

IPAM: An Improved Penalty Aware VM Migration Technique

Garima Singh[1(✉)], Ruchi Tailwal[2], Meenakshi Kathayat[2], and Anil Kumar Singh[1]

[1] Motilal Nehru National Institute of Technology Allahabad, Prayagraj, UP, India
singhgarima4688@gmail.com, ak@mnnit.ac.in
[2] Govind Ballabh Pant Institute of Engineering and Technology, Pauri, UK, India

Abstract. In today's era, more and more businesses are moving on the cloud. The customer needs to be assured of the quality-of-service they desire from the cloud service provider. Service level agreement (SLA) is a significant medium for building this confidence. The services are often running on virtual machines (VMs) provided by service provider. Various cloud management activities require transferring of VMs among the servers. They make use of live migration to achieve these objectives. With multiple VMs being migrated, it is essential to assign the available bandwidth among the VMs to minimize the penalty incurred due to SLA violation during migration. However, depending upon the assigned transfer rates, some VMs in the group might complete the migration process earlier than the others. In this paper, we propose an algorithm and a controller to assign freed bandwidth to the remaining VMs in the group. With extensive simulations, it is found that the penalty, migration time, and downtime for VMs is significantly reduced compared to the state-of-the-art approaches.

Keywords: VM migration · Service Level Agreement (SLA) · SLA violation · Cloud service provider

1 Introduction

Cloud computing has surfaced as a major paradigm in the field of IT. It is not just because of the web services and high-power data centers. The onus of this shift goes to virtualization, which is the key enabling technology behind the cloud. One of the salient feature offered by virtualization is VM migration, allowing transfer of virtual machine (VM) between physical servers without any change to the physical infrastructure.

VM migration is the core technology for key management activities in the cloud, such as load balancing, fault tolerance, maintenance, etc. It can be accomplished in a live or non-live manner. In non-live/cold migration, the VM is first stopped, and then the VM memory is transferred to the destination host. Once the transfer is complete, the VM is resumed at the destination. With live/hot

© The Author(s), under exclusive license to Springer Nature Switzerland AG 2022
O. Gervasi et al. (Eds.): ICCSA 2022, LNCS 13375, pp. 324–336, 2022.
https://doi.org/10.1007/978-3-031-10522-7_23

migration, the VM is running while the migration takes place. It has a short downtime during which the services running on the VM remains unavailable and is hardly noticed by the user.

If service level agreement (SLA) [8] regulates the availability of services on the VM, the overheads due to VM migration can not be disregarded [13]. Whenever VM is migrated, downtime is inevitable. During the downtime, the services hosted on the VM will remain inaccessible. Depending on the price paid for the SLA satisfaction, for the same downtime, different penalties would be charged for different VMs [11].

This work is an extension of the penalty aware VM migration (PAM) technique [11], which tries to minimize the penalty incurred while migrating multiple VMs. The initial transfer rates are assigned to VMs using PAM for the proposed algorithm. However, depending upon the assigned transfer rate, it is possible that some of the VM might complete the migration earlier than the other VMs in the same group, and the bandwidth that was allocated to VMs which completed the migration will be freed. If there are no new tasks that require bandwidth, this freed bandwidth can be allocated to the remaining VMs in the group which has not completed the migration yet. The main contributions of this work are:

- An algorithm to assign additional bandwidths to VMs that are still in the pre-copy phase while some of the VMs have completed the migration process.
- By allocating additional bandwidth to the remaining VMs, the bandwidth utilization is improved, and the total penalty charged is reduced.
- Simulations are conducted to analyze the effect of various parameters on performance matrices for the proposed approach.

The remaining paper is arranged as follows. Section 2 discusses VM migration techniques and related work. A brief description of a model for SLA aware migration is discussed in Subsect. 2.3. Section 3 discusses the proposed algorithm to improve the performance of penalty aware migration. The experimental results for various performance matrices is discussed in Sect. 4. Finally, Sect. 5 presents the conclusion.

2 Background

2.1 VM Migration Techniques

VM migration leverages process migration and tries to solve the problem of residual dependencies faced by the latter. It can be carried out in a live/non-live manner. As the downtime and consequently service un-availability duration is more with non-live migration, the research community focuses more on live VM migration. So, in this work, we will be discussing live migration techniques only.

Live VM migration can be further carried out in pre-copy, post-copy, or hybrid manner. Any live VM migration technique consists of a *push* and/or *pull* phase with a *stop-and-copy* phase. With *pre-copy* [2], there is a *push* phase followed by a small *stop-and-copy* phase. During the *push* phase, the VM memory

data is transferred from the source host to the destination host in iterations. Each subsequent iteration transfers the memory that has been dirtied in the previous round. When some terminating condition is reached, the VM is stopped, and the remaining dirty pages are transferred, after which the VM is resumed on the destination host.

Post-copy [5] migration first executes the *stop-and-copy* phase followed by the *pull* phase. During *stop-and-copy* phase, a small amount of data containing the necessary information to start the VM is copied from the source to the destination host, while the VM is paused at the source host. Once the data is copied, the VM is resumed at the destination host. The remaining pages are copied from the source host during the *pull* phase. *Hybrid* migration is a blend of *pre-copy* and *post-copy* migration technique. There are a limited number of iterative rounds as *pre-copy* migration followed by a *stop-and-copy* phase, after which VM is resumed at the destination host. The remaining pages are transferred during the *pull* phase as in *post-copy* migration.

2.2 Related Work

Pre-copy migration is more robust in comparison to post-copy and hybrid migration. That is the reason why majority of the hypervisors like XEN and VMware uses pre-copy to migrate the VMs. A great amount of research focused on improving the performance of pre-copy migration, and this subsection aims at discussing some of them.

Compression is one technique that can be used to reduce the network traffic during live migration. Data compression exploits data regularities to enhance the capability of pre-copy migration. Jin et al. proposed *MECOM* [6] that adaptively compresses the pages of the VM using different compression techniques depending upon the similarity ratio of the page. The work in [10] proposes to optimize the total downtime of multiple migrating VMs by allocating transfer rates and compression rates to VMs depending upon the available bandwidth. To shorten the VM migration duration and total data transfer on the network, many techniques try to avoid sending duplicate pages while migrating VMs by checking for inter and intra VM similarity [3,9].

Migrating multiple VMs is a strenuous task compared to single VM migration and requires careful planning. The migration resources need to be allocated among the migrating VMs in an intelligent manner, and a proper migration schedule needs to be determined to reduce the impact of migration. Cerroni and Esposito [1] proposed a transfer rate assigning scheme for multiple VMs to optimize the total migration time and total downtime. Similarly, Liu and He proposed *VMBuddies* [1] for migrating multi-tier applications over WAN. They proposed an arbitrator for assigning bandwidth to VMs so as to limit the service interruption time. It ensures that all VMs finishes migration nearly at the same time.

Whenever VMs are migrated, there might be a possibility of SLA violation. Many works have considered the placement and consolidation of VMs to minimize the chances of SLA violation. An economically enhanced resource manager

(EERM) is proposed by Macias et al. [7] to escalate the returns of the service provider. Tsakalozos et al. [12] proposed a time-constrained live migration technique to minimize the possibility of SLA violation and service degradation experienced by the customer.

2.3 A Model for SLA Aware VM Migration

Before discussing IPAM - an improved penalty aware migration technique, it is essential to discuss modeling live migration in the light of SLA. Although it is already mentioned in the previous works [11], this section discusses penalty aware migration in a nutshell for a better comprehension of the proposed strategy.

Assume VM_j has memory size, dirty rate, and transfer rate as M_j, d_j, and r_j respectively. The size of dirty memory generated during the ith pre-copy round ($V_{i,j}$) and the time span of the ith pre-copy round ($t_{i,j}$) can be computed for VM_j using Eq. 1.

$$V_{i,j} = \begin{cases} M_j, & \text{if } i = 0; \\ d_j t_{i-1,j}\,, & \text{else} \end{cases} \quad t_{i,j} = \begin{cases} \frac{M_j}{r_j}, & \text{if } i = 0; \\ \frac{d_j t_{i-1,j}}{r_j}, & \text{else} \end{cases} \tag{1}$$

Assume for VM_j, ρ_j is the ratio of memory dirty rate (d_j) and transfer rate (r_j), i.e., $\rho_j = d_j/r_j$. With pre-copy migration, the dirty memory transferred during the different pre-copy iterations and the duration of each iteration can be enumerated as follows:

Iteration 0: $V_{0,j} = M_j$ and $t_{0,j} = \frac{M_j}{r_j}$
Iteration 1: $V_{1,j} = d_j t_{0,j} = M_j \rho_j$ and $t_{1,j} = \frac{V_{1,j}}{r_j} = \frac{M_j \rho_j}{r_j}$
Iteration 2: $V_{2,j} = d_j t_{1,j} = M_j \rho_j^2$ and $t_{2,j} = \frac{V_{2,j}}{r_j} = \frac{M_j \rho_j^2}{r_j}$

$$Finally : V_{i,j} = d_j t_{i-1,j} = M_j \rho_j^i \text{ and } t_{i,j} = \frac{V_{i,j}}{r_j} = \frac{M_j \rho_j^i}{r_j} \tag{2}$$

The migration time (TM_j) and downtime (TD_j) for VM_j with n number of pre-copy iterations, can be calculated using the following equation.

$$TM_j = \sum_{i=0}^{n} t_{i,j} = \frac{M_j}{r_j} \sum_{i=0}^{n} \rho_j^i \quad \text{and} \quad TD_j = \frac{M_j \rho_j^n}{r_j} \tag{3}$$

To remove the complexities in modeling the problem, it is presumed that the VMs start migrating concurrently. So the total migration time is the largest of the migration time of m VMs, i.e., $TMT = max\ \{\ TM_j \mid 1 \leq j \leq m\ \}$

Total downtime(TDT) determines the service inaccessibility duration for m VMs during migration and can be calculated as

$$TDT = \sum_{j=1}^{m} DT_j \tag{4}$$

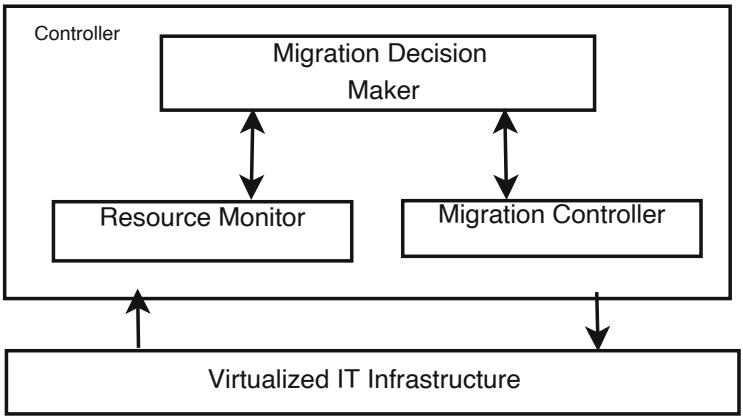

Fig. 1. Controller modules

The penalty incurred for migrating VM_j with $Price_j$ paid by the customer for it's SLA satisfaction is assumed to be Pen_j. Pen_j follows the properties of a Gompertz curve [14]. For this work, if the downtime is less than a threshold TD_{min}, no penalty is incurred. The value of Pen_j is computed as follows.

$$Pen_j = \begin{cases} 0, & \text{if } TD_j \leq TD_{min}; \\ \dfrac{75}{100} e^{-e^{-100\frac{M_j \rho_j^n}{r_j} + \frac{e}{2}}} Price_j, & \text{otherwise.} \end{cases} \tag{5}$$

Further, the total penalty (TP) for migrating m VMs is:

$$TP(r_1, r_2, ..., r_m) = \{Pen_1 + Pen_2 + ... + Pen_m\} \tag{6}$$

Finally, the optimization problem that need to be solved is as given below:

$$\min_{r_1, r_2, ..., r_m} TP(r_1, r_2, ..., r_m) \tag{7}$$

such that

$$r_1 + r_2 + ... + r_m \leq B \tag{8a}$$

$$0 < r_j \leq B \tag{8b}$$

$$0 < d_j < MinRate \leq r_j \tag{8c}$$

Penalty aware migration uses Lagrange multiplier [4] for allocating transfer rates to VMs such that the objective function given in Eq. 7 is minimized, and the constrains described using Eqs. 8a–8c are satisfied. Here, $MinRate$ is the minimum guaranteed transfer rate for migrating each VM and is set to 30 Mbps in this works.

Algorithm 1: Re-computeTxRate

Input: m, i, $Rate$, $Finish$, α

1 $Finish[i] \leftarrow 1$ ▷ VM_i has completed the migration

2 $Num_Remaining_VMs \leftarrow m - \sum_{k=1}^{M} Finish[k]$

3 $Free_BW \leftarrow Rate[i]$ ▷ Bandwidth freed by VM_i

4 $Temp_BW \leftarrow \alpha * \frac{Free_BW}{Num_Remaining_VMs}$ ▷ Extra Bandwidth to be allocated to remaining migrating VMs

5 **foreach** VM_j **do**

6 **if** $Finish[j] \neq 1$ **then**

7 *Find min k s.t.* $\sum_{i=0}^{k} t_{i,j} >= TM_i + \tau$

8 $r_j = r_j + Temp_BW$

9 Migrate VM_j with the new transfer rate r_j for round k onwards

3 Improved Penalty Aware VM Migration

Here, we put forward an algorithm to enhance the functioning of penalty aware migration, as discussed in the previous subsection. We propose a controller, as shown in Fig. 1 to regulate the allocation of transfer rates to migrating VMs depending upon the network bandwidth availability.

When a request comes to migrate multiple VMs in parallel, the migration decision maker module uses PAM to assign the initial transfer rates to VMs. The resource monitor module periodically monitors the VM performance and bandwidth availability. As some of the VMs complete migration, the resource monitor module informs the migration decision-maker about the freed bandwidth. The migration decision-maker then determines if additional bandwidth can be allocated to the remaining VMs or not.

When VM_i is migrated successfully, the migration controller evokes the procedure $Re - computeTxRate$ as shown in Algorithm 1. Here, $Finish[i]$ denotes if the VM_i has successfully migrated or not. Initially, $Finish[i]$ is set to 0 $\forall$i $\in \{1, 2, ..m\}$. When VM_i successfully completes the migration, the algorithm sets the value of $Finish[i]$ to 1 (step 1). Next, we estimate the bandwidth allocated to VM_i and the number of VMs which have not completed the migration, as shown in step 2 and 3. Assume α be the fraction of the bandwidth that the migration controller decided to assign to the remaining VMs. The additional bandwidth that can be assigned to the remaining migrating VMs are migrated is computed in step 4. Finally, step-5 to 8 assign the additional bandwidth as computed in step 4 to VMs. For simplicity, we assume that in case more bandwidth is available during the middle of pre-copy transfer round, the VM does not immediately switch to the new transfer rate. Rather, the transfer rate is updated in the subsequent rounds. Moreover, we assume that some constant time τ is required to switch to this new bandwidth. For experimental purposes, the value of τ is set to 1 sec in this work.

Table 1. Experimental parameters

Parameter	Value
Max bandwidth for migration (B)	1 Gbps
No. of pre-copy iterations (n)	10
Memory size of VM_1, VM_2 and VM_3	[1, 1, 1] GB
Average memory dirtying rate for 3 VM's	[3000, 3000, 3000] pps
Price for VM_1, VM_2, and VM_3	[1000, 500, 100]
Min. transfer rate ($MinRate$)	30 MBps

Allocating a portion of freed bandwidth to the VMs which are still migrating decreases the penalty, downtime, and migration time. However, it is possible that the migration controller decides not to assign the freed bandwidth to the remaining VMs, in such a case, the value of α will be 0 and the performance of IPAM will be the same as of PAM.

4 Experimental Results

Here we analyze the functioning of the following three approaches:

- Downtime Aware Migration (DTAM): This migration scheme tries to reduce the total downtime by efficient allocation of transfer rate as discussed in [1].
- Penalty Aware Migration (PAM): This approach tries to minimize the total penalty charged while migrating VMs as discussed in [11].
- Improved Penalty Aware Migration (IPAM): The suggested migration scheme to improve the performance of PAM.

The working of DTAM, PAM, and IPAM is analyzed in terms of penalty, migration time, and downtime experienced by the VMs during migration. The simulation parameters used in this paper are as in Table 1 until stated explicitly. Three VMs of 1 GB size each with price as 1000, 500, and 100 units are migrated under different environments. The value of TD_{min} is 1 nanosecond. So, no penalty would be charged for migrating the VM if the downtime is less than or equal to 1 nanosecond. The value of α is 1/3 for Algorithm 1. Hence once a VM completes the migration, only one-third of the bandwidth assigned to that VM can be allocated to the remaining VMs. All the simulations are conducted in Matlab. To find the initial transfer rate for the optimization problem, *fmincon* and *globalsearch* functions provided in Matlab are used.

4.1 Effect of Pre-copy Rounds

This subsection studies the effect of pre-copy iterations on various parameters. Here all the 3 VMs have memory size and dirty rate fixed to 1 GB and

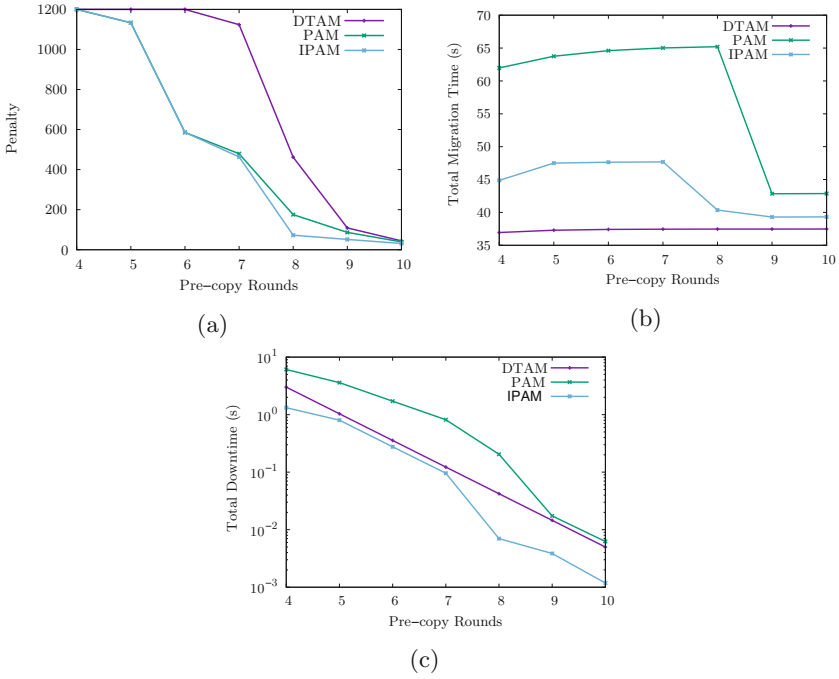

Fig. 2. Effect of pre-copy transfer rounds on different parameters for migrating 3 VMs with memory size and dirty rate as 1 GB and 3000 pages per second respectively. The maximum bandwidth for migration is set to 1 GB while the pre-copy rounds are varied from 4 to 10.

3000 pps[1], respectively. The maximum bandwidth available for migration is 1 Gbps, and the pre-copy iterations are varied from 4 to 10.

Figure 2a shows the penalty charged by the three approaches. With a small number of pre-copy iterations (4–6), the penalty charged by PAM and IPAM is almost the same. This is because, with a smaller number of rounds, IPAM can not reduce the downtime time of remaining VMs sufficiently. Further, as the number of rounds is increased from 6 to 9, the penalty charged by IPAM is found to be less compared to PAM. With IPAM, once a VM completes migration, the bandwidth allocated to that VM is assigned to the remaining migrating VMs, which reduces the penalty for the remaining VMs. Moreover, when the pre-copy round is 10, DTAM, PAM, and IPAM approximately impose the same penalty. All the approaches transfer the VMs with sufficiently small downtime resulting in a small penalty.

Figure 2b shows the total migration time for the different numbers of pre-copy rounds. The total migration time for IPAM is less compared to PAM but higher than DTAM. With IPAM, the VM with less price (VM_2 and VM_3) are migrated

[1] PPS is page dirtied per second. The size of each page is assumed to be 4 KB.

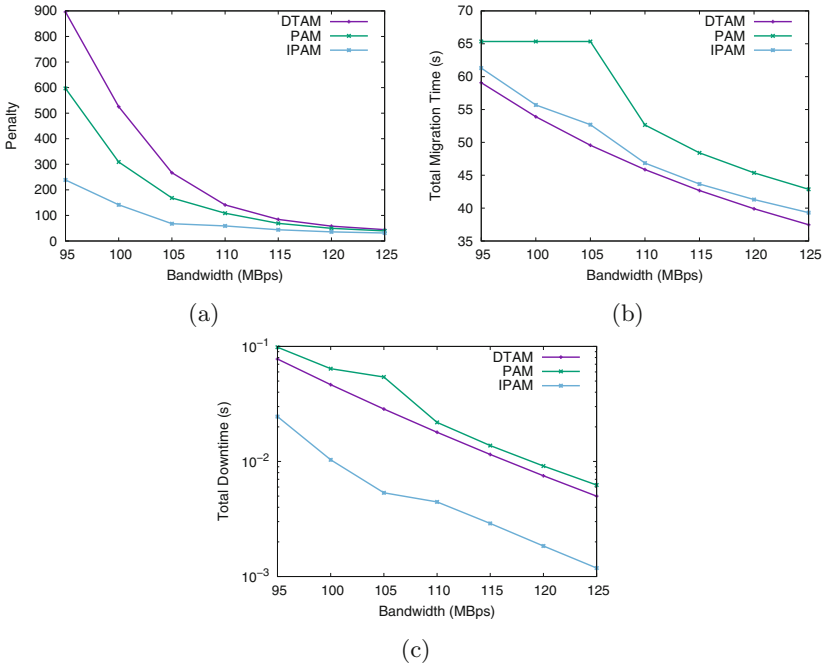

Fig. 3. Effect of available bandwidth for migration on different parameters for migrating 3 VMs with memory size and dirty rate as 1 GB and 3000 pages per second respectively. The number of pre-copy iterations is fixed to 10 while the maximum bandwidth available for migration is varied from 95 MBps to 125 MBps.

with less bandwidth during the initial pre-copy rounds. This results in a longer migration duration for the initial rounds. However, as VM_1complete migration, the bandwidth assigned to it is available for the remaining VMs. IPAM assigns this freed bandwidth to VM_2 and VM_3 for the remaining pre-copy rounds. This results in a reduction in the migration time for VM_2 and VM_3 for the remaining pre-copy rounds.

Figure 2c shows that the total downtime for different number of pre-copy rounds. It can be seen that the total downtime for IPAM is less compared to DTAM and PAM because assigning more bandwidth to VM in the last round can significantly reduce its downtime and, consequently the total downtime will be reduced. In case multiple VMs are to be migrated and the controller decides to assign additional bandwidth to the remaining VMs, IPAM can migrate VMs with less total penalty, total migration time, and total downtime in comparison to PAM.

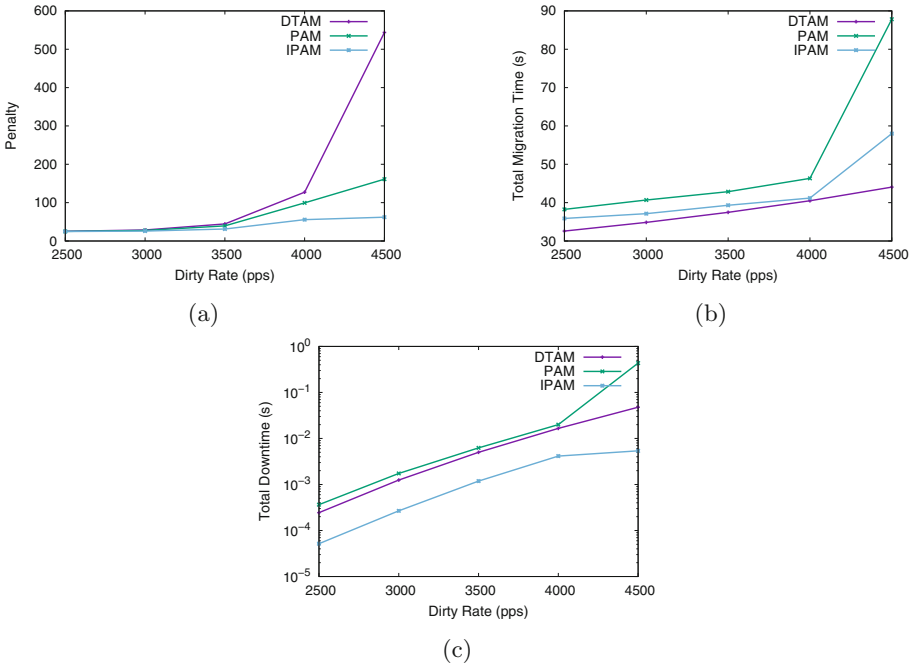

Fig. 4. Effect of dirty rate on different parameters for migrating 3 VMs with memory size as 1 GB. The number of pre-copy iterations and maximum bandwidth available for migration is fixed to 10 and 1 Gbps respectively. The average dirty rate is varied from 2500 pps to 4500 pps.

4.2 Effect of Available Bandwidth

Here, the simulation environment is the same as in Sect. 4.1 except for the number of pre-copy rounds, which is fixed to 10, while the bandwidth available for migration is varied from 95 MBps to 125 MBps.

Figure 3a presents the penalty imposed for transferring 3 VMs with different available bandwidths. It can be observed that for the same migration bandwidth, the penalty imposed by IPAM is less followed by PAM and DTAM. It is self-evident that with an increase in the bandwidth, the penalty is reduced. When more bandwidth is available for migration (125 MBps), the penalty imposed by all the approaches is approximately the same. This is because adequate bandwidth was allocated to VMs to transfer them with a small penalty. However, when the bandwidth available is reduced, the penalty imposed by IPAM was found to be less compared to PAM and DTAM. With less bandwidth, PAM allocated more bandwidth to VM with a higher price. The VMs with less price were migrated with low bandwidth, resulting in a surge in the penalty for such VMs. With IPAM, once few VMs finish the migration, a fraction of bandwidth allocated to them can be assigned to the remaining VMs for the further left pre-copy rounds.

Figure 3b shows the total migration time for different available bandwidths. The total migration time is highest for PAM followed by IPAM and DTAM. In the initial pre-copy rounds, VM_2 and VM_3 were migrated with less bandwidth which prolonged the duration of those rounds. However, as VM_1 completed the migration, a portion of bandwidth allocated to it was assigned to the VM_2 and VM_3, which reduced the duration of later pre-copy rounds resulting in an overall reduction in the migration time.

Figure 3c shows the total downtime for different available bandwidths. The penalty charged by IPAM is the lowest among the three approaches for the same bandwidth. This is again because of assigning some additional bandwidth to VM_2 and VM_3 in the last pre-copy rounds, which lowers the downtime for VM_2 and VM_3. Due to this, the overall downtime also drops. In case VMs need to be migrated with less bandwidth, and the migration scheduler decides to allocate the freed bandwidth to the remaining migrating VMs, IPAM can efficiently migrate them with a significantly reduced penalty.

4.3 Effect of Dirty Rate

The impact of memory dirty rate on different parameters is shown in Fig 4. Here, all the simulation parameters are the same as in Sect. 4.1 except that the number of the pre-copy rounds is fixed to 10, and the average rate is varied from 2500 pps to 4500 pps.

With a low dirty rate of 2500–3500 pps, the penalty imposed by PAM, IPAM, and DTAM is almost the same as shown in Fig. 4a. This is because all three approaches were able to migrate the VMs with sufficiently small downtime. This resulted in a significant reduction in the total penalty for all three. But as the dirty rate is increased beyond 3500 pps, the penalty imposed by IPAM was found to be less in contrast to PAM and DTAM. This was again because of assigning additional bandwidth to VM_2 and VM_3 during the last iteration, which reduced the downtime and consequently the penalty incurred for VM_2 and VM_3.

Figures 4b and 4c present the total migration time and total downtime for various dirty rates. The total migration time for IPAM is less compared to PAM and higher than DTAM. This is again due to less assignment of bandwidth during the initial rounds to VM_2 and VM_3 as explained in Sect. 4.2.

5 Conclusion and Future Work

The key technology behind cloud management activities is VM migration, providing features like resource optimization, fault tolerance, and load balancing. However, VM migration confronts various challenges. Service level agreement (SLA) is a significant factor while moving the business to the cloud and determines the degree of trust the customer puts on the service provider.

Whenever service level agreement (SLA) regulates the availability of services on the VM, the overheads due to VM migration can not be disregarded. The penalty aware migration (PAM) assigns transfer rate to the VMs to safeguard

SLA during migration. This work improves the performance of penalty aware migration by proposing an algorithm that efficiently utilizes the freed bandwidth. As few VMs in the group complete the migration process, the bandwidth allocated to them is free. This bandwidth is assigned to the remaining VMs in the group which are still in the pre-copy phase. It was found that the total migration time, total downtime, and the penalty incurred are sufficiently reduced with the proposed approach.

Although this work mathematically models live migration in the light of SLA to efficiently utilize the available bandwidth, real-time implementation is left for the future. Further, the work can be extended to consider multiple SLAs for a service and agreed-upon penalty clauses in case of their violation.

Acknowledgment. This work was supported by the Ministry of Electronics and Information Technology (MEITY), Government of India under Visvesvaraya PhD Scheme for Electronics and IT (Grant no. PhD-MLA-4(20)/2015–16).

References

1. Cerroni, W., Esposito, F.: Optimizing live migration of multiple virtual machines. IEEE Trans. Cloud Comput. **6**(4), 1096–1109 (2016)
2. Clark, C., et. al.: Live migration of virtual machines. In: Proceedings of the 2nd Conference on Symposium on Networked Systems Design & Implementation, vol. 2, pp. 273–286 (2005)
3. Deshpande, U., Wang, X., Gopalan, K.: Live gang migration of virtual machines. In: Proceedings of the 20th International Symposium on High Performance Distributed Computing, pp. 135–146 (2011)
4. Gavin, H.P., Scruggs, J.T.: Constrained optimization using Lagrange multipliers. CEE 201L. Duke University (2012)
5. Hines, M.R., Deshpande, U., Gopalan, K.: Post-copy live migration of virtual machines. ACM SIGOPS Operating Syst. Rev. **43**(3), 14–26 (2009)
6. Jin, H., Deng, L., Wu, S., Shi, X., Chen, H., Pan, X.: MECOM: live migration of virtual machines by adaptively compressing memory pages. Future Gener. Comput. Syst. **38**, 23–35 (2014)
7. Macías, M., Rana, O., Smith, G., Guitart, J., Torres, J.: Maximizing revenue in grid markets using an economically enhanced resource manager. Concurrency Comput. Pract. Experience **22**(14), 1990–2011 (2010)
8. Parkin, M., Badia, R.M., Martrat, J.: A comparison of SLA use in six of the european commissions FP6 projects. Institute on Resource Management and Scheduling, CoreGRID-Network of Excellence, Tech. Rep. TR-0129 (2008)
9. Riteau, P., Morin, C., Priol, T.: Shrinker: improving live migration of virtual clusters over wans with distributed data deduplication and content-based addressing. In: Jeannot, E., Namyst, R., Roman, J. (eds.) Euro-Par 2011. LNCS, vol. 6852, pp. 431–442. Springer, Heidelberg (2011). https://doi.org/10.1007/978-3-642-23400-2_40
10. Singh, G., Singh, A.K.: Optimizing multi-VM migration by allocating transfer and compression rate using geometric programming. Simul. Model. Pract. Theory **106**, 102201 (2021)

11. Singh, G., Singh, A.K.: Optimization of SLA aware live migration of multiple virtual machines using lagrange multiplier. Future Gener. Comput. Syst. (2022)
12. Tsakalozos, K., Verroios, V., Roussopoulos, M., Delis, A.: Live VM migration under time-constraints in share-nothing IaaS-clouds. IEEE Trans. Parallel Distrib. Syst. **28**(8), 2285–2298 (2017)
13. Voorsluys, W., Broberg, J., Venugopal, S., Buyya, R.: Cost of virtual machine live migration in clouds: a performance evaluation. In: Jaatun, M.G., Zhao, G., Rong, C. (eds.) CloudCom 2009. LNCS, vol. 5931, pp. 254–265. Springer, Heidelberg (2009). https://doi.org/10.1007/978-3-642-10665-1_23
14. Winsor, C.P.: The Gompertz curve as a growth curve. Proc. Nat. Acad. Sci. U.S. Am. **18**(1), 1 (1932)