Informed Live Migration Strategies of Virtual Machines for Cluster Load Balancing

Xing Li, Qinming He, Jianhai Chen, Kejiang Ye, Ting Yin

▶ **To cite this version:**

Xing Li, Qinming He, Jianhai Chen, Kejiang Ye, Ting Yin. Informed Live Migration Strategies of Virtual Machines for Cluster Load Balancing. 8th Network and Parallel Computing (NPC), Oct 2011, Changsha„ China. pp.111-122, 10.1007/978-3-642-24403-2_9 . hal-01593006

HAL Id: hal-01593006

https://inria.hal.science/hal-01593006

Submitted on 25 Sep 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Distributed under a Creative Commons Attribution 4.0 International License

Informed Live Migration Strategies of Virtual Machines for Cluster Load Balancing*

Xing Li, Qinming He, Jianhai Chen, Kejiang Ye, Ting Yin

College of Computer Science, Zhejiang University,
Zheda Rd. 38, Hangzhou 310027, China
{lix,hqm,chenjh919,yekejiang,yintingeye}@zju.edu.cn

Abstract. Virtualization technology brings great conveniences to cluster and data center management. By using this technique, we can reconstruct a new computing environment quickly and easily. Compared to the traditional cluster environment, load balancing in a virtualized environment needs to address several new problems. This paper focuses on live migration strategies for load balancing in the virtualized cluster. We first divide the whole balancing process into three sub-problems, namely, the selection of the VM being migrated to, the choice of destination host and the determination of the migration execution sequence. Then we perform a series of experiments to investigate the particular features of the live migration of virtual machines in the balancing scenario. Based on our experiment results, we propose an informed live migration strategy which includes affinity-aware decision making and workload-aware migration to improve the efficiency of configuration of the virtualized cluster.

Keywords: Virtualization, Live Migration, Workload Aware

1 Introduction

Virtualization has recently emerged as an essential technology to the modern computing cluster and data center mainly due to its capabilities of virtual machine isolation, consolidation and migration [1]. The enabled hardware independence and rapid deployment using QCOW technology [2], permit one to construct a computing cluster within minutes.

Live migration is an extremely powerful tool for cluster management. It facilitates typical functions such as: load balancing, online maintenance, power saving, and high availability. For example, if a physical machine needs to be removed from a cluster for maintenance, we can simply migrate the virtual machine (VM) from the original host to another available host. Similarly, VMs can be reallocated across the physical servers in the cluster to relieve the workload

* This work is funded by the National 973 Basic Research Program of China under grant NO.2007CB310900 and National Natural Science Foundation of China under grant NO. 60970125.

on congested servers. In these situations the combination of virtualization and migration significantly improves the manageability of clusters.

At present, most of the virtualization platforms use memory Pre-Copy[3] as the default algorithm to implement live migration due its ability to decrease the VM downtime to the order of milliseconds which is imperceptible to users. However, the Pre-Copy algorithm will cause workload performance degradation during live migration. It will also produce an inevitable overhead.

In fact, not only the memory transferring mechanism affects the cost of live migration, the choice of the actual migration strategy also impacts greatly on the performance of a migrating VM. Especially, in the scenario of performing load balancing for virtualized cluster, there are often multiple VMs that must be migrated to multiple destination hosts . In such case, the migration strategy of these multiple VMs will be even more important.

In this paper, we study the behaviors of various live migration strategies. We found that some particular features will impact the performance of load balancing. These features include the affinity relationship between migrating VMs and the actual sequence of migration. We then propose an informed live migration strategy that incorporates affinity-aware decision and workload-aware migration. The strategy makes migration decisions according to the VMs' characteristics and the migration sequence. It will improve the reconstruction efficiency of virtualized cluster.

The rest of this paper is organized as follows. Section 2 introduces the background of load balancing in the virtualized cluster and the motivation to study the effective balancing approach. In Section 3, we present our experiments and results, followed by the proposed strategies. Section 4 describes related work and Section 5 presents our conclusions and future work.

2 BACKGROUND&MOTIVATION

In this section, we briefly introduce the virtual machine technology and the motivation to investigate the live migration strategies.

Live-migration technology. At present, many commercial corporations have their own live migration technology available within their software products. For example, Microsoft has Hyper-V [4] live migration, Cirtix integrates XenMotion [6] with their xenserver, and VMware uses VMotion [5]. Among a number of live migration algorithms, Memory Pre-Copy is the most prevailing one. The main advantage of the Pre-Copy algorithm is that it produces the minimum downtime. The actual time taken could be as low as 60ms[3], so users will not detect the event that the services have been interrupted and the virtual machine is migrating from one host to another. This algorithm copies and sends memory pages from the current host to the destination host iteratively, while ensures that services on the migrating VM are still available. Until the writable working set becomes small enough, will the virtual machine monitor (VMM) suspend the VM and then send the last remaining dirty pages and the

CPU state. The network bandwidth being occupied may affect other VMs when network connection is involved in migration.

The performance metrics of live migration. To measure the performance and efficiency of live migration, four typical metrics are used: migration time, downtime, data transferred and workload performance degradation [7]. The migration time measures the time elapsed from the start of a migration process to the end of it. The downtime measures the time period when the migrating VM is stopped to transfer the remaining data. The VM is not responsible during the downtime . The data transferred reflects the total size of virtual machine memory data that has been sent during the migration. The workload performance degradation reflects the performance lose caused by live migration compared to the no-migration case.

Three problems of reconstructing the virtualized cluster. In order to balance the load of a virtualized cluster, three important problems must be addressed in making a wise choice on a good live migration strategy. Firstly, how to select the candidate virtual machine from the overloaded host to migrate out? The different VM selections will result in different outcomes. Criterions of minimum migration costs and earliest completion of reallocation procedure are both very necessary. Secondly, how to choose the destination host for each candidate migrating virtual machines? In other words, how to find the mapping relationship between migrating VMs and those idle hosts. Lastly, if there are many VMs to be migrated at one time, what would be the migration execution sequence of these candidate VMs, that will produce less marginal impact the other VMs. It can be expected that the migration sequence plays a significant role in the overall performance.

3 EXPERIMENTAL EVALUATION&ANALYSIS

In this section, we describe our detailed migration experiments for different scenarios, and then analyze the measurement results. These results will reveal key characteristics of many migration algorithms when used for load balancing purpose.

3.1 The Migrating VMs Selection Problem

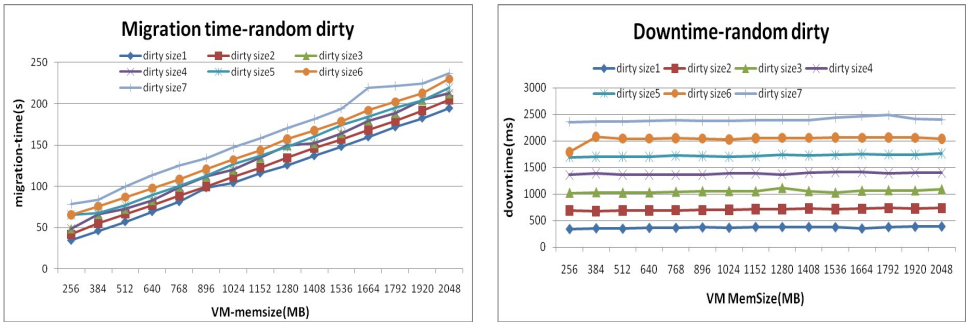
In order to release the overloaded host fast, we need to choose the correct candidate VMs for migration to assure the minimum migration costs and make the reallocation procedure finished as quickly as possible. Table 1 displays the migration time, down time and overheads of selected typical workloads.

As we can see, the migration of different workloads will use different lengths of time and cause different overhead. Workloads such as OLTP and SPEC-jbb, which have many memory accesses, need long periods of time to migrate and generate more overhead. The network-intensive workload like webbench will prolong the migration procedure, and cause extra overheads. The pure CPU-intensive workload like SPEC-CPU will have the least impact in all three metrics.

Table 1. The typical workload behavior in migration

workloads	Migration-time(s)	Downtime(ms)	Overhead(%)
IOZONE	214.51	1028	7.93%
WEBBENCH	47.67	113	29.03%
OLTP	69.32	998	33.26%
SPEC-CPU	26.69	88	3.31%
SPEC-jbb	31.25	232	8.29%

In the next experiment, we run a scalable memory dirty workload in the testing VM. We change the VM memory size during the process. The dirty workload was implemented by updating an array continuously and randomly. This was written using the C programming language so that the memory would have a consistently high dirtying rate and would be hard to migrate.



(a) time (b) downtime

Fig. 1. Factors of Migration time&downtime

The Figure 3 shows that the migration time of a virtual machine is influenced by both the memory size and the memory dirty size. This is because that Xen uses a pre-copy algorithm to transfer the memory data to the destination host iteratively in its live migration implementation.

Figure 4 illustrates that the migration downtime is proportional to the memory dirty size and is unrelated to the memory size. When Xen detects that some parts of the memory are getting dirty frequently, it considers it not worthy to send it repeatedly. In this case, Xen will stop this virtual machine and turn to the Stop-and-Copy phase. Xen then transfers the remainder of the dirty data during the Stop-and-Copy phase. However, this approach leads to the case that the downtime is proportional to size of the dirty memory of the migrating VM.

When there is a need to select several virtual machines to migrate out, both the procedure completion time and migration overhead have to be taken into account. As the above experiments illustrated, we can choose some virtual machines which are running CPU workloads to migrate, and consider additionally

criteria such as the small memory size and small dirty size. The dirty logging switch tool we have mentioned in previous section are especially useful in monitoring the dirty rate and dirty size in each domains to support the migration strategies.

3.2 The Destination Host Choose Problem

Deciding which host be the suitable destination will greatly impact the migration perfformance. We have designed a group of experiments, the first one runs the SPEC-CPU workload in two virtual machines which are placed on different physical hosts. Then one of the virtual machines will be migrated from its original host to the other, making the two virtual machines running together. To get clear experiment results, one additional condition imposed was that the two virtual machines are pined to a single physical CPU, and share the same L2 cache.

Table 3 shows that, there are visible affinity relationship between some workloads. Theworkload pairs like libquantum and gamess can work well together, but not the other workload pairs like two bwaves not. This is because that the latter two workload would mutual restrain each other and need very long time to complete.

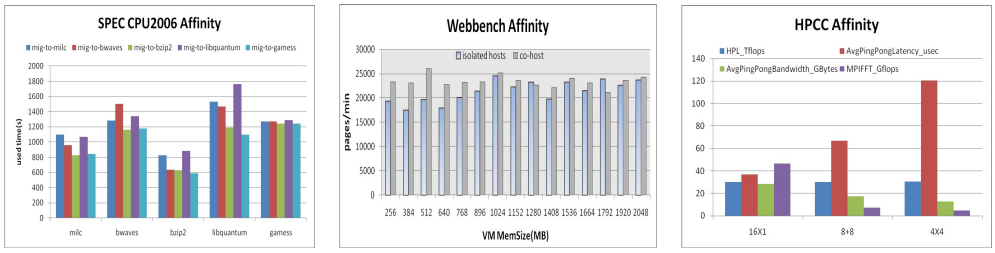
Table 2. SPEC-CPU workload affinity, the values in the table display the running time of each workloads

Destination	**Workloads be migrating**				
	milc	bwaves	bzip2	libquantum	Games
mig-to-milc	1100	1280	824	1530	1270
mig-to-bwaves	961	1500	633	1470	1270
mig-to-bzip2	824	1160	628	1190	1240
mig-to-libquantum	1070	1340	886	1760	1290
mig-to-gamess	843	1180	588	1100	1240
max-difference	25.1%	22.7%	33.6%	37.5%	3.9%

Table 3 and Figure 3(a) show that choosing different migration destination hosts will dramatically impact the virtual machine performance. The difference may even reach 37.5% in some cases. The reasons for these differences are that these CPU-bounded workload have different cache accessing types: some workloads like bwaves and libquantum will pollute cache block, while some other workloads like bzip2 and gmaess will not.

Figure 3(b) shows that when placing the two virtual machines, which have a large number of communications, together on the same host, they will get better performance.

We also designed experiments to deploy a virtual cluster of e 16 virtual machines in different number of physical hosts, from one physical host, to 2 hosts each has 8 VMs, and to 4 hosts each has 4 VMs. The comparisons of the

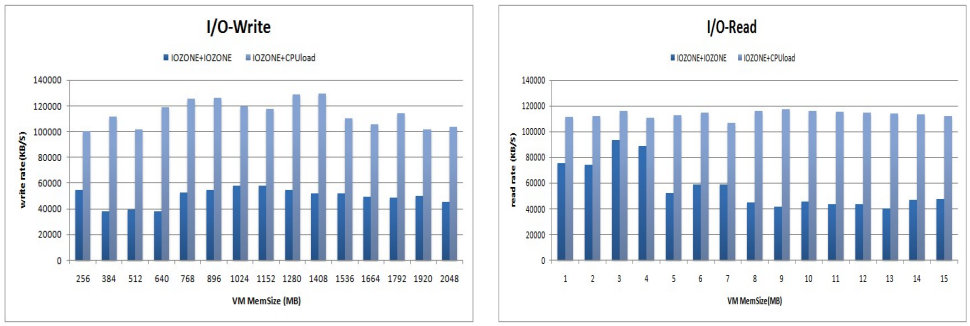


(a) SPEC CPU workload (b) Webbench affinity (c) HPCC affinity

Fig. 2. (a)The affinity of SPEC CPU workload.(b)Comparison of webbench on 2 isolated-hosts VMs with co-host.(c)Comparison of HPCC VMs isolated and co-host.

three cases are shown in Figure 3(c). In general, the three configurations show similar HPL performance. But the communication performance decreases a lot when deploying these virtual machines to more different hosts. When there are 4 physical hosts, the MPIFFT shows the worst performance. It means that the communication performance of virtual machines is more effective when deployed on the same physical host.

On the other hand, the Xen memory sharing mechanism and grant table could be used to communicate between domains. By using this mechanism, the domain could lease one of its memory block to another domain temporarily. This method will be more effective than communication through a physical network. However theis memory sharing mechanism is only implemented in the virtual machine monitor Xen.



(a) (b)

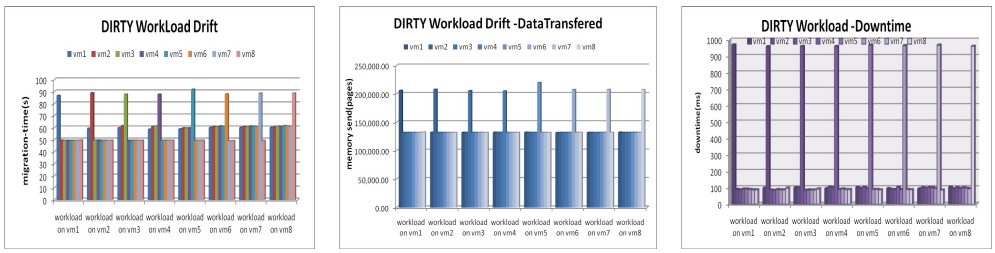
Fig. 3. Two I/O type VMs working together

Figure 4 illustrates that, when two disk-I/O-bounded virtual machines working together, there will be a significant degradation in performance. In contact, a disk-I/O-bounded virtual machine and a CPU-intensive one could work well to-

gether. It is easy to see that consolidating multiple disk-I/O-bounded workloads has to consider the nonsharable usage of the physical disk I/O operations which restricted the capacity. In summary, we have the following observations when determining a suitable server to host a migrating VM: (1) it would be better to choose the kinds of virtual machines as neighbors, the kinds that have well affinity with the migrating one; (2) it would be better to make the virtual machines that communicate with each other to work on the same host; (3) it would be better to avoid migrating two workloads that have the same nonsharable type of resources together.

3.3 The Migration Sequence

When migration involves multiple VMs, the migration sequence becomes a critical issue because it will dramatically affect the whole reconfiguration procedure of the cluster. Our next experiments will show the influence. In this set of experiments, we migrate 8 virtual machines one after the other. One of the 8 virtual machines has a workload running on it, others are idle. By altering the location of the workload VM in the migration sequence, we can observe clearly that the different migration order of the workload virtual machine will affect the other virtual machine that are co-hosted with it. Firstly, we place a memory-dirty workload on the first virtual machine, named VM1, in the migration sequence, and migrate these 8 virtual machines one by one from VM1 to VM8. Then, we place the dirty workload on the second virtual machine VM2 and migrate all the virtual machines in succession. Repeatedly, we place the memory-dirty workload on the VM3, VM4 until VM8.



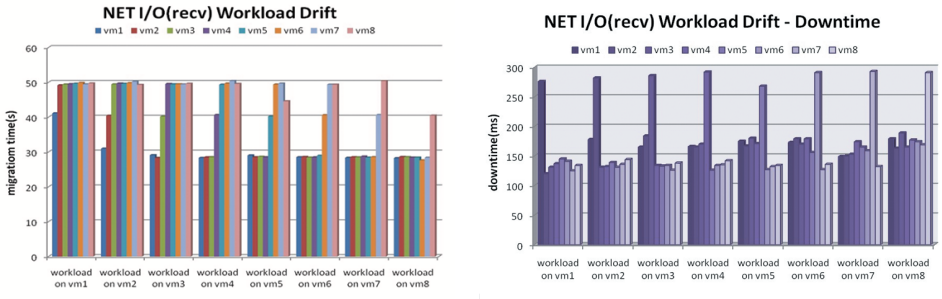
(a) The impact of memory dirty workload (b) Data transfer (c) migration downtime

Fig. 4. (a)Memory dirty workload VMs will affect these VMs migrated before it.(b)Data transferred of each migrated VMs.(c)Downtime of each migrated VMs

Figure 5(a) illustrates memory dirty workload virtual machine will significantly affect the migration time of these co-hosting virtual machines migrate before it, and these virtual machines which migrate behind the dirty workload VM don't suffer impact. As Figure 5(b)(c) show, the memory dirty workload

will not affect the performance of the migration downtime and migration data transferred of the other VMs. It is clear that the memory dirty workload causes large page fault and slows down the other virtual machines' memory copy procedure. The conclusion is simple, We must ensure that the memory dirty workload always have the higest priority to migrate. However, not all the cases own this phenomenon as we can see a complete different situation in the next experiment.

This time we changed the memory dirty program into a NET I/O workload.Thus the workload virtual machine continuously receives the data sent from remote server. In the remainder, we just repeat the process as described in previous experiment.



(a) The impact of migration time (b) The impact of migration downtime

Fig. 5. The impact of NET I/O(receive) VMs on (a)migration time or (b)migration downtime of the other VMs migrated before

Figure 6(a) shows NET I/O(receive) virtual machine will greatly impact the migration time of the other virtual machines migrated after it, this is because when the net I/O receiver workload migrates to the destination host, it reduces the destination host's ability to receive the memory data being transferred, and also slows down the migration time of other followed virtual machines. However, this is not the same as previous case. It seems that we need to migrate these virtual machine with NET I/O(receive) workload as late as possible. Figure 6(b) shows that NET I/O will prolong the migration downtime, because when it comes to stop-copy step, it will take a longer time to send the remaining memory data.

In this case, we need a compromise approach to balance the migration time and downtime. For these virtual machines tend to have short migration time. We choose to migrate it before the NET I/O (recv) workload virtual machine. On the other hand, for these virtual machine tend to have short downtime, we choose to migrate it after the workload virtual machine migration.

Then, we change the workload into web server application and repeat the experiment.

(a) migration time (b) migration downtime

Fig. 6. WebServer VMs will impact the migration (a)time,(b)downtime of some other VMs migrated before it

As Figure 7 shows, the web server workload VM will dramatically impact both of the migration time and migration downtime of its co-hosting virtual machines. Obviously, toese virtual machines migrated before the workload VM will have a long migration time and downtime. This is because the web server load sends data through the network continuous,which occupies large bandwidth, and will slow down both the migration time and downtime. It is easy to reach the conclusion: for these web server workloads, we need to migrate them out as soon as possible.

Our experiments also show that the disk I/O load and CPU workload will not affect the neighboring virtual machine migration performance.

At last, we can assemble these experiments described above together and take an overview. With the workload virtual machine migrating backwards in the migration order, the average migration time of whole migrating VMs sequence will have different features. For the sequence having memory dirty workload, the workload virtual machine gets migrated later the average migration time gets larger?"For the sequence which having NET I/O (recieve) workload virtual machine, the average migration time will be smaller but will have a longer average downtime; For these sequence having web server workload, the later migration will cause both larger migration time and down time. For these having disk I/O or CPU workload sequences, the migration order causes no effect.

3.4 Informed Live Migration Strategies

By analyzing the above experimental results, we have found interesting features of the live migration in the scene of virtual cluster load balancing: (1) Different kinds of workloads result in a diversity in the migration time and different performance overheads; (2) Besides the total memory size, the memory dirty rate and dirty area size will also affect the migration time and downtime; (3) There are certain affinity relationship between some workloads, e.g., some kinds

of workload consolidation will cause large performance degradation, while others will not; (4) Migrating two virtual machines which communicate through network on the same physical host will obtain performance improvement; (5) The migration sequence of a group of VMs with different kinds of workloads will greatly affect the whole reconfiguration efficiency.

Based on these specific features, we propose affinity-aware decision making and workload-aware migration to improve the efficiency of live migration.

The strategy contains a set of rules: (1) choosing proper virtual machines to achieve shorter migration procedure time and less migration overheads; (2) taking the virtual machine affinity into consideration to decide suitable destination host; (3) Understanding the virtual machine workload features and determining an optimal migration sequence. In our implementation of the strategy, we tend to choose the VMs that have less total memory and smaller memory dirty size as candidate migrating VMs./ The types of the workloads are also considered (see Table 4.2), the CPU workload and Disk I/O are currently preferred in our testing. When choosing the migration destination host, it's important to avoid consolidating the same type of virtual machine with nonsharable ; but the VMs having network communications should be set to work together. When determining the migration sequence of a group of VMs, the VMs which have dirty memory or web server workload should be considered first. For the VM which have Network I/Os (receiving) and web application running on it, for example, online video, webgame, etc, we need to migrate it at the end of the migration process. This is because it will impact the destination server's bandwidth, slow down the pre-copy operation and increase the downtime of the subsequent migration. So, the VMs which have NET I/O(receive) workload need to be migrated at last. For the VMs which have CPU or disk I/O workload, we just need to ensure that the smaller VM is migrated early using the principle of SJF (Shortest Job First). This will lead to the least average completion time of each migrating VM.

Based on the aforementioned discoveries and mechanisms, we are in a position to implement the features of affinity-aware decision making and workload-aware migration into the balancing system, which establishes our informed live migration strategies for live migration.

4 RELATED WORK

Many efforts have been made to improve the efficiency of live migration. Some research has been done into improving the efficiency of the live migration mechanism. H.Jin et al. [14] presents an implementation of a novel memory-compression based VM migration approach to improve the migration efficiency ; Liu et al.[16] described a design and implementation of a novel approach that Adopts checkpointing-recovery and trace-replay technology to Provide fast, transparent VM migration. Luo et al. [11] describe a whole-system live migration scheme, which transfers the whole system run-time state, including CPU state, memory data, and local disk storage, of the virtual machine (VM). There are also many

efforts have been made to the dynamic placing and consolidation of the virtualized resources. Hermenier et al. [15] provide substantial improvement over static server consolidation in reducing the amount of required capacity and the rate of service level agreement violations. Choi et al. [13] describe a learning framework that autonomously finds and adjusts thresholds at runtime. Verma et al. [8] presented the pMapper architecture and placement algorithms to minimize the power used to a fixed performance requirement. Hermenier et al. [15] propose the Entropy resource manager for homogeneous clusters, which performs dynamic consolidation and takes migration overhead into account. Differing from the above work, This paper studied the live migration strategies in the scenario of load balancing in a virtualized cluster.

5 CONCLUSIONS AND FUTURE WORK

In this paper, we study the live migration features in the load balancing scenario of a virtualized cluster, our experiments showed several interesting observations that help to establish rules for live migration to achieve the goal of load balancing. Our proposed informed live migration strategy includes affinity-aware decision making and workload-aware migration. It improves the efficiency of reconfiguration a virtualized cluster.

Our future work will include more comprehensive performance metrics and use workload automatic monitoring to support the informed live migration strategy. Additionally, we will use mathematical modeling methods to evaluate and integrate the multiplicity of factors that influence live migration.

References

1. Barham, P., Dragovic, B., Fraser, K., Hand, S., Harris, T., Ho, A., Neugebauer, R.,Pratt, I., War?eld, A.: Xen and the art of virtualization. *Proceedings of the 19th ACM Symposium on Operating Systems Principles*, pp. 164-177 (2003)
2. QCOW,`http://en.wikipedia.org/wiki/Qcow`
3. Clark, C., Fraser, K., Hand, S., Hansen, J.G., Jul, E., Limpach, C., Pratt, I.,Wareld, A.: Live migration of virtual machines. *Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation*,pp. 273-286 (2005)
4. Hyper-V,`http://www.microsoft.com/hyper-v-server/en/us/default.aspx`
5. VMware VMotion,`http://www.vmware.com/products/vmotion/overview.html`
6. XenServer XenMotion,`http://www.citrix.com`
7. D. Huang, D. Ye, Q. He, J. Chen, K. Ye, Virt-LM: A Benchmark for Live Migration of Virtual Machine. *Proceeding of the second joint WOSP/SIPEW international conference on Performance engineering*,pp.307-316.(2011)
8. A. Verma, P. Ahuja, and A. Neogi, pMapper : Power and Migration Cost Aware Application Placement in Virtualized Systems??*Ifip International Federation For Information Processing*, pp. 243-264 (2008)
9. K. Ye, X. Jiang, Q. He, and X. Li, Evaluate the performance and scalability of image deployment in virtual data center. *Network and Parallel Computing*, pp. 390-401 (2010)

10. W. Voorsluys, J. Broberg, and S. Venugopal.:Cost of virtual machine live migration in clouds: A performance evaluation. *IEEE International Conference of Cloud Computing*, pp. 254-265 (2009)
11. Y. Luo, B. Zhang, X. Wang, Z. Wang, Y. Sun, and H. Chen, Live and incremental whole-system migration of virtual machines using block-bitmap. in *Cluster Computing, 2008 IEEE International Conference on*, pp. 99–106 (2008)
12. N. Bobroff, A. Kochut, and K. Beaty, Dynamic placement of virtual machines for managing sla violations. in *Integrated Network Management. IM'07. 10th IFIP/IEEE International Symposium on*, 2007, pp. 119–128(2007)
13. H. Choi, H. Kwak, A. Sohn, and K. Chung, Autonomous learning for efficient resource utilization of dynamic VM migration. in *Proceedings of the 22nd annual international conference on Supercomputing*, pp. 185–194(2008)
14. H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan,Live virtual machine migration with adaptive, memory compression. in *Cluster Computing and Workshops, 2009. CLUSTER'09. IEEE International Conference on*, pp. 1-10 (2009)
15. F. Hermenier, X. Lorca, J. Menaud, G. Muller, and J. Lawall, Entropy: a consolidation manager for clusters. in *Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on Virtual execution environments*, pp. 41-50(2009)
16. H. Liu, H. Jin, X. Liao, L. Hu, and C. Yu, Live migration of virtual machine based on full system trace and replay. in *Proceedings of the 18th ACM international symposium on High performance distributed computing*, pp. 101-110 (2009)
17. X. Jin, H. Chen, X. Wang, Z. Wang, X. Wen, Y. Luo, and X. Li, A Simple Cache Partitioning Approach in a Virtualized Environment, *Proceedings of IEEE International Symposium on Parallel and Distributed Processing with Applications*, pp. 519-524.(2009)