion of CDC, virtualization technologies, VM migration process, and VM migration use cases within a CDC. For the ease of readers, we provided a list of the most frequently used acronyms in the paper in Table 1.

Table 1 List of acronyms

Symbol	Description
SLA	Service level agreement
VM	Virtual machine
CDC	Cloud Data Centre
ICT	Information and communication technology
CC	Cloud computing
KVM	Kernel-based virtual machines
3TA	Three tier architecture
QoS	Quality of Service
MDC	Modular Data Centre
SaaS	Software as a Service
PaaS	Platform as a Service
IaaS	Infrastructure as a Service
NAS	Network-attached Storage
SVM	Secure virtual machine
HPC	High-performance computing
VMM	Virtual machine monitor
PM	Physical machine
WAN	Wide area network
LAN	Local area network
ISR	Internet Suspend and Resume
NSF	Network file system
Pod	Process domain
DM	Dirty memory
IRLM	Inter-raked live migration
RLE	Run length encoding
SSA	Slowdown scheduling algorithm
LZO	Lempel–Ziv–Oberhumer
MBFD	Modified best fit decreasing algorithm
GA	Genetic algorithm
LP	Linear programming
RDMA	Remote direct memory access
IDTS	Inter-domain data transfer system

2.1 Cloud Data Centers

Cloud Data Centers is a pool of heterogeneous computing and storage components clustered together using very fast communication links to host numerous applications and to store data [34]. A cloud operator charges the customer for cloud resource usage based on the "pay-as-you-go" service model [35]. This service model certifies that the customer pays only for the resources they have actually used in the specified time duration [36]. Moreover, CDC offers a wide range of service types to be accessed via web links [36–40]. However, cloud computational cost and application performance are profoundly affected due to internal resource fragmentation [41] (low server utilization) and bandwidth constraints imposed by the CDC network architecture design [34,42].

Network architecture is a vital element in CDC network design since it significantly influences CDC throughput [34,43]. Modern data center network architecture designs are based on the hierarchical tree-based Three-Tier architecture (3TA). In three-tier architecture, the core layer connects CDC to the Internet backbone, the aggregation layer implements assorted functionalities such as firewalls and content switching, and the access layer ensures inter-rack connectivity [34,44]. According to network routing protocol design, CDC architecture is classified as switch-centric (e.g. fat tree architecture [10,34,45]), server-centric (e.g. B-Cube [46]), and hybrid models (e.g. DCell [44]).

Figure 1 highlights a set of services offered by CDC to its consumers. As depicted in Fig. 1, the cloud operator controls and manages the cloud services, including PaaS, SaaS, and IaaS. Among others, the SaaS service model offers accounting applications, including e-commerce, office automation, and knowledge management services. Likewise, IaaS model encompasses hardware resources such as storage, processor, and network. Alternatively, PaaS service model offers several cloud execution platforms, such as developer studio, database management software (DBMS), groupware, and operating systems (OS) to help the IT professionals to develop, deploy, test, debug, and host sophisticated web applications. Within a CDC, virtualization proficiently manages the cloud resources to effectively offer aforementioned services to the customer.

2.2 Virtualization

The primary objective of virtualization technology is to proficiently share the highly expensive hardware resources among a number of OSs. To efficiently share system resources, virtualization abstracts the underlying hardware resource by placing a software layer between OS and the hardware. VMM evades application privacy violation by running numerous VMs in an isolated, secure environment to meet the SLA. The key managerial responsibilities of VMM include hardware resource control, resource allocation, OS instruction translation, OS handling, and OS interrupt processing. Based on VMM architectural design, the hypervisors are categorized either as Type 1 or Type 2 [20,47]. Compared with Type 2, Type 1 hypervisors are remarkably efficient as they directly access hardware resources. However, Type 2 hypervisors run as a process within a conventional OS environment and rely on host OSs to access hardware resources [20,21,47–52].

Xen [19], VMware ESX [20], Oracle VirtualBox [21], and kernel-based Virtual Machine (KVM) [52] are highly accepted virtualization technologies with common features [21,25,26]. Hypervisors support para and full virtualization. Furthermore, all hypervisors [19–21,52] mentioned above have x86-64 system architecture built-in support. Besides architectural support, Xen is more complicated (hypervisor usability feature) relative to the other aforementioned hypervisors due to its highly complex I/O interrupt handling method [19]. Xen upholds large-size memory and a fine number of processor cores than KVM [52,53], VirtualBox, and VMware [20], to host more

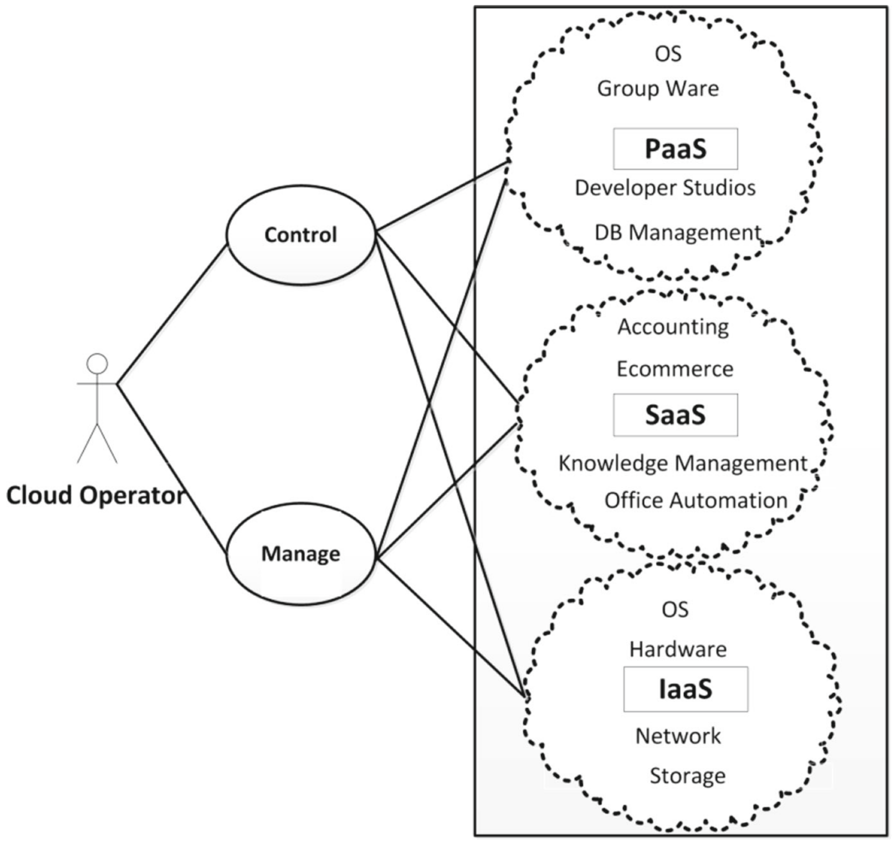

Fig. 1 Cloud service models

VMs. Likewise, Xen is more energy efficient than KVM due to its inner structure for handling I/O operations [19]. Hypervisors including Xen, KVM, VMware ESX, and VirtualBox exploit the VM migration feature to capably manage cloud resources within inter- and intra-CDC architectures [54].

2.3 Virtual machine migration

Virtualization has become a fundamental element in today's CDC owing to the support of isolating, consolidating and migrating server workload. VM migration schemes migrate the state of virtual devices (e.g. memory, CPU, I/O) between physical hosts during the VM migration process [55,56]. The hypervisors migrate a VM either in non-live or live communication mode [57]. The non-live VM migration mode does not service the migrant applications during VM migration [58] but the live VM migration mode does [54].

Virtual machine migration schemes seek to upgrade manageability, fault tolerance, and application's performance within a CDC. Figure 2 illustrates a general overview

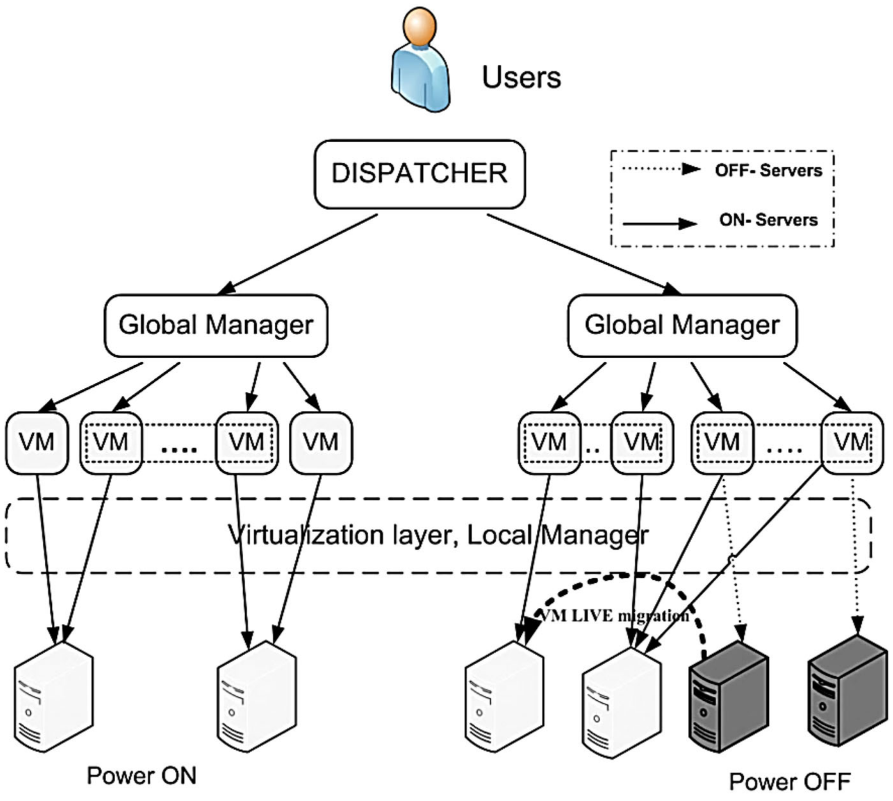

Fig. 2 An overview of VM migration process on distributed CDC

of a distributed CDC framework, wherein a VM is migrated (live mode) from an underutilized server to a resource-rich server to power off the former for efficient resource utilization. The global manager manages a set of physical servers (preferably a cluster) and creates VM for the new application in response to the request by the dispatcher module. However, the local manager examines the system components' health and recommends the global manager to migrate a few VMs if required. In response, the global manager signals the VMM to trigger VM migration. Figure 2 presents VM migration between servers located under same administrative domain; however, VM migration can be triggered across the CDCs owned and managed by different providers (e.g. WAN migration) [3,59].

The VM migration technology assists to achieve various resource management objectives as shown in Fig. 3. A detailed discussion on the applications of VM migration is as explained below:

- *Power management* To attain power efficiency within a CDC, VM migration process shifts complete server workload from an under-loaded server (e.g. when resource usage below a threshold) to an underutilized server to switch off the former. Server consolidation and DVFS enabled VM migration methods aggressively co-locate the VMs and decrease CPU clock rate, respectively, to achieve

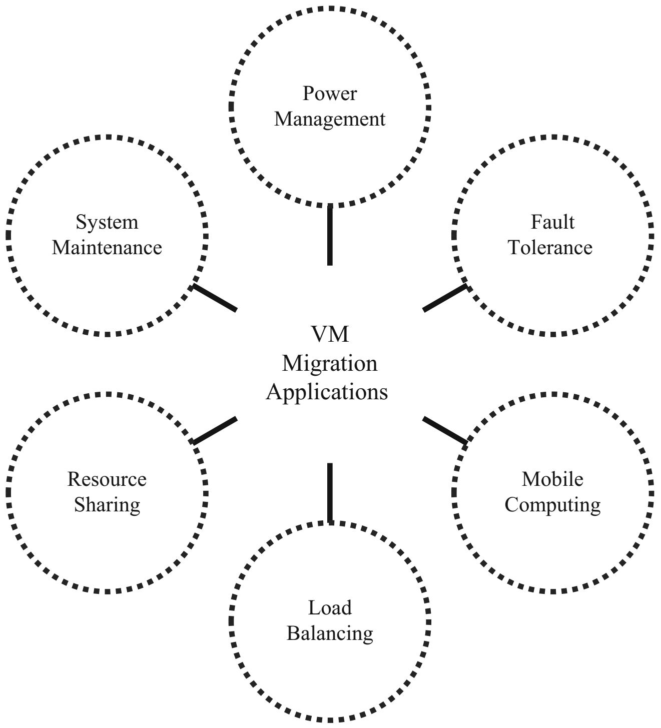

Fig. 3 VM Migration applications

power efficiency within a CDC at the cost of application's performance degradation [1,60].

- *Resource sharing* The application performance degradation issue due to sharing limited system resources, such as system memory, cache, or CPU cycles, can be resolved by relocating resource hungry VM to a resource-rich server [61,62]. However, high system resource sharing reduces cloud operational cost as unnecessary servers can be switched off [63,64].
- *Fault tolerance* A fault-tolerant system triggers VM migration prior to fault happening. It migrates back VM to the original server after system maintenance endowment, if necessary [65–67]. A fault-tolerant system vastly improves the system availability to enhance the CDC reliability feature [56].
- *System maintenance* Provisioning the periodic/dynamic maintenance extends system life time [25,68]. VM migration technology shifts running application to another host to continue servicing the application during system maintenance time [69].
- *Load balancing* Load balancing helps cloud operator to avoid single point of failure by distributing the server workload across several physical hosts within a CDC. Server's workload beyond its capacity degrades system performance; thus, load balancing approaches (using VM migration) reduce the possibility of application performance degradation by eliminating hot-spots within CDC [70].

- *Mobile computing* Mobile computing exploits VM migration technology to augment portable computing capabilities. Nowadays, users do not prefer to work on desktop computers only. Rather, they prefer to work on smart phones while they are on the move. VM migration technology helps a user to migrate running applications along OS states from a desktop server to smart phone or vice versa [35].

3 A taxonomy of VM migration schemes

This section highlights and discusses a thematic taxonomy for the classification of current VM migration schemes. We classified the existing VM migration schemes based on seven prominent characteristics, which are VM migration pattern, objective function, network link, hypervisor type, execution resource constraints, migration granularity, and network management as shown in Fig. 4.

VM migration pattern specifies the main method employed to migrate a VM from a source server to the target server. Migration patterns are categorized as live or non-live VM migration. A live VM migration pattern does not suspend application service during VM migration, whereas a non-live pattern follows pause, copy, and resume methodology to migrate a VM. An objective function highlights the primary objective of VM migration schemes. VM migration schemes aim for a number of objectives, including (1) minimizing service downtime, (2) optimizing the migration duration, (3) minimizing the duration of application QoS degradation, and (4) optimal bandwidth resource utilization during a VM migration process. The downtime attribute defines the time during which service is unavailable for current hosted applications. Similarly, migration duration specifies the time period between the start of VM migration and

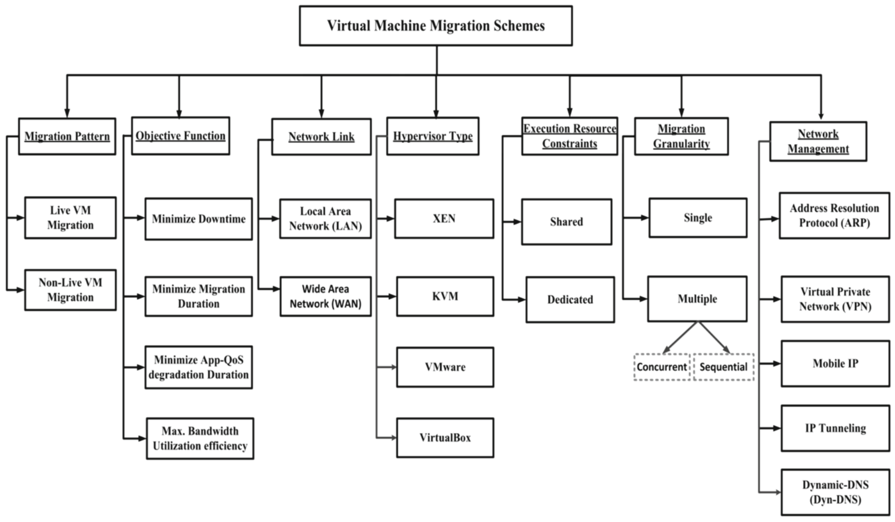

Fig. 4 VM Migration thematic taxonomy

the time when a migrant VM becomes consistent with the VM at the source server. During a VM migration process, the service quality of hosted applications is rigorously affected if migration is not properly handled. Application QoS degradation duration is the time period during which the application consistently experiences performance degradation due to service response delay, high packet drop rate, jitter variations, and extended migration time.

The network link metric specifies the network type selected to migrate a VM. VM migration approaches migrate the VM within LAN or WAN boundaries to scale the CDC. The WAN attribute of the network link metric offers limited bandwidth, higher latency, and physically distant servers. However, the LAN attribute is considered ideal for migration schemes as there is no need to migrate the storage. The hypervisor type parameter specifies the hypervisor technology that manages server resources during VM migration. The majority of migration schemes choose Xen as a management layer because Xen is flexible, reliable, and efficient in terms of resource management. Furthermore, the execution resource constraints metric specifies whether the proposed VM migration scheme borrows required system resources from co-hosted applications (shared resources) or dedicated resources are assigned to the migrant VM. The migration granularity describes the granularity level at which the migration scheme migrates the VM(s). For example, some VM migration schemes migrate a single VM while others migrate multiple VMs at a time. The attributes of the network management parameter identify the method of planning, tracking, and managing network addresses upon VM migration completion. Migrating a VM across a WAN link requires network redirection due to changes in IP attachment.

3.1 Live VM migration

Live VM migration pattern un-interruptedly services the running applications during the elapsed VM migration time. The key objectives of live VM migration pattern include: (1) optimizing the application performance during VM migration process [71], (2) augmenting the bandwidth utilization efficiency [32], and (3) minimizing the downtime [72]. Bandwidth optimization methods augment the application's QoS while putting extra burden on system resources (e.g. CPU cycles) [56].

Figure 5 illustrates taxonomy of live VM migration pattern. Live VM migration schemes have three categories, which are pre-copy VM migration, post-copy VM migration, and hybrid methods. The network performance and application performance categories of the pre-copy VM migration patterns are based on the objectives of pre-copy VM migration schemes [58]. Pre-copy VM migration pattern iteratively copies memory pages to the destination server to minimize application service downtime. Alternatively, post-copy method transfers minimum system state to optimize the total migration time and outperforms when write intensive applications are hosted within migrant VM [56]. Similarly, a hybrid method augments application performance by considering characteristics of both pre-copy and post-copy VM migration pattern. However, a non-live VM migration design suspends the application execution during VM migration process that implicates the Internet suspend resume and process domain [73].

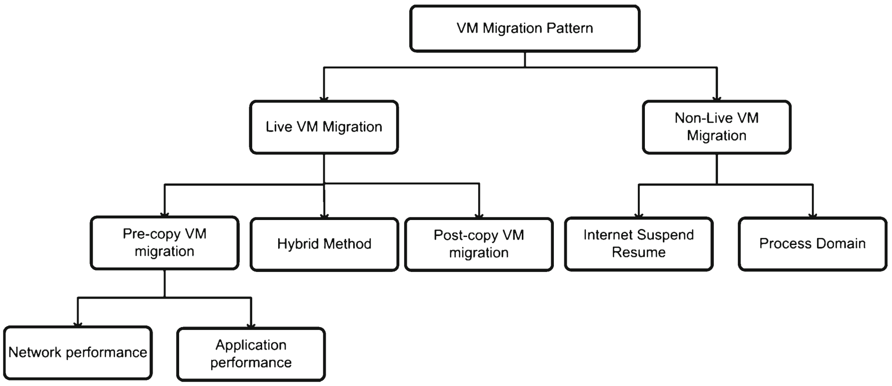

Fig. 5 VM migration pattern-based taxonomy

3.1.1 Pre-copy VM migration

A pre-copy VM migration pattern iteratively copies VM memory pages (intra-servers) until some suitable termination criterion is met. Using a pre-copy VM migration design, total migration time is relatively higher; subsequently, network and system resources remain occupied for the longer period of time to transfer dirty memory pages [57,71,74]. The complete pre-copy VM migration process consists of several rounds, wherein dirtied memory pages are proactively updated on the receiver server when dirty memory pages are pushed from the source server [29,54,75]. Further, a memory page is marked as dirtied (during pre-copy stages) if it is updated by the application during the recent round. The entire pre-copy VM migration process is divided into six key stages, including (1) target host selection, (2) resource reservation, (3) iterative pre-copying rounds, (4) stop and copy phase, (5) commitment, and (6) VM activation at the target server [29].

An abstract overview of pre-copy VM migration working flow is illustrated in Fig. 6. In the stated figure, VM memory is labelled as a set of memory pages (e.g. P1, P2, P3), whereas dirty memory is marked as DM (e.g. Dp1, Dp2, Dp3). After VM migration initiation, during the first round, the complete VM memory is marked as dirty memory. In the subsequent rounds, previous round dirty memory pages are migrated in parallel to dirty pages logging [57]. Furthermore, during iterative rounds, the hypervisor pursuits for some appropriate timings (e.g. the amount of total dirty memory is beyond a pre-defined threshold) to trigger stop and copy phase to terminate pre-copy rounds. During stop and copy phase, the migration controller identifies and migrates the remaining dirty memory pages along with the system state to the receiver server. Commitment and activation phases redirect the network links once VM is resumed at destination [71].

Based on the objective parameter, pre-copy VM migration schemes can be categorized into network performance and application performance classes as discussed in the following section.

a. Network Performance VM migration process migrates several hundred gigabyte (GB) of memory during VM migration process. However, migration of such a massive

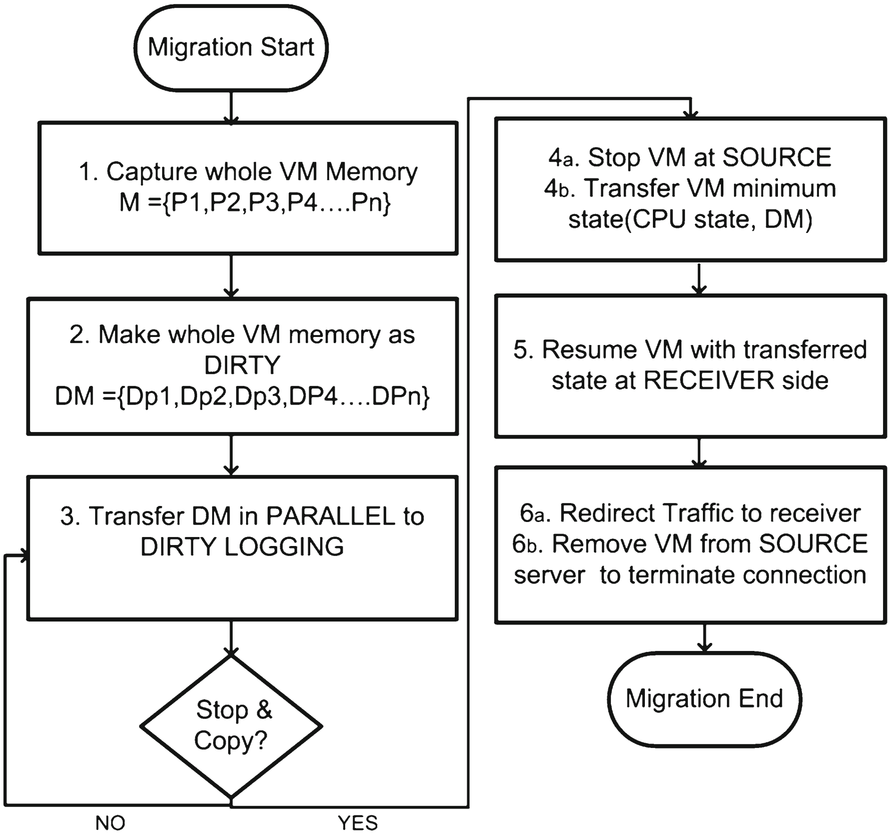
Migration Start
1. Capture whole VM Memory M ={P1,P2,P3,P4….Pn}
2. Make whole VM memory as DIRTY DM ={Dp1,Dp2,Dp3,DP4….DPn}
3. Transfer DM in PARALLEL to DIRTY LOGGING
Stop & Copy?
NO
YES
4a. Stop VM at SOURCE
4b. Transfer VM minimum state(CPU state, DM)
5. Resume VM with transferred state at RECEIVER side
6a. Redirect Traffic to receiver
6b. Remove VM from SOURCE server to terminate connection
Migration End

Fig. 6 Abstraction of pre-copy VM migration

amount of data engages entire network capacity for several minutes and degrades application performance (if the migration process is not properly managed). The VM migration schemes exploit network bandwidth optimization methods to reduce VM memory size [54].

b. Application Performance In case of improper resources management, pre-copy migration pattern adversely affects the system resources. Due to high resource sharing among migration daemon and hosted applications, the probability of SLA violation is high due to application performance degradation during VM migration process. Similarly, selecting a non-optimal pre-copy's stop and copy triggering time also degrades the application performance.

3.1.2 Post-copy VM migration

A post-copy VM migration process is initiated by suspending the VM execution at the source server. During the suspension stage, the minimum state (e.g. non-page-able memory, registers, CPU state, etc.) of VM is captured. Afterwards, the complete VM migration process is divided into three phases. During the first phase, the captured

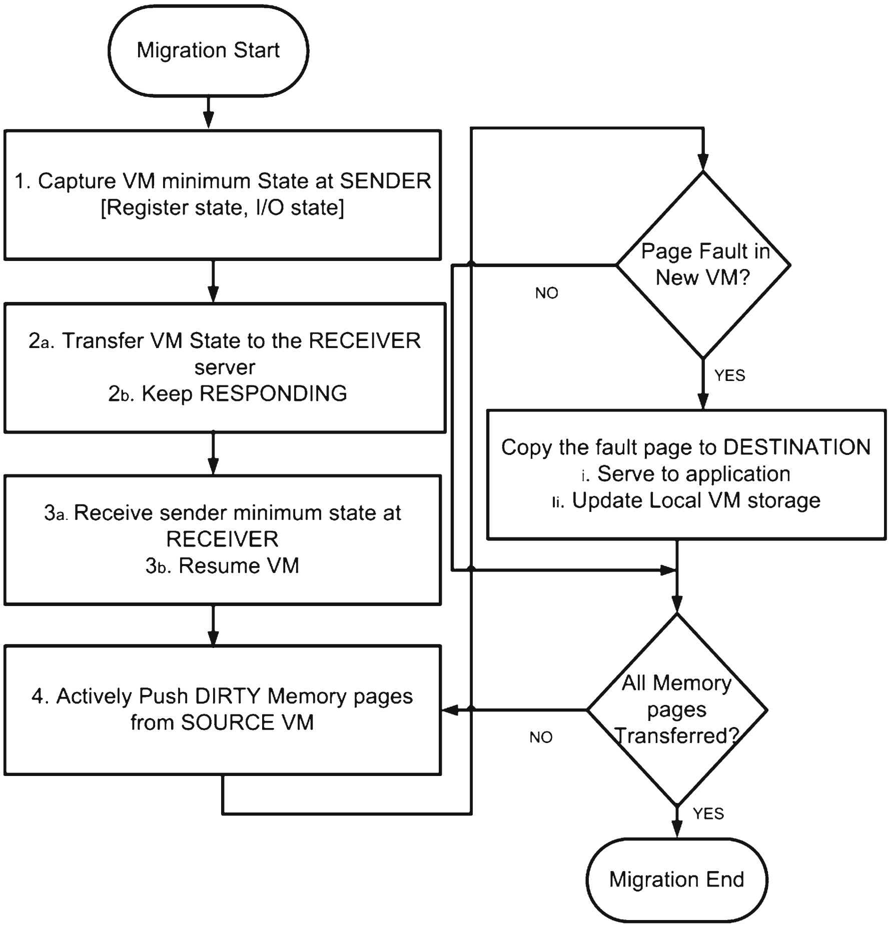
Migration Start
1. Capture VM minimum State at SENDER [Register state, I/O state]
2a. Transfer VM State to the RECEIVER server
2b. Keep RESPONDING
3a. Receive sender minimum state at RECEIVER
3b. Resume VM
4. Actively Push DIRTY Memory pages from SOURCE VM
Page Fault in New VM?
NO
YES
Copy the fault page to DESTINATION
i. Serve to application
Ii. Update Local VM storage
All Memory pages Transferred?
NO
YES
Migration End

Fig. 7 Post-copy VM migration flow diagram

minimum state is transferred to the target server. In the second phase, VM is resumed at the target server with transferred VM states. The third step is highest time consuming (among three phases) as it fetches memory pages from the source server based on the read/write requests by the migrant application. Furthermore, during this phase, based on read/write request, VM service is temporarily suspended until required memory pages are fetched from the source server (if the pages are not already fetched) [29,57,76].

As illustrated in Fig. 7, to service application's request, the target server VM fetches memory pages from the source server (if pages are not already fetched). Furthermore, at receiver server, memory page is served to the running application after updating the local storage database. The connection to the source server remains alive until complete synchronization of the corresponding VMs. When complete memory is updated at receiver side, then connection to the source server is detached.

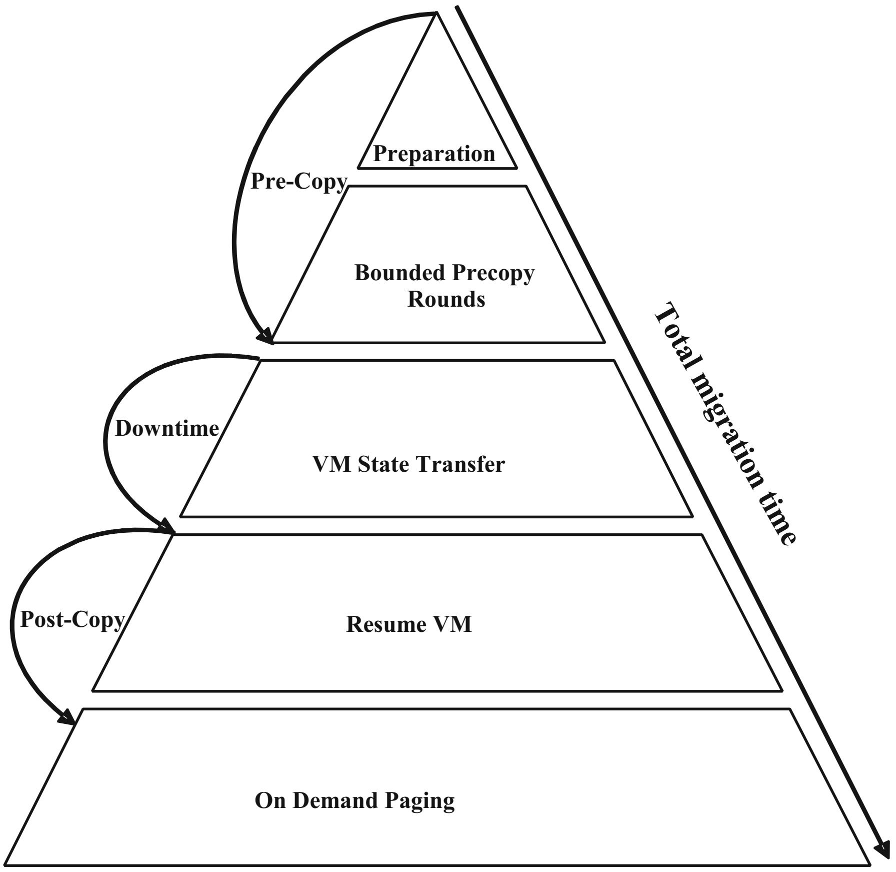

Fig. 8 Hybrid live VM migration overview

3.1.3 Hybrid methods

A hybrid VM migration scheme combines the characteristics of both pre-copy and post-copy VM migration patterns to improve VM migration performance. In comparison to post-copy VM migration pattern, a hybrid method exploits bounded pre-copy rounds to identify and transfer VM working-set to lessen the network I/O pages fault rate. After completing bounded pre-copy rounds, post-copy VM migration method captures and transfers VM minimum state to the target server to resume VM.

The hybrid VM migration process is based on five phases, including preparation, bounded pre-copy rounds, VM state transfer, VM resume, and on demand paging, as illustrated in Fig. 8. During the preparation phase, the required system resources are reserved at the target server. Delimited pre-copy rounds identify and transfer VM working-set to the receiver server. Furthermore, after bounded iterative rounds, VM minimum state is captured and transferred to the receiver server to resume VM at the target server. VM resume phase launches VM from the transferred state at the receiver server. In the final phase, based on application read/write requests, VM memory pages are fetched from the source server to synchronize both VMs to demolish connection to the source server as soon as possible.

3.2 Non-live VM migration

A non-live VM migration process discontinues the application execution while migrating VM memory state from the source server to the target server. Non-live VM migration pattern [77] does not resume the VM (at target server) until complete VM is transferred. Process migration [78] provides the foundation for non-live VM migration pattern. Conversely, non-live VM migration schemes as compared to process migration migrate application along all its execution states to eliminate the residual dependency issue [79]. Non-live migration schemes such as the Internet Suspend Resume [31] and Process Domain [73] lead to application QoS degradation (especially for interactive web applications) due to service disconnection during VM migration process [58,80]. Considering the advantages, a non-live VM migration pattern offers predictable migration time and also guarantees that VM memory pages are transferred exactly once during the VM migration process [21,66,71].

Figure 9 illustrates an abstract overview of non-live VM migration process using a state diagram, wherein VM halts (suspends) a running application when it receives a VM migration request. During suspend state, the hypervisor estimates and reserves required system resources for the VM at the receiver host. Hypervisor, at the source server, migrates VM memory until all memory pages reach the receiver server. When complete VM memory is transferred, then VM enters into the resume state to restart VM at the receiver side. The connection with the source is demolished when VM is resumed.

4 Review on VM migration schemes using thematic taxonomy

This section discusses state-of-the-art live and non-live VM migration schemes.

a. Pre-copy VM migration schemes

This section discusses the pre-copy-based VM migration schemes to optimize the network performance or application performance during VM migration process.

i. Network performance The consolidated server workloads share the network bandwidth capacity during their I/O activities within CDC. Due to larger VM memory size, VM migration daemon demands plenty of bandwidth capacities to migrate VM across the CDC. Consequently, the VM migration daemon suppresses the VM memory contents to reduce the required bandwidth to effectively utilize the network bandwidth capacity.

Inter-raked live migration (IRLM) [30] has efficiently utilized the network bandwidth capacity while concurrently migrating a bunch of VMs across the CDC. IRLM employed data deduplication (eliminating duplicate data) to reduce the VM memory size by identifying, tracking, and avoiding transfer of identical memory pages. During

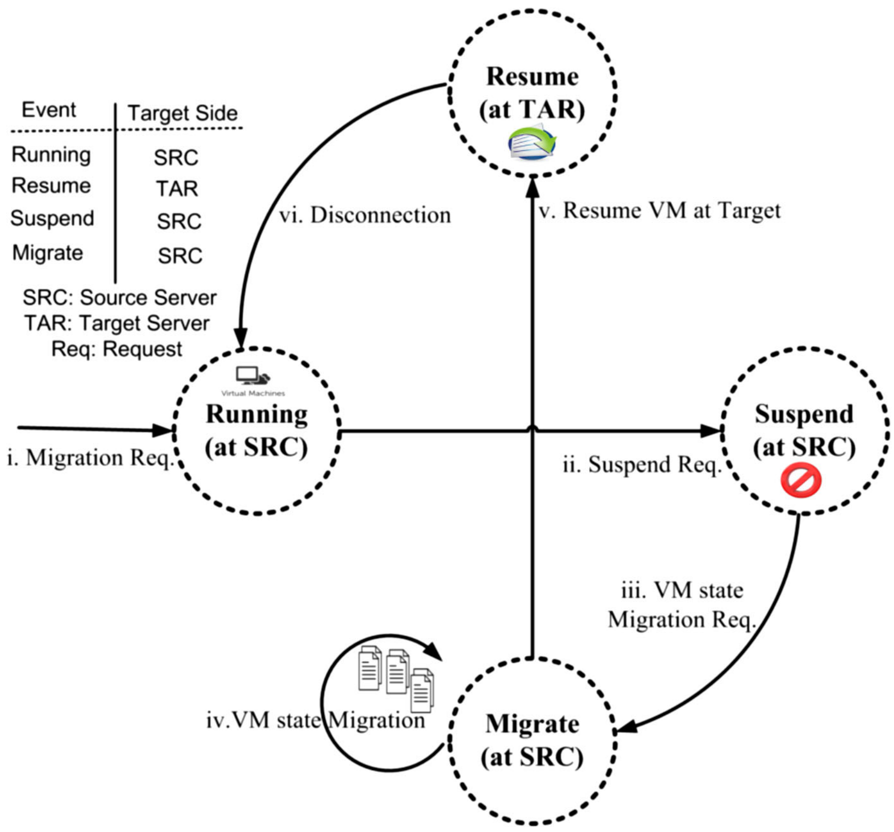

Fig. 9 Non-live VM migration events diagram

pre-copy preparation phase, IRLM exploited a controller module (at sender side) to identify similar memory pages both within and across servers using QEMU/KVM process. Furthermore, to identify the memory similarity ratio at rack level, IRLM implemented an index server to hold hash values of memory pages. Prior to memory page transferring, QEMU/KVM process captures VM memory page status by querying index server. In response, controller module transfers a memory page identifier if the similar memory page is already sent by any of the VM located within the source rack. On the other hand, at the receiver side, the controller module copies the received memory page or identifier into the shared memory space of legitimate target servers. However, the proposed scheme is complex and compound as it calculates 160 bit hash value of every memory page within each VM. Furthermore, it degrades the application QoS due to resource contention and also it did not consider partial page similarity.

In [81] (similar to [30]), a deduplication-based bandwidth optimization scheme is proposed to concurrently migrate a bunch of VMs across the racks. The proposed scheme has exploited a controller thread to identify the page similarity ratio among co-hosted VMs. Further, to evade likelihood of hash collisions, identical memory pages are compared at byte level. However, compared to IRLM [30], the authors considered partial contents similarity among VM memory pages to further optimize bandwidth capacity. To calculate partial page similarity, the proposed scheme generates a delta page using current and reference pages (using cache). Furthermore, during VM migration phase, the controller ensures that only one instance of the identical memory pages is transferred. For the subsequent dirty memory pages, rehashing is exploited either during preparation time (periodic deduplication) or migration time (online deduplication). Although, dirty page rehashing and delta compression have improved the network performance but it consumes extra CPU cycles. Furthermore, the migration preparation phase, being resource-intensive operation, affects the application performance during synchronizing the VMs. However, the proposed migration scheme is lacking in considering VM migration across CDC hosting heterogeneous OSs and workloads.

Pre-copy migration pattern degrades application performance while hosting write intensive workloads within migrant VM during VM migration process. Dynamic rate limiting [29] lessens the application write rate to prioritize the VM migration process. However, prioritizing VM migration over running applications badly impacts application performance. Empirically study revealed that during application execution (within a VM) same portion of VM RAM is dirtied again and again in many cases. Therefore, delta-based compression can efficiently improve the overall network performance [32]. The proposed scheme has employed binary XOR-based run length encoding (XBRLE) delta compression method to improve VM migration performance. Furthermore, RLE compression method is integrated to compress delta pages to further optimize the network bandwidth utilization. Alternatively, at the target server, the controller module decompresses the VM memory pages accordingly. However, XBRLE-based migration scheme requires adequate system cache while hosting memory intensive workloads. Moreover, compressing/decompressing the VM memory pages also consumes system resources. A similar approach is discussed in [82] that exploits VM self-similarity ratio and hashing-based fingerprints to identify and track memory similarity.

The compression enabled VM migration scheme, called MEMCOM (memory compression) [75], assumes that adequate abundant system resources are available (during VM migration process) to compress memory pages using multithreading approach. As one compression method cannot work proficiently for all types of workloads, therefore, MEMCOM has selected the characteristics-based compression (CBC) method to condense VM memory size. The CBC has exploited 16 word dictionaries to classify VM memory pages into zero bytes, high similar and low similar memory pages. Furthermore, to optimize the VM migration process, CBC adaptively selects the fast compression algorithm to compress high similar and zero byte pages. Alternatively, to compress low similar memory contents, CBC chooses the high compression rate standards. Furthermore, based on the network behaviour, MEMCOM adaptively selects compression standards to optimize the effect of shared bandwidth on VM migration performance. However, the MEMCOM migration scheme requires plenty of system resources to compress the memory contents and also affects the total migration time due to classifying memory pages into similar, highly similar, and low similar classes. As MEMCOM exploits abandon system resources to compress VM memory, therefore, perceived application downtime and total migration time are limited.

Uncontrolled VM migration process affects the migrant application's performance during VM migration process. To improve network bandwidth utilization efficiency, "sonic migration" framework [83] detects and prevents the transfer of soft memory pages during VM migration process. The proposed scheme tolerates the effect of VM migration process on the throughput of co-resident applications by optimizing total migration time. Soft memory pages include OS free-pages and soft state-kernel objects. In proposed scheme, guest kernel communicates to VMM (in advance) to prevent the transfer of soft pages by communicating soft pages address to the VMM. Furthermore, to optimize the CPU resource usage, a shared memory is created between VMM and guest kernel to store and update the addresses of soft memory pages. Moreover, before triggering stop and copy phase, VMM interrupts the VM to update the shared memory. In response, VM appraises the soft pages and generates a hyper call for VMM to trigger stop and copy phase. However, the hyper calls bring additional overhead to CPU and memory resources.

Cloud net [84]-based migration schemes efficiently utilize network bandwidth capacity while triggering VM migration across WAN links. Cloud net employs block-based contents-based redundancy (CBR) method to split memory pages into fixed size blocks. Cloud net has exploited "super-fast hash" algorithm to generate memory page's hash values to optimize the total VM migration time by suppressing redundant data. Moreover, Cloud net has employed delta-based compression method to transfer delta difference pages (instead of complete memory pages). However, during hosting dis-similar workloads within source and destination servers, memory pages' compression significantly affects the co-hosted application performance due to limited ROI. For instance, if the application hosts dis-similar workloads, then compression/decompression process will lead to inflated migration time. VM migration within a LAN does not require storage migration as storage is shared between communicating nodes. Cloud net manages the network connection using virtual private network (VPN). For storage migration, synchronous storage migration followed by asynchronous dirty blocks migration is opted to ensure high storage consistency. However, during VM migration process, WAN latencies and large-sized storage severely affect the total migration time.

Clone-based VM migration technology has upgraded the CDC reliability feature (system crash scenario). Using cloning-based services, application code and memory records among VMs hosted on different servers are alike since they are generated from the same template. MV-motion [85] has used hash-based fingerprinting to highlight the similarity ratio among VM memory blocks. MV motion generates metadata to represent the VM's memory pages. Metadata structure holds the information about memory pages hash values (using superfast hash) and page identifiers. Prior to VM migration, MV motion generates VM metadata on both sender and receiver hosts, respectively. During migration preparation phase, sender module transfers metadata to the receiver server to identify the set of memory pages that are already available at the receiver host (s). During the VM migration process, consistency of metadata is certified by exploiting copy on write (COW) approach at the receiver server. However, in case of hosting dis-similar workloads across the servers, the performance of the proposed approach is degraded due to limited ROI during memory similarity discovery phase [85]. Furthermore, clone-based schemes heavily use network bandwidth to synchronize the corresponding VMs.

The checkpoint recovery-based VM migration scheme, as discussed in [86], has exploited memory content's similarity ratio (within VM's memory images [87]) to lessen network traffic using density-based hash functions. In the initial round, the proposed scheme captures the replication epo (snapshot) and applies XOR method to identify the identical memory area between non-dirty memory and system cache to transfer compressed memory difference between primary and backup servers. Likewise, at the receiver host, the controller module receives memory difference pages and reconstructs the original memory image from it. To respect the applications QoS, the proposed scheme migrates the replication epo (s) in asynchronous communication mode. Asynchronous data-transfer mode reduces VM downtime due to eliminating the likelihood of transferring similar memory contents over and over. Furthermore, implemented COW dirty logging offers uninterruptible services to assure the service availability during VM migration process. Transferring compressed difference in addition to I/O deduplication has significantly reduced migration time and application degradation duration. However, the performance of designed approach is affected by varying snapshot capturing rate.

The VM migration scheme (called Shrinker) as discussed in [33] leveraged the deduplication optimization model to efficiently migrate virtual cluster (VC) across the CDC boundaries. Shrinker has implemented a service at the sender cluster (server hosts the service) to log memory pages identifier prior to transferring them to the target server. During the VM migration process, hypervisor accesses the service to acquire the status of memory pages prior to transferring them to the target server. The hypervisor transfers memory page identifier if memory page is already transferred by any of the VM. Alternately, if the memory page has not already been transferred, then the hypervisor registers (at the service server) and transfers the complete memory page to the receiver server. Alternatively, the receiver module exploits the distributed content addressing approach to transfer memory pages to the legitimate target host (s). Furthermore, on the receiver side, the index server logs the IP address of the legitimate target server (s) against memory page hash values before it delivers the memory pages to the target server. Moreover, on receiving the hash values, Shrinker pings the index server entity to locate IP address of the host containing the similar memory page. The receiver module registers the target server against page hash value at the index server once the required memory page is received from the target server. However, the proposed scheme enlarges total migration time by visiting, searching, and comparing memory pages at the service server before transferring the memory page to the receiver. Further, being managed by a centralized server, the proposed approach is a victim of single point of failure.

The RLE compression and dynamic reordering [88] based integrated VM migration scheme has improved the network performance by optimizing total migration and application downtime. The proposed scheme dynamically reorders the memory pages during live VM migration process to reduce the likelihood of re-transferring habitually updated memory pages. The proposed scheme dynamically assigns a weight to the memory pages based on their frequency of access. For instance, the most frequent updated memory pages are assigned higher weights compared to the others. During

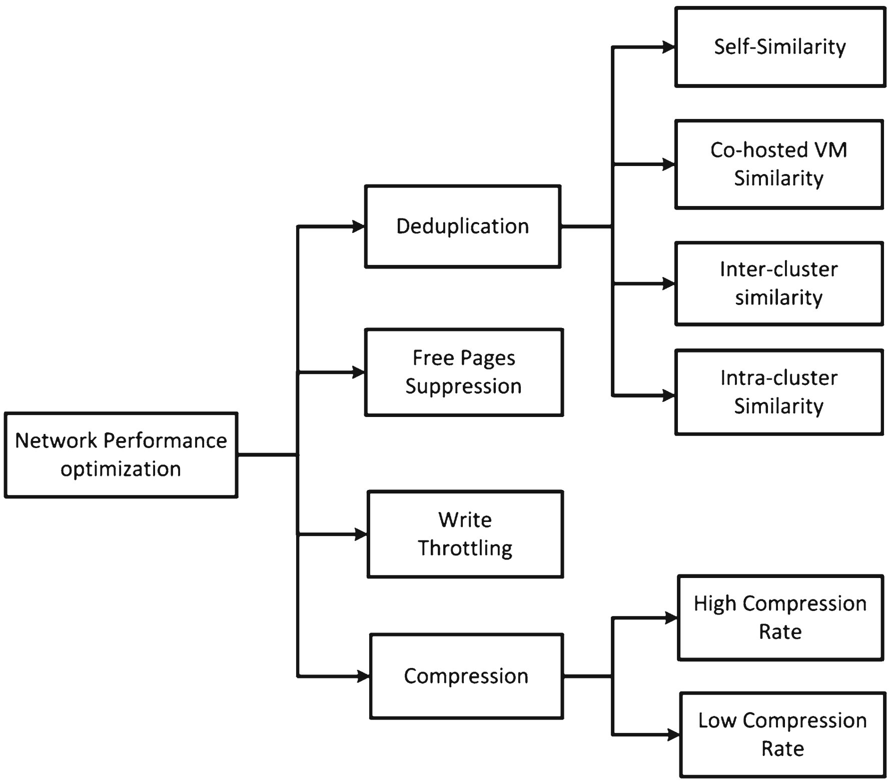

Fig. 10 Taxonomy on network performance-based VM migration schemes

the VM migration process, the proposed scheme migrates lower weighted memory pages ahead of higher order to prioritize the earlier. Further, instead of transferring the complete original dirty page, the proposed scheme transfers delta-compressed memory pages to reduce the total downtime and migration time. However, the authors have not considered spatial locality heuristics while assigning weights at the page level to optimize the network performance.

Concluding the above-discussed state-of-the-art VM migration schemes, migrating large-sized VM memory over shared bandwidth is a challenging task, especially when strict SLA violation business goals are to be met. A number of migration schemes have employed optimization methods to optimize the bandwidth utilization efficiency. For instance, deduplication [33], free pages suppression [83], throttling [29] assisted VM migration, and application of light weight high-speed compression algorithms [23] to mimic the transferred memory size have been exploited by VM migration schemes as shown in Fig. 10. Deduplication [30] based approaches have investigated memory contents similarity ratio either within a VM [30], co-hosted VM, inter-cluster VMs [81], or intra-cluster (across rack) VMs as shown in Fig. 10. As VMs are generated from the same hypervisor templates, therefore, it is needless to migrate free pages (e.g. kernel pages) during VM migration elapsed time. Throttling-assisted approaches

[11,29] have reduced the amount of data to be transferred by slowing down application write rate. The exploitation of compression [23] schemes is supportive only when the compression algorithm quickly compresses the memory contents. All the above-discussed optimization schemes require extra CPU cycles to process memory.

ii. Application performance VM migration process severely affects application performance during VM migration process due to aggressive pre-copy iterative rounds termination, concurrent VM migration, high resource contention, unpredictable workload nature, and high memory dirty rate. The following section thoroughly discusses the state-of-the-art pre-copy VM migration schemes that considered application performance optimization during VM migration process.

Checkpoint/Recovery (CR) and Trace/Replay (TR)-based integrated VM migration system [56] has deliberately reduced the VM memory size, total migration time, and elapsed service downtime due to the transfer of VM execution log rather than VM memory pages. During pre-copy's first round, CR/TR migrates the complete system state to the receiver server. For the subsequent rounds, CR/TR iteratively migrates the system execution log which is replayed at the receiver server to reconstruct the original memory. System execution log records the non-deterministic system events at the source side. Furthermore, while capturing VM memory snapshots, incorporating COW optimization minimizes the probability of packet's loss. During iterative rounds, VM migration controller signals final iterative round when the size of execution log climbs above the pre-defined threshold. During the last round, the migration controller transfers VM execution log (generated during the final round) followed by network redirection phase. However, CR/TR method consumes sufficiently larger CPU cycles while replaying dirty logs instantly to keep pace with dirty log generation rate. Furthermore, the CR/TR method works efficiently only if the network transfer rate is higher than the log growth rate.

The performance of VM migration process while hosting high-performance computing (HPC) applications (e.g. MPI) has been extensively analysed in [71]. Considering HPC applications migration, a static threshold-based iterative round's termination criterion is not ample due to the high dirty rate of HPC workloads. Furthermore, implementing a static threshold degrades VM migration performance due to aggressive pre-copy termination or enlarges VM migration duration. The proposed scheme calls for VM migration termination when pre-copy rounds are no more progressing. The authors considered the knowledge of application dirtying behaviour to terminate pre-copy iterative rounds. The migration daemon triggers stop and copy phase only if (1) dirty rate is constantly increasing, (2) memory dirty rate is low, or (3) repeatedly same memory portion is migrated during the recent iterative rounds. Furthermore, HPC application enlarges the total migration time which affects the application QoS. Moreover, the proposed approach has chosen the dedicated bandwidth for VM migration to optimize service downtime and total migration time. However, deciding VM migration termination based on condition (1) severely affects the application performance as it aggressively aborts iterative rounds.

Service level management (SLM) [89] migration scheme evades application QoS degradation during VM migration process by adopting low-level and high-level service management using Xen-patched KVM. The proposed scheme periodically estimates the remaining migration time and forecasts its impact on the application service level.

In low level method, KVM terminates VM migration when either number of iterative rounds reaches a pre-defined threshold or three-time VM memory has been copied to the receiver. However, in high-level method, the proposed scheme uses a counter variable and a pre-defined onset to terminate the pre-copy iterative rounds. Counter variable is incremented when either the service response delay violates the allowed limit or memory pages dirtying rate becomes higher than the page transfer rate. The SLM has considered the application QoS degradation during VM migration, which is missing in the majority of the literature [24,29,56,71,84]. Conversely, the proposed scheme behaves awfully when migrant application is memory intensive. Furthermore, the authors have not discussed the criteria to define the threshold.

To avoid aggressive pre-copy termination, a comprehensive empirical study has investigated the cost-profit analysis for adaptive and non-adaptive VM migration [24] schemes. Adaptive method proactively adjusts the memory page transfer rate based on the VM behaviour, whereas non-adaptive method transfers VM memory pages at maximum possible network speed. The combined and improved approach called improved live migration (ILM) has reflected application's high dirty rate and server's limited resource concerns during VM migration process. ILM optimizes the non-managed VM migration method by triggering stop and copy phase when (1) a number of iterative rounds reach a pre-defined threshold, (2) three times VM memory is copied, (3) dirty rate becomes lower than a pre-defined threshold, or (4) in the previous iterative round, dirty rate was higher than threshold, and in current round transferred data size is larger than previous round. Furthermore, ILM has eliminated the free memory pages during the VM migration process to optimize the network bandwidth. ILM has significantly improved bandwidth utilization efficiency, VM migration time, and total service downtime.

Pre-copy VM migration pattern degrades the application performance when memory dirtying rate rises higher than network transfer capacity. To handle the said issue, in [90], an optimized pre-copy VM migration scheme is proposed, which varies the virtual CPU (VCPU) frequency to adjust the application dirty rate. Furthermore, to avoid application QoS degradation, the proposed scheme adjusts the VM dirtying rate and mutes the VCPU frequency to the desired downtime limit when application dirty rate is high. Consequently, memory writing rate becomes lower. The authors have examined (empirically) the downtime under several bandwidth conditions while considering variable memory writes rate. However, the proposed scheme adversely affects the performance of running applications. However, the proposed scheme can be used only for a limited set of applications. Typically, the proposed approach is acceptable for games application. As, for the game applications, slowing the VCPU rate to reduce frame rendering rate does not stop game functioning but only affects the visual objects for a while.

Table 2 highlights the comparisons among aforementioned VM migration schemes to deliberately present the overhead associated with the VM migration process. The overhead describes the effect of VM migration process on the server resources that can lead to application QoS degradation. Optimization methods, such as write throttling [91], rate limiting [90], and smart stop copy [71], have improved the application performance running within a VM. Furthermore, network optimization techniques [71,72,90,91] and fast VM migration schemes [56] have amended the application

Table 2 Comparisons of application performance-related VM migration schemes

Scheme (Ref.)	Focus of study	Resources effected	Optimization	Method used
VCPU [90]	Application, bandwidth optimization	N/A	CPU network	CPU rate limiting, stop and copy convergence protocol
ILM [24]	Application, resource availability-based migration	CPU memory	Network	Finite resource availability-based migration
HPC [71]	Application, bandwidth optimization	N/A	Network	Migration termination
CR/TR [56]	Application, fast VM migration	CPU	Network, memory	Log transfer
SLM [89]	Application performance	N/A	N/A	QoS-based iterative rounds termination

performance while forfeiting the server resources such as CPU [91], memory [71], or cache [91]. Likewise, to optimize the network resource usage, optimization methods such as write throttling [91], smart stop and copy [71] schemes with in-time convergence property, CPU rate limiting [90], log transfer-based optimization [56], and slow down scheduling algorithms [92] have been proposed. Further, many of the approaches have optimized network resources [71,72,91,92], CPU resources [72,90,92], or memory [56,71] usage as presented in Table 2.

b. Post-copy VM migration

Post-copy VM migration pattern reduces the total migration time by transferring minimum system state to resume VM at target server. However, only a few of VM migration schemes have considered post-copy pattern to transfer VM memory image. This section briefly discusses state-of-the-art post-copy-based VM migration schemes within CDC.

Network page faults significantly degrade application performance during post-copy-based VM migration process. In [76], an optimized post-copy VM migration scheme is proposed that exploits on demand paging, active push, pre-paging, and dynamic self-ballooning optimization strategies to mitigate the network I/O faults. Active push method proactively transfers memory pages (to the target server) based on temporal locality heuristic. Likewise, at the sender host, on demand paging strategy transfers memory pages based on the request by the target server. Alternatively, pre-paging is a pre-emptive strategy that pre-fetches the memory pages on the target server. The proposed scheme reduces the network-bounded page faults by growing bubbles around the current page fault (pre-paging) in both forward and backward directions. Dynamic self-ballooning lemmatizes the page transfer rate to lessen the network load. Ballooning approach permits a guest kernel to release and handover the free memory pages back to the hypervisor. However, the performance of the proposed scheme depends on the accuracy of prediction heuristic (e.g. spatial locality).

In [93], a lightweight extension of KVM hypervisor is proposed to migrate a VM without amending guest OS. The VMEM (already existing method) accesses and manipulates memory pages within OS's kernel domain by instrumenting the memory access queries. VMEM process is appropriate for debugging the application (s), security inspection, and KVM-based VM migration execution. To handle the memory designated queries at the source server, a special device driver is instrumented inside the host OS. Prior to VM memory migration, the QEMU thread captures and transfers the VM state to the receiver host. Alternatively, at receiver end, QEMU process exploits VM transferred state to create and resume VM. For the remaining migration time, based on the application request pattern, the kernel uses a page fault handler to request VMEM to pull the contents from source server (if pages are not already fetched). VMEM driver is a Linux-based kernel module that implements page fault handler. In response, VMEM pulls the faulted page (s) from source server to update the shared memory at receiver side. The advantage of the proposed scheme is its flexibility to execute VM migration without requiring any special driver to execute VM migration.

Memory intensive workload's performance is rigorously affected when migrating a VM using pre-copy migration pattern due to its inability to transfer the memory faster than memory dirty rate. The post-copy VM migration pattern works well for memory intensive workloads. The proposed scheme as discussed in [57] has considered VM migration over LAN links. The design of the proposed scheme has considered RDMA stack to lower the latency of faulted memory pages. The RDMA link eliminates and by-passes the need of OS intervention during VM migration process. Furthermore, to reduce VM memory size, the proposed scheme exploited the pre-paging optimization strategies to fasten the VM migration process. To augment the application performance, the post-copy scheme is linked with memory management unit (MMU) such that only the thread waiting on the page fault is paused while others can continue their execution. MMU-enabled Linux strengthens the proposed scheme to handle the page faults directly in the kernel space much like swapping of pages from a disk-based swap device incurring no context switches into a user mode. The proposed scheme has significantly improved application performance due to the chosen optimization strategies in terms of optimized downtime, total migration time, and application degradation time.

c. Hybrid methods

This section discusses the state-of-the-art hybrid VM migration schemes.

The delta compression empowered hybrid memory data copy (HMDC) scheme [23] has incorporated active push and on demand paging strategies to optimize VM memory transfer. The optimization methods have eliminated the network-bounded page faults to optimize the application performance. HMDC iteratively pushes the memory pages in parallel to dirty page logging during the initial phase of VM migration (e.g. the pre-copy stage) process. During the second phase, the generated bitmap list of dirty pages is transferred to the receiver server to synchronize the VMs. During the third phase, the resumed VM accesses the dirty memory pages from the sender server based on the bitmap list. At source side, HMDC employs the RLE-based delta compression to professionally utilize the network resource capacity. Alternatively, at target side, HMDC updates the running VM accordingly. However, being a resource-intensive operation, delta compression method affects the co-hosted application's performance due to high resource sharing such as CPU cycles, cache, and physical memory. Further, HMDC is not robust as a power outage during migration phase can lead to system crash.

A hybrid VM migration scheme is proposed in [94] wherein a VM is migrated across the Ethernet link. The entire VM migration process is divided into three stages, including preparation, downtime, and resume stage. During the preparation stage, the proposed scheme exploits "access bit scanning" method to identify the VM working set. VM working set specifies a subset of VM memory which is frequently accessed by the running application. "Access bit scanning" instruments the flags in page table to ascertain the most frequently accessed pages. During the subsequent stages, the working set along CPU register's states is migrated to the destination server to resume VM. After VM resume phase, the hypervisor actively pushes the memory pages (from source host) to reduce the number of network I/O page faults. Moreover, the adaptive pre-paging is optimized by growing search space around the faulted memory pages. Further, Lempel–Ziv–Oberhumer (LZO) [95] compression method is opted to compress the memory pages. However, applying compression/decompression process consumes significant system resources [75]. The proposed scheme has significantly reduced the downtime, total migration time, and VM memory size.

d. Non-live VM migration

The Internet suspend and resume (ISR) [31] scheme has followed the non-live VM migration pattern to migrate VM across the servers. In comparison to the process migration, VM migration process is artless as there is no issue of residual dependency. However, "Suspend" and "Resume" phases in ISR are complex, heavy weight, and slower. ISR combines two of-the-shell technologies, including virtual machine and distributed file system, to carry out migration process. A distributed file system works as the transport medium to transfer the suspended VM state between corresponding servers. Furthermore, the ISR [31] has evaluated the performance of suspend and resume operations by emulating the VM migration over LAN links. To reduce the VM memory size, memory pages are compressed to augment the network performance. VM migration is trivial than process migration because of larger-sized VM memory, resource heterogeneity, network management, and migration transparency. VM migration process affects the application performance severely as cloud service is non-responsive during VM migration process. However, the major issues with ISR framework include: (1) VM state is transferred over the insecure channel, (2) locality heuristics are not used to send data smartly, and (3) portability issue.

The Zap virtualization incorporated with checkpoint restart tool has been investigated in [73] to transfer a process domain (PoD) between two servers while eliminating residual dependency. A PoD represents a group of processes with a common private namespace. Zap resides on top of the operating system layer for transparent migration of unmodified application without needing any modification to the Linux kernel. Zap is the very first system that indemnified transparent migration of legacy applications across physical machines running heterogeneous OSs by leveraging loadable kernel and thin virtualization layer. To migrate a PoD, Zap suspends execution of all the processes prior to capturing virtualization mappings and process states (e.g. file descriptor, CPU registers, memory, and buffers). In the next stage, the digitally signed PoD state is migrated to new host. Zap can provide general-purpose process migration functionality with low overhead, but it has left many questions un-explored, such as (1) when to migrate a PoD and (2) which PoD to migrate. In addition, Zap virtualization is vulnerable to all security concerns which are applicable to process migration.

5 Comparisons of VM migration schemes using thematic taxonomy

This section compares recent VM migration schemes (published after 2010) based on the thematic taxonomy as discussed in Sect. 3. The selected parameters for the comparison include Migration Pattern, Downtime, Migration Duration, App QoS Degradation Duration, Bandwidth Utilization Efficiency, Network Link, Migration Granularity, Execution Resource Constraints, Hypervisor type, and Network Management as illustrated in Table 3.

Service disturbance time describes the time interval during which cloud service is inaccessible for cloud service users. Service disturbance time is also termed "downtime" in the literature. The attributes of downtime are affected by network bandwidth capacity, aggregate of system resources assigned to the VM migration daemon, and application dirtying rate. The downtime attribute substantially affects application QoS, SLA, and network connectivity (TCP connection loss). The attributes of downtime are categorized as low, high, or medium based on the selected pre-copy/post-copy optimization methods. The perceived downtime is relatively low if the VM's memory contents are suppressed during the VM migration process (using deduplication/compression) [33,75,88] or an optimal termination point is selected to terminate the pre-copy iterative rounds [24,29,90,91]. Additionally, several VM migration schemes [56,86] transfer a VM execution log (instead of actual VM memory pages) to optimize application downtime. The downtime is considerably higher when (1) the pre-copy is not progressing [84], (2) write-intensive [81] workload is migrated using the pre-copy method, or (3) a non-optimized post-copy [76] migration pattern is picked out. Migrating multiple VMs simultaneously over a shared network link extends the application service downtime due to high resource contention [30,33,81]. Notwithstanding, while migrating mixed-mode workload within a migrant VM, the perceived downtime is relatively medium [32,71]. In comparison to the live VM migration patterns, non-live VM migration schemes exhibit higher downtime as the latter halt service during VM migration [31,73,77]. However, higher downtime is dangerous for hosted applications as it can even destroy the TCP/IP network connection. Furthermore, longer downtime [71,81,89] severely affects the negotiated SLA.

Table 3 Comparisons of VM migration schemes

Category	Scheme (Ref.)	Migration pattern	Downtime	Migration duration	App QoS degradation duration	Bandwidth utilization efficiency	Network link	Hypervisor type	Execution resource constraints	Migration granularity	Network management
Post-copy	IRLM [30]	Live	H	H	H	YES	LAN	KVM	SHARED	MULTIPE (Parallel)	N/A
	Gang Migration [81]	Live	H	H	H	YES	LAN	KVM	SHARED	MULTIPLE (Parallel)	N/A
	XBRLE [32]	Live	M	M	M	YES	LAN	KVM	SHARED	SINGLE	N/A
	Sonic Migration [83]	Live	N/A	L	M	YES	LAN	Xen	SHARED	SINGLE	N/A
	MECOM [75]	Live	L	L	L	YES	LAN	Xen	Extra CPU Cycles	SINGLE	N/A
	Cloud-Net [84]	Live	H	H	H	YES	WAN	Xen	SHARED	SINGLE	VPN
	MV motion [85]	Live	NA	L	M	YES	LAN	Xen	SHARED	SINGLE	N/A
	Checkpoint Recovery [86]	Live	L	L	M	YES	LAN	KVM	SHARED	SINGLE	N/A
	Shrinker [33]	Live	H	H	H	YES	WAN	KVM	SHARED	MULTIPLE (Sequence)	N/A
	Pre-copy-Delta [88]	Live	L	L	M	YES	WAN	KVM	SHARED	SINGLE	N/A
	CR/TR [56]	Live	L	L	L	YES	LAN	Xen	Extra CPU Cycles	SINGLE	ARP & Dyn-DNS
	HPC [71]	Live	M	M	M	NO	LAN	KVM	Dedicated Bandwidth	SINGLE	N/A
	SLM [89]	Live	H	M	L	NO	LAN	KVM	SHARED	SINGLE	N/A
	ILM [24]	Live	L	M	L	YES	LAN	Xen	SHARED	SINGLE	N/A
	VCPU [90]	Live	L	M	H	YES	LAN	Xen	SHARED	SINGLE	N/A

Table 3 continued

Category	Scheme (Ref.)	Migration pattern	Downtime	Migration duration	App QoS degradation duration	Bandwidth utilization efficiency	Network link	Hypervisor type	Execution resource constraints	Migration granularity	Network management
Post-copy	DSB [76]	Live	H	L	M	YES	LAN	Xen	SHARED	SINGLE	N/A
	HMDC [23]	Live	L	L	M	YES	LAN	Xen	SHARED	SINGLE	N/A
	VMEM [93]	Live	N/A	M	M	NO	LAN	KVM	SHARED	SINGLE	N/A
Hybrid	RD-Pos [57]	Live	L	L	L	YES	LAN	Xen	N/A	SINGLE	N/A
	Hybrid [94]	Live	L	L	M	YES	LAN	Xen	SHARED	SINGLE	N/A
Non-live VM migration	ISR [31]	Non-Live	H	M	H	YES	LAN	N/A	N/A	SINGLE	N/A
	POD [73]	Non-live	H	H	H	NO	N/A	N/A	N/A	SINGLE	N/A

M medium, *H* high, *L* low

The attribute of migration duration specifies the total time taken by the VM migration daemon to complete VM migration between source and target server. The major factors influencing migration duration (also called migration time) include configured VM memory size, nature of hosted workload, memory dirtying rate, and network link topology [96]. During the VM migration process, the migration daemon shares system resources such as network bandwidth, CPU cycles, physical memory, and I/O devices assigned to the co-hosted applications. Hence, total migration time severely affects the hosted application(s). The attributes of migration time can be categorized as low, medium, or high for VM migration schemes. The migration time attribute is low when a VM migration scheme exploits (1) fast compression algorithms [75], (2) lightweight deduplication [33,86,97], (3) dedicated resources for the VM migration daemon [32,56,75], or (4) write throttling to reduce application write rate [29]. Conversely, the migration duration attribute is high when (1) the migration scheme does not employ deduplication/compression methods [30], (2) VM memory exhibits a wide range of data [30,81], (3) the migration scheme aggressively terminates iterative pre-copy rounds [29], (4) there is high resource contention, (5) write-exhaustive applications are migrated using non-optimized pre-copy migration patterns [29], or (6) a non-optimized post-copy approach is selected to migrate read-intensive workload [93]. However, in numerous migration schemes, the attribute of migration duration is medium, as they exploit limited optimization technologies during the VM migration process or inadequate dedicated resources are assigned to the migration daemon [71,90]. Also, in case of non-live VM migration, migration duration is considerably higher as a complete VM is migrated across the CDC [31,73,77].

Application performance is a measure of QoS parameter as perceived by the user during VM migration process. The application QoS can be quantitatively ranked on the basis of application's query response delay time, query dropped rate, throughput, jitter, and service degradation time. Likewise, application QoS (shortly App QoS) degradation duration specifies the time period during which application QoS is consistently humiliated. The attribute of app QoS degradation duration is affected by total elapsed VM migration duration and application service downtime [30,33,72,81,83]. In the reported literature, the attributes of app QoS degradation duration for the proposed migration schemes are categorized as *low*, *medium*, or *high*. The app QoS degradation duration time interval is relatively low [56,75,82,84] when (1) migration controller allocates adequate dedicated system resources to VM migration daemon, (2) light weight, efficient, and fast compression algorithm are chosen, (3) deduplication optimization is employed, or (4) selected workload is ideal for chosen VM migration pattern. Alternatively, employing the write throttling methods (during VM migration) results in higher app QoS degradation duration. The attribute of app QoS degradation duration is medium [71,85,88] when only a few optimization methods are chosen (during VM migration) or closely enough bandwidth is allocated to the VM migration daemon. Similarly, while using the post-copy VM migration pattern, App QoS degradation duration is relatively higher due to frequent network I/O page faults. However, post-copy migration schemes have improved the App QoS degradation duration by choosing several optimization methods including active push and pre-paging [56,71,75,82]. Furthermore, workload type too affects App QoS degradation duration. For instance, in case of migrating a write intensive application [71] using pre-copy migration method, app QoS degradation duration period is very high. Furthermore, for non-live VM migration, App QoS degradation duration is much higher due to prolonged downtime and migration time [31,73,77].

The attribute of bandwidth utilization efficiency characterizes whether the proposed VM migration scheme has efficiently utilized the network bandwidth capacity or not. Efficient VM migration management (within a DC) is challenging due to larger VM size and constrained system shared resources [57]. Being a shared resource, non-effective network bandwidth utilization (by VM migration process) affects co-hosted application's performance [71]. Assorted VM migration schemes [30,31,33,56,81,83,86,94] have proficiently utilized the network bandwidth capacity during VM migration process by implementing deduplication [30,81,97], memory compression [23,31,32,75], free memory pages filtering [83], and dynamic rate limiting [30,32,75,81,85,86]. Additionally, the performance metrics such as application QoS degradation duration, total migration time, and service downtime are affected by the attribute (e.g. "NO") of bandwidth utilization efficiency. VM migration schemes such as one presented in [76] have highly improved network performance by exploiting numerous bandwidth optimization methods, whereas other schemes such as [23,30,94], and [81] have implemented limited bandwidth optimization methods. Likewise, both [30] and [81], have opted deduplication optimization and dedicated resource assignment policies to optimize network performance; however, the former has considered inter-rack deduplication, whereas the later has implemented deduplication at a server level. Among [83] and [85], former evaded transfer of soft pages, whereas later exploited metadata-based optimizations to improve application QoS. Further, [86] and [23] have improved the bandwidth utilization efficiency by employing memory compression methods and smart hybrid memory copying pattern (for mixed mode workload), respectively. In comparison to [83], [71] has optimized the pre-copy termination criterion in addition to adaptation of parallelization programming model [98]. Furthermore, bandwidth utilization efficiency affects VM migration transparency [23,75,86,97].

The attribute of the network link specifies the type of the network link chosen to migrate a VM. VM migration schemes have considered either LAN [86] network or WAN [91] links for VM migration. Bulk of migration schemes have considered VM migration within LAN [29,76] boundaries. Migrating a VM over LAN links does not necessitate storage migration as NAS is equally accessible to both source and target servers [99]. To surge CDC scalability, the hypervisor triggers VM migration across the WAN links. In comparison to LAN links [29,71,72,94], VM migration over WAN channel [33,56,84,88,91,100] affects application performance due to limited bandwidth capacity, higher latencies, and high packets dropped rate [84,86,91]. However, VM migration over WAN links severely humiliates the application QoS degradation duration, downtime, and total migration time [84,89]. The probability of SLA violation is also high while considering VM migration across the WAN links due to the above-stated reasons. The attribute of the hypervisor type metric stipulates the underlying hypervisor tool preferred to manage the cloud-hosted applications [19,26,92]. Numerous hypervisors [19–21,52] support live VM migration feature [18]. The majority of schemes have considered Xen [23,24,82,83,94] and KVM [30,32,71,81,89] (both are LINUX based) hypervisors as underlying virtualization technologies for resource management.

The attribute of execution resources constraints (ERC) parameter ascertains the resource (both system and network) assignment policies for VM migration daemon. VM migration daemon either shares the already existing system resources assigned to the running application (s) or exploits absolutely dedicated system/network resources during VM migration process. In case of sharing existing system resources, VM migration daemon steals the required system resources (for VM migration) from the current hosted applications to migrate a VM. Bulk of VM migration schemes [23,72,84,90,91,94] do not assign dedicated resources (e.g. dedicated bandwidth, memory, and CPU) to a migration daemon as it increases the CDC computational cost. Furthermore, depending on the type of workload being hosted within a VM, the shared resource-based category of VM migration schemes [23,76,94] affects the VM performance significantly. Moreover, the shared attribute of ERC parameter affects the total migration time and application service downtime [56,71,82,88]. Alternatively, assigning dedicated system resources [56,71,75] to VM migration daemon minimizes the total migration time and downtime [23,30,32,76,81,92,94].

The attribute of VM migration granularity defines the granularity level selected by the VM migration framework during VM migration process. VM migration controller either migrates a single VM [72,82,88,90,94] or multiple VMs (at the same time) [30,33,81] across the CDC for effective resource management. The 'multiple' attribute of migration granularity affects the app QoS degradation duration, total service downtime, and co-hosted application's QoS [30]. Furthermore, during 'multiple' VMs migration, a set of VMs are migrated in a pre-defined order (e.g. sequential, parallel, concurrent) between CDC clusters. Migrating bunch of VMs at the same time degrades the applications performance. Moreover, the accumulated total migration time by each VM is also larger due to queuing delay at the sender side while considering 'multiple' attribute. To uniquely identify a VM within a CDC, every VM is assigned a unique IP address at the time of VM creation. After VM migration completion, the point of attachment changes for migrant VM. The attribute of network management metric specifies the mean/protocol chosen to update the location of migrant VM [91]. For VM migration over LAN [30] links, the network management is easy to handle as IP does not change for the external world [84,100]. ARP-based broadcasting [29,56] method seamlessly upgrades the new point of attachment for VM migration within a LAN. However, considering WAN links, network management is a challenging and complex task [91,99]. The VM migration schemes [91] have considered Dyn-DNS [29,56,91], VPN, IP tunnelling, and Mobile IP-based protocols [91,99] to update the IP address of migrant VM over WAN links. Once VM is resumed, the VM migrator re-directs the packets to the receiver server [33,84,101].

6 Open research issues and challenges

This section discusses open research issues and challenges in VM migration domain that are significant in developing optimal VM migration schemes for computational clouds as shown in Fig. 11.

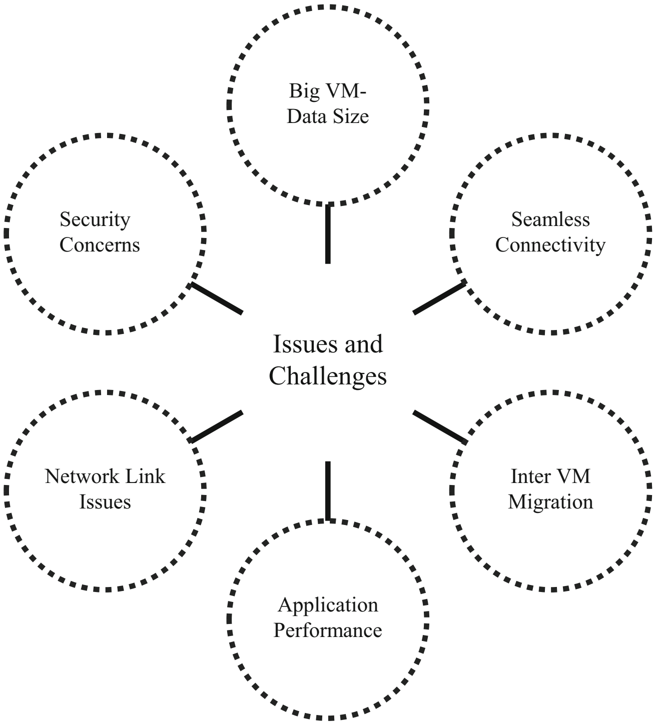

Fig. 11 Research issues and challenges in VM migration

6.1 Big data transmission and VM migration optimization adaptation issues

VM memory size is truly dynamic and ranges from a couple of GBs to several hundred, which is extremely difficult to transfer on shared bandwidth while respecting the application SLA. The issue with transferring large-sized VM memory has been addressed [23,30,32,75,81,83,97] either by ruthlessly exploiting fast network communication links [102] or by dispensing dedicated bandwidth [23,30,32,57,71,75, 76,81,83,97,100,103] to the VM migration daemon. VM migration schemes have greatly reduced VM memory size by employing deduplication [30], memory compression [75,104], write throttling [91], smart and efficient pre-copy termination, and memory self-ballooning [90] methods. Nevertheless, the proposed optimizations (e.g. deduplication, compression) disgrace application and network performance due to dissimilar workload hosted among geographically distant CDC servers (hypervisors, OS, etc.) [1,60,91]. The mentioned critical issue can be successfully mitigated by designing an adaptive optimization method to augment the VM migration performance. The adaptive method, based on machine learning algorithms, implicitly and proactively learns the application behaviour for a specified period of time and formally adopts the most suitable optimization from a list of optimization methods (e.g. deduplication, compression, throttling, etc.).

In the literature, several optimization methods have substantially improved the pre-copy iterative rounds termination selection criteria. However, the proposed schemes do not function adeptly when source and target servers host dissimilar workloads. The fairly aggressive termination of unbalanced iterative rounds likely leads to SLA violation. To avoid such aggressive termination, a QoS-based dynamic threshold (to limit the iterative rounds) can significantly optimize migration performance. Another innovative approach to optimize VM migration performance is to use a fast communication link between corresponding servers. The remote direct memory access (RDMA) [102] approach generally allows the direct transfer of data from one server's memory to another server while by-passing OS intervention. Including RDMA links to transfer VM memory pages can significantly optimize system throughput, migration time, application downtime, and co-hosted VMs performance. However, applying RDMA-based VM migration incurs various stimulating challenges, including protocol design, network QoS, memory registration, and non-contiguous data transfer.

6.2 VM migration links and application performance issues

Network link carrying capacity within a CDC rigorously affects application performance during VM migration. Recently, to vastly improve CDC scalability, VM migration schemes have considered cloud resource management beyond LAN boundaries. Though, in the majority of VM migration schemes, migration is considered within LAN links [1,29,60,69,70]. VM migration over WAN links faces several challenges due to limited bandwidth, greater latencies, unpredictable network behaviour, heterogeneous network architecture design, larger communication distances, and higher packet drop ratio [103]. All stated issues increase the possibility of SLA violation during VM migration. In addition, storage migration (over WAN links) increases VM migration complexity as live storage migration exploits synchronous and asynchronous modes of communication to transfer storage blocks [64,105–107]. VM migration schemes migrate VM memory and storage over a shared communication link. However, the performance degradation issue throughout VM migration over WAN links can be handled by migrating storage and VMs over different communication channels. While considering WAN links, VM migration schemes should exploit high compression rate algorithms (compression speed can be low) to optimize VM migration. So far, VM migration schemes have not thoroughly explored VM migration issues encountered while migrating VMs between two CDCs operated by different energy sources (e.g. electricity, wind energy).

The pre-copy VM migration pattern transfers VM memory several times the size of the VM if the VM migration termination point is not carefully managed. The final round phase in pre-copy [23,86,108,109] impacts the performance of applications running within a VM. A pre-copy VM migration scheme either aggressively terminates iterative (pre-copy) rounds or waits for a suitable termination condition to occur. However, in the latter case, the VM migration time surges with limited chances to achieve considerable benefit by delaying VM migration termination in many cases. The un-predictable workload behaviour and random variations in application dirtying rate increase the complexity of finding a suitable termination point. The matter can be addressed by embedding workload characteristics knowledge in the pre-copy's stop and copy module to find a suitable termination point. For instance, while a VM hosts a real-time application (e.g. banking application), it would be fruitful to wait for a suitable length of time to terminate iterative VM migration rounds. Conversely, in case of hosting non real-time applications, it is beneficial to dismiss the iterative pre-copy phases as soon as possible to optimize co-hosted application performance. However, provisioning explicit knowledge about hosted application behaviour is a big challenge.

To optimize application performance, VM migration schemes mitigate the migration noise that is measured in terms of delayed response, high packet drop rate, and limited application throughput. The adverse effect of a VM migration process on running application performance is known as migration noise. Migration noise can be reduced if resource demands of the VM migration process are predicted in advance. However, predicting application resource demands in advance is challenging as some applications such as social network applications exhibit unpredictable behaviour. To capture the behaviour of running applications, application and system profiling methods can be exploited.

6.3 Inter-VM migration and seamless connectivity

Virtualization technology isolates the co-resident workloads [29] to improve application privacy and system throughput. Sometime, co-resident VMs communicates with each other (e.g. parallel processing) to share the data. Recently, a number of VM migration schemes in the reported literature [110–113] have focused on inter-VM migration issue. For instance, communication libraries which are based on shared memory have implemented socket style APIs to exploit message passing interface for inter-VM communication (IVMC). Other approaches, including inter-domain data transfer system (IDTS) [112] and Virtual machine communication interface (VMCI) [111,112], expedite to host inter-dependent applications on the same physical machine. However, so far, the researchers have not exhaustively focused on standardizing the APIs for inter-VM communication (e.g. hypervisors/VMs) to enhance application performance. Standardize API interfaces will improve overall system throughput.

Seamless VM migration ensures that application's connection remains alive during VM mobility. To manage a VM, VMM assigns a unique IP address to every VM for the unique identification [33]. However, VM's IP address changes when VM is migrated to a new location. For a LAN, IP management [114] is simple as hypervisor (LAN links) exploits ARP broadcast to update the MAC and IP address of the migrant VM [115]. For VM migration over WAN links, the hypervisor updates the network IP address using Mobile IP [115], Dynamic DNS [91], and tunnelling [99] based standards. Moreover, during network management over WAN links, a significant amount of network packets are dropped due to resource's heterogeneity and network speed incompatibility issues within heterogeneous CDC [91,100,116]. The stated issue can be handled by designing a packet re-ordering method to handle resource's heterogeneity issue (e.g. network speed difference).

6.4 VM isolation and security issues

The hypervisor multiplexes guest OSs to the underlying hardware resources (CPU, memory, storage) for efficient CDC resource utilization. However, co-hosted VM interference degrades application performance when VMs are poorly isolated [117]. Strict VM isolation policies such as physical resource partitioning (e.g. disk drives, network interface) empower hypervisors to eliminate the threat of denial-of-service (DoS) attacks. Alternatively, logical resource partitioning-based isolation methods share underlying physical resources such as memory and processor while eliminating VM migration noise (e.g. semaphores for memory access) [116]. Furthermore, VM isolation mitigates side-channel masquerade attacks that exploit CPU and memory usage patterns to reveal cryptographic keys. Consequently, isolating co-residing VMs hinders the impact of one VM's failure on the performance of other VMs by efficiently isolating the faults. Implementing VM isolation policies and VM resource introspection (by the hypervisor) helps to design profound security systems for secure VM migration [28,118,119].

Securing VM migration on heterogeneous CDCs is a challenging task, as data travel across erroneous network links due to longer communication distances. Consequently, hijacking a VM during migration enables hijackers to acquire OS kernel states, currently hosted applications, application sensitive data, and hardware states, for malicious activities [54,117,120,121]. Moreover, the VM migration process deteriorates isolation boundaries as it exposes its complete system state to the network. Therefore, hijackers can compromise the VM data, hypervisor module, and VM migration control plane (e.g. false resource advertising) [114]. Remote attestation along with embedding encryption keys and secret data within the hardware platform helps secure the hypervisor and hosted VMs [122]. Similarly, the Security Assertion Mark-up Language (SAML) standard authenticates legitimate entities (migrating VMs) and controls access to cloud resources [121].

Co-hosted VMs share existing physical infrastructure to consolidate server workloads; thus, co-residing applications face several security threats such as virus propagation and denial-of-service (DoS) attacks. Co-hosted VMs communicate with each other to implement parallel processing and information sharing services [51,88]. However, inter-VM communication using shared system resources compromises the security and transparency of applications. Inter-VM migration solutions including XenLoop [123], Xen-Sockets [124], XWAY [125], Inter-OS [126], Inter-Domain Data Transfer System (IDTS) [112], and Virtual Machine Communication Interface (VMCI) [111] exploit shared memory for communication while preventing DoS attacks. Besides all these solutions, Xen's Virtual Network method exploits standard network interfaces to offer the highest isolation as it routes data through standard network interfaces rather than physical resource sharing at the cost of application performance and data integrity [111,117]. Moreover, as shared resources offer limited isolation, malicious VMs can access other VMs' address spaces (e.g. via the cache) to carry out harmful activities (e.g. application privacy theft) [114,120]. In addition, data integrity [121] should also be guaranteed during a VM migration process using highly complex cryptographic functions. Malicious entities can relocate a VM for their own purposes if the hypervisor is compromised. Moreover, as communication takes place at different trust levels, it is important to ensure that any entrusted VM migration or inter-VM migration should not cause resource starvation [53,110,114,120,121,127].

7 Conclusions

The paper discussed the notion of Cloud Data Centers, VM migration process, and thoroughly explained various VM migration schemes to improve application and network performance within Cloud Data Centers. It analysed current VM migration schemes based on a thematic taxonomy and highlighted the commonalities and variances among VM migration schemes based on the selected performance parameters. It has also discussed the issues and challenges in existing VM migration schemes to design an optimal live VM migration scheme.

VM migration is an expensive process owing to the adequate system resources required for handling large-sized VMs, the unpredictable workload nature, and possibility of SLA violation. Particularly, the performance of co-resident applications is adversely affected in scenarios of inappropriate VM migration process management. Therefore, various optimization techniques have been proposed to assist cloud data center operators with optimizing resource utilization technologies to reduce VM migration noise. The pre-copy VM migration pattern improves application performance by optimizing the stop-and-copy phase during iterative memory transfer rounds. Alternatively, the post-copy-based migration method improves application performance by pre-fetching memory pages (to the receiver server) to reduce network I/O page faults. The pre-copy and post-copy migration schemes perform well when hosting read-intensive and write-intensive applications, respectively. Nevertheless, VM migration over WAN links suffers on account of limited shared bandwidth, a need for storage migration, and network connection c redirection.

The challenges faced by VM migration schemes are very dynamic. Cloud resource heterogeneity, unpredictable workload nature, system workload, VM memory size, and degree of SLA violations call for vigorous, resource-aware and computationally inexpensive VM migration schemes. Large-sized VM memory extends migration time and service downtime. Incorporating optimization methods such as memory contents compression, fine granular deduplication, and dynamic write throttling instrumentation improves application performance at the cost of required system resources. Finally, security is another major threat to the VM migration process and it can be safeguarded by (1) preventing compromised entities' access to VMM, (2) isolating VM boundaries, and (3) securing the network connections.

Acknowledgments This work was fully funded by Bright Spark Unit, University of Malaya, Malaysia and partially funded by Malaysian Ministry of Higher Education under the University of Malaya High Impact Research Grant UM.C/625/1/HIR/MOE/FCSIT/03. We also would like to thank Tahir Maqsood (COMSATS Institute of IT, Abbottabad) and Junaid Shuja (University Malaya, Malaysia) for their valuable comments.

References

1. Beloglazov A, Buyya R (2013) Managing overloaded hosts for dynamic consolidation of virtual machines in cloud data centers under quality of service constraints. IEEE Trans Parallel Distrib Syst 24:1366–1379
2. Uddin M, Shah A, Alsaqour R, Memon J (2013) Measuring efficiency of tier level data centers to implement green energy efficient data centers. Middle East J Sci Res 15:200–207
3. Beloglazov A, Buyya R (2010) Energy efficient resource management in virtualized cloud data centers. In: Proceedings of the 2010 10th IEEE/ACM international conference on cluster, cloud and grid computing. IEEE Computer Society, New York, pp 826–831
4. Zhou M, Zhang R, Zeng D, Qian W (2010) Services in the cloud computing era: a survey. In: 4th international universal communication symposium (IUCS). IEEE, New York, pp 40–46
5. Server S (2013) Storage Severs Word Press
6. Koomey J (2011) Growth in data center electricity use 2005 to 2010. A report by Analytical Press, completed at the request of The New York Times
7. Moura Silva L, Alonso J, Silva P, Torres J, Andrzejak A (2007) Using virtualization to improve software rejuvenation. In: International symposium on network computing and applications (NCA). IEEE, New York, pp 33–44
8. Mishra R, Jaiswal A (2012) Ant colony optimization: a solution of load balancing in cloud. Int J Web Semant Technol 3:33–50
9. Pop CB, Anghel I, Cioara T, Salomie I, Vartic I (2012) A swarm-inspired data center consolidation methodology. In: Proceedings of the 2nd international conference on web intelligence, mining and semantics. ACM, New York
10. Juniad Shuja SAM, Bilal K, Hayat K, Khan SU, Sarwar S (2012) Energy-efficient data centres. Computing 94(12):937–994
11. Jeong J, Kim S-H, Kim H, Lee J, Seo E (2013) Analysis of virtual machine live-migration as a method for power-capping. J Supercomput 66:1629–1655
12. Beloglazov A, Buyya R (2010) Energy efficient resource management in virtualized cloud data centers. In: Proceedings of the 10th international conference on cluster, cloud and grid computing. ACM/IEEE, New York, pp 826–831
13. Kim K, Lee S, Yoo H, Kim D (2014) Agriculture sensor-cloud infrastructure and routing protocol in the physical sensor network layer. Int J Distrib Sensor Netw 2014. (in press). doi:10.1155/2014/437535
14. Wang J, Fan ZH (2014) Family health telemonitoring system based on WSN. Adv Mater Res 860:2762–2765
15. Whaiduzzaman M, Sookhak M, Gani A, Buyya R (2013) A survey on vehicular cloud computing. J Netw Comput Appl 40:325–344
16. Huang J, Du D, Duan Q, Zhang Y, Zhao Y, Luo H, Mai Z, Liu Q (2014) Modeling and analysis on congestion control for data transmission in sensor clouds. Int J Distrib Sensor Netw 2014. (in press). doi:10.1155/2014/453983
17. Kremer J, Cloud Computing and Virtualization. White paper on virtualization
18. Xing Y, Zhan Y (2012) Virtualization and cloud computing. In:Zhang Y (ed) Future wireless network and informatiion systems. Springer, Berlin, Heidelberg, pp 305–312
19. 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. ACM SIGOPS Oper Syst Rev 37:164–177
20. 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 (TOCS) 30:12
21. Younge AJ, Henschel R, Brown JT, von Laszewski G, Qiu J, Fox GC (2011) Analysis of virtualization technologies for high performance computing environments. In: International conference on cloud computing (CLOUD). IEEE, New York, pp 9–16
22. Medina V, García JM (2014) A survey of migration mechanisms of virtual machines. ACM Comput Surv (CSUR) 46:30
23. 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 7:639–646
24. 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 International conference on performance engineering. ACM, New York, pp 387–398

25. Asberg M, Forsberg N, Nolte T, Kato S (2011) Towards real-time scheduling of virtual machines without kernel modifications. In: 16th IEEE conference on emerging technologies & factory automation (ETFA), pp 1–4
26. Habib I (2008) Virtualization with kvm. Linux J 2008:8
27. Ferreto TC, Netto MA, Calheiros RN, De Rose CA (2011) Server consolidation with migration control for virtualized data centers. Future Gener Comput Syst 27:1027–1034
28. Xu F, Liu F, Jin H, Vasilakos A (2014) Managing performance overhead of virtual machines in cloud computing: a survey, state of the art, and future directions. Proc IEEE 102:11–31
29. Christopher Clark KF, Hand S, Hanseny JG (2005) Live migration of virtual machines. In: Proceedings of the 2nd conference on symposium on networked systems design & implementation, vol 2
30. 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, New York, pp 19–26
31. Kozuch M, Satyanarayanan M (2002) Internet suspend/resume. In: Proceedings fourth IEEE workshop on mobile computing systems and applications, 2002. IEEE, New York, pp 40–46
32. Svärd P, Hudzia B, Tordsson J, Elmroth E (2011) Evaluation of delta compression techniques for efficient live migration of large virtual machines. ACM Sigplan Notices 46:111–120
33. 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: Euro-Par 2011 parallel processing. Springer, Berlin, pp 431–442
34. Bilal K, Khan Samee U, Kolodziej J, Zhang L, Madani S, Min-Allah N, Wang L, Chen D (2012) A comparative study of data center network architectures. In: 26th EUROPEAN conference on modelling and simulation, ECMS
35. Shiraz M, Gani A, Khokhar RH, Buyya R (2013) A review on distributed application processing frameworks in smart mobile devices for mobile cloud computing. IEEE Commun Surv Tutor 15:1294–1313
36. Liu L, Wang H, Liu X, Jin X, He WB, Wang QB, Chen Y (2009) GreenCloud: a new architecture for green data center. In: Proceedings of the 6th international conference industry session on Autonomic computing and communications industry session. ACM, New York, pp 29–38
37. Garg SK, Versteeg S, Buyya R (2013) A framework for ranking of cloud computing services. Future Gener Comput Syst 29:1012–1023
38. Mosbah MM, Soliman H, El-Nasr MA (2013) Current services in cloud computing: a survey. arXiv preprint arXiv:1311.3319
39. Malawski M, Kuzniar M, Wojcik P, Bubak M (2013) How to use Google App engine for free computing. IEEE Internet Comput 17:50–59
40. Wilder B (2012) Cloud architecture patterns: using microsoft azure. O'Reilly Media, Inc., Sebastopol
41. Greenberg A, Lahiri P, Maltz DA, Patel P, Sengupta S (2008) Towards a next generation data center architecture: scalability and commoditization. In: Proceedings of the ACM workshop on programmable routers for extensible services of tomorrow. ACM, New York, pp 57–62
42. Arabnia H (1995) A distributed stereocorrelation algorithm. In: Proceedings of fourth international conference on computer communications and networks. IEEE, New York, pp 479–482
43. Zhang Q, Cheng L, Boutaba R (2010) Cloud computing: state-of-the-art and research challenges. J Internet Serv Appl 1:7–18
44. Guo C, Wu H, Tan K, Shi L, Zhang Y, Lu S (2008) Dcell: a scalable and fault-tolerant network structure for data centers. In: ACM SIGCOMM computer communication review. ACM, New York, pp 75–86
45. Al-Fares M, Loukissas A, Vahdat A (2008) A scalable, commodity data center network architecture. In: ACM SIGCOMM computer communication review. ACM, New York, pp 63–74
46. Guo C, Lu G, Li D, Wu H, Zhang X, Shi Y, Tian C, Zhang Y, Lu S (2009) BCube: a high performance, server-centric network architecture for modular data centers. ACM SIGCOMM Comput Commun Rev 39:63–74
47. Alam N (2009) Survey on hypervisors. Indiana University, Bloomington, School of Informatics and Computing
48. Vaquero LM, Rodero-Merino L, Caceres J, Linder M (2008) A break in the clouds: towards a cloud definition. ACM SIGCOMM Comput Commun Rev 39(1):50–55

49. Li C, Raghunathan A, Jha NK (2010) Secure virtual machine execution under an untrusted management OS. In: IEEE 3rd international conference on cloud computing (CLOUD). IEEE, New York, pp 172–179
50. Ganesan R, Murarka Y, Sarkar S, Frey K (2013) Empirical study of performance benefits of hardware assisted virtualization. In: Proceedings of the 6th ACM India computing convention. ACM, New York
51. Muraai M, Furuya T, Imai T, Kimura S (2013) Application of server virtualization technology to communication services. FUJITSU Sci Tech J 49:286–291
52. Zhang B, Wang X, Lai R, Yang L, Wang Z, Luo Y, Li X (2010) Evaluating and optimizing I/O virtualization in kernel-based virtual machine (KVM). In: Network and parallel computing. Springer, Berlin, pp 220–231
53. Bhandarkar SM, Arabnia HR (1995) The REFINE multiprocessor—theoretical properties and algorithms. Parallel comput 21:1783–1805
54. Kapil D, Pilli ES, Joshi RC (2013) Live virtual machine migration techniques: survey and research challenges. In: 3rd international advance computing conference (IACC). IEEE, New York, pp 963–969
55. Yao L, Wu G, Ren J, Zhu Y, Li Y (2014) Guaranteeing fault-tolerant requirement load balancing scheme based on VM migration. Comput J 57:225–232
56. Liu H, Jin H, Liao X, Yu C, Xu C-Z (2011) Live virtual machine migration via asynchronous replication and state synchronization. IEEE Trans Parallel Distrib Syst 22:1986–1999
57. Shribman A, Hudzia B (2013) Pre-Copy and post-copy VM live migration for memory intensive applications. In: Euro-Par 2012: parallel processing workshops. Springer, Berlin, pp 539–547
58. Kozuch M, Satyanarayanan M (2002) Internet suspend/resume. In: Proceedings of fourth workshop on mobile computing systems and applications. IEEE, New York, pp 40–46
59. Beloglazov A, Buyya R (2010) Energy efficient allocation of virtual machines in cloud data centers. In: 10th IEEE/ACM international conference on cluster, cloud and grid computing (CCGrid), pp 577–578
60. Beloglazov A, Buyya R (2010) Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In: Proceedings of the 8th international workshop on middleware for grids, clouds and e-Science. ACM, New York
61. Shrivastava V, Zerfos P, Lee K-W, Jamjoom H, Liu Y-H, Banerjee S (2011) Application-aware virtual machine migration in data centers. In: Proceedings of INFOCOM. IEEE, New York, pp 66–70
62. Mishra M, Das A, Kulkarni P, Sahoo A (2012) Dynamic resource management using virtual machine migrations. IEEE Commun Mag 50:34–40
63. Dong J, Jin X, Wang H, Li Y, Zhang P, Cheng S (2013) Energy-saving virtual machine placement in cloud data centers. In: 13th IEEE/ACM international symposium on cluster, cloud and grid computing (CCGrid), pp 618–624
64. Zheng J, Ng TSE, Sripanidkulchai K (2011) Workload-aware live storage migration for clouds. In: ACM SIGPLAN notices. ACM, New York, pp 133–144
65. Nagarajan AB, Mueller F, Engelmann C, Scott SL (2007) Proactive fault tolerance for HPC with Xen virtualization. In: Proceedings of the 21st annual international conference on supercomputing. ACM, New York, pp 23–32
66. Thein T, Park JS (2009) Availability analysis of application servers using software rejuvenation and virtualization. J Comput Sci Technol 24:339–346
67. Nguyen TA, Lee D, Park JS (2013) Towards virtualization technology on satellite on-board computer system with hardware redundancy, software rejuvenation and virtual machine live migration techniques: modeling, analysis and implementation proposal (space, aeronautical and navigational electronics). IEICE Tech Rep 113:157–162
68. Wu C-M, Chang R-S, Chan H-Y (2013) A green energy-efficient scheduling algorithm using the DVFS technique for cloud datacenters. Future Gener Comput Syst 37:141–147
69. Zhou R, Liu F, Li C, Li T (2013) Optimizing virtual machine live storage migration in heterogeneous storage environment. In: Proceedings of the 9th SIGPLAN/SIGOPS international conference on virtual execution environments. ACM, New York, pp 73–84
70. Liu S, Ren S, Quan G, Zhao M, Ren S (2013) Profit aware load balancing for distributed cloud data centers. In: 27th international symposium on parallel & distributed processing (IPDPS). IEEE, New York, pp 611–622

71. Ibrahim KZ, Hofmeyr S, Iancu C, Roman E (2011) Optimized pre-copy live migration for memory intensive applications. In: Proceedings of 2011 international conference for high performance computing, networking, storage and analysis. ACM, New York
72. Zhu L, Chen J, He Q, Huang D, Wu S (2013) ITC-LM: a smart iteration-termination criterion based live virtual machine migration. In: Network and parallel computing. Springer, Berlin, pp 118–129
73. Osman S, Subhraveti D, Su G, Nieh J (2002) The design and implementation of Zap: a system for migrating computing environments. ACM SIGOPS Oper Syst Rev 36:361–376
74. Wood T, Tarasuk-Levin G, Shenoy P, Desnoyers P, Cecchet E, Corner MD (2009) Memory buddies: exploiting page sharing for smart colocation in virtualized data centers. In: Proceedings of the SIGPLAN/SIGOPS international conference on virtual execution environments. ACM, New York, pp 31–40
75. Jin H, Deng L, Wu S, Shi X, Pan X (2009) Live virtual machine migration with adaptive, memory compression. In: IEEE international conference on cluster computing and workshops, CLUSTER'09. IEEE, New York, pp 1–10
76. Hines MR, Deshpande U, Gopalan K (2009) Post-copy live migration of virtual machines. ACM SIGOPS Oper Syst Rev 43:14–26
77. Kozuch M, Satyanarayanan M, Bressoud T, Ke Y (2002) Efficient state transfer for Internet suspend/resume. Intel Research Pittsburgh, Technical Report IRP-TR-02-03
78. Glazer DW, Tropper C (1993) On process migration and load balancing in time warp. IEEE Trans Parallel Distrib Syst 4:318–327
79. Milojičić DS, Douglis F, Paindaveine Y, Wheeler R, Zhou S (2000) Process migration. ACM Comput Surv (CSUR) 32:241–299
80. Aikema D, Mirtchovski A, Kiddle C, Simmonds R (2012) Green cloud VM migration: power use analysis. In: IEEE international green computing conference (IGCC), pp 1–6
81. 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, New York, pp 135–146
82. Zhang X, Huo Z, Ma J, Meng D (2010) Exploiting data deduplication to accelerate live virtual machine migration. In: International conference on cluster computing (CLUSTER). IEEE, New York, pp 88–96
83. 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, New York
84. Wood T, Ramakrishnan K, Shenoy P, Van der Merwe J (2011) CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines. In: ACM SIGPLAN notices. ACM, New York, pp 121–132
85. Zhang Z, Xiao L, Zhu M, Ruan L (2013) Mvmotion: a metadata based virtual machine migration in cloud. Cluster Comput 17(2):441–452
86. Gerofi B, Vass Z, Ishikawa Y (2011) Utilizing memory content similarity for improving the performance of replicated virtual machines. In: Fourth IEEE international conference on utility and cloud computing (UCC), pp 73–80
87. Arabnia HR, Oliver MA (1989) A transputer network for fast operations on digitised images. In: Computer graphics forum. Wiley Online Library, New York, pp 3–11
88. 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 (CloudCom). IEEE, New York, pp 542–548
89. Treutner T, Hlavacs H (2012) Service level management for iterative pre-copy live migration. In: 8th international conference on network and service management (CNSM). IEEE, New York, pp 252–256
90. 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 34:1088–1096
91. Bradford R, Kotsovinos E, Feldmann A, Schiöberg H (2007) Live wide-area migration of virtual machines including local persistent state. In: Proceedings of the 3rd international conference on Virtual execution environments. ACM, New York, pp 169–179
92. Liu Z, Qu W, Liu W, Li K (2010) Xen live migration with slowdown scheduling algorithm. In: International conference on parallel and distributed computing, applications and technologies (PDCAT). IEEE, New York, pp 215–221
93. 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 (CCGrid), pp 73–83

94. Sahni S, Varma V (2012) A hybrid approach to live migration of virtual machines. In: International conference on cloud computing in emerging markets (CCEM). IEEE, New York, pp 1–5
95. Lann J, Van Bockhaven C (2014) Cryptanalysis of, and practical attacks against E-Safenet encryption
96. Arabnia HR, Oliver MA (1987) A transputer network for the arbitrary rotation of digitised images. Comput J 30:425–432
97. Al-Kiswany S, Subhraveti D, Sarkar P, Ripeanu M (2011) VMFlock: virtual machine co-migration for the cloud. In: Proceedings of the 20th international symposium on high performance distributed computing. ACM, New York, pp 159–170
98. Arabnia HR (1990) A parallel algorithm for the arbitrary rotation of digitized images using process-and-data-decomposition approach. J Parallel Distrib Comput 10:188–192
99. Hirofuchi T, Ogawa H, Nakada H, Itoh S, Sekiguchi S (2009) A live storage migration mechanism over wan for relocatable virtual machine services on clouds. In: Proceedings of the 2009 9th IEEE/ACM international symposium on cluster computing and the grid. IEEE Computer Society, New York, pp 460–465
100. Travostino F, Daspit P, Gommans L, Jog C, De Laat C, Mambretti J, Monga I, Van Oudenaarde B, Raghunath S, Yonghui Wang P (2006) Seamless live migration of virtual machines over the MAN/WAN. Future Gener Comput Syst 22:901–907
101. Lin H-P, Chuang C-C, Tseng H-W, Pang A-C, Lin P, Jeng J-Y (2012) A study of network infrastructure optimization for data center servers. In: 15th international symposium on wireless personal multimedia communications (WPMC). IEEE, New York, pp 164–168
102. Liu J, Wu J, Panda DK (2004) High performance RDMA-based MPI implementation over InfiniBand. Int J Parallel Program 32:167–198
103. Voorsluys W, Broberg J, Venugopal S, Buyya R (2009) Cost of virtual machine live migration in clouds: a performance evaluation. In: Cloud computing. Springer, Berlin, pp 254–265
104. Lu P, Ravindran B, Kim C (2011) Enhancing the performance of high availability lightweight live migration. In: Principles of distributed systems. Springer, Berlin, pp 50–64
105. Hirofuchi T, Nakada H, Ogawa H, Itoh S, Sekiguchi S (2009) A live storage migration mechanism over wan and its performance evaluation. In: Proceedings of the 3rd international workshop on virtualization technologies in distributed computing. ACM, New York, pp 67–74
106. Acs S, Gergely M, Kacsuk P, Kozlovszky M (2013) Block level storage support for open source IaaS Clouds. In: 21st Euromicro international conference on parallel, distributed and network-based processing (PDP). IEEE, New York, pp 262–268
107. Arabnia HR, Smith JW (1993) A reconfigurable interconnection network for imaging operations and its implementation using a multi-stage switching box. In: Proceedings of the 7th annual international high performance computing conference, pp 349–357
108. Daniel Versick DT (2010) Reducing energy consumption by load aggregation with an optimized dynamic live migration of virtual machines. In: International conference on P2P, parallel, grid, cloud and internet computing. IEEE, New York, pp 164–170
109. Ramakrishnan K, Shenoy P, Van der Merwe J (2007) Live data center migration across WANs: a robust cooperative context aware approach. In: Proceedings of the SIGCOMM workshop on Internet network management. ACM, New York, pp 262–267
110. Gebhardt C, Tomlinson A (2010) Challenges for inter virtual machine communication. Technical Report RHUL-MA-2010-12. Royal Holloway, University of London, Department of Mathematics
111. Huang W, Koop MJ, Gao Q, Panda DK (2007) Virtual machine aware communication libraries for high performance computing. In: Proceedings of the 2007 ACM/IEEE conference on supercomputing. ACM, New York
112. Li D, Jin H, Shao Y, Liao X (2009) A high-efficient inter-domain data transferring system for virtual machines. In: Proceedings of the 3rd international conference on ubiquitous information management and communication. ACM, New York, pp 385–390
113. Heninger IM, Hrischuk C, Jones ZH, Quirk AJ (2012) Automatically selecting optimal transport protocol in a cloud computing environment. In: Google patents
114. Xianqin C, Han W, Sumei W, Xiang L (2009) Seamless virtual machine live migration on network security enhanced hypervisor. In: 2nd IEEE international conference on broadband network & multimedia technology, 2009 (IC-BNMT'09). IEEE, New York, pp 847–853
115. Harney E, Goasguen S, Martin J, Murphy M, Westall M (2007) The efficacy of live virtual machine migrations over the internet. In: Proceedings of the 2nd international workshop on Virtualization technology in distributed computing. ACM, New York

116. Sharath Venkatesha SS, Kintali S (2009) Survey of virtual machine migration techniques. University of California, Santa Barbara
117. Vouk MA (2008) Cloud computing—issues, research and implementations, CIT. J Comput Inf Technol 16:235–246
118. Rimal BP, Choi E, Lumb I (2009) A taxonomy and survey of cloud computing systems. In: Fifth international joint conference on INC, IMS and IDC, 2009 (NCM'09). IEEE, New York, pp 44–51
119. Khan A, Othman M, Madani S, Khan S (2014) A survey of mobile cloud computing application models. IEEE Commun Serv Tutor 16(1):393–413
120. Garfinkel T, Rosenblum M (2005) When virtual is harder than real: security challenges in virtual machine based computing environments. In: HotOS
121. Danev B, Masti RJ, Karame GO, Capkun S (2011) Enabling secure VM-vTPM migration in private clouds. In: Proceedings of the 27th annual computer security applications conference. ACM, New York, pp 187–196
122. Wang W, Zhang Y, Lin B, Wu X, Miao K (2010) Secured and reliable vm migration in personal cloud. In: 2nd international conference on computer engineering and technology (ICCET). IEEE, New York, pp 705–709
123. Wang J, Wright K-L, Gopalan K (2008) XenLoop: a transparent high performance inter-vm network loopback. In: Proceedings of the 17th international symposium on High performance distributed computing. ACM, New York, pp 109–118
124. Zhang X, McIntosh S, Rohatgi P, Griffin JL (2007) XenSocket: A high-throughput interdomain transport for virtual machines. In: Middleware 2007. Springer, Berlin, pp 184–203
125. Kim K, Kim C, Jung S-I, Shin H-S, Kim J-S (2008) Inter-domain socket communications supporting high performance and full binary compatibility on Xen. In: Proceedings of the fourth ACM SIGPLAN/SIGOPS international conference on Virtual execution environments. ACM, New York, pp 11–20
126. Youseff L, Zagorodnov D, Wolski R (2008) Inter-OS communication on highly parallel multi-core architectures. Technical Report, University of California, Santa Barbara
127. Shuja J, Bilal K, Madani SA, Othman M, Ranjan R, Balaji P, Khan SU (2014) Survey of techniques and architectures for designing energy-efficient data centers. IEEE Syst J PP (99):1–13. doi:10.1109/JSYST.2014.2315823