Sindh Univ. Res. Jour. (Sci. Ser.) Vol.50(001) 107-114 (2018)
http://doi.org/10.26692/sujo/2018.01.0018
SINDH UNIVERSITY RESEARCH JOURNAL (SCIENCE SERIES)

State-Full Virtual Machine Live Data Migration For Improved Load Balancing

M. SHAIKH, A. SHAIKH*++, M. A. MEMON**, A. A. SHAH** R. H. SHAH**

Department of Computer Systems, Mehran University of Engineering and Technology, Jamshoro,Sindh, Pakistan

Received 30th January 2017 and Revised 4th March 2018

Abstract: Cloud resembles with Virtualization to achieve the shared environment and shared resources. In virtualized shared environment, the availability of the shared resources for the shared multiple clusters/Virtual Machines is always matters. To provide the better performance to the end users of the cloud VMs and for the always availability of shared resources, virtual machine migration is always a challenging task in virtualized environment. The aim of this research is to investigate various parameters of live data migration between the shared clusters/virtual machines in state-full context and overall load balance level. The proposed migration strategy ensures the avoidance of attacks/anomalies which causes the load increase on the VMs which effects the entire system. This could be done by the resource allocation from one cluster to another via checkpointing method of VM to maintain the state of the VM, load of the VM including resource availability. Using Xen hypervisor memory technique which dynamically allows the migration of the configured memory while the allocated memory could be discarded for a while. By this the bad memory remains un-migrated only the good memory consisting the used data would be migrated by means of Real-time.

Keywords: Cloud, VMM,VM, VM MIgration, Xen, Linux

1. INTRODUCTION

The term Cloud resembles to the Internet in network diagrams. In simple words, the cloud can be defined as any computer entity which is being hosted on the web as a service (Maohua 2009). Internet has various parts of schematic clouds. Through Cloud computing many applications of different types and services are being delivered through the internet cloud. The idea of cloud computing is gaining a lot of success in entertainment and business applications due to its virtualized softwares, which enables the sharing of infrastructure such as physical services, storage, provision of services according to users demand, network access and other networking capabilities etc. (Sagar *et. al.*, 2007)

Despite the success, popularity and availability of providers of the virtualized cloud computing paradigm, there are significant number of involved challenges and risks as well. User can face privacy issues, data unavailability, attenuation of network, lack of resources, performance lack, scalability clash, huge work load complexity, non-robustness, slow up migration and programmability issues. In this work work, the issues are discussed and analyzed

– Network availability
– Resources availability
– Data Integrity and Consistency
– Prescribed storage available on the sink VM during the migration
– Seamless Migration process to the end user
– State Maintenance of the VM
– Attack detection

Easy to enhance and upgrade a Virtual Machine
– Real-time Availability in data context
– Facilitates for fault Diagnosing and its management

2. LIVE MIGRATION

Virtualization is an essential element for the shared environment and cluster, distributed computing. One of the main key points of the shred environment is the high availability of the computing resources at any time with the least cost and sometimes free. Also in the shared computing the resources are always be there for any guest in the cloud based distributed environment in the operational form. This can be done by several ways but one of the main motivation of this sharing of the shared resources between the multiple guests is known as virtual machine live migration. By taking the advantage

++Corresponding Author Email: shaikhasad@hotmail.com; Tel.: +92-333-2859550 muhammad.ali@usindh.edu.pk
*++Department of Software Engineering, ILMA University, Korangi Creek, Sindh, Pakistan
**Institute of Information and Communication Technology, University of Sindh, Pakistan

live migration of the guests systems the shared computing made quite easy without any load balancing and expensive computing and processing power. The factors involved in motivation leads to the virtual machine migration in Real-time are:

– Optimization of Migration Downtime in Multiple data context
– Network Congestion
– Recovery from Host/Guest failure
– Seamless Maintenance of Virtual Machines to the user
– Virtual Machine/Guest Robustness

3. RELATED WORK:

3.1 MIGRATION WITHOUT VIRTUAL MACHINE MONITOR

Most of the hypervisors likewise KVM, VMware, Xen ensures hypervisor-based migration elucidation (Jin Heo, et. al., 2009). However, quite a few demerits that has recently get significant attention due to rely on the hypervisor (DSN 09, 2009). The main issue is that the virtual machine monitor plays a very vital role throughout the whole migration procedure, and considers many "responsibilities" form the beginning till end of the entire migration process. This innermost role of the hypervisor has widely turn out to be source of several foremost security apprehensions. Szefer et al. shows that the hacker or attacker can easily access memory, expose authenticated information as well as make modification in the software used by a host VM if the malicious party attacks on hypervisor successfully. Note that inspection of such security threads is outside form scope of this study. Thus, the possible security issues caused by hypervisor reflect us to discover the VM live migration techniques that do not involve any hypervisor for migration.

As far as these issues are concerned, for their solutions a technique such as migration without hypervisor is required. Besides pointing out the security concerns, in an attempt to solve these issues, the solution must involve use-case developments that come up from the VM common practice. One of them is known as data center setting, which represents and offers to each user with an isolated VM. Likewise in this setting, a wholly customized VM is often initiated and created for every user, with custom application and environment. Also in this setting, there are numerous reasons to show why a system administrator needs to take decision of VMs migration, like as for fault-tolerance and load balancing purposes. In such cases, what are the required metrics that should be considered to provide always resources huge availability?

As discussed earlier, the downtime occurrence by the usey ensures the time intervals on which the user cannot access the system, is a transparent essential metric. Downtime should be as less as possible. Although the minimal downtime ensures by most of the hypervisor-based migrations. In accumulation with downtime, another essential metric is total waiting time to accomplish the migration, as it involves the provision of resources on both of the sink and target backup host(Maohua Lu, 2009). For instance, for hardware updates, load-balancing and policies of energy savings etc., a host with running state is usually be shut down. In these cases, the alert and active VMs should be migrated to other backup VMs and also ensures their resumptions on the target hosts before the replacement of the resources on the sink and target host machines. Most previous reviews only focus to provide minimal downtime. Besides downtime, this wok proposes for the optimization of minimization of total migration time.

3.2 Method For Vc Checkpointing

Checkpointing methods for multiple VMs—a challenging target space—have also been discussed. The efforts in this space, the awareness of including VCCP,VirtCFT, VNsnap. VirtCFT ensures the always availability by checkpointing of individual VMs for virtual clusters to additional backup host. A two phase commend coordinated blocking algorithm by predicting FIFO-based communication channels as the checkpointing global algorithm Michael (Litzkow, *et. al.,* 2009). Thus, checkpoints request broadcasts to all the VMs by checkpoint coordinator and waits for a while to ensure two-phase acknowledgments. Since due to use of the FIFO-channel based checkpointing algorithm, in compatibility the network also must be FIFO, which must make scope of the work limited and or such channels must be eliminated which uses overlays i.e. increasing overheads. In addition, as VirtCFT by using a checkpoint coordinator which enables communication and make contact with VMs several times with each individual VM during the checkpointing, due to the additional communications delays the downtime is increased for checkpointing. VCCP also depends upon the trustworthy FIFO transmission to deploy the algorithm blocking coordinated checkpointing. VCCP faces the problems of overheads of checkpointing before the coordination with VM and detaining in-transit frames of Ethernet, due to the coordination algorithm.

VNsnap captures the global snaps of virtualization-based networked systems and no any reliable FIFO data transmission required, and it is completely based on unblocked distributed snapshot algorithm (Litzkow, et. al.,2009). VNsnap works outside from the shared networked environment and one of the key ideas is no any modification is needed in the software running in the VMs. The VNsnap presents the two daemons for

checkpointing, one is VNsnap-disk and another is VNsnap-memory. These daemons produce a large checkpoint size, which is equal to the guest memory size. Also, checkpoints will stored in memory by VNsnap-memory, which copies the memory in duplicate snapshot of memory, resulting of memory overhead roughly of up to 100%. In addition, its distributed algorithm by using the receive-but-drop approach, which caused the back off for a while of the TCP connections suite at the virtual network level after the checkpointing. And the back off of the TCP is completely intolerable and critically changes the downtime.

4. CONTRIBUTIONS:

4.1 Live Migration with Load Balancing

One of the common and generic problem in live migration of VMs is load balancing, which have also been studied and discussed at the different levels, at multiple and different decades by using a range of strategies (Elmore, *et. al.,* 2011) (Singh, *et.al.,* 2008). Our goal and a general motivation is the optimization of the use of computing resources i.e., CPU (processing power), memory storage in a shared computing environment. Conventionally the process migration is used to share the workload form the heavy loaded processors to light load processors in the cluster systems. By taking the advantage of changes in execution of process of utilization of network process migration allows to perform its functions at the user level and/or kernel level. Strategies at the user level by using checkpointing allow the dynamic migration of processes (Michael *et. al.,* 1999) . According to few proceedings, the successful implementation of live migration always needs strong cooperation between the co elements of migration and process (Freedman, 1991). A common trouble faced by this user level implementation is to access without kernel support, they are incapable for the process migration with inter-process communication and location aware information. On the other hand, implementation at the kernel level enables the process migration quite quicker as well as capable to migrate multiple types of processes (Christopher *et. al.,* 2005) (Carlo *et.al.,* 2011). In comparison with user level strategies, kernel level ensures better performance. Thus kernel level strategies are more efficient process migration techniques, they needs some modification at the kernel level of operating system.

As nowadays virtualization turn out to be more and more common, these limitation can be overcome during process migration by making an individual VM as a load balancing unit. In the virtualized and shared world, a VM runs all the processes and applications, so now it could quite probable to carry the load balancing based on entire-system replication (DSN 09, 2009), (Freedman, 1991). At present multiple existing techniques are available for the live migration on virtualization based system (Ellard T Roush, 1995), (Fred *et. al.,* 1991). By using live migration, lots of new findings have been proposed by many researchers in the cluster based systems to improve the performance of load balancing. Some of them work by using the prediction technique to anticipate the future resources and their demands based on the current resources utilization (Jin *et. al.,* 2009) (Xu, *et.al.,* 2008). Though, to get accurate prediction, these multiple works requires to attain and analyze the performance response all the time, which establishes overhead. As in comparison of this work, this work does not predict and make predication in advance, instead, our goal is to always refer and update the past record to help in order to achieve the final decisions. Some other previous literatures highlights on the designing a set of instruction (algorithm) to investigate where and what to migrate and amount of resources allocated after migration (Zhang, *et. al.,* 2010) (Michael *et.al.,* 1999). Conversely, they do not care about the migration cost and also not consider the downtime and migration performance time which are not evaluated in experimental results.

4.2 Adaptive Live Migration To Overcome Load Balancing

In virtualization based shared environment, there are numerous physical machines i.e., VMs in the running state are virtually inter-connected via high data rate internetworking technologies. They are capable to provide increasingly on demand high availability of computational resources and services. One of most motivating feature is lake of available computing resources i.e. (CPU processor, Secondary Disk storage and virtual memory storage, processor state etc.). They could be shareable through the active interconnections between all the VMs. Many previous researchers have observed that multiple computational resources are remained unused for a substantial quantity of the operational time (Maohua *et. al.,* 2009) (Sagar *et.al.,* 2007). Hence, avoidance of load balancing gained huge interest in order to avoid some more critical situations of overloading where some machines are running with huge load and others are in idle state i.e. under or within workload limit.

Usually, there are multiple existing methods to gain load balancing in interconnected systems. One of the simple and attractive methods is static load

balancing solution, and this allocates to machines with applications at the be- ginning. The efficiency of this type of strategy relies on accuracy dependency of prior prediction of load. While, another strategy i.e. dynamic load balancing could be exploited by run-time migrating applications and processes among different physical machines. A dynamic load balancing approach is more effective and efficient than it limits some applications to process or run on the VMs where those were initially assigned. However, because of the applications strongly bounds to the host operating system (i.e. sockets and file descriptors) as well as platform dependent (i.e. natively compiled codes and device drivers), so due to these it is too much a complex task to implement a mechanism for process migration. Moreover, for communication some kind of process may relies on shared memory, which suffers from residual dependencies problems and founds further complications.

Limitation coming from process migration are overcome by through virtual machine live migration. In contrasts with process migration, VM live migration migrates CPU virtual state, guest OS data and memory and state of emulated devices, which removes platform or OS dependencies. A guest VM with multiple running applications, which is heavy in load can be migrated to the hypervisor of another machine. The same machine can be considered as idle machine in order to utilize huge resources availability. Furthermore, in contrast to VM migration sop-and-resume strategy, natively VM live migration guarantees minimal downtime as well as minimal interruption in VM users interaction. Therefore, a better strategy of load balancing must also ensures minimal down- time to users.

5. VIRTUAL MACHINE SAVE AND RESTORE MECHANISMS: VM CHECKPOINTING

In order to provide some benefits such as rapid and dynamic resource allocation, high availability with improved load balancing, a functional feature in virtualization is the saving and restoring of VM is presented in this work. The proposed strategy ensures an entire virtual machine by using the transparent checkpointing is captured by taking snapshot of the VM. After that the snapshot will be restored in the target VM and the restore mechanism with the configured memory. Many latest virtualization systems gives very basics of resumption and VM checkpointing mechanism for saving the current running state of an active guest VM in the form of a checkpoint file, and also, then to resume that the same saved VM by from checkpoint file to correct and same consistent suspended state (for e.g., KVM, Xen, VMware).

From figure 4, we also summarize, for the effective VM-level checkpointing and suspension; the hypervisor should have ability to resume the VM rapidly from the check-pointed state. Users and clients of network based applications are more tending to suspend an idle guest VM if the latency magnitudes for VM resumption are leads to in order as seconds than minutes. The capability to quick and rapidly restore a VM from a pre saved checkpointed image could also make possible many other effective features, including quick recovery form crash, quick reallocation of VM, debugging, and testing etc.

Traditional virtual machine resumption techniques can be classified in two solutions. First solution initially restores each and everything whatever saved in checkpoint state, and then starts guest execution. As the VM memory size and checkpoint size dominates with each other, this type of solution works well for less memory sizes (i.e. in MBs). On the other hand, whenever size of memory becomes large (i.e., in GBs), the time of VM resumption significantly increases (i.e. in 10s of seconds). From figure 1 (a) it is illustrated the time consumed by native mechanisms for save and restore of Xen VMM as a function of memory size. We observed that time consumed by save/restore mechanisms of Xen are in the form of multi-digit order when the size of memory approaches to 1GB.

To start the VM as soon as possible, an alternative mechanism is to restore the device and CPU states that are necessary for VM booting, and after that restore the saved memory data from checkpoint file after VM starts. In this manner, suspended VM can start very quickly. Figure 1 (b) illustrates time consumed Xen for VM resumption when only necessary device/CPU states have been restored. We observed that, with 1GB RAM configuration it takes 1.3 seconds to resume a VM.

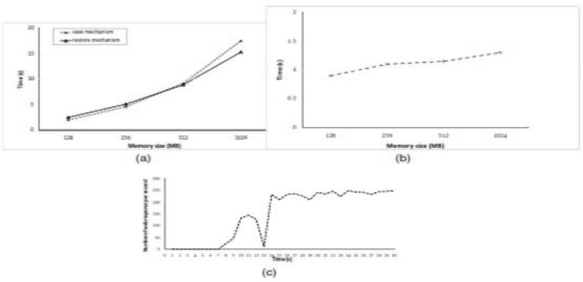

Fig. 1. Comparison of mechanisms for VM resumption.

Since while restoring the memory data the VM state is still running, so its performance would not be get influenced by the size of VM memory; However, while using this approach, performance degrades immediately due to faults of cascaded paging, because no any page available and loaded for use. Figure 1 (c) illustrates responses achieved per second for Apache server running in the Xen, after the VM has been restored using this mechanism. We observed that, to resume normal activity the VM must needs to wait for 14 seconds. Therefore, in order to further reduce the downtime, a check-pointed VM should be resumed rapidly, while degrading performance congestion after the start of VM.

6. EXPERIMENTAL SETUP:

We have designed an experimental setup which includes two hosts. One host is primary or master and second one is used as backup. The two hosts are Intel core 2 Duo processor 2.6 GHz and 4 GB RAM. The two hosts are connected through a 2 Mbps network connection. The network connection is used for migration of the Primary host to the second host.

Live migration can be done by so many ways i.e. by using XEN Hypervisor, KVM Hypervisor, and Qemu Hypervisor etc. To perform live data migration, we prepared a XEN cloud based environment. In the XEN cloud based environment we have created a host VM of size 5 GB, with 1 GB RAM, running Ubuntu Server 12.04 OS. The host VM can execute all the services and applications just as our desktop system. The host VMs runs PV (Para-Virtualized) guests. The PV guests are known as Guest VMs. Three guest virtual machines (VMs), based on Ubuntu sever 12.04 OS, are created in the Host VM of the system. The VMs are named as Testvm1, Testvm2 and Testvm3. The VMs are of size 1GB, 2 GB and 3 GB respectively. We assigned RAM as 256 MB for Testvm1; 512 MB RAM for Testvm2 and RAM 1024 MB for Testvm3. The file system due to which an image file consist of 3 GB could be shared by two machines by using NFS. The (Table 1) shows the detailed specifications of the guest VMs.\

Parameter / VM	Guest vm1	Guest vm2	Guest vm3
ID:	2	3	4
Name:	Testvm1	Testvm2	Testvm3
Hypervisor:	Xen	Xen	Xen
OS Type:	Hvm	Hvm	Hvm
State:	Running	Running	Running
CPU:	1	1	1
CPU Time:	11.2 s	12.5 s	14 s
Virtual Memory	524288 KiB	524288 KiB	524288 KiB
Allocated Memory	1048576 KiB	2097152 KiB	3145728 KiB
Disk Space:	1 GB	2 GB	3 GB
UUID:	192.168.0.60	192.168.0.30	192.168.0.20

6.1 PERFORMED EXPERIMENTS AND RESULTS ANALYSIS:

VM Migration with Load Balancing Apart from previous work, here the intention is to provide load balancing with negligible downtime by virtual machine live migration. In our earliest investigations, we observed that a common VM live migration technique doesn't work in all situations, for instance, when dealing with memory-intensive application. Thus, we purposed a workload- adaptive migration mechanism.

Here DCbalance, OSVD and DLB are our workload-adaptive live migration mechanisms. OSVD depends on VM live migration and incorporates performance estimate technique. DLB implements a dynamic load balancing algorithm (Minjia *et. al.*, 2010) which is installed on Xen original live migration mechanism. We fragment the performance evaluation of load balancing and downtime of migration mechanism in the succeeding.

As we have discussed earlier, Apache is a static web application. While having different loads, the hosts will be down till the state and memory migration takes place.

Fig. 1 demonstrates the downtime results of workloads for the Apache benchmark with different sizes of VMs. We note that all mechanisms sustain slightly low downtime i.e. within 1 second. Also we observed that the OSVD system experiences additional time due to the prediction overhead. The total migration time is from when

the migration is activated to when the resumed VM is entirely operational by users. the findings of total migration time are mostly similar to the previous ones. We observe that our proposed mechanism reduce the total migration time in DLB and OSVD by up to 33% and 38%.

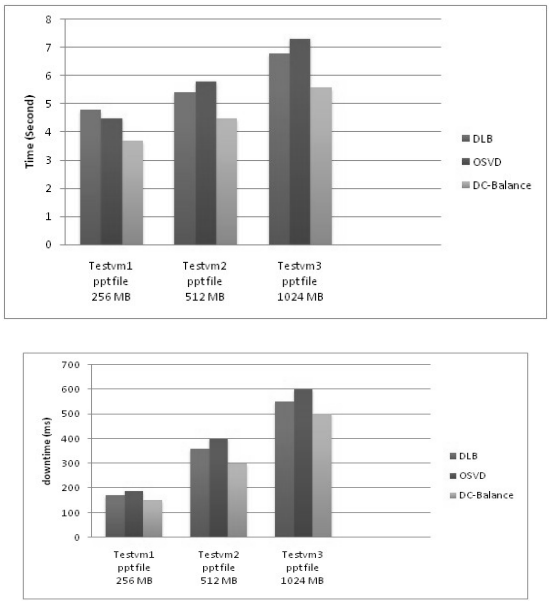

Fig. 2. Downtime comparison under Apache

The VMs are of different sizes. Greater the virtual hard disk, longer the time will taken by Xen Hypervisor to migrate. When migration is in process, VMs are idle. This is known as the VM downtime. VM downtime is the time from when the VM pauses to save for the checkpoint to when the VM restarts. Here downtime is calculated in two situations:

– When the VM is idle i.e. does not have any workload.

– When the VM runs the Apache web server workload (Aaron *et. al*, 2011).

Since downtime is also part of total migration time and our proposed mechanism incurs smaller downtime.

From the Experimental results, we observed that for live migration, high speed networks are required, so that the live migration can be done efficiently. Live migration through Xen Hypervisor depends on the size of virtual RAM and virtual hard disk. Larger the virtual disk is, longer the time Xen Hypervisor takes for live migration. We have used Xen Hypervisor because it transfers the whole configured memory even though the least memory is used. Xen Hypervisor also maintains the state of the VM during migration. Live migration consists of three parameters: Real, User and Sys. The parameters maintain the live migration, state of VM and memory migration.

In other case, migration is initiated when the VMs are running Apache and have some workloads of different applications. DCbalance is the load which uses history record to support scheduled VM migration. It is very efficient load balancing strategy. Our estimation illustrates that DCbalance speed up the load balancing decision process. DCbalance is capable to attain minimal downtime for different kind of applications with different sizes of memory.

In figure 2, here results illustrates that the suggested migration mechanism shrinks the downtime by up to 73% and reduces the total migration time by up to 38% as compared to other techniques.

6.2 OBSERVATIONS

After setting up the environment, the hosts VMs are forced to run some applications like web service, simple file editing service etc. The applications that run in the system are specified below. Besides applications that run on the guest VMs, some other benchmarks are also used.

Static web application: Apache 2.0 is used as static web application. Apache is the world's most widely used web server. It provides the network connection between the guest VMs. The hosts have 10 simultaneous connections, and repetitively downloaded a 256KB file from the web server.

– Dynamic web application: SPECweb99 is a Dynamic web application which is used for calculating workload on the web servers and on the host systems. This benchmark includes a web server which serves static and dynamic requests. The two host VMs in our system create a workload of 10 immediate connections to the web server (Christopher Clark, et. al., 2005).

NPB-EP: This benchmark is a standard technique designed for estimating parallel programs. NPB consists of 5 kernels, from which the Kernel EP program is selected (Fred Douglis, et. al., 1991). Therefore, this example comprises great workloads on the guest VM.

SPECsys: It is a Memory-intensive Application, used to evaluate the load on the RAM. This benchmark is used to measure output of NFS file server. It also calculates response time when the

load is increased due to NFS operations (i.e. lookup, read, write, etc.).

7. CONCLUSIONS

The summary of this paper aims to describe the primary phenomenon of the implemention live migration from the sink VM to the backup destination VM through checkpointing of VMs. Multiple mechanisms for the VM checkpointing at the decades of the abstraction exits earlier and discussed briefly along with the some previous preceding and finding in order to measure the checkpointing performance. Furthermore the checkpointing for the multiple VMs such as VC is also discussed in related work of this paper. Also we discussed the few ways to implement the live migration in enhanced and efficient manner. To get the better optimization and performance measure during the live migration of VMs, Load balancing is the important parameter and the main motivation, for its avoidance and control some methods are also discussed here. As in our proposed strategy, live migration provides always availability of the resources by means of shared and clusters system of VMs, to achieve the minimal load.

We will consider some directions for the future work on strategies for load balancing exploits the live VM migration exits. Adding some knowledge of net- work design and topology in the DCbalance algorithm and joining the current implementation with sniffer (network monitor), it will enable DCbalance with mapping based on network-awareness to overcome the network traffic while performing the migrations. Another problem that is worth to consider in migrations that how to tackle faulty control nodes in virtualized internetworking environment i.e., recovery form fault and some techniques for the fault tolerance.

REFERENCES:

Aaron J., S. Das, D. Agrawal, and A. El Abbadi. (2011) Zephyr: live migration in shared nothing databases for elastic cloud platforms. In Proceedings of the 2011 ACM SIGMOD International Conference on Management of data, SIGMOD 11, 301312, New York, NY, USA, ACM.

Aameek S., M. Korupolu and D. Mohapatra. (2008) Server-storage virtualization: integration and load balancing in data centers. In Proceedings of the ACM/IEEE conference on Supercomputing, SC 08, 53:153:12, Piscataway, NJ, USA,. IEEE.

Anna H., (1989) Load balancing in distributed systems: a summary. SIGMETRIC Perfor Eval. Rev., 16 (2-4):1719.

Bobro, A. K and K. Beaty. (2007) Dynamic placement of virtual machines for managing sla violations. In Integrated Network Management, 2007. IM 07.10th IFIP/IEEE International Symposium on, 119128,.

Christopher C., K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. War_eld. (2005) Live migration of virtual machines. In Proceedings of the 2nd conference on Symposium on Networked Systems Design and Implementation - Volume 2, NSDI05, 273286, Berkeley,CA, USA, USENIX Association.

Carlo C., E. Jones, (2011) Raluca Ada Popa, Nirmesh Malviya, Eugene Wu,Samuel Madden, Hari Balakrishnan, and Nickolai Zeldovich. Relational Cloud: A Database Service for the Cloud.In 5th Biennial Conference on Innovative Data Systems Research, Asilomar, CA.

Changbin L., B. T. Loo, and Y. M.. Declarative (2011) automated cloud resource orchestration. 26:126:8.

DSN 09. IEEE/IFIP International Conference on, 534543, 2009.

Freedman. D. (1991) Experience building a process migration subsystem for unix. In USENIX Winter 91, 349356.

Feldmann A., R. Bradford, E. Kotsovinos and H. Schioeberg. (2007) Live wide-area migration of virtual machines including local persistent state. In VEE07: Proceedings of the third International Conference on Virtual Execution Environments, 169116, San Diego,CA, USA,. ACM Press.

Ellard T R., (1995) The freeze free algorithm for process migration. Technical report, Champaign,IL,USA.

Fred D. and John (1991). Ousterhout. Transparent process migration: design al- ternatives and the sprite implementation. Softw. Pract. Exper. 21(8):757785.

Jin H., X. Zhu, P. Padala, and Z. Wang. (2009) Memory overbooking and dynamic control of xen virtual machines in consolidated environments. In Proceedings of the 11th IFIP/IEEE INM, IM09, pages 630637, Piscataway, NJ, USA, IEEE Press

Jing X., M. Zhao, J. Fortes, R. Carpenter, and M. Yousif. (2007) Autonomic resource anagement in virtualized data centers using fuzzy logic-based approaches. Cluster Computing, 11(3):213227,

Jing X., M. Zhao, J. Fortes, R. Carpenter and M. Yousif. (2008). Autonomic resource management in virtualized data centers using fuzzy logic-based approaches, Cluster Computing, 11 (3): 213-227,

Minjia Z., H. Jin, X. Shi, and S. Wu. (2010.) Virtcft: A transparent vm-level faulttolerant system for virtual clusters. In Parallel and Distributed Systems (ICPADS), 2010 IEEE 16th International Conference on, pages 147-154.

Michael Litzkow and M. Solomon. (1999) Mobility. chapter Supporting checkpointing and process migration outside the UNIX kernel, pages 154162. ACM Press/Addison-Wesley Publishing Co., New York, NY, USA,.

Maohua L. and T. Cker (2007) Chiueh. Fast memory state synchronization for virtualization-based fault tolerance. In Dependable Systems Networks.

Sagar D., M M. Hayat, J. E. Pezoa, Cundong Yang, and D. A. Bader. (2010.) Dynamic load balancing in distributed systems in the presence of delays: A regenerationtheory approach. IEEE Trans. Parallel Distrib. Syst., 18(4): 485-497 .

Timothy W., P. Shenoy, A. Venkataramani, and M. Yousif (2007) Black-box and gray-box strategies for virtual machine migration. In Proceedings of the 4th USENIX conference on Networked systems design and implementation, NSDI07, 1717, Berkeley, CA, USA. USENIX Association.

Wei H., Q. G. J. Liu, and D. K. Panda. (2007) High performance virtual machine migration with RDMA over modern interconnects. In CLUSTER 07: Proceedings of the, IEEE International Conference on Cluster Computing, 1120, Washington, DC, USA,. IEEE Computer Society,