Prediction-Based Optimization of Live Virtual Machine Migration

Changyuan Chen, Jian Cao

Prediction-based Optimization of Live Virtual Machine Migration

Changyuan Chen, Jian Cao*
Department of Computer Science and Engineering, Shanghai Jiao Tong University, China
changych@sjtu.edu.cn, cao-jian@cs.sjtu.edu.cn

Abstract. Virtual Machine (VM) migration is an important technology to support Infrastructure as a Service (IaaS). Traditional pre-copy and post-copy strategies could function well in LAN but will need considerable time to migrate between remote hosts in WAN. In this paper, we propose a prediction-based strategy to optimize cloud VM migration process over WAN. In this strategy, information about size increments of snapshots is used to determine appropriate time points for migration in order to reduce the downtime during migration. Specifically, we utilize Markov Chain Model to predict the future increasing speed of snapshots. The experiments on KVM showed our approach could achieve satisfying results.

1 Introduction

Cloud computing helps enterprises take advantage of resources provided by large cloud service vendors. Typically, enterprises need to expand their IT capabilities during workload peaks; meanwhile migrating a VM to a cloud is a cost-efficient choice. As a result, attention is being attracted to live VM migration.

The entire process of VM migration can be divided into three stages: the pre-copy, the down time and the synchronization stage [1]. During the pre-copy stage, a VM keeps running while the modified data is transferred [2]. After that, the VM shuts down and synchronizes the latest data [3]. In post migration, the VM resumes on the destination host before all the modified data is transferred [4]. So data on both sides should be synchronized. The durations of these three periods are important metrics and most of the migration strategies are designed for optimizing these metrics.

* Corresponding Author

There are three classic basic algorithms for VM migration, namely pure stop-copy, pre-copy and post-copy algorithm. Pure stop-copy algorithm is designed to shut down the VM and copy all its state to the destination host [5, 6, 7]. Although pure stop-copy algorithm can minimize the total migration time, it creates long down time. In order to reduce the down time, pre-copy algorithm is widely used. For example, Khaled Z. presents a pre-copy based algorithm on-line (OL) to provide minimal downtime [2]. Post-copy algorithm is another way to reduce the down time during VM migration. Michael designs a post-copy based strategy using adaptive pre-paging across a Gigabit LAN [8]. Pre-copy algorithm and post-copy algorithm could reduce down time, but they both require a high bandwidth environment like LAN.

From the strategies above, we learn that the strategy to reduce the down time during VM migration is a critical issue. Lots of strategies work well in LAN, where the need of high bandwidth is meet. But they could hardly perform well in WAN. In this paper, we propose a prediction-based migration strategy, aiming to minimize the down time during VM migration. The prediction-based strategy could initiatively learn the VM's state and select the optimal points to complete the migration. While a VM running on a host, snapshots are taken and transferred to the destination host iteratively. Every time one snapshot is transferred, we predict an increasing curve of snapshot sizes using Markov Chain. Based on the prediction, we can capture the growth platform, which is the optimal time to finish the whole migration.

The rest of this paper is organized as follows. In the following section, we describe some related work about our problem. Then, we analyze the characteristics of snapshots on KVM platform in Section 3. Section 4 discusses the actual design and implementation of our migration strategy. Section 5 describes the experiments and their results. Finally, we draw some conclusions and describe the future work.

2 Related Work

VM migration technology enables most of the cloud services to work for a surge of customers. Lots of achievements about VM migration have been gained in recent years. XenMotion [9] is the migration module in Xen which adopts a pre-copy algorithm to address the issue, and VMotion [10] developed by VMware also allows a running VM to be moved from one host to another. They both aim at the LAN environment [11]. Especially, Xen implements live migration but it requires shared

storage between hosts [12]. But migration in LAN can no longer meet the demand, so in this paper we propose a VM migration strategy which is adapted for WAN.

Liu proposed a novel approach to provide fast, transparent VM migration for both LAN and WAN environments, which is called CR/TR-Motion[11]. Liu's experiments demonstrated that CR/TR-Motion works well in LAN environment, but its performance in WAN is unsatisfactory. Timothy presented architecture, namely CloudNet, as a cloud framework with a VPN based network infrastructure to provide VM migration in WAN [13]. He optimizes the cost for transferring storage and VM memory in WAN environment, but CloudNet he implemented is built on the base of VPN. As is known, most VM migrations work in the general Internet environment, and we can hardly transfer data through VPN. On contrary, the VM migration strategy we propose is suitable for the general Internet environment. In our strategy, we make use of the incremental characteristic of snapshots and use pre-copy mechanism to reduce the down time during migration. In order to get the minimum snapshot increment during migration, we propose a prediction-based strategy using Markov chain as a theoretical basis. VM snapshot is a collection of all the states of the VM, including storage data, memory pages and CPU states. So we propose a prediction strategy to forecast the growth trends of VM snapshots, which will help to optimize the down time during migration.

3 Prediction-based Model

In this section we describe our prediction-based model, which will smooth the way to our migration strategy. Two core aspects will be presented in the following sub-sections: snapshot size growth and the prediction model.

3.1 The Growth of the Size of a Snapshot

VM snapshots are files containing storage data, memory pages and CPU states at some time. A traditional snapshot at time t is defined as $SN_t = D \cup M \cup R$, where D represents the storage data, M represents the memory pages and R represents the CPU state. An incremental snapshot means the differences between the current and the former ones. So an incremental snapshot created at time t_i is defined as $sn_{t_i} = SN_{t_i} - SN_{t_{i-1}}$, and all the states of a VM at time t_i is $SN_{t_n} = \bigcup_{t_0}^{t_n} sn_{t_i}$.

3.2 The Prediction Model for Snapshot Size Growth

The growth of the size of a snapshot can be modeled as a time series and we try to find a prediction model to predict its future trend. We adopt Markov Chain as the prediction model.

Markov Chain & Transition Matrix. A Markov Chain is a mathematical system that undergoes transitions from one state to another on a state space [15]. It is a random process that the next state depends on the current one. The growth curve of snapshot size is a time series with some regular characteristic (Fig. 1 to Fig. 4).

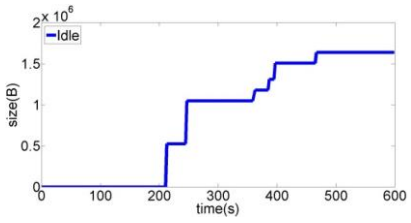

Fig. 1. No extra program on VM.

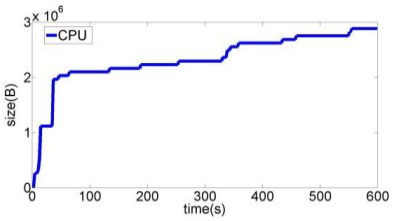

Fig. 2. CPU intensive program on VM.

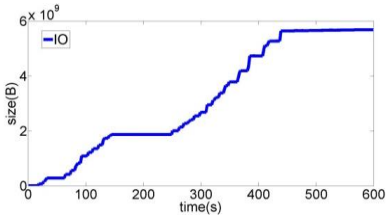

Fig. 3. IO intensive program on VM.

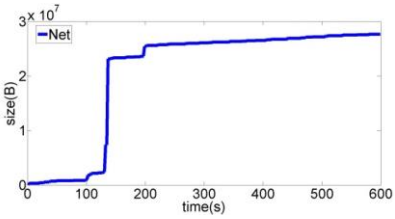

Fig. 4. Network intensive program on VM.

In order to analyze and forecast the increasing curve, we set an *n-sized* window to capture the continuous discrete states of *n* as a status (Fig. 5). Each state in an *n-sized* window represents a size of an incremental snapshot in a time slot, and the *n* states compose a status, which is the base unit in our model. Optimal value of *n* depends on the learning data and the migration platform. The optimal value we set in experiments will be detailed in the evaluation section.

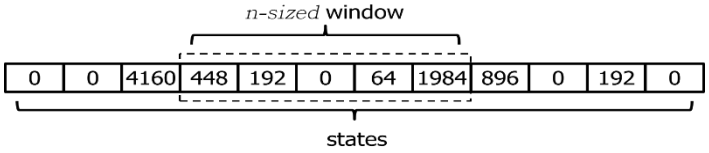

Fig. 5. N-sized window.

Step 1. We extract patterns using *n-sized* window and build transition matrix using Markov Chain. Patterns are some typical snapshots growth sub-sequences, each of which represents a cluster of original growth curves. We extract the patterns from the historical data using a pattern fusion model which is based on Euclidean distance [14]. Then we make up the pattern set, P=$\{P_1, P_2, \ldots\ldots, P_N\}$, where N is the number of patterns. We define pattern $P_i = \{s_1, s_2, \ldots\ldots, s_n\}$, in which s_i is a single state representing the size of an incremental snapshot. The length of pattern P_i is determined by the size of the window. The transition matrix is defined as M = $\begin{pmatrix} p_{11} & \cdots & p_{1N} \\ \vdots & \ddots & \vdots \\ p_{N1} & \cdots & p_{NN} \end{pmatrix}$, which stores all the transition probabilities. In the matrix, the rows R = $\{R_1, R_2, \ldots\ldots, R_N\}$ represent the current statuses while the columns C = $\{C_1, C_2, \ldots\ldots, C_N\}$ represent the following one. So each value in the transition matrix means a probability from one status to the successor. For instance, the i_{th} row in the transition matrix is $R_i = \{p_{i1}, p_{i2}, p_{i3}, \ldots\ldots, p_{iN}\}$, where $p_{ij} = Probability: P_i \rightarrow P_j$.

Step 2. In this step we formalize the prediction process based on the transition matrix M. The growth of the snapshot size can be represented as L = $\{s_1, s_2, \ldots\ldots, s_l\}$, each $s_i (1 \leq i \leq l)$ is a size increment while the curve L represents the snapshot growing from t_1 to t_l. The latest status is $S_i = \{s_{l-n+1}, s_{l-n+2}, \ldots\ldots, s_{l-1}, s_l\}$. The best matched pattern P_{best} will be found according to S_i, where P_{best} is a pattern P_j that meets such condition $\min\limits_{1 \leq j \leq N} \{Dist(S_i, P_j)\}$. Here, we use Euclidean distance to calculate $Dist(S_i, P_j)$. Then we will forecast the next status $S_{i+1} = \{s_{l-n+2}, s_{l-n+3}, \ldots\ldots, s_l, s_{l+1}\}$ according to P_{best} (Fig. 6). The status S_{i+1} is a status that satisfies the condition $\max\limits_{1 \leq j \leq N} \{M[P_{best}][S_j]\}$. After that we get the new curve $L' =$

$\{s_1, s_2, \ldots\ldots s_l, s_{l+1}\}$, where the state s_{l+1} is what we predict.

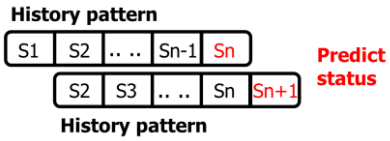

Fig. 6. Iterative prediction process.

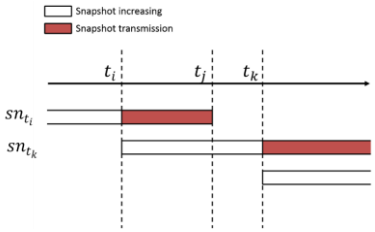

Fig. 7. Part of migration process.

So far, we make a prediction. We can repeat the predictions to obtain a long future curve $L'' = \{s_1, s_2, \ldots\ldots, s_l, s_{l+1}, \ldots\ldots, s_m, s_{m+1} \ldots\ldots\}$.

We find the curve sometimes go steep and sometimes go slow, so we could perform the last transmission during slow segment. Therefore, we need to identify these segments, which we call them growth platforms. We define $d_{ij} = \sum_i^j s_k$ $(i \leq k \leq j)$, meaning the whole size increment from time i to j. Given a length of period m and a curve L'' with length n, a segment L_{ij} that meets the condition $min\{d_{ij}, (j - i = m, 1 \leq i, j \leq n)\}$ is the growth platform L^* of L''.

4 Prediction-based Migration Strategy

During migration the efficiency depends on the snapshots' sizes with a given bandwidth. We define an increasing curve of a snapshot as $\mathrm{L} = \{s_{t_0}, s_{t_1}, \ldots\ldots, s_{t_n}\}$, which represents the growing size of a snapshot. The element s_{t_i} in L is the size of sn_{t_i}: an incremental snapshot at t_i. Part of the migration is as follows (Fig. 7).

We consider the process starts at time t_i with sn_{t_i} and a given stable bandwidth $\mathcal{B}$. Snapshot sn_{t_i} is created at t_i. Let $\Delta \mathrm{t} = \frac{s_{t_i}}{\mathcal{B}}$ and $t_j = t_i + \Delta t$, snapshot sn_{t_i} starts being transmitted at t_i and completes at t_j. At the same time, the VM keeps running. Thus, at time t_k $(j \leq k)$, the next snapshot sn_{t_k} will be transmitted. And so forth, snapshots are transmitted to the destination host until the VM shuts down.

4.1 Feedback-based Migration Strategy

Based on the prediction model described above, we propose a VM migration strategy: feedback based migration (FM) strategy. It is mainly composed of four steps. First, to transmit the base image and forecast a snapshot increasing curve using the prediction model. Second, to capture the time when the incremental snapshot is the smallest. Third, to adjust the predicted curve according to real-time feedback. Finally, to shut down the VM and synchronize the status when it reaches the time we predicted.

Predicting snapshot increasing curve is described in section 3, this section would describe the snapshots transmission process. Given a snapshot size growth curve and a bandwidth, a period that the smallest incremental snapshot is generated could be captured using depth-first search and greedy algorithm, which is described here.

Algorithm 1. Feedback-based migration algorithm

```
Input : a snapshot size growth curve p_list and a base_size
Output : finish_t, the proper point to shut down the VM
FindFinishTime (p_list, base_size)
begin
  min_size = MAX  finish_t = 0
  DFFind(p_list, base_size, 0)
  return finish_t
end

DFFind (p_list, base_size, start_t)
begin
  if(base_size == 0)
    min_size = 0  tf = start_t
    return
  current_size = base_size
  while current_size not reach finish time
    update next_t and next_size
    DFFind(p_list, sub_size, start_t + next_t)
    if(min_size > next_size)
      min_size = next_size  finish_t = start_t + next_t
end
```

The algorithm FindFinishTime (FFT) would find the finish time of the migration with $O(n^2)$ time. Every time an incremental snapshot is transmitted, a predicted curve and a real-time would be compared. If the two curves match, the migration will work as predicted. Otherwise, a new predicted curve would be made and another finish time would be calculated. FM strategy works efficiently if the prediction is accurate. But when the predicted curve deviates from the actual curve, the finish time should be calculated every time a snapshot is transmitted. Thus, the efficiency would be lower. And an enhanced strategy is proposed below.

4.2 Adjustment-based Migration Strategy

We enhance the former strategy by adding the adjustment factors during prediction and propose another strategy: adjustment based prediction (AM) strategy.

Algorithm 2. Adjustment-based migration algorithm

```
Input : markov_matrix, a base_size
Output : p_list, the predicted curve
Predict(markov_matrix, base_size)
begin
  current_size = base_size  p_list = null  pattern_time = 0
  build history_list from current
  while not reach finish time
    pattern = getPattern(history_list)
    if pattern_time > threshold
      pattern = getFollowPattern(history_list)
    next_status = predictNextStatus(markov_matrix)
    update p_list and history_list
    if pattern equals next_pattern
      pattern_time ++
    pattern = next_pattern
  return p_list
end
```

Every time we make prediction, the times of continuously repeated patterns is recorded. Once the time exceeds the threshold (one single pattern repeats for more

than m times, which will be detailed in evaluation), we get the second popular status as the next status instead of the most popular one. The complexity of the algorithm is $O(n^2)$, and the length of the increasing curve is n. The algorithm improves the prediction efficiency, and the transmission is the same as FM strategy.

5 Experiments

In this section, we present an evaluation of our prediction-based migration. The experimental platform we used is built between SJTU, China and UFL, USA. We use KVM as the virtualization layer and lib-virt as the control layer.

We extract a pattern set through learning from history data. In Section 3, we know that the length of a pattern will affect the migration. In Fig. 8, we choose different lengths of patterns to compare the prediction accuracy and the efficiency. Finally, we select 50 as the pattern length according to our experiment.

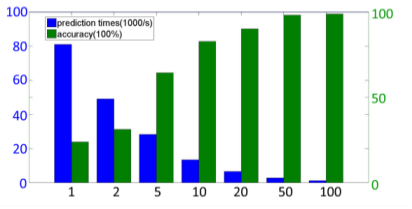

Fig. 8. Pattern length experiment.

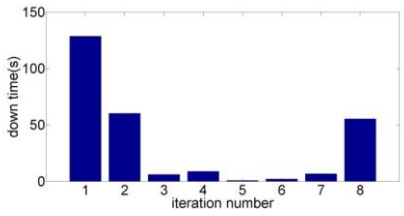

Fig. 9. Down time on iteration number.

Considering migration over WAN, the strategy pure stop-copy (PSC) and the strategy fixed number iterations (FNI) are suitable. We find that the PSC strategy can minimize the whole migration time while its down time is long. The FNI strategy can reduce the down time, but it depends on the iteration numbers (Fig. 9).

The FNI strategy cannot detect the size of snapshot automatically. Fig. 9 reveals that snapshots become smallest during the 5^{th} iteration. The FNI strategy cannot minimize the down time, since the iterations number is fixed. Compared with PSC and FNI strategies, FM and AM strategies can minimize the down time. We evaluate the performance of FM and AM strategies, compared with PSC and FNI strategies. In Fig. 10 to Fig. 13, we analyze the performance with different types of snapshots.

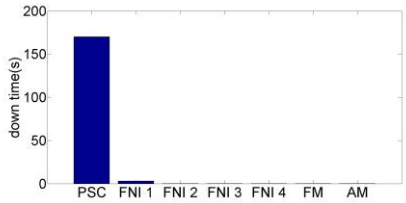

Fig. 10. Result on CPU intensive program.

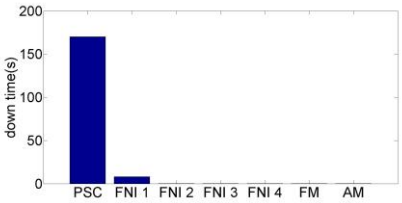

Fig. 11. Result on memory intensive program.

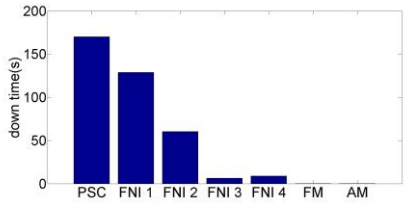

Fig. 12. Result on network intensive program.

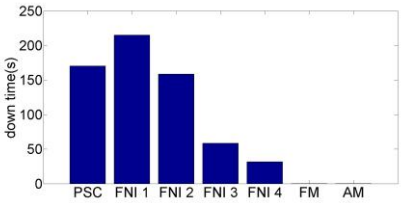

Fig. 13. Result on IO intensive program.

From the figures above, we know that the network and IO intensive snapshots reflect the real performance about our strategy. What's more, we consider the size of the base image as a factor in our evaluation. In Fig. 14 and Fig. 15, we give a performance comparison using FM and AM strategies with PSC and FNI strategies.

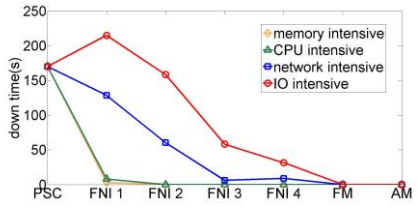

Fig. 14. Down time with 500M base image.

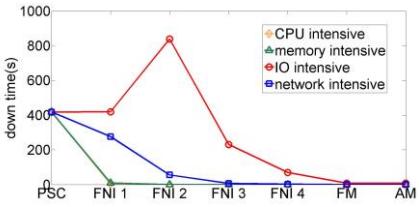

Fig. 15. Down time with 1G base image.

VMs in evaluation run different programs, including CPU intensive, memory intensive, network intensive and IO intensive programs. In addition, two sizes of base image are considered. In evaluation, four migration strategies are taken, and the migration iterations from 1 to 4 are selected in FNI strategy. PSC strategy always produces a constant down time and the down time varies for FNI strategy. We can see

that FM and AM work well and stably in all cases with almost zero down time.

As mentioned above, FM strategy is less efficient than AM strategy whenever the predicted curve deviates from the actual curve. AM strategy could adjust the predicted result so that the predicted curve matches the actual curve better. Here, we set m=5 as the threshold to avoid patterns repetition considering the snapshots size and bandwidth in our evaluation platform. The prediction times of AM strategy is fewer, and the effective prediction ratio is higher. Effective Prediction Ratio is defined as EPR $= N_{correct}/N_{total}$, where $N_{correct}$ is the times of correct prediction and N_{total} is the times of total prediction. In fig.16, it is indicated that the EPR of AM strategy is higher in different types of VMs and it is 21.1% higher than FM strategy overall.

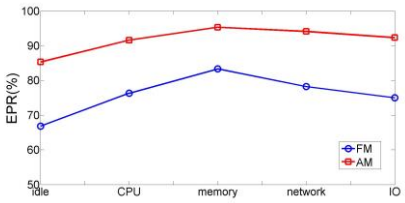

Fig. 16. EPR comparison between FM and AM strategies.

6 Conclusion and Future Work

In this paper, for optimizing VM migration over WAN, we propose a prediction-based strategy which can forecast the increasing curve of snapshots about VMs. Our main contribution is to predict the increments of VM snapshot and select the proper segment to shut down the VM which minimizes the VM down time. Compared with two migration strategies, the evaluation shows that our PB strategy works well and stably during migration, which minimizes the down time among all the strategies.

In the future, there are two parts of work we can focus on. First, more migration metrics can be considered like the whole migration time and the bandwidth limitation. Second, we could split the snapshot finer, such as dirty page in memory and storage.

Acknowledgement

This work is partially supported by China National Science Foundation (Granted Number

61073021, 61272438), Research Funds of Science and Technology Commission of Shanghai Municipality (Granted Number 14511107702, 12511502704).

References

1. Christopher Clark, Keir Fraser, Steven Hand, Jacob Gorm Hansen, Eric Jul, Christian Limpach, Ian Pratt, Andrew Warfield:Live Migration of Virtual Machines. NSDI 2005
2. Khaled Z. Ibrahim, Steven A. Hofmeyr, Costin Iancu, Eric Roman: Optimized pre-copy live migration for memory intensive applications. SC 2011: 40
3. Ma F, Liu F, Liu Z: Live virtual machine migration based on improved pre-copy approach. ICSESS 2010: 230-233
4. Michael R. Hines, Umesh Deshpande, Kartik Gopalan: Post-copy live migration of virtual machines. Operating Systems Review 43(3): 14-26 (2009)
5. M. Kozuch and M. Satyanarayanan: Internet suspend/resume. In Proc. IEEE Workshop on Mobile Computing Systems and Applications, Washington, DC, USA, 2002, pp. 40–46
6. C. P. Sapuntzakis, R. Chandra, B. Pfaff, J. Chow and M. S. Lam: Optimizing the migration of virtual computers. ACM SIGOPS OSDI 2002, pp. 377–390, 2002
7. Andrew Whitaker, Richard S. Cox, Marianne Shaw, and Steven D. Gribble: Constructing Services with Interposable Virtual Hardware. NSDI 2004: 169-182
8. Michael R. Hines, Kartik Gopalan: Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. VEE 2009: 51-60
9. M. Nelson, B.H. Lim, and G. Hutchins: Fast Transparent Migration for Virtual Machines. Proc. USENIX Ann. Technical Conf., pp. 391-394, Apr. 2005
10. M. Nelson, B.H. Lim, and G. Hutchins: Fast Transparent Migration for Virtual Machines. Proc. USENIX Ann. Technical Conf., pp. 391-394, Apr. 2005
11. H. Liu, H. Jin, X Liao: Live Virtual Machine Migration via Asynchronous Replication and State Synchronization. IEEE Trans. Parallel Distrib. Syst. 22(12): 1986-1999 (2011)
12. Dan Williams, Hani Jamjoom, Hakim Weatherspoon: The Xen-Blanket: virtualize once, run everywhere. EuroSys 2012: 113-126
13. T. Wood, K. K. Ramakrishnan, Prashant J. Shenoy: CloudNet: dynamic pooling of cloud resources by live WAN migration of virtual machines. VEE 2011: 121-132
14. Yang D, Cao J, Fu J: A pattern fusion model for multi-step-ahead CPU load prediction. Journal of Systems and Software, 2013, 86(5): 1257-1266
15. Geyer C J.: Practical markov chain monte carlo. Statistical Science, 1992: 473-483