ISSN No. 0976-5697

Volume 8, No. 5, May – June 2017

International Journal of Advanced Research in Computer Science

RESEARCH PAPER

Available Online at www.ijarcs.info

Performance Optimizing Factor Analysis of Virtual Machine Live Migration in Cloud Data-centers

Anu V.R.,
Research scholar
School of Computer Science
Mahatma Gandhi University, Kottayam, India

Dr.Elizabeth Sherly,
Professor
Indian Institute of Information Technology and Management-Kerala Trivandrum, India

Abstract: Cloud computing holds its importance in almost every field of technology. It makes provisioning, scaling, and maintenance of applications and services with sophisticated deployment and management of resources. Currently, many clouds, such as Amazon's EC2, Google's AppEngine, IBM's Blue- Cloud, and Microsoft's Azure are providing various cloud services.VM migration is a widely used cloud feature for better performance and efficiency. Allocation of jobs in an intelligent manner is essential for the efficient VM migration in any cloud environment. This paper analyses different factors which has direct impact on VM migration and thus on cloud performance. With the extensive simulations in cloudsim environment, we found that memory dirtying rate has a vital role in determining the performance of live migration in cloud data centres. We used different techniques to vary the memory dirtying rate and studied its impact on down time, total migration time and thus scope of performance improvement. Along with the dynamic parameters, static parameters are also contributed to performance improvement of live migration characteristics.

Keywords: cloud computing, virtualization, virtual machine, live migration

I. INTRODUCTION

Currently many cloud venders like Google compute, VMware, HP and IBM are providing various online services based on VM (Virtual Machine) migration. The inevitable demand of scalable resources in cloud applications make virtual machine migration a mandatory procedure between the servers in a data center. The fundamental infrastructural operations in a data center like load balancing, server maintenance, fault tolerance etc. are successfully executed with VM migration.

One of the major challenge faced by current data centres are the rate of power dissipation while running. The main cause of power dissipation in data centres are due to power consumption by cooling mechanisms adopted and secondly due to the power consumed for routine data centre operations. Power dissipation increases drastically because of inefficient utilization of resources which escalates operational cost and carbon emission rate. Rapidly growing energy demand of ICT (Information and communication technology) is a major cause of global warming due to the exponential CO2 emissions [1]. Virtualization is a key solution to utilize the resources in a distributed environment effectively. It helps to run numerous VMs on a single hardware resource while respecting the application's privacy by implementing strict isolation policies (server virtualization) based on the performance requirements of VM-hosted workloads [2].Through virtualization, VM migration performs a lot of benefits like balancing the load in servers and consolidating the server by migrating virtual machines between under loaded or overloaded physical hosts [3], [6]. Thus, it will reduce the number of servers which are active in datacenters.VM migration helps to consolidate VMs from under loaded servers to a destination server and keep all these servers in inactive mode. This will reduce the power consumption of servers and ultimately improve the power efficiency. Also if a server requires a hardware maintenance or software updation, the services running in the physical host can be migrated into another.

Migrating an entire operating system along with its running application is a strong management tool in multiple VM based environment [5], [6]. It helps us to overcome many difficulties with process level migration, which always cause a residual dependency, in which the original host has to remain available and network accessible for completing certain services particularly some system calls. But in the case of VM migration, the original host can be moved from the scenario, once VM migration has completed. OS migration allows separation of hardware or software consideration and consolidating server hardware into a single management domain. Mainly two methods of migration are adopted. In offline migration or cold the currently running services will be terminated first and then VM is transferred to new physical host. But in live migration all services will be alive while migration taking place between the servers. Whether it is cold or live migration, the migrations are not equal in terms of factors affected. When one gives emphasis to reduce the down time, another could seek to minimize the end to end migration time or the bandwidth consumed by the migration.

Technologies used in live migration may different from vendor to vendor. But generally two methods adopted in live migration are pre-copy and post-copy [2], [18], [21]. In post-copy migration method, initially the VM is suspended at source, then copies processor state to the destination. After that operation, VM resumes at destination and begins fetching modified memory pages over the network from the source server. But in pre-copy method, the modified pages are transferred to the destination iteratively till it reaches into a threshold convergence point. At this point VM suspended at source machine, then transferred to the

destination and this period is called Down Time. After that period, the VM restarted at Destination. With bounded iterative push step and final stop & copy step, *pre- copy* migration tries to overcome the problems associated with earlier designs. Such a design minimizes both *TMT* and *down time*.

In this work we study and analyze the live migration strategies in detail. Live migration is done by sending snapshots of OS instances with all its currently running applications between the servers [1], [19]. In all forms of migration the VM snapshots containing I/O configurations, saved segment definitions, spool files, user directory entries, SFS file pool servers, application programs so on. VM snapshots can be generally categorized in terms of the status they transferred, such as CPU state, memory state and storage content. As compared with the other elements, CPU state is the least weight content which can be transferred between the physical servers without taking much time. Second type of transfer is the memory state, which includes the memory state of both the guest OS running on VM and all running processes within the VM. Like normal physical machines, VM may be configured with more memory than in active use. If the hypervisors can effectively identify the unused memory, then transfer of this memory from source t destination can be avoided [2], [16]. This will strongly improve the performance of migration. The third category is storage transfer, as compared with the first two categories storage is too huge, and most datacenters follows a centralized storage area like NAS (Network Attached Storage) or SAN (Storage Attached Network). So instead of transferring VM snapshots between the servers, the shared storage can be accessed equally by the virtual machine through physical machines.The following are the general steps for live migration.

1) Initially, both source and destination hosts are prepared for migration process. A TCP connection has to be established between source and destination. Memory is allotted in the destination host for the coming VM and skeleton of VM is also set up.
2) In this step, the memory state and CPU status of VM from source host will be transferred to destination host. For the migration process, either post-copy or pre-copy strategies are adopted.
3) Here, a mass storage transfer from source host to destination will be taking place. Normally a common storage systems like NAS or SAN is used, so source host or destination host can access it equally. At the end of this stage an up-to-date virtual machine is ready to use at destination host.
4) For a transparent migration process, all network connections that were open before the migration, must also keep open after migration process completes. Since each VM will have Virtual Network Interface Card identified by a MAC address, the VM required to update the switches in the network so that the virtual machine traffic will be forwarded through the corresponding switch port after migration.

Fig.1 shows the general steps in pre-copy algorithm.The target server selection and resource reservation steps are common to pre-copy and post-copy algorithm.

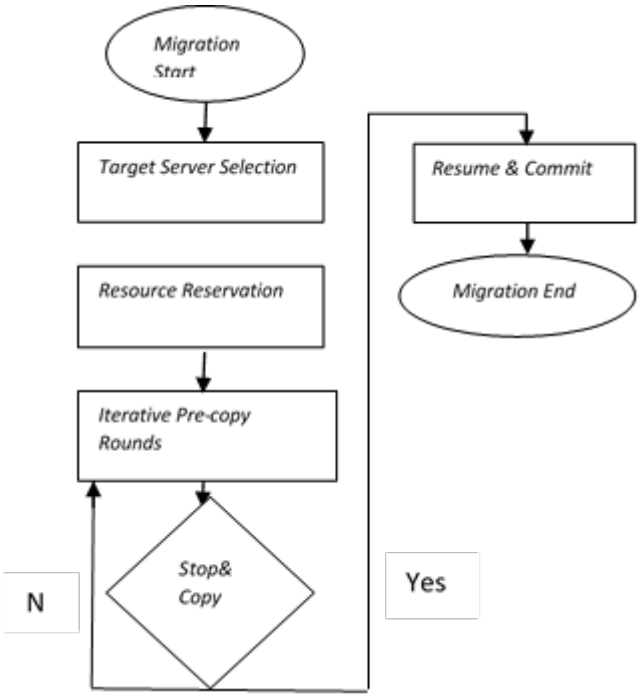

Fig 1. Pre copy algorithm

Post-copy migration strategy works as an incremental model. Initially the minimum status of VM will be transferred from source to destination server [4]. After that if any updation happened, that will be transferred to destination.

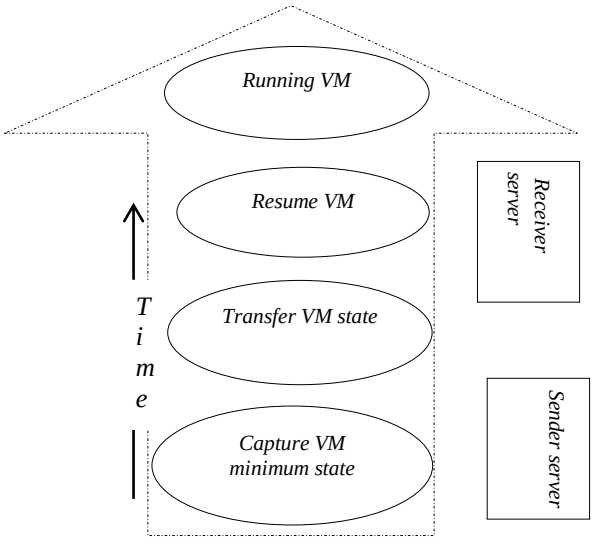

Fig 2. Post-copy algorithm

The rest of this paper is organized as follows. Section 2 presents a concise view of a detailed literature survey and a brief narration to the related research works. Section 3 deals with the motivations behind this work. Section 4 includes factor analysis. Section 5 comes with the conclusion and future work.

II. LITERATURE REVIEW

Even though dynamic resource management is a fundamental asset of cloud computing, it is far from optimization because of being an NP – complete problem. So scope of optimization is always relevant in resource management of data centers. Migration plays a major role in managing these resources effectively in a cloud computing scenario [4]. It is an effective tool to manage resources which are still over- or underutilized, code and data which are not properly distributed. Even the potential benefits for energy efficient computing are not exploited to their best yet – in fact, there is hardly any quantifiable data that allows for measuring energy efficiency aspects to their minute details.

Cloud computing are often considered as high performance computing systems because of their high availability of resources. The effective performance depends highly on the degree of scalability, the utilization of resources and the communication strength. In the past few years, researchers have undergone studies and formulated various algorithms and techniques in order to undergo more optimized and efficient algorithms for virtual machine migration, where VM can be migrated with minimum downtime. In this section, we have presented some of the research works related to live migration.

Some of the optimizations methods proposed by Raja Wasim Ahmad and Abdullah Gani [4] for pre-copy live migration are delta compression, page skip, de-duplication ballooning and data compression. Another method introduced by Senthil Nathan and Umesh Bellur [5] combines check pointing, logging, and roll-forward recovery with CPU scheduling which can reduce migration overheads compared with pre-copy algorithm. VM to be migrated on the source host generates log files continuously, these log files are transferred to the target host in sequence while the target host replays with the received log files. A log file will be generated every time unit. After consumption of the received log files in each round, the time units used for replaying are estimated as value *K*. Then a request is sent for *K* log files to the source host which will be transferred in the next round.

During live migration, the complete contents of VMs' RAM are transferred from source to the destination host. Since the RAM sizes of several gigabytes, live migration often involves transferring large data volumes. Besides that, if any pages are updated at source side, then that has to be resend to destination. Both these processes lead to long total migration time. Even though compression techniques reduce the amount of data transferred, they do not necessarily reduce the number of page re-sends. To manage this issue, P. Sv¨ard, J. Tordsson, B. Hudzia and E. Elmroth.[8] proposed and evaluated a page reordering technique that reduces the amount of transmitted data by sending the memory pages in reverse order of usage frequency to avoid re-transfers.

Jing *et al.* [7] proposes an optimization migration algorithm to reduce the migration downtime, which deals with analysis of the memory transfer in the real-time migration of current Xen virtual machine. This algorithm uses layered copy algorithm and memory compression algorithm, and it optimizes the time and space complexity of real time migration and reduces the migration downtime which improves the migration performance.

Akoush S, Sohan R, Rice A, Moore A.W and Hopper A [1] experimented that the link speed and page dirty rate are the major factors impacting migration behaviour. These factors have a non-linear effect on migration performance. Migration time can be accurately predicted by enabling more dynamic and intelligent placements of VMs without degrading performance.

III. MOTIVATION

Virtualization technology allows multiple operating systems run concurrently on the same physical machine. Migration feature of virtualization provides facility to migrate virtual machines from one host (source) to another physical host (destination). If VM can migrate between servers without degrading currently running services, then it will improve the overall performance of the system. Live migration will keep the services live without interrupting its execution, handling the background issues like server consolidation, fault tolerance, load balancing between servers etc. at background with transparency.

Since it is a real time process, there exist different challenges based on the cloud infrastructure, selection of network, selection of host or even selection of VMs. There are some fundamental properties desired for live migration. The migration down time need to be ideally minimum, to offer continuous application services without any interruption by VMs. Particularly for interactive services, the migration process should not cause any sort of disconnection or performance degradation or delay for the users.

Most of the live migration data centers assumed to be happened in LAN. In that case VM disk image can be stored in a network storage device, which can be accessed by both source and destination. This helps to reduce the amount of memory state transfer between the servers. The other resources are CPU state and VM's memory state. The amount of data transferred for these resources are very small as compared with VM disk image. If the migration consumes huge amount of resources it leads to overall performance degradation and delay. So bottleneck occurs when simultaneous migration requires various resource allocation. Due to mismatch between the memory and the network in two strategies of live migration (pre-copy, post-copy) the down time increases thus increase the total migration time [5], [22]. If the migration time and down time can be predicted in advance, the optimum usage of resources and transfer rate can be selected, and this in turn improves the performance. It is actually a two-way solution, if there is no bottleneck in fixing the transfer rate and resources then the down time also get reduced.

In the above mentioned properties the uninterrupted service property keeps the migration 'live'. However, the other criteria are more flexible and can be compromised at smaller levels.

A. Challenges for keeping migration 'live'

Major issues which are challenges to keep migration alive are described as follows. If the selected migration process is pre-copy migration then three steps are performed to complete the migration process namely iterative copy, stop & copy and activation. In iterative stage while running the

source, pages are transferred from source to destination and at this time memory is also updated continuously [3]. However, the rate of updation of memory is higher than the speed of page transfer through the network. This scenario creates bottle neck in the network and more pages are buffered cumulatively and waiting time is prolonged for transfer. This led to drastic increase in the number of pages for stop and wait operation which ultimately extend the downtime and thus total migration time [17], [20]. Downtime extension leads to issues like interruption of services, disconnection of active users, database connectivity issues and so on. Major memory access from a common storage point like SAN or NAS reduces the issues related to storage transfer. In pre-copy iterative stage, [10] since source machines are working during transfer, a large amount of pages are dirtied again and these pages are also added to the queue for transfer. This issue will reach in its peak, if already sent pages get dirtied again. Sometimes the size of dirtied pages to be transferred, are equal to or more than the size of source VM. This also causes the reason to shoot up the down time and fails to predict downtime and thus total migration time.

Post migration algorithms resume the destination VM before its memory content have been transferred from the source VM to destination. As compare with the rate of dirtying at source VM transfer rate through the network is low and this cause a high risk of missing pages. Due to the same transfer rate problem if it tries to resend pages which are missing, it could not be successful and cumulated page fault occurs and that leads to severe performance degradation.

Thus, in this study we analyze the factors of live VM migration to retain the desired properties and overcome the challenges for real live migration.

IV. FACTOR ANALYSIS

The main objective of live migration is no service interruption would be perceived by the users of applications on the migrated VM. Performance of live migration involves several factors such as size of VM to be transferred, memory dirtying rate, the algorithm used to implement migration, network transmission rate, down time and total migration time, low resource consumption and so on. In the above factors reduced down time keeps migration as 'live' migration. Although, live migration down time is trivial but the transfer time is still affordable as compared with cold migration, which leads to the importance of improving live migration's performance and understand how to best utilize the feature in cloud management [2], [9]. In pre-copy strategy migration, it includes six stages such as *initialization, reservation, iterative pre-copy, stop & copy, commitment and activation*. In these steps iterative pre-copy is a repeated procedure and the selection of stop condition for that stage is a critical decision parameter in terms of overall performance of algorithm and may cause non-linear trends in the total migration time and downtime experienced by VMs.These stop conditions are highly dependent on the design of both the hypervisor and the live migration subsystem. But it is normally defined by the threshold amount of data copied between physical hosts while minimising VM downtime. Akoush S *et al*. [1] describe the factors affecting live migration and categorized it in to two, static and dynamic. In the abo.ve mentioned six steps, compared with other methods *iterative copy* is dynamic in nature. The effects of static parameters are considered as unavoidable migration overheads for applications which do not have high memory modification rate. But in most of applications the memory modification rate is high and the parameters having dynamic effect, which also contribute equally to the performance improvement of the whole migration process.

According to P. Sv¨ard, J. Tordsson *et al.* [8] the factors which affect the performance of live migration are broadly classified as page dirty characteristics and page content characteristics of the application. In the methods like delta compression and page skipping, the page transfer rate has a negative relation with the performance of migration. But false dirty pages don't affect much in the above mentioned methods. In strategies like de-duplication both page content characteristics and page dirtying characteristic has a negative effect on performance.

If we consider pre copy algorithm it proceeds with 'n' rounds. Initially the VM memory is transferred iteratively and transfers only the dirtied memory during the following rounds. At some point the number of modified pages will be small and at that time the source VM will be halted temporarily, copy the (normally small number) remaining pages from source to destination, and restart it on the destination host. That halted period of source VM and commencement of destination VM measures as *down time*. At that time user application in the VM is stopped temporarily both in source and destination machine. So down time is crucial for defining the performance of the services running in virtual machines.

The following model can be considered for analyzing the factors affecting live migration [10]. We denote the data volume transmitted at each round as *Di* ($0 \le i \le n$), and the elapsed time at each transferring round as *Ti* ($0 \le i \le n$). *D0* is equivalent to the VM memory size *Vm*. *T0* represents the time consumed to transfer the data of VM memory image and *Ti* is the time to transfer the dirty memory generated during previous rounds. Let V_m be the current size of VM, N_m be the total network traffic, K be the memory dirtying rate and T_{mig} be the migration latency. The data transmitted in round *i* can be calculated as:

$$D_i = \begin{cases} V_m, & if\ i = 0 \\ K.T_{i-1}, & Otherwise \end{cases} \quad (1)$$

Elapsed time at each round can be calculated as

$$T_i = \frac{K - T_{i-1}}{R} = \frac{V_m . K^i}{R^{i+1}} \quad (2)$$

Initially, the scenario we consider is that, the memory dirtying rate is much smaller than the memory transmission rate. Let α denote the ratio of K to R:

$$\alpha = K/R \quad (3)$$

Combining equations (1), (2) and (3), we have the network traffic during the round *i*.

$$N_i = K.\frac{V_m}{R}.\ \alpha^{i-1} = V_m.\ \alpha^i \quad (4)$$

Then the total network traffic during the migration can be summed up as:

$$N_m = \sum_{i=0}^{n} N_i = V_m \cdot \frac{1-\alpha^{n+1}}{1-\alpha} \quad (5)$$

$$T_{mig} = \sum_{i=0}^{n} T_i = \frac{V_m}{R} \cdot \frac{1-\alpha^{n+1}}{1-\alpha} \quad (6)$$

From equation (6) it is clear that the migration time (T_{mig}) is the cumulative sum of time taken to transfer dirtied pages in each iterative stage. So in any stage the dirtied rate is high then Ti for that stage increases thus increases total migration time (T_{mig}). Due to the mismatch in memory accessing speed and network transfer rate, some of the pages were missed.

In majority of systems the vendors always keep a tradeoff between minimizing total migration time and downtime [11], [12]. Fast synchronizing of dirty pages minimizes the downtime, but it will generate more migration data and prolong the migration time [13]. While Synchronizing lazily causes more memory dirty memory pages to be migrated during the migration downtime and will introduce more downtime. So an optimal approach may be better according to the current scenario, which would work better than selecting a plane migration algorithm. Another factor of migration which affects the performance is Preparation Time [15]. It is the time to start migration and transferring the VM's state to the target node. Ideally preparation time has to be minimized and it depends the transfer rate and band width of the network. The VM continues to execute and dirty its memory at this time. Another criterion is Resume Time. Since it is the time between resuming the VM's execution at the target and the end of migration, all dependencies on the source are eliminated and does not show direct relation with bandwidth or other major criteria. Pages Transferred can also considered as criteria since it is the total amount of memory pages transferred, including duplicates, across all of the above time periods. Pages Transferred rate directly related to network speed, bandwidth, dirtying rate and so on.

The effect of amount of memory used in the allotted memory hierarchy by VMsis a key improvising factor for performance optimization [2], [14]. The amount of memory allocated by hypervisor to VM is called **configured memory**. This **configured** memory act as the physical memory for the VM user application. But this memory cannot be fully allotted to VM due to various issues and the actual allotted memory is called **allocated memory** and it is less than the configured memory. Even this allotted memory is not completely utilized by VMs in the server. That memory which is actually used by VMs are called **used memory**. When different applications are running on VM, these applications request memory from VM. So VM allocate memory for them from this used memory. When memory migration is performing, the status of this used memory is transferred from source server to destination server. If an updation is reported in an already sent page, then it is required to transfer that change only (delta) to the destination, instead of the whole page. As compared with used memory this delta is very small. This method will reduce the size of data transferred from source to destination, thus the migration time also get reduced.

V. EXPERIMENTAL SET UP

Factors we analyzed in the previous section were evaluated through the event driven simulator called *cloudsim* simulator. The simulations were done to evaluate the effect of each factor in the performance of migration process. To evaluate the performance of Cloud, the virtual migration scenario were simulated through *cloudsim* in a Window 7 OS basic (64-bit), i3 Processor, 370 M Processor, 2.40 GHz of speed with memory of 3 GB and the language used is Java. Clark's [3] Pre-copy method is implemented as the migration algorithm here. The simulation model involves 40 VMs with cloudlets in it. It includes one data center and 20 numbers of hosts in it. Bandwidth assigned is 2 GB/s and VMM selected is Xen server.

VI. RESULT ANALYSIS

In migration, mainly CPU status, Memory status and storage status are transferred from source to destination. In this study we consider storage as common central store such as SAN or NAS. So the effect of storage migration is not included here. From literature survey and our study, it is observed that the amount of CPU status information needed to be transferred is very small as compared with other parameters. Even if the CPU is loaded with CPU intensive jobs then also there is no much change in migration data or time. It is because in CPU intensive jobs also the status information required to be transferred is same. So we can consider that live migration is independent of CPU loads.

While in memory migration both memory pages and the dirty bits (will set if there is a modification in already transferred pages) are to be transferred. Since continuous updation is taking place, this is a huge amount of information as compared with other fields. To study the effect of stress on memory migration, we have dirtied almost all memory pages, which leads to the drastic growth of dirty memory size and it prolongs the migration time, down time and migration data.

When we perform live migration in *cloudsim* using Xen server, it took around 35.38 seconds as total migration time (TMT) and down time is 5.07 seconds and transfers an amount of 2048 MB of data.

The fig. 3 show the relation between migration-time versus dirty memory size (in MB). Dirty memory size represents the change or updation happened to the page whose copy is already transferred to the destination. Here instead of transferring the entire page again, a bit which maps the change is only transmitted (dirty bit). As the time progresses some pages are missed because of speed mismatch between the networks and due to reasons like network congestion. This lead to dirtying of more bits and thus total dirty page size increases. As the graph progresses, the size of dirtying memory rate doubles according to the migration time and reaches a maximum value and then start decreasing. The reason for increase in the size of dirtying memory is because, while migration process progresses the amount of data resends increases and thus increases

migration time. After reaching a peak value it starts decreasing. This indicates the starting of down time where the source VM stops its working at source machine and sends all the pages of data to destination. At this time, VM being migrated is unresponsive to ping requests. So the amount of dirtying start decreasing, thus the graph bends downward. Again, in this graph the relation between dirty memory size and total migration data can also be shown. Initially the data required updation in the source machine is small and thus the size of dirtying memory is also small. Thus in migration, amount of data transferred is also small. So initially the graph doesn't show any clear linear progression. But when the dirty memory size increases into a reasonably big value (around 50 MB) the migration data also increases and shows a linear change.

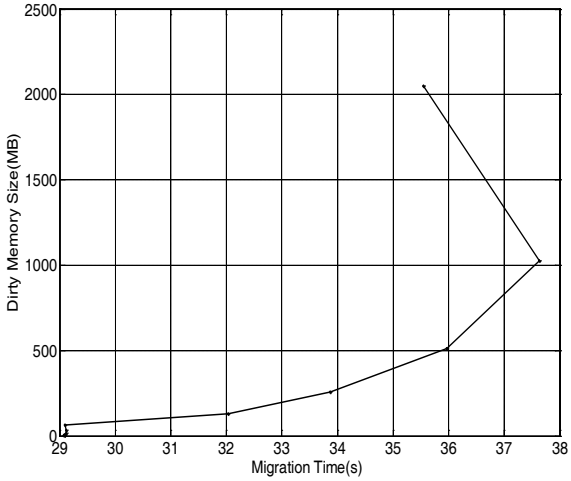

Fig. 3 Effect of dirty memory size on migration time

Total Migration time is composed of transfer time and down time. The transfer time is the time when the pages were transferred from source to destination when both the machines are alive. Initially, as the size of dirty memory increases there is no much change in migration time. It is because the dirty rate is low at initial stage and amount of data transferred is also low. As the migration data size increases, migration time also get increased.

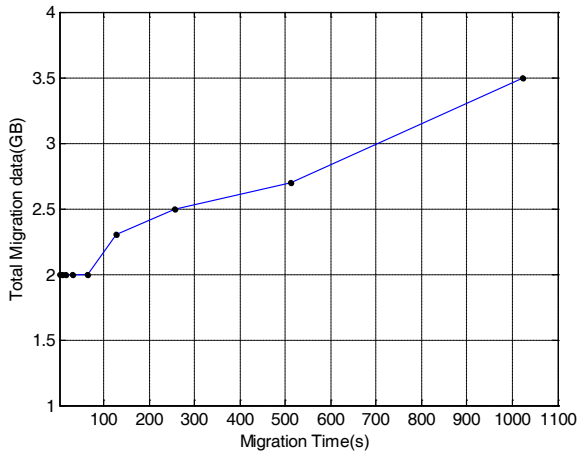

Fig. 4 Effect of migration data on migration time

Fig: 4 show the relation between migration time and total migration data. This is an extension to relation between page dirty rate and migration time. If the page dirty rate increases, amount of data migrated also increases. It is clear from fig. 4 that there is no major drift at initial points, but gradually a predominant change occurs due to heavy page dirtying rate. Selection of down time initiation in VMs and duration of down time has to be managed carefully. If it took more time to synchronize dirty pages, then down time prolongs in a negative way. So in most of the hypervisors considered a time out option for this situation. Sometimes here after, a cold migration is initiated. Fast synchronization of dirty pages reduces the down time, but it increases the migration data and thus migration time. If a slow synchronization is applied, it create more dirty pages thus prolongs the down time. So a tradeoff has to be adopted while selecting the dirty page synchronization.

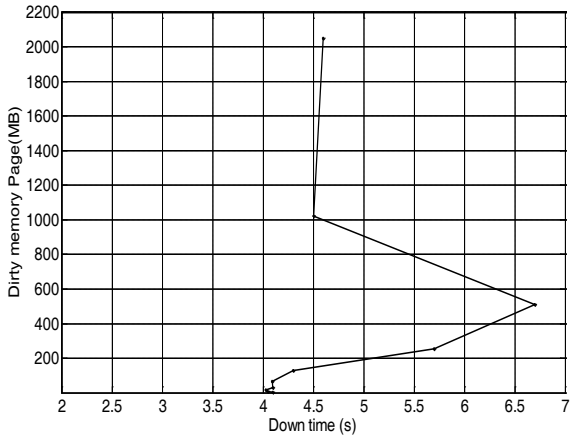

Fig. 5 Effect of dirty memory rate on down time

Fig. 5 indicate the relation between dirty memory page and down time. At down time, the source is temporarily stopped the VM execution and transfers the pages from source to destination and ideally down time has to be very small. The size of dirty memory increases means there are more number of changes in the already transferred pages from source to destination through migration. So that changes have to be reflected into the corresponding pages. This prolongs the down time. So the graph shows a linear drift as the size of dirty page increases. Reaching into a maximum point the graph moves downwards. It is due to the time out option.

Table I. Memory Migration Comparison

Hypervisor	***Performance matrices***		
	Down Time(s)	*Migration Time(s)*	*Migration Data(MB)*
KVM	0.18	12.45	250
Hyper-V	0.59	24.01	2255
VMware	1.01	18.05	2050
Xen	5.07	35.38	2048

In Table 1, we compare the migration time and down time of Xen hypervisor in virtual machine memory migration with other available hypervisors. The migration scenario and system specification considered for these hypervisors are

also the same as in the case of the experimental setup .The results for downtime and migration time for other hypervisors are collected from other researchers' experiments [15], [23], [24] ,[25]. The result shown that network throughput of Hyper V is the maximum, even though the migration time is not as good as other hypervisors values. A common observation is that when the amount of data increases migration time required is also increases. It is due to network congestion and more bandwidth utilization for transferring more amounts of pages. Another observation from the table is that the variation between downtime with migration data and migration time with migration data are not proportional especially in the case of KVM hypervisor for large values of data. The reason for same may be the change of methods adopted for migration in each hypervisor. Normally if a migration does not finish in a fixed time, cloud administrators initiate a time out option and complete it with cold migration.

The effect of dirty page rate on migration time, down time and migration data can be clearly identified from Table 2. Observation from Table 2 is that, the influence of dirty page rate is negligible up to a value of 128MB in most of the hypervisors.

When the dirty page rate is 1024MB and above, bandwidth utilization is huge for total data transfer. Thus migration time and down time substantially increases. Here only in VMware, the total migration data increases proportionally as the dirty page rate increases. In Xen and Hyper-V instead of continuous increase, the migration data decreases after a certain point (2048 MB).

This is be due to some of the dirty pages will be swapped into the storage disk, which saves the synchronization time for the memory. When the pages are in the storage disk, it is shared rather than transferred through memory migration. In Table 2, KVM is not observed because, for large values of page dirty rate, this could not complete the live migration process. So for large values of dirty page rate, a time out option is initiated and completes the migration process through cold migration. The dirty memory size becomes a crucial factor in Xen's downtime. For large values of dirty memory rate the downtime goes up to or over 20 seconds and which is not favorable for most of the web services.

The impact of CPU activity is trivial on live migration. Our simulation results show that even though CPU intensive workloads are allocated to migrating virtual machines, which will not affect the speed of migration process. But Memory write operation impacts more on memory migration because of increase in the dirty page rate. The dirty page rate increases the amount of total migration data and thus migration time.

Table II. Effect of dirty page rate on migration time, down time & migration data

Dirty Memory Size (MB)	***Total Migration data(GB)***			***Migration Time(s)***			***Downtime (s)***		
	Hyper-V	**VM ware**	**Xen**	**Hyper-V**	**VM ware**	**Xen**	**Hyper-V**	**VM ware**	**Xen**
2	2.01	0.68	2.01	21.3	6.5	30.6	0.52	0.5	4.1
4	2.05	0.68	2.01	20.05	6.2	30.5	0.55	0.51	4.01
8	1.95	0.65	2.05	20.0	7.02	31.5	0.51	0.6	4.44
16	2.05	0.62	2.25	22.2	6.00	32.4	0.55	0.55	4.02
32	2.25	1.12	2.2	19.08	5.0	30.7	0.51	0.52	4.05
64	2.2	1.45	2.7	24.01	5.7	30.5	0.59	0.51	4.9
128	2.7	1.23	2.45	25.5	6.6	33.3	0.6	1.0	5.01
256	2.75	1.55	2.25	28.8	8.1	33.2	0.66	2.0	5.5
512	3.00	1.9	3.00	30.1	16.05	35	0.61	1.02	7.01
1024	4.09	2.01	4.6	35.1	18.05	36.05	0.60	1.01	14.25
2048	2.69	3.27	2.45	27.5	25.6	35.3	0.65	1.02	5.07

VII. CONCLUSION

Live migration is an important feature of VM management in data centers. In this paper different factors affecting the live VM migration are discussed. Different challenges to keep the live VM migration also analyzed. By reducing the downtime into a negligible value keep the migration process as live migration. By analyzing migration characteristics, it found to be clear that the factors such as total migration time and down time are not linearly related to the page dirtying rate and link bandwidth rate. So a tradeoff has to be maintained between these parameters to get optimum performance. As compared with CPU status migration, memory migration is more affected by the page dirty rate thus migration speed gets reduced. Like dynamic parameters, the effects of static parameters are also equally important especially for applications which do not have high memory modification rate.

We plan to analyze the effect of static parameters in live migration along with dynamic parameters. Currently, the effects of dynamic parameters are only considered to analyze the live migration characteristics. In future, we plan to analyze and improvise the static parameters, so that it will improve performance of live migration.

VIII. REFERENCE

[1] Akoush S, Sohan R, Rice A, Moore A.W & Hopper A., Predicting the performance of virtual machine migration. In IEEE International Symposium on Modelling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS), pp. 33-46, 2010.

[2] Wenjin Hu and Andrew Hicks, A Quantitative Study of Virtual Machine Live Migration,InProceedings of the ACM Cloud and Autonomic Computing Conference, pp. 11-15 ,2013.

[3] Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, EricJul, Christian Limpach, Ian Pratt, and Andrew Warfield, Live migration of virtual machines. In NSDI '05: 2nd Symposium on Networked Systems Design and Implementation, USENIX Association, Vol.2, pp. 273-286, 2005.

[4] Raja Wasim Ahmad& Abdullah Gani,A survey on virtual machine migration and server consolidation frameworks for cloud data centers.In Journal of Network and Computer Applications, Vol.52, pp. 11-25, 2015.

[5] Senthil Nathan, UmeshBellur, On Selecting the Right Optimizations for Virtual Machine Migration,InInternational Conference on Virtual Execution Environments on 12th ACM SIGPLAN/SIGOPS ,pp. 37-49,2016.

[6] Robert Jay Creasy. The origin of the VM/370 time-sharing system.In IBM Journal of Research &Development, Vol.25, pp. 483–490, 1981.

[7] Zhao,Jia,LiangHu,GaochaoXu,DechengChang,Yan Ding &XiaodongFu,A Fast Live Migration Algorithm of Virtual Machine with CPU Scheduling.In Proceedings of the Internatonal Conference on Grid Computing and applications(GCA),In Proceedings of the International Conference on Grid computing and Applications(GCA),pp. 115-120,2013.

[8] P. Sv¨ard, J. Tordsson, B. Hudzia, and E. Elmroth,High Performance Live Migration through Dynamic Page Transfer Reordering and Compression, In CloudCom '11: 3rd IEEE International Conference on Cloud Computing Technology and Science, pp. 542–548, 2011.

[9] J. Yang, Key technologies and optimization for dynamic migration of virtual machines in cloud computing, In International Conference on Intelligent Systems Design and Engineering Applications, pp.643-64, 2012.

[10] Liu,Haikun,HaiJin,Cheng-ZhongXu&Xiaofei Liao ,Performance and Energy modelling for live migration of virtual machines. In the Journal of Cluster Computing 16, Vol.2, pp.249-264, 2013.

[11] Maziku,Hellen&SachinShetty,Towards a network aware vmmigration:Evaluating the cost of vm migration in cloud data ceters,In IEEE 3rd International Conference on Cloud Networking(CloudNet),pp.114-119,2014.

[12] Chen J,QinY,Ye y & Tang Z,A Live Migration Algorithm for Virtual Machinein a Cloud computing Environment.InUbiquitious Intelligence and Computing and IEEE 12th International Conference on automatic and Trusted computing,pp.1319-1326,2015

[13] Forsman,Mattias,AndreasGlad,Lars Lundberg &DragosIlie,Algorithms for Automated Live Migration of Virtual Machines,In Journal of systems iand software ,Vol. 101,pp.110-126,2015.

[14] Thaman J & Singh M, Improving Performance of Cloud Datacenters Using Heuristic Drive VM Migration. In International Conference on Next Generation Computing Technologies (NGCT), pp.41-45, 2015.

[15] Salfner,Felix,PeterTroger&andreasPolze,downtime Analysis of Virtual Machine Live Migration.In fourth International Conference on Dependabilty ,IARIA,pp.100-105,2011.

[16] KivityAvi,YanivKamay, dorLaor,Uri Lublin & Anthony Liguori,kvm: the Linux Virtual Machine Monitor,In Proceedings of the Linux Symposium,Vol.1,pp.225-230,2007.

[17] M. R. Hines and K. Gopalan.Post-copy Live VirtualMachineMigrationUsing AdaptivePre-paging and DynamicSelf-ballooning, In VEE, 2009.

[18] Shirinbab,Sogand,Lars Lundberg &DragosIllie,Performance Comparison of kvm, vmware and xenserver using a large telecommincation application, In cloud Computing IARIA XPS Press,2014.

[19] Song X, Shi J, Liu R,Jian Yang &Haibo Chen ,Parallelizing Live Migration of Virtual Machines,In ACM Sigplan Notices,Vol.48,pp.85-96,2013.

[20] Svard,PetterB,Hudzia S Walsh,JohanTordsson& Erik Elmroth. The Noble Art of Live VM Migration-Principles and Performance of Pre copy, Post copy and Hybrid Migration of Demanding Workloads, Techncal Report UMINF, Vol. 14.10,2014.

[21] Paul Barham, Boris Dragovic, Keir Fraser, Steven Hand, Tim Harris, AlexHo, Rolf Neugebauer, Ian Pratt, and Andrew Warfield, Xen and the art of virtualization. In SOSP '03: Proceedings of the nineteenth ACMsymposium on Operating systems principles, Vol.37, pp.164–177, 2003.

[22] Eric Harney, SebastienGoasguen, Jim Martin, Mike Murphy, and MikeWestall, The efficacy of live virtual machine migrations over the internet, In VTDC '07: 2nd International Workshop on Virtualization Technologies in Distributed Computing, pp 1-7. ACM, 2007.

[23] www.Linux-kvm .org

[24] Why Hyper-V? Competitive-advantages-of-windows-server-hyper-v-over-vmware-vsphere.pdf, October 2013 Vo1.0.White Paper.

[25] Virtual Machine Migration Comparison: VMware vSphere vs Microsoft Hyper-V.A principled technologies test report, commissioned by VMware Inc.Whitepaper, 2011.