Virtual Machines' Migration for Cloud Computing

Mohamed Riduan Abid
Alakhawayn University
Ifrane, Morocco
R.Abid@aui.ma

Karima Kaddouri
Alakhawayn University
Ifrane, Morocco
Ka.Kaddouri@aui.ma

Moulay Driss El Ouadghiri
Moulay Ismail University
Meknes, Morocco
dmelouad@gmail.com

Driss Benhaddou
University of Houston
TX, USA
dbenhadd@central.uh.edu

Abstract— **Virtualization is strongly emerging back as a fundamental Cloud Computing (CC) technology enabler whereby CC services are mainly provided via the instantiation of Virtual Machines (VMs). These instantiations follow a stochastic pattern, which is mainly dictated by the nature of the CC services requests and Cloud "elasticity". Consequently, a load-balancer emerges as indispensable to intervene in situations where VMs need to be dynamically migrated from a data center site to another in order to sustain optimal CC operation. In this paper, we briefly survey available VM migration techniques, delineate their pros and cons, and shed further light into the novel aspects to consider when approaching, these VM migration techniques, from a CC perspective, e.g., considering Mobile Cloud Computing (MCC) and Network Function Virtualization (NFV). In addition, we propose a novel VM migration scheme (soft-migration) inspired from mobile communication.**

Keywords-Virtual machines; Cloud computing; Live migration; Soft migration

I. INTRODUCTION

Virtualization has its roots in the mainframe era. The late 1960s witnessed the release of a novel memory time-sharing operating system known as the IBM 360 mainframe model 67 (a.k.a CP-67) to share scarce computing resources among multiple users. This was a major innovation: Personal users and organizations were actually able to use computing capabilities at a lower cost without having to own a computer. Some of the key customers to benefit from these time-sharing capabilities were MIT, Princeton University, Bell labs and General Motors [1]. Still, this early project encountered several issues, one of them being thrashing [2].

Optimizing resource utilization, in an era where computing demands are dramatically increasing, is a must. This can only be met through resource sharing and underutilization avoidance. Cloud Computing (CC) leverages optimal use of resources via the promotion of *computing as a utility* instead of a *product*. As a utility, users have on-demand access to computing resources in a similar way to other public utilities, e.g., electricity, water, and natural gas. Besides, users are charged only on what they have used, i.e., pay-per-use. To implement "pay-per-use", CC services need to be dynamically allocated and released, i.e., on-demand, and this is where virtualization. The latter is the main technology enabler behind CC. CC services are, in fact, provided via the instantiation of VMs whose "sizes" can be dynamically decided on, and can be created and released whenever needed. VMs are but image files that can be stored, updated, retrieved for execution, and even migrated from a physical station to another.

CC provides three basic service models: Software as a Service (SaaS), Platform as a Service (PaaS), and Infrastructure as a Service (IaaS) [4]. IaaS is provided via the instantiation of virtual machines (VMs). VMs refer to several virtual instances of an operating system running in isolated partitions within a physical machine [5]. In analogy with "time-sharing", which was developed to optimize resource utilization while giving each user the illusion of having access to a complete set of system resources, VMs take this idea further by providing users with complete system environments, each with its own operating system that manages virtualized hardware resources.

Due to the high and ever-growing demand that strains Cloud resources, the maintenance and management of CC operations against the heavy demand on CC services is a primary concern, especially that the requests on the CC services follow mostly a stochastic pattern which depend mainly on the time when the user requests a CC service and when it releases it. As a consequence, CC providers will witness a dynamic load on their data centres which can only be mitigated via the deployment of optimal load-balancing schemes. Besides, leveraging "elasticity", which is a fundamental aspect in CC, would further worsen the situation.

Elasticity is a major "pillar" in CC [4]. With elasticity, CC users can be allocated VMs with elastic sizes depending on the demand. For instance, an e-Commerce web service would need to be allocated further extra resources (e.g., number of vCPU, memory, etc.) during peak periods (e.g., week-ends and holidays). These extra resources need to be released once the need is over. The acquisition and release of these extra resources should be instantaneous, a fact that puts further pressure on the load dynamicity of CC data centers, and thus renders the deployment of an optimal Load-balancer (LB) indispensable. Besides deciding on whether to admit a VM request or not, the LB would often

 ISBN: 978-1-61208-607-1

need to move (i.e., migrate) a VM from one Cloud site to another one as a consequence of an instantaneous increase (elasticity) in the size of the current running VMs.

The process of moving a virtual machine from one physical host to another is labelled migration [6]. This merely consists on the transfer of the VM state, which is dictated by the actual memory image, virtual CPUs states, and the states of attached IO devices. There are basically two main migration techniques: pre-copy [7] and post-copy [8]. Initially, these techniques were not tailored to fit the CC services. Thus, they need to be adapted and fine-tuned to fit in the CC contest. Two key performance metrics are considered to evaluate a migration technique: downtime and migration time. Downtime is the time during which the VM is unreachable to the user because the VM is in the period of transiting from a site to another, and migration time, which is the total amount of time that is needed to transfer the VM from source to destination while keeping it accessible. With the arousal of VM migration, and in order to move a VM between two physical machines, it was obligatory to completely shut down the VM, prepare the destination host resource-wise, move the VM files and then start the VM in the new machine. Nowadays, thanks to several migration techniques, we can move VMs with minimum downtime.

In this paper, we shed further light into the subtleties of VM Migration. We survey available VM migration techniques, present their strengths and weaknesses, and advocate considerations to account for when approaching it from a Cloud Computing point of view, mainly Mobile Cloud Computing (MCC) which is rapidly increasing domain marrying CC and Mobile Computing, and Network Function Virtualization (NFV) which is deemed as the key towards the Cloudification of the Telecommunication world.

Besides, we present a novel VM migration (soft-migration) scheme that is inspired from mobile computing. This promotes the complete elimination of the downtime by managing a time interval whereby VM requests are served simultaneously by the source and target VMs in a similar way to the soft hand-off process in cellular telephony. This will assure the elimination of the downtime.

The rest of the paper is organized as follows: Section II briefly surveys the different VM migration techniques In Section III, we address VM migration from a CC perspective and present relevant live migration use cases, e.g., MCC and NFV. In Section IV, we present our novel soft-migration scheme, and finally, we conclude and set future work in Section V.

II. VM MIGRATION TECHNIQUES

There are two main techniques for moving VM's memory state: pre-copy [7] and post-copy migration [8]. In a memory transfer we have three phases. First, we have the (i) *Push phase* where the original VM keeps its running status whereas some of the memory pages are being pushed through the network to the target host. To make sure the transfer is consistent, updated pages have to be retransmitted thereafter. Second, in the (ii) *Stop-and-copy phase*, the original VM is stopped, all the remaining dirty pages are copied to the destination, and the VM is resumed on the destination host. Finally, in the (iii) *Pull phase*, the copied VM begins its execution. If it comes across a page that has not yet been transferred, this results in a page fault from the VM.

Pre-copy migration combines both the push phase and the stop-and copy phase. The post-copy approach combines the pull phase and the stop-and-copy phase. In *pre-copy memory migration* (figure 1), the hypervisor copies all the memory pages in an iterative fashion from source to destination while the VM is still running at the source. If some memory pages change (i.e., they become dirty) during this process, they will be re-copied. Once enough pages are transmitted (Threshold on the maximum number of iteration is defined by the user at run time.), the VM is suspended at the source and the remaining state is relocated to the destination [7]. In *post-copy migration* (figure 2), the transfer is initiated by suspending the VM at the source. With the VM suspended, a minimal subset of the execution state of the VM (CPU state, registers, non-pageable memory) is transferred to the target. The VM is then resumed at the target. At the destination, if the VM tries to access a page that has not yet been transferred, it generates a page fault. Concurrently, the source dynamically pushes the remaining memory pages of the VM to the target - a technique known as *pre-paging*, which minimizes page faults [8].

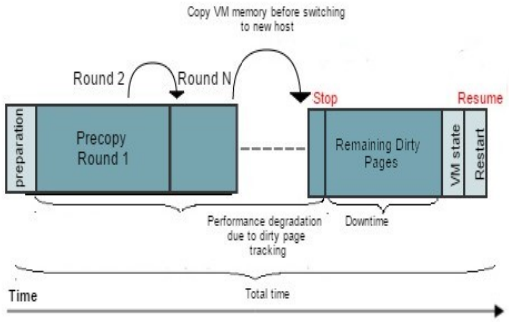

Figure 1. Pre-copy migration

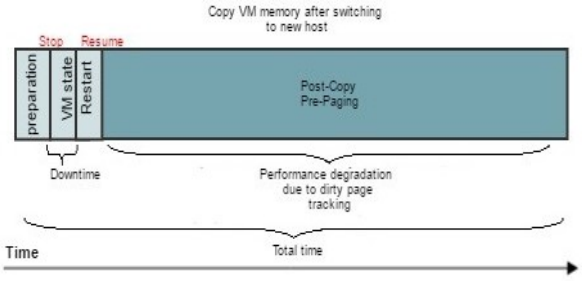

Figure 2. Post-copy migration

 ISBN: 978-1-61208-607-1

The techniques discussed above have been remodelled and readapted by a number of researchers to minimize downtime when live migrating VMs.

E. Zaw et al. [9] propose an updated version of the pre-copy approach. The designed framework is built to include a pre-processing phase in order to decrease the size of transferred data. A working set prediction algorithm is used to implement the pre-processing step. The suggested algorithm predicts the least recently used memory pages that directly affects the total migration time. As a result, the transferred memory page size is diminished. Evaluation of the algorithm showed that the proposed solution –compared with traditional pre-copy- can decrease the total migration time by 11.45%.

With the objective to optimize virtual machine migration based on pre-copy as well, H. Deng et al. suggest a memory compression solution. Similar to the previously mentioned work, the idea of reducing the size of migrated data to improve the performance of VM migration is applied. In the source node, data being transferred is first compressed by a compression algorithm then, upon arrival to the destination node, it is decompressed. The added metric here, compression time, is an extra overhead caused by the compression jobs. In [11], J. Changyeon et al. propose a shared storage based technique for live migrating VMs. Only unique memory pages are sent directly from the source to the destination. Pages that are replicated and found on shared storage are fetched directly by the destination node, so duplicated data is not sent by the source node. The authors demonstrated the efficacy of their suggested technique by running a set of experiments with the XEN hypervisor. There was an improvement in total migration time between 30%-60% with minimal downtime rise, hence improving the migration performance.

T. Wood et al. [12] present CloudNet, a CC platform that attempts to deliver smooth connectivity between enterprise and datacenter sites. It also implements wide area network (WAN) migration of VMs. The objective once more was to minimize the size of transferred data, migration time, and user downtime. This framework uses asynchronous and synchronous disk duplication to reduce downtime. The performance of this platform was evaluated in a setting composed of three geographically separated data centers and a local testbed. The result showed that memory transfer time was decreased by 65%.

III. LIVE MIGRATION IN CC

Live migration refers to the process of moving a virtual machine from one physical host to another while the VM is continuously powered-up. When properly done, this process takes place without any perceptible effect from the point of view of the end user. When a VM is running a live service, it is important that this transfer occurs in a manner that balances the requirements of minimizing both downtime and total migration time. Thus, live VM migration is crucial for dynamic resource management and proper carrying of CC services in Cloud-based systems.

The idea of viewing computing resources given by Cloud providers as a single unified pool is ideal. However, the reality is far from this vision because these resources are distributed across geographically separated and interconnected datacenters. This presents a real challenge for live VM migration in CC. The majority of authors presented previously tackle live migration techniques from the local area network (LAN) perspective, under the assumption of a shared file system that allows migrating only memory data and evading disk state transfers. The situation is further worsened with the arousal of MCC [13] whereby mobile devices will be mostly seeking Cloud services in order to mitigate their inherent limitation on the computer power and energy/battery.

MCC is strongly arising as a promising technology leveraging Mobile Computing and Cloud computing. MCC is driven by the emergence of novel IT ecosystems alike IoT (Internet of Things) [14], Smart Cities [15], and Smart Grids [16] whereby mobile devices, e.g., sensors/actuators, will be all the time connected to the Cloud.

Due to the inherent limitations in energy and compute power, mainly due the restriction in device size, these devices will be mostly sending data for processing in the Cloud. The mobility and ad-hoc topology of these sensors/actuators will generate a dynamic load on the Cloud, thus requiring an optimal load-balancer. The load-balancer needs to account for this dynamicity by migrating VMs (that will process mobile devices requests) into “closer” locations. This is crucial for providing and maintaining requested QoS, especially the delay.

Most of the previously proposed VM migration techniques were not accounting for the novel mobile Cloud services. With the rising demand for mobility of resources, the requirement for MCC also increased, and nowadays users rely on mobile devices. These devices can be effortlessly connected to the Cloud, and accordingly, mobile applications can easily access Cloud resources. Next section delineates the most prominent contributions for VM migration in MCC.

A. VM Migration for Mobile Cloud Computing

1) State of the Art

The development of mobile agents plays an important role in remote access, data retrieval, and most importantly mobile Cloud computing. M. Zaa et al. [17] tackle the issue of migrating resources between the Cloud and mobile devices using mobile agents. They can migrate from one host to another host in search of resources. In particular, they can be used to transport resources such as the VM’s state from one environment to another, with its data remaining intact and capable of executing in a new environment.

Chun et al. [18] focus on VM migration using VM state cloning. Their system, CloneCloud, duplicates the runtime environment and then executes the application-level VM either on the Cloud or the device. The goal is to achieve better performance with a boosted CPU and memory resources that can be exploited proficiently. However, the application on the Cloud needs to access physical hardware on the mobile devices. Henceforth, reproducing a device and then executing it on the Cloud also adds more complexity.

K. Ma et al. [19] highlight VM state migration using the Stack on Demand (SOD) concept. Instead of using live VM migration which can be too "massive" (i.e., bulky data unfit for mobile devices), they propose a compressed migration scheme intended for stack-based virtual machines. This mechanism migrates the minimal portion of the VM state to the destination host for continued execution. Inspired by the stack concept, it chops the stack into segments and only transfers the top segment at a time. M. Islam et al. [20] propose a Genetic Algorithm based VM migration scheme for a heterogeneous MCC system. Their genetic algorithm leverages both user mobility and the load of the Cloud servers to enhance the efficacy of VM migration. It chooses the fittest Cloud server from the pool for a mobile VM and decreases the total number of VM migrations. Thus, it ensures a smaller task execution time. In [21], a technique called dynamic VM synthesis was presented. This is based on Cloudlets. A Cloudlet is a small-scale Cloud intended for delivering computing resources to high-performing mobile applications. In this scheme, a VM overlay (i.e. file that captures a VM state) is sent by a mobile device to a Cloudlet that has the base VM from which this overlay was created. The Cloudlet merges the overlay with the base to synthesize the ready-for-launch VM, which starts execution at the exact state the mobile communicated.

2) Discussion

In [17], the mobile agents used to transport the VM state from one environment to another also need to be migrated, and depending on their availability, there can be some downtime. Although the solution in [18] boosts performance and considerably decreases user response time, there still is minimal downtime when a migration point is reached. The VM thread is suspended and its state sent to a clone. There, the thread state is instantiated into a new thread and execution resumes. In [19] when the top stack segment finishes and pops, the return values are sent to the next site for continued execution. However, there are often freeze times between the multiple hops from one site to another. For [20], when an adequate Cloud is found for the mobile VM, there still is suspend time occurring as the VM state is loaded on the Cloud. While significantly decreasing response time, the solution in [21] still generates a few milliseconds of downtime before the application is executed on the Cloudlet. Furthermore, in the case when the Cloudlet is not available nearby, the mobile device would need to connect to a distant Cloud, which degrades performance.

In all of the previously mentioned contributions, the main object of migration, the VM, is mainly an OS, a server or an application. However, this is not always the case. There are other use cases for live VM Migration not limited to this. Thus there is a need to delve deeper and investigate other use cases that justify the need for live migration. This is particularly relevant for the paradigm shift we are witnessing nowadays in networks, in what is referred to as Network Functions Virtualization (NFV) [31].

B. NFV and Live VM migration

Network operators are becoming saturated with an increasingly large quantity of network hardware appliances. Launching a new network service usually necessitates finding the space and power needed. Accommodating these resources is becoming more and more challenging due to the increasing costs of energy and capital investment, but also the scarcity of skills essential to design, integrate, and operate such complex hardware.

Furthermore, dedicated hardware quickly reaches end of life, which implies that the purchase-integrate-deploy cycle to be repeated is with little or no revenue benefit at all. Even worse, with the current technological innovation acceleration, hardware lifecycles are becoming even shorter as dedicated hardware becomes rapidly obsolete. This highly impedes on revenues innovation in a progressively network-focused connected world.

NFV's goal is to leverage one particular technology that enables CC: Virtualization. Hardware Virtualization is needed to link traffic between VMs and physical interfaces. This connection is possible with the use of hypervisors and other virtualized resources such as virtual Ethernet switches (vSwitches). Cloud infrastructures provide mechanisms to enhance resource availability, organization, and administration. It also delivers automatic forking of VM resources, the re-launch of failed VMs, and the migration of VMs. These provide a much needed boost for incorporating NFV in the cloud infrastructure.

NFV is a radical adjustment to the way network operators design networks. It applies virtualization technologies to consolidate network hardware onto virtualized servers, switches and storage. These resources might be located in datacenters, network nodes, or in the end user location. It requires the implementation of network functions at the software level made to run on standard server hardware, also called "commodity of the shelf" (COTS). These network functions can be migrated to, or forked in various places in the network as required by demand, without the need ever for installing and deploying new hardware equipment.

Software-Defined Networking (SDN) [23] is a concept related to NFV. SDN is a model that decouples the data plane from the control plane, in such a way that the control plane is central and the forwarding components remain distributed. NFV is not dependent on SDN. It is completely

 ISBN: 978-1-61208-607-1

feasible to implement NFV as a standalone entity using existing networking technologies. However, the two are complementary and there are benefits to using SDN concepts to develop and orchestrate an NFV infrastructure.

Last but not least, the network functions to virtualize (e.g., BBU (BaseBand Unit), switches, routers) might need to be migrated as well, and the trend is not the same as with ordinary VMs which consist on an OS on top of a virtual hardware.

IV. SOFT MIGRATION

Our contribution in this survey is our proposition of a soft-handover inspired framework for VM migration, that we named soft-migration. Soft-handover [24] is a scheme where a mobile phone is concurrently connected to two or more radio base stations during a phone call. The cell receiver combines the signals of both base stations for a bit stream of better quality. If any one of the signal fades, there will still be acceptable signal strength from the remaining radio station.

We can use this concept of simultaneous connection with the VM migration scheme (see Figure 3), where the VM is connected to both hosts during migration time. This allows it to run continuously regardless of the transfer state, permitting a seamless VM migration. Thanks to unceasing memory transfer from both servers, there would ideally be very minimal disruptions and downtime.

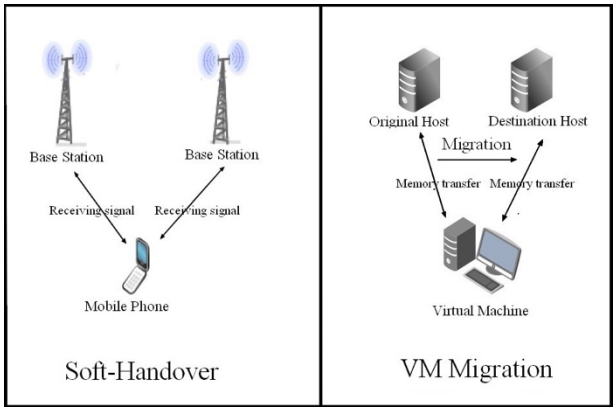

Figure 3. Soft-Migration

Our soft-handover inspired idea aims to minimize downtime as its primary objective. The pre-copy and post-copy approaches have different suspend time gaps.

In post-copy, the suspend phase is at the very beginning, when the VM is stopped and a minimal subset of it gets copied to the target. Then, VM resumes execution at the new host whilst the remaining pages get sent. The problem here would be the high number of page faults that might get generated if the user only accesses the pages that have not yet been copied. This would result in further suspend time when the wanted pages are being pulled from the source. With pre-copy, the memory pages are copied without stopping the execution of the VM. Then occurs the suspend phase where the remaining and dirtied pages get sent, and finally, execution is resumed on the new host. Here, there is no page faults issue. Since the suspend phase does not occur at the beginning, the majority of memory pages are copied before the VM gets paused. Our proposed scheme can be deemed as a "hybrid" approach between the pre-copy and post-copy techniques, and consists of 2 phases, see figure. 4:

Phase 1: Similar to the pre-copy approach, memory pages are copied from the original host to the target host without stopping the execution of the VM.

Phase 2: After the maximum number of iterations is reached (defined by the user), we switch to the new host and resume execution there. There is no suspend phase. Instead, and similar to the post-copy approach, the rest of the pages will be dynamically pushed. If the user tries to access a non-copied page or a dirty page being replaced, this will generate a page fault. The missing page is dynamically pulled from the source.

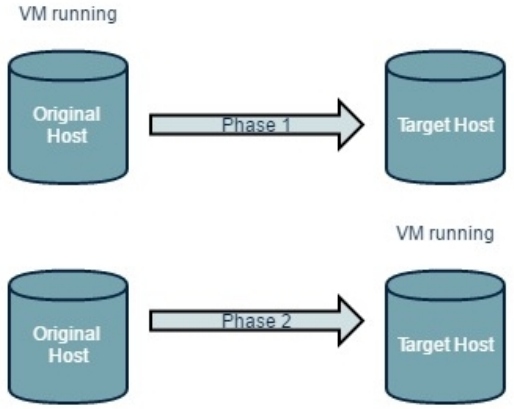

Figure 4. Proposed Migration Process

This approach ensures that the user is continuously connected to either host during the VM transfer (see Figure 5), just as mobile users do in the hand off scenario.

The main advantage of the proposed scheme is that there is *no* suspend time, and thus we have less *down time*: VM execution is continuous even when the original host replaces dirty pages and transfers remaining ones. Still, we have to keep track of which pages are dirty and which are not in order to minimize page faults, and this mandates the implementation of module, within the Load-balancer, that logs relative VM pages migrations.

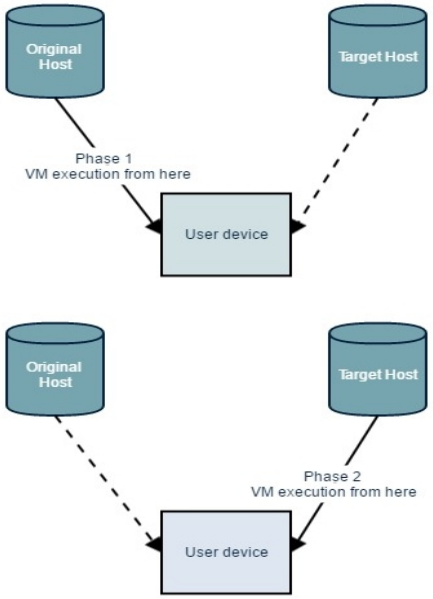

Figure 5: User is never disconnected

V. CONCLUSION AND FUTURE WORK

In this paper, we surveyed main VM Migration and addressed its particularity from a Could Computing perspective, mainly when dealing with MCC services, and when migrating network function either using NFV or SDN., We also presented our own soft-migration technique inspired from the soft-handoff mechanism in mobile communication.

Our future work consists of implementing the soft-migration scheme and study its performance. In addition, we will investigate plausible schemes for network function migration in 5G whereby most of the (telecommunications) functions are to be virtualized.

REFERENCES

[1] D, Morton, “IBM Mainframe Operating Systems: Timeline and Brief Explanation for the IBM System/360 and Beyond”, IBM, 2015.

[2] P. J. Denning, “Thrashing: Its causes and prevention”. Fall Joint Computer Conference, pp. 915-922, 1968

[3] D. Dale, “Server and Storage Virtualization with IP Storage”, Storage Networking Industry Association (SNIA), 2008.

[4] P. Mell and T. Grance, “The NIST Definition of Cloud Computing”. National Institute of Standards and Technology, pp. 1-3, 2011.

[5] J. Smith and R. Nair, Virtual Machines: Versatile Platforms for Systems and Processes, Elsevier, 2005.

[6] C. Christopher, et al, “Live migration of virtual machines”. In Proceedings of the 2nd conference on Symposium on Networked Systems Design & Implementation pp 273–286, 2005.

[7] D. Botero, “A Brief Tutorial on Live Virtual Machine Migration from a Security Perspective”, Princeton University, 2011.

[8] Hines, U. Deshpande, and K. Gopalan, “Post-Copy Live Migration of Virtual Machines”, SIGOPS Operating Syst. Review, 43(3):14–26, July 2009.

[9] E. Zaw and N. Thein, “Improved Live VM Migration using LRU and Splay Tree Algorithm”, International Journal of Computer Science and Telecommunication, vol. 3, no. 3, 2012.

[10] H. L. Deng, S.W. Shi and X. Pan, “Live Virtual Machine Migration with Adaptive Memory Compression”, IEEE, 2009.

[11] J. Changyeon, E. Gustafsson, J. Son and B. Egger. "Efficient Live Migration of Virtual Machines Using Shared Storage", in the 9th ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp 41-50, 2013.

[12] T. Wood, P. Shenoy, K.K. Ramakrishnan and J. Van der Merwe CloudNet, “Dynamic Pooling of Cloud Resources by Live WAN Migration of Virtual Machines”. In Proceedings of the 2011 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments, pp 51–60, 2011. ACM.

[13] T. Dinh et al, "A Survey of Mobile Cloud Computing: Architecture, Applications, And Approaches", Wireless Communications and Mobile Computing Wirel. Commun. Mob. Comput, pp: 1587–1611, 2013.

[14] F.Xial et al, "Internet of Things", International Journal of Communication System Int. J. Commun. Syst. pp: 1101–1102, 2012.

[15] H. Chourabi, T. Nam and S. Walker, "Understanding Smart Cities: An Integrative Framework", 45th Hawaii International Conference on System Sciences, pp: 2289-2297, 2012.

[16] J. R. Roncero, "Integration Is Key to Smart Grid Management", CIRED Seminar 2008: SmartGrids for Distribution Frankfurt, 2008.

[17] M. Zaa, J.P, Gabhane and A.V Dehankar. “A Survey on Migration of Task between Cloud and Mobile Device”. International Journal of Advanced Research in Computer Engineering & Technology (IJARCET), pp. 610-613, vol 2, no. 2, 2013.

[18] B. Chun et al, "CloneCloud: Elastic execution between mobile device and Cloud," In Proc. Of Eurosys, 2011.

[19] K. K. Ma et al, "A Stack-On-Demand Execution Model for Elastic Computing," In Proc. of the 39th Intl. Conf. on Parallel Processing (ICPP2010), pp. 208-217, 2010.

[20] M. Islam, A. Razzaque and J. Islam, "A genetic algorithm for virtual machine migration in heterogeneous mobile Cloud computing," 2016 International Conference on Networking Systems and Security (NSysS), pp. 1-6, 2016.

[21] M. Satyanarayanan et al, “The Case for VM-based Cloudlets in Mobile Computing”, IEEE Pervasive Computing, 2009.

[22] B. Han, V. Gopalakrishnan, L. Ji and S. Lee, "Network function virtualization: Challenges and opportunities for innovations", IEEE Commun. Mag., vol. 53, no. 2, pp. 90-97, 2015.

[23] “Software-Defined Networking: The New Norm for Networks”, ONF White Paper, April 13, 2012.

[24] Juan et al, “Verification of Mobility-Based Soft Handover Algorithm using WCDMA Measurements Data”, IEEE 63 rd Vehicular Technology Conference, 2006.