Live Migration Time Estimation

Arunachalam Ramanathan
VMWare, Inc.
aramanathan@vmware.com

Ka Wing Ho
VMWare, Inc.
miho@vmware.com

Gabriel Tarasuk Levin
VMWare, Inc.
glevin@vmware.com

Abstract

We propose a mechanism to estimate the time for live migration of a Virtual Machine (VM). Migration time estimation can inform a customer the amount of time that is required for live migration of a VM. It can inform Virtual Center (VC) on the cost of doing a migration thereby, allowing Distributed Resource Scheduler (DRS) to make better load balancing decisions [1]. Migration time estimation can be used to improve migration scheduling at VC so that the migration time of a single VM is reduced.

We discuss the method to estimate migration time based on two key metrics which are guest VM page dirty rate and vMotion network transmit rate. We discuss the issues associated with the two metrics and present techniques to better estimate them. We maintain a history of vMotion network transmit rate and use trend estimation [2] to determine the future dirty rate of a VM. Finally, we discuss the results of the estimation, limitations of the techniques and future work.

General Terms: Resource management, scheduling and performance

Keywords Live Migration, memory, time estimation

1. Introduction

The ability to do live migration of a VM is one of the key advantages of virtualization. Live migration has many applications, some of which include reduced downtime during maintenance, load balancing resource pools and starting Fault Tolerant VMs [1].

Today, there is no reliable way to estimate the time required for live migration. When a customer wants to change a host into maintenance mode, the customer has no clue on when the operation is scheduled to finish. If the customer could get an estimate of the time required for the live migration of a VM and hence the entire maintenance mode operation, it would help to take an informed decision.

Live migration is used by DRS to balance resource pools. DRS load balances resource pools based on the cost of the live migration. It computes the cost of the live migration by using active memory from VM memory statistics. ESX defines active memory of a VM based on the fraction of the guest physical pages touched in the last one minute. The calculation does not hold good in cases where the time to migrate a VM takes less than a minute or in cases where a memory precopy iteration takes less than a minute. A more accurate time estimation of a live migration would help DRS to make better load balancing decisions.

Migration time estimation can inform VC on the time it takes to live migrate a VM from ESX's perspective and can lead to better migration scheduling decisions at the VC. The current migration scheduling policy in VC is based on minimum resource limits for a live migration. For example, in case of vMotion, 250 mbps is the minimum required network bandwidth. If a 1 GbE vMotion network is available between source and destination ESX hosts, VC can schedule 4 simultaneous migrations. But a single vMotion is designed to make use of the entire network bandwidth available, say 1 GbE in the above case. The time taken for a vMotion is inversely proportional to the vMotion network transmit rate. So if migrations are scheduled sequentially, a single migration will run to completion faster. A reduced migration time is beneficial because it means fewer changed pages and lesser amount of data to be transmitted over the network. Also, during the time of migration, the VM config file and NVRAM file are in read only mode, VM's performance is affected due to page tracing and VM's large pages are broken. So with migration time estimation, VC can make optimized scheduling decisions with respect to migration running time and VM's performance.

The paper is divided into 6 sections. Section 2 describes the method to estimate migration time, lists the required metrics and the issues with existing metrics. Section 3 discusses techniques to improve existing metrics. Section 4 discusses the experiments, results, limitations and alternative approaches. Section 5 talks about migration scheduling based on migration time estimation. Section 6 is conclusion and future work.

2. Migration Time

Migration of a VM involves several phases namely memory precopy, stun, checkpoint, checkpoint transfer, page in and execution switch from source to destination. During migration, majority of the time is spent in precopying guest physical memory from source to destination host and is called memory precopy time (T_{MP}). The switchover time (T_{SW}) is the time taken for stun, checkpoint, checkpoint transfer, page in and execution switch from source to destination host. T_{SW} is fairly constant as it happens during guest downtime which is limited to 1 sec during live migration. So the total migration time is the sum of memory precopy time and switch over time.

$$T_M = T_{MP} + T_{SW} \quad (1)$$

As switch over time is limited, this section focuses on estimating memory precopy time.

2.1 Memory Precopy Time

Memory precopy time is defined as the time taken to send physical memory of the VM from source to destination host. Memory is tracked in units of pages. Memory precopy is an iterative process and involves tracing guest physical pages. When guest tries to write to a page which is traced, it takes an exit into virtual machine monitor (VMM) where the trace is fired and migrate is notified about the page state change. The first phase of

precopy is called trace phase where traces are installed on all pages of the guest memory. After trace installation, the pages whose traces remain installed are transmitted to the destination. After the trace phase, in subsequent iterations the pages whose traces have been fired are transmitted and new traces are installed to track the transmitted pages.

In some cases, memory precopy will converge to a small number of outstanding pages which are sent during guest down time and this process is called Pagein. The time taken for Pagein is not accounted in MPT as it runs in parallel with checkpoint transfer. The time taken to precopy memory is the sum of time taken to install page traces (TPT) and the time taken to send all pages (TSP) given by equation 2.

$$T_{MP} = T_{PT} + T_{SP} \quad (2)$$

$$T_{PT} = \text{Trace install time} * \text{num of pages} + \text{Trace throttling} \quad (3)$$

$$T_{SP} = \text{Memory size} / (\text{Transmit rate} - \text{Page Dirty Rate}) \quad (4)$$

TPT, the time taken to install page traces is the sum of time taken to install traces for all pages and trace throttling. Page traces are installed after stopping vCPUs and guest instructions are not running at this time. For VMs with large memory, continuous trace installation leads to starvation of guest. So trace throttling was introduced which is a sleep of 10 ms after continuously installing traces for 400 MB of memory. So the time taken to install page trace is a function of memory size and is simple to estimate.

TSP, the time taken to send all pages from source to destination is a function of memory size, vMotion network transmit rate and page dirty rate. vMotion network transmit rate is the amount of network bandwidth available to live migration during precopy. Live migration's definition of dirty rate is different from VM's definition of dirty rate. For live migration, if the page trace fires, the page is dirtied and has to be transmitted. But if the page is dirtied again between the time the trace is fired and the time the page is transmitted, it is not accounted as it does not affect live migration. Even if the the page changes multiple times in this interval, it needs to be transmitted once.

For cases where page dirty rate is greater than vMotion network transmit rate, memory precopy cannot convergence. In such cases, we rely on Stun During Page Send (SDPS) to slow down vCPUs of the VM to reduce the page dirty rate below transmit rate.

2.2 Issues with existing metrics

ESX does not compute or maintain vMotion network transmit rate. As a result, the transmit rate is not available at the start of a migration to determine migration time. Live migration is programmed to get the link speed of the vMotion network on the host but is not guaranteed to get the transmit rate at link speed as customers may multiplex other traffic such as NFS, iSCSI, FT on the same link. There can be another migration in progress and the available bandwidth needs to be shared.

ESX computes VM's dirty rate using statistical sampling by tracing a fraction of guest physical memory over each minute of guest time. The existing memory dirty rate/min provided by ESX is insufficient in cases where the memory precopy time is less than a minute as it can lead to incorrect estimation of the total amount of memory that needs to be transmitted. The approach to determine dirty rate/min installs page traces at the start of the minute. If the trace fires, the page is marked dirty and the trace is not reinstalled for the rest of the minute. So if the page is dirtied multiple times a minute, it is not accounted in the dirty rate. If we consider a live migration where a precopy iteration runs for 5 secs and if the page is dirtied once every 5 secs, it needs to be transmitted in multiple times. In the above case, the total amount of memory transmitted and time taken for memory precopy will be different from the estimated time using page dirty rate/min.

3. Techniques to improve metrics

3.1 Separate vMotion network with transmit rate history

Reservation: To get a reliable estimate of vMotion network transmit rate, vMotion traffic needs to be separate from other traffic. Today we recommend customers to have vMotion network isolated from management network for security reasons as we transmit VM memory over vMotion network. In addition to this we need to isolate vMotion network traffic from other vmkernel application traffic such as NFS, iSCSI, FT etc. This can be done by limiting vMotion traffic to a separate network infrastructure. If this is not feasible, vMotion port group at DVS should get a reservation for the allocated bandwidth so that vMotion network transmit rate is not affected by other traffic. Similarly bandwidth can be reserved at the switching layer using QoS, isolation etc.

Migration Scheduling Policy: Migration scheduling policy at VC should be changed from scheduling migrations simultaneously based on resource reservation limits to scheduling migrations sequentially for performance reasons. This will help to better estimate vMotion network transmit rate for migrations.

Transmit Rate History: We need to maintain a history of vMotion network transmit rate between the source and destination ESX hosts. Initially, the transmit rate can be assumed to be a function of link speed and it can be refined over time by maintaining a transmit rate history of migrations between ESX hosts.

3.2 High resolution dirty rate

To estimate migration time we require dirty rate to be sampled at a higher resolution than the existing resolution of one minute. The resolution of the dirty rate depends on migration time estimation model and the overhead involved in sampling. We plan to estimate migration time in seconds. So we need dirty rate at a resolution of one sec or less. Increasing the sampling rate from one minute to one second has a high overhead as installing page traces have impact on guest performance. A one sec sample can be collected once or few times every minute thereby minimizing the overhead associated with high sampling interval [3]. The sampling rate and interval can be determined based on how well we are able to estimate the migration time. For the experiments the sampling rate and interval was set to one sec. We haven't experimented or analyzed various possibilities to determine the exact sampling rate and interval.

3.3 Page dirty rate modelling and forecasting

Migration time estimation requires average dirty rate of VM for the running time of the migration to give a close estimate. So a high resolution dirty rate sampling alone is not enough. Because the last sample is a point in time data and is not a good representation of the workload. We can take the average of the last few samples but if the workload had a burst or was idle during last few samples, the time estimate will be off.

Ideally, we want to model the workload using the collected samples. From the model, predict future values of dirty rate. Then take an average of the predicted dirty rate and use it to estimate the migration time. The idea here is that the model would capture the characteristic of the workload and help in better estimation of dirty rate and migration time.

4. Experiments

The sample period was set to one second using the vsish option /config/Mem/intOpts/SamplePeriod. This provided the page dirty rate of VM at a one second resolution. Page dirty rate samples were collected every sec from the vsish cache through vmkernel. These samples were used to build a model of the workload to estimate the future page dirty rate.

Experiment setup consisted of two ESX hosts each with 4 Intel CPUs and 128 GB of main memory. The hosts had dual 10 GbE nics for vMotion network. Page dirty rate samples were collected from 2 VMs. The first VM had Windows 2008 server operating system with SPECjbb 2005 workload [4]. The VM had 4 vCPUs and 20 GB of memory out of which 16 GB was configured as SPECjbb heap. The second VM had Freebsd operating system with Write Page workload. The VM had 4 vCPUs and 10 GB of memory. The Write Page workload was designed to test vMotion where it steps over memory in a tight loop and changes one character per page to generate high dirty rate. It had 8 GB of memory and for this experiment it was programmed to sleep randomly for random seconds while stepping over memory. This was done to model a bursty workload.

4.1 Page dirty rate forecasting using Trend Estimation

Trend estimation [3] was used to model the workload from the collected samples. This involved estimating a trend from the samples and using the trend to forecast the future dirty rate of the VM.

The initial step is to compute exponentially weighted moving average (EWMA) from the collected samples. This helps to smooth the page dirty rate time series [5]. The initial value for EWMA is computed as the mean of the previous 5 samples. Alpha is known as the smoothing constant. It is computed by equation 5 where N is the number of samples taken for trend estimation. The EWMA is used to find the trend of the time series.

$$\text{Alpha} = 2 / (N + 1) \quad (5)$$

$$\text{EWMA} = \text{alpha} * \text{sample} + (1 - \text{alpha}) * \text{EMWA} \quad (6)$$

The next step is to compute noise (e) for each sample from its corresponding EWMA. Then do a regression analysis [6] on EWMA time series to find the intercept (b) and slope (a).

$$e = \text{Sample}/\text{EWMA} \quad (7)$$

$$Y = at + b \quad (8)$$

$$Y_t = (at + b) * e_t \quad (9)$$

The slope and intercept gives the trend (Y) of the sample. The future trend can be obtained by equation (8) where t stands for the time. Noise is a characteristic of the workload and is the only variable used in forecasting the future dirty rate. We multiply noise with the trend to get values for the future page dirty rate. Microsoft Excel was used for trend estimation and regression analysis.

Page dirty rate samples were collected from both SPECjbb2005 and Write Page workloads to forecast dirty rate for different periods of time (N) where N was 10, 30 and 60 secs. Different periods were used to understand how effectively the workload was modelled.

For each forecast period N, 2N samples were collected. The first N samples were used to estimate the trend of the workload. For each of the first N samples, its corresponding EWMA and noise are computed. Regression analysis is run on EWMA values to get the slope and intercept which reflects the trend of the workload. The slope and intercept along with noise from the collected N samples were used to forecast N future page dirty rates of VM. The forecasted N future page dirty rates are compared against the second half of the (observed) collected 2N samples. We do a time series plot of the observed and forecasted dirty rates to see effectiveness of the model. We also compare the average of the observed and forecasted dirty rates to get forecast error %. The page dirty rate is expressed as the percentage of memory that was written in the last one sec.

4.2 Results

Table 1 displays the results for different experiments conducted with SPECjbb2005 workload. Trend estimation was able to forecast page dirty rate within 5 % of error for all the periods. Figure 1 shows a time series plot for the forecasted period of 30 secs. Here the forecasted page dirty rate plot closely resembles the observed page dirty rate plot. For this case, the forecasted average was within 0.66% of the observed average. From the collected samples for different periods, the SPECjbb2005 workload did not express any major changes in the dirty rate.

The average standard deviation and variance for the collected samples for different periods were around 10% and 25% from the mean. For workloads which do not have big variations in dirty rate, trend estimation is effective.

Table 2 displays the results for different experiments conducted with Write Page workload. For smaller forecast periods, trend estimation was unable to predict trend and resulted in high percentage of errors.

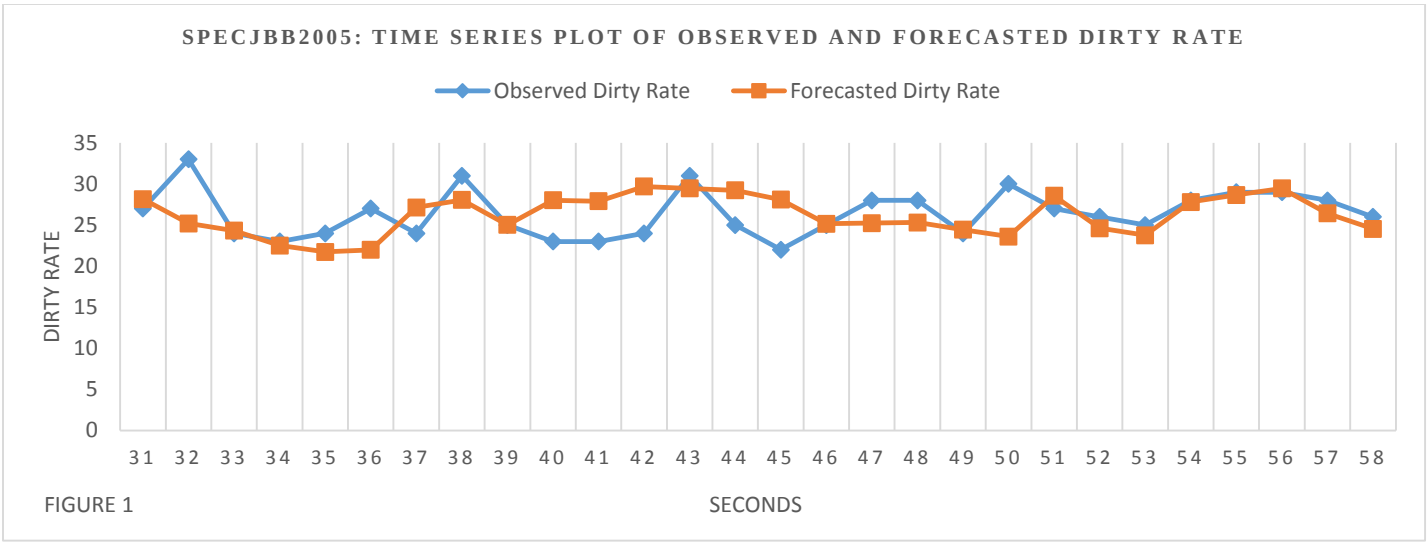

FIGURE 1

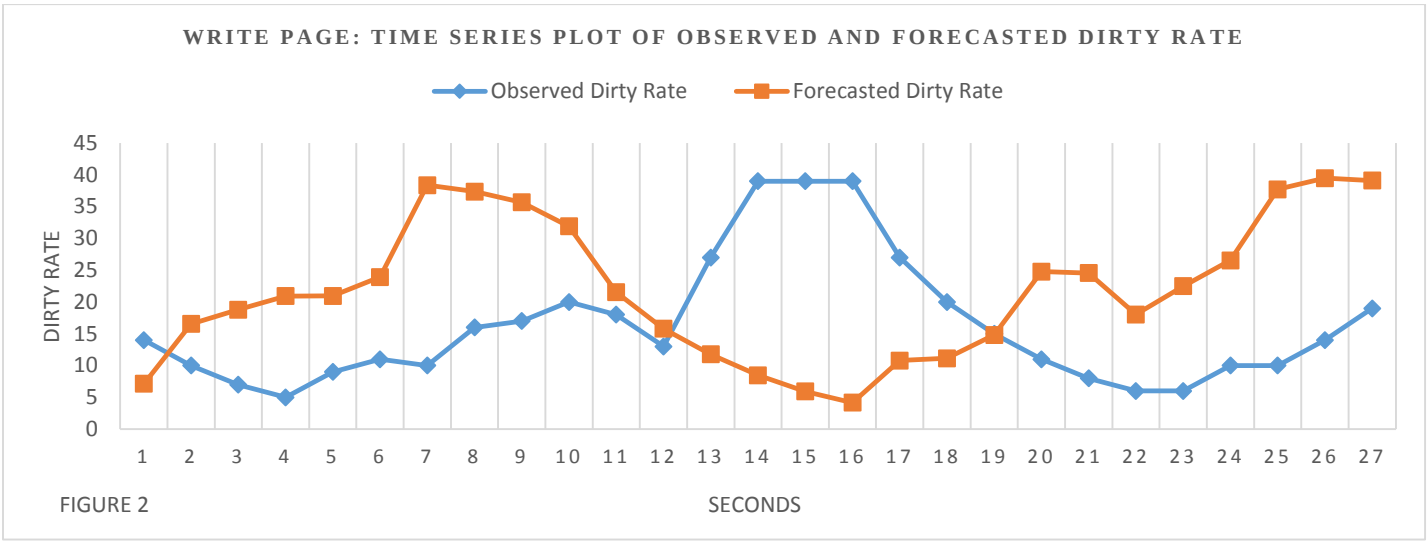

FIGURE 2

Table 1: SPECjbb 2005 Workload

S. No	Forecasted time	Average of dirty rate/sec		
		Observed	Forecasted	% Error
1	10	25	25.19	0.76%
2		25.7	27	5.06%
3	30	26.39	26.22	0.64%
4		25.21	26.42	4.80%
5	60	25.68	24.82	3.35%
6		24.72	26.12	5.66%

Table 2: Write Page Workload

S. NO	Forecasted time	Average of dirty rate/sec		
		Observed	Forecasted	% Error
1	10	9.1	38.65	324.73%
2		13.8	35.34	156.09%
3	30	24.96	18.75	24.88%
4		16.39	22.21	35.51%
5	60	19.28	19.69	2.13%
6		19.91	21.12	6.08%

Write Page workload was designed to have bursts in dirty rate. So trend estimation was unable to model the workload for periods which only consists of the burst or no burst. But for longer periods, trend estimation is able to learn the characteristic of the workload. Even if it learns for longer period, it cannot successfully forecast for a smaller period of time.

Figure 2 shows a time series plot for the forecasted period of 30 secs. In this case, the forecasted average has 35% error compared to observed average. The forecasted page dirty rate plot widely varies from the observed page dirty rate plot. The average standard deviation and variance for the collected samples for different periods was 60% and 750% from the mean .For workloads which have big variations in dirty rate, trend

estimation is not effective. In the next section we discuss the limitations of trend estimation model.

4.3 Limitations and alternative approaches

Trend estimation is a linear model and can be trained to learn trends of a workload. If a workload is complex and does not follow a trend, then forecasting through trend estimation will have high error rates as observed in the experiments. One of the problems was that forecasting was based on one variable which was the noise of the workload as slope and intercept were constants. A complex workload cannot be expressed as a function of one variable. It requires a sophisticated model like Markov model [6] which associates the probability for the given variable. Another approach is to have sub second sampling rate so that complex curve can be represented as a line and the model can learn the slope and intercept to determine the trend.

5. Migration Scheduling

One of the important uses of migration time estimation is to improve migration scheduling in VC. This section discusses the scheduling improvements that can be done in VC with the help of migration time estimation.

A migration starts with migration request passed down from VC to the ESX. There is a significant overhead involved in passing the migration request from VC to the migrating VM on ESX. The migration request has to traverse the management stack which comprises of VC, host agent and host daemon before getting to the VM to be migrated. The overhead time (T_O) is the time taken when the VM is selected for migration to the time the actual migration request is received at the VM. It typically takes around 3 seconds but can take several seconds depending upon the load and queuing at the intermediate layers. The VC scheduling policy should be designed in such a way as to mask this overhead while scheduling multiple migrations. The total migration time from an end user perspective is the sum of the overhead time and the migration time.

$$T = T_O + T_M \quad (10)$$

As per VC's current migration scheduling policy which is based on resource limits for a live migration, it can start 4 simultaneous migrations on 1 GbE link or 8 on a 10 GbE link. While starting 4 or 8 migrations, all of the migration requests traverse the management stack at the same time and may not appear as a big overhead. The introduction section described the benefits of sequential migration scheduling. So while moving towards sequential scheduling, this overhead needs to be minimized as this may be applicable for every single migration.
VC needs to estimate the migration time before scheduling the migration. This requires the above discussed metrics such as VM page dirty rate and vMotion network transmit rate be exported and maintained at VC. Once VC estimates migration time, it can start the request. If the next migration is between different pair of hosts then it can be started in parallel. Even if one of the hosts is involved in another migration, VC needs to wait. If not, VC should start next migration after time T_M instead of waiting till T. This will help eliminate T_O for successive migrations between the same hosts.

In the absence of sophisticated estimation models for dirty rate, it may not be possible to accurately estimate the time of migration. In that case, we wish to err on the conservative side by providing migration time estimation based on the worst case metrics. For this, VC needs to maintain a long history for both vMotion network transmit rate and VM page dirty rate.

To summarize, migration scheduling based on migration time estimation will result in overall reduced migration time in case of scheduling multiple migrations such as changing host into maintenance mode.

6. Conclusion and Future Work

We presented a mechanism for live migration time estimation. The main component of migration time was memory precopy time which is a function of VM memory, vMotion network transmit rate and VM page dirty rate. We discussed the existing issues with above metrics and provided techniques to improve them. Presented experiment results for page dirty rate forecasting based on trend estimation. Finally, we discussed migration scheduling improvements at VC.

In future, we plan to work on techniques to better estimate vMotion network transmit rate and VM page dirty rate. Explore sampling techniques to minimize sampling overhead and collect high resolution samples. Research on sophisticated techniques like Markov Model to better forecast VM page dirty rate. We plan to export the metrics to VC to make improvements to migration scheduling, DRS cost benefit analysis and show expected migration time to end users.

References

[1] http://www.vmware.com/products/vsphere.

[2] http://en.wikipedia.org/wiki/Trend_estimation.

[3] A Sampled-LRU Approach for VM Working-Set Estimation. Carl A.Waldspurger, Rajesh Venkatasubramanian.

[4] http://www.spec.org/jbb2005/

[5] http://en.wikipedia.org/wiki/Exponential_smoothing

[6] http://en.wikipedia.org/wiki/Markov_model