Bandwidth Provisioning for Virtual Machine Migration in Cloud: Strategy and Application

Uttam Mandal, Pulak Chowdhury, Massimo Tornatore[†], Charles U. Martel, and Biswanath Mukherjee
University of California, Davis; [†]Also with Politecnico di Milano, Italy
{umandal; pchowdhury; cumartel; bmukherjee}@ucdavis.edu, {tornator}@elet.polimi.it

Abstract—**Physical resources are highly virtualized in today's datacenter-based cloud-computing networks. Servers, for example, are virtualized as Virtual Machines (VMs). Through abstraction of physical resources, server virtualization enables migration of VMs over the interconnecting network. VM migration can be used for load balancing, energy conservation, disaster protection, etc. Migration of a VM involves iterative memory copy and network re-configuration. Memory states are transferred in multiple phases to keep the VM alive during the migration process, with a small downtime for switchover. Significant network resources are consumed during this process. Migration also results in undesirable performance impacts. Suboptimal network bandwidth assignment, inaccurate pre-copy iterations, and high end-to-end network delay in wide-area networks (WAN) can exacerbate the performance degradation. In this study, we devise strategies to find suitable bandwidth and pre-copy iteration count to optimize different performance metrics of VM migration over a WAN. First, we formulate models to measure network resource consumption, migration duration, and migration downtime. Then, we propose a strategy to determine appropriate migration bandwidth and number of pre-copy iterations, and perform numerical experiments in multiple cloud environments with large number of migration requests. Results show that our approach consumes less network resources when compared with maximum and minimum-bandwidth provisioning strategies while using an order of magnitude less bandwidth than maximum-bandwidth strategy. It also achieves significantly lower migration duration than minimum-bandwidth scheme.**

I. Introduction

Public and enterprise Internet services are increasingly running over cloud-computing systems hosted in geographically-distributed datacenters interconnected by wide-area networks (WANs) [1]. Datacenter physical resources and networks are generally virtualized [1] [2] [3]. Servers are virtualized to support one or more Virtual Machines (VMs), which in turn host applications and services. Among other benefits, virtualization enables seamless migration of VMs across physical servers. Live migration has been shown to be beneficial for cloud providers with single or multiple datacenters [4] [5] [6]. While migration within a single datacenter over a local-area network (LAN) has been commonly used, recently migrations across datacenters over WANs are also becoming popular [7]. Due to much larger network delays, difficulty in migrating VM configurations, and, in many cases, unavailability of sufficient bandwidth, migrations over a WAN require different considerations than over a LAN. Different aspects of WAN migration are addressed in several studies [4] [5] [7] [8].

VM migration is performed in multiple steps. A network connection is first established between source and destination locations. VM memory, and optionally disk-storage, states are then transferred in multiple iterations, called pre-copy iterations. In the first iteration, the full memory, and optionally disk storage, is transferred. Successive iterations transfer only the delta, or the modified/dirtied part of the memory (see Section III for details). Finally, the VM is stopped at the source location, and it is re-configured and resumed at the destination location. The last step incurs a small downtime during which the VM is unavailable. The rate at which a VM modifies or dirties its memory (or disk storage) is called memory-dirtying rate, a key characteristic impacting migration. Each of these migration steps requires a significant amount of computation which can cause performance degradation of the VM. Allocating sufficient network bandwidth for migration can reduce migration downtime and performance degradation.

Depending on the constraints and objectives of migrations, cloud providers may wish to minimize any of three performance metrics: migration duration, downtime, and network resource consumption. For some cloud applications, limiting migration downtime is of utmost importance [9] [10], while for others, optimizing network resource consumption is necessary [11] [12]. Additionally, VM migrations may have to meet a constraint on acceptable downtime [13], in which case reducing migration duration to meet this constraint is important. While higher network bandwidth can reduce migration downtime, it can increase resource consumption. Also, as network bandwidth is scarce and an expensive resource, provisioning very high bandwidth for migrations may not be possible. On the other hand, bandwidth can be decreased to reduce resource consumption but will increase migration duration and downtime. Number of pre-copy iterations has similar effect, with increased number of iterations causing high migration duration but low downtime. Some state-of-the-art solutions use a minimum possible bandwidth that could achieve a certain constraint (e.g., downtime constraint) to perform migrations [4]. While this strategy performs well under certain conditions, it is far from optimal, as we demonstrate through our experimental results in this study.

In this study, we first analyse and understand the characteristics of a VM migration over a WAN. To this end, we derive comprehensive models – based on existing studies – to

*This work has been supported by NSF Grant No. CNS-1217978.

quantify three characteristics of a VM migration over a WAN, namely total duration, migration downtime, and resource consumption. We then perform numerical analysis to demonstrate the applicability of these models in a cloud environment. These results also help us to identify control parameters to perform migrations more efficiently. We then propose a strategy to identify an appropriate value of this control parameter, which is then used to calculate an effective migration bandwidth and number of pre-copy iterations to optimize performance characteristics and achieve any desired performance goals. Finally, we use this strategy in multiple cloud environments to perform migrations and discuss experimental results. In a nutshell, we propose a bandwidth-provisioning strategy that receives migration request information as input, and determines bandwidth and pre-copy iteration count as output to find a balance among all three metrics to meet performance goals.

The rest of this study is organized as follows. In Section II, we discuss recent related studies. In Section III, we derive models that quantify VM migration performance metrics. In Section IV, we identify key aspects and controllable parameters of these models; and based on these observations, we propose a bandwidth–provisioning strategy. In Section V, we present results of our strategy in multiple simulated cloud environments. Finally, Section VI concludes our study.

II. Related Work

VM migration over WAN has been investigated in previous research works. Refs. [4] [8] demonstrate the usefulness and practicality of live VM migration over a WAN, and discuss results on migration duration and downtime. A practical example of VM migration over WAN with acceptable downtime is presented in [8]. Experimental results with a range of VMs and applications are presented in [4], where the authors propose a rate-limiting strategy to dynamically adapt migration bandwidth depending on the memory-dirtying rate during one iteration. A special case of this approach is explored in our previous work [12]. However, re-provisioning migration bandwidth in each iteration can add significant overhead for resource allocation and de-allocation. Ref. [7] proposes VM migration over a WAN employing several optimization techniques, such as stop-and-copy, use of memory-page delta, content-based redundancy, etc. Experimental results are presented after applying these techniques to applications with different workload characteristics.

Other research studies have analysed VM migration characteristics [9], [14], [15], [16]. Ref. [9] presents models to predict migration performance for specific workloads on a Xen virtual machine monitor, noting that network bandwidth and VM memory-dirtying rate are key factors impacting migration characteristics. Ref. [15] presents performance models for varying amount of available CPU resources and VM characteristics. Ref. [14] provides an intuitive model for migration duration, migration energy, and resource consumption, based on provisioned bandwidth and VM characteristics. However, they are specifically applicable to migrations over a LAN.

We took inspiration from [14] and proposed initial models for migrations over a WAN in [12]. In our current study, we generalize the approach in [12] by proposing models which depend on provisioned bandwidth, VM characteristics, and inter-iteration delay (which may depend on end-to-end network delay). Several existing studies consider a range of bandwidths for migration over a WAN, for example, Ref. [7] considers from few Mbps to few Gbps, and finds that migration performance metrics vary for different bandwidths. Similar results are presented in [9], where three bandwidth values are considered: 100 Mbps, 1 Gbps, and 10 Gbps. The authors suggest that network bandwidth and memory-dirtying rate have non-linear effect on migration performance, and note that the ratio of memory-dirtying rate and provisioned bandwidth is a key factor. In our previous study [12], we proposed a two-stage bandwidth provisioning strategy as a special case for heterogeneous bandwidth (vs. homogeneous bandwidth) provisioning. In our current study, we extend the models presented in [12] to quantify VM migration characteristics in a more comprehensive manner. Based on the models and numerical experiments, we emphasize that the main controlling parameter to optimize migration performance is the ratio of bandwidth and memory-dirtying rate. We then propose a strategy to identify suitable values of this parameter and, as a result, value of migration bandwidth.

Ref. [13] proposes a scheme to limit the number of iterations in the pre-copy phase to satisfy a downtime constraint or a desired 'progress' of memory copy in successive iterations. This scheme performs the minimum number of iterations to meet this constraint. While minimum number of iterations is necessary, it is not clear if it is also sufficient. Ref. [9] takes a similar approach. In addition, it restricts the number of iterations to a maximum value in case other conditions fail. Our work demonstrates that the minimum number of iterations (to achieve a downtime constraint) is also sufficient to optimize migration performance.

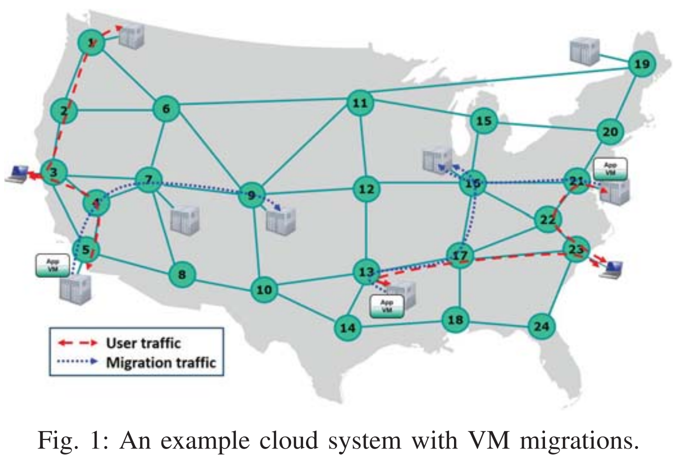


Fig. 1: An example cloud system with VM migrations.

III. Modeling VM Migration

VM migration across datacenters requires additional network resources in a WAN. Figure 1 shows an example of a nationwide cloud system with multiple datacenters, connected by a WAN. VMs are migrated in this cloud system across datacenters. These migrations generate significant amount of

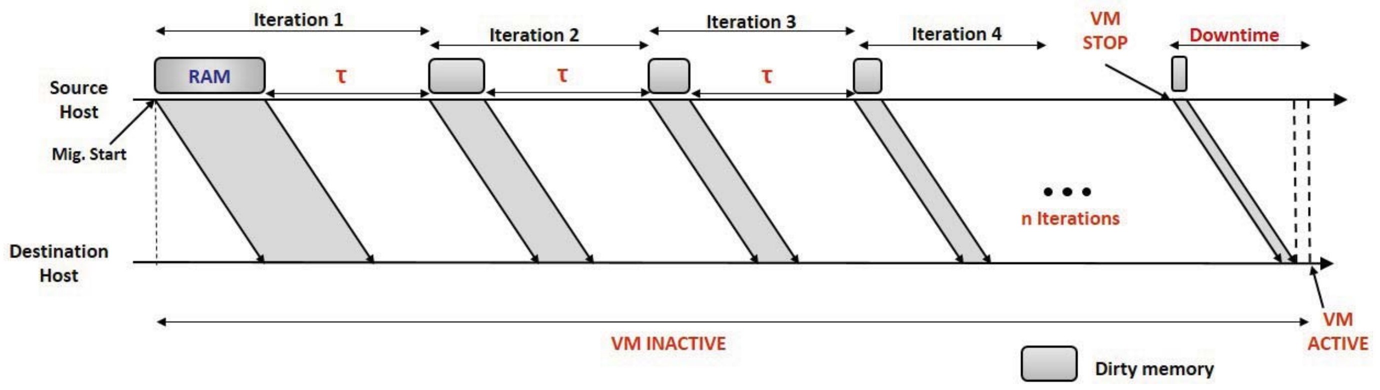


Fig. 2: Timing diagram of iterative pre-copy-based VM migration technique.

traffic in the WAN, adding to the already-existing user traffic. As VM migrations become key operational technique, WAN will require significant additional network resources [6] [7].

We consider an iterative pre-copy-based live migration [7] [8] [14]. Figure 2 shows an example timing diagram of this process. In the first iteration, the entire VM memory, say V_M Mb, is copied to the destination. During this iteration, part of the VM memory at the source gets modified, or 'dirtied', again. The next iteration, therefore, copies only the dirtied memory from the previous iteration. Similarly, the following iteration copies the dirtied memory from this iteration, and so on. We consider a small delay, τ seconds, between subsequent iterations caused by a combination of end-to-end network delay (round-trip time (RTT)) and data-processing delay. Duration of each iteration, therefore, consists of memory transmission time and the inter-iteration delay, τ. The iterative copy phase is stopped when a predefined number of iterations has been performed or the amount of dirtied memory is low enough to achieve a desired downtime. A final stop-and-copy phase is then started. In this phase, the VM is first stopped at its source, dirtied memory is copied, and the network is re-configured before bringing the VM up at the destination location. This phase may also involve request rerouting and redirection [5]. This last stop-and-copy phase causes VMs and hosted applications/services to be unavailable. Duration of this downtime largely depends on the amount of dirtied memory that needs to be copied in the last iteration.

A. VM Migration Modelling

To develop an analytical model of VM migration, we need to identify the duration and/or amount of dirtied memory in each iteration. How fast a VM dirties its memory depends on the characteristics of the VM and its hosted applications. While this amount may vary for different iterations, a constant average dirtying rate can be assumed for a particular VM [6] [9] [14]. We denote this dirtying rate by D Mbps, number of iterations in the pre-copy phase by n, amount of memory transferred in iteration i by V_i Mb, and duration of this iteration by T_i seconds. Note again that T_i consists of memory transmission time and inter-iteration delay, τ. Memory copied during iteration one, V_1, is equal to the size of VM memory, V_M. If the provisioned bandwidth for migration is R Mbps, data transmitted in round i can be stated as:

$$V_i = \begin{cases} V_M, & \text{if } i = 1 \\ D \times T_{i-1}, & \text{otherwise} \end{cases}$$

Duration of iteration i can be calculated as:

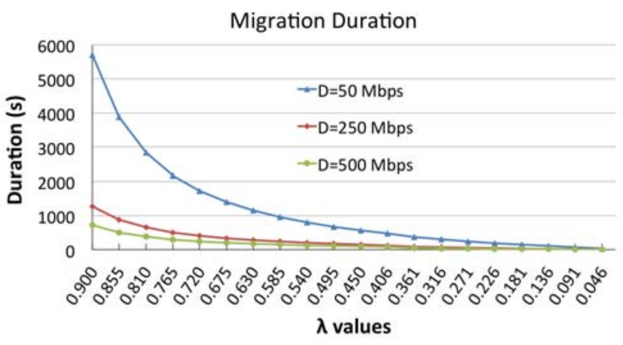
$$T_i = \frac{V_i}{R} + \tau = \begin{cases} V_M/R + \tau, & \text{if } i = 1 \\ (D \times T_{i-1})/R + \tau, & \text{otherwise} \end{cases}$$
$$= \begin{cases} V_M/R + \tau, & \text{if } i = 1 \\ T_{i-1} \times \lambda + \tau, & \text{otherwise} \end{cases}$$

Here, $\lambda = D/R$, which is the ratio of the memory-dirtying rate and the provisioned bandwidth, and is a key parameter in this study. We assume the value of λ is always less than 1, i.e., the migration bandwidth, R, is always larger than memory-dirtying rate, D. Otherwise, the iterative phase will make no progress as the memory will be dirtied faster than transfer of the dirty memory. Now, the duration of iteration 1, 2, 3, 4 ... is given by:

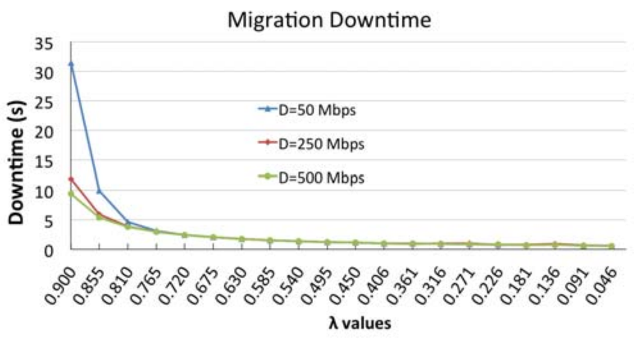
$$\begin{aligned} T_1 &= \frac{V_M}{R} + \tau \\ T_2 &= T_1 \times \lambda + \tau = \frac{V_M}{R}\lambda + \tau\lambda + \tau \\ &= \frac{V_M}{R}\lambda + \tau(1+\lambda) \\ T_3 &= T_2 \times \lambda + \tau = \frac{V_M}{R}\lambda^2 + \tau(1+\lambda+\lambda^2) \\ &= \frac{V_M}{R}\lambda^2 + \tau(\frac{1-\lambda^3}{1-\lambda}), \lambda < 1 \\ T_4 &= \frac{V_M}{R}\lambda^3 + \tau(\frac{1-\lambda^4}{1-\lambda}) \quad \text{etc.} \end{aligned}$$

Additionally, T_1 can be represented as $T_1 = \frac{V_M}{R}\lambda^0 + \tau(\frac{1-\lambda^1}{1-\lambda})$. Therefore, we can generalize that the duration of iteration i is given by:

$$T_i = \frac{V_M}{R}\lambda^{i-1} + \tau\frac{1-\lambda^i}{1-\lambda}, i = 1...n \tag{1}$$



(a) Migration duration.



(b) Migration downtime.

Fig. 3: Migration duration and downtime for different D and λ.

Total migration time, T_{mig}, can now be expressed as:

$$\begin{aligned} T_{mig} &= \sum_{i=1}^{n} T_i + t_{down} \\ &= \frac{V_M}{R}\sum_{i=1}^{n}\lambda^{i-1} + \frac{\tau}{1-\lambda}\sum_{i=1}^{n}(1-\lambda^i) \\ &= \frac{V_M}{R}\frac{1-\lambda^{n+1}}{1-\lambda} + \tau\frac{n(1-\lambda)-\lambda(1-\lambda^{n+1})}{(1-\lambda)^2} \\ &\qquad + t_{down}, \end{aligned} \tag{2}$$

where t_{down} is the duration of stop-and-copy phase, during which the VM is unavailable. This downtime includes final memory copy and network reconfiguration time. Given that the iterative phase performs n iterations, amount of dirtied memory to be transferred after VM is stopped, V^s, equals:

$$V^s = T_n \times D = V_M\lambda^n + \tau\frac{1-\lambda^n}{1-\lambda}D$$

Therefore, duration for which the VM is down is given by:

$$t_{down} = \frac{V^s}{R} + t_g = \frac{V_M}{R}\lambda^n + \lambda\tau\frac{1-\lambda^n}{1-\lambda} + t_g \tag{3}$$

where t_g is the network re-configuration time, denoting the time required to perform necessary reconfiguration to route requests to the new VM location. Finally, network resource consumption can be represented by:

$$W_{mig} = T_{mig} \times R. \tag{4}$$

For sake of clarity, let us evaluate these models using some numerical examples. Following Eqs. (2) and (3), Fig. 3 presents total migration time and migration downtime for VMs with 4 GB (32768 Mb) of memory and dirtying rates of 50 Mbps, 250 Mbps, and 500 Mbps as a function of $\lambda = D/R$. We note here that, for a particular D, λ is inversely proportional to provisioned bandwidth, R. Each point in Fig. 3 is calculated by applying Eqs. (2) and (3) for a given value of λ and D. Inter-iteration delay, τ, and network reconfiguration time, t_g, are considered on the order of few hundred milliseconds. The y-axes in these figures represent migration duration and downtime in seconds, while x-axes represent λ values. Note that the trend of migration duration (and downtime) with decreasing value of λ follows a similar pattern for VMs with varying dirtying rates. These results (and additional unreported results) exhibit close similarity to published data [4] [7] [8] [9] [15] [13] [17], further validating usefulness of these models.

B. Migration Control Parameters

Numerical results from our models, specifically from Eqs. (2) and (3), show that, for a particular VM, migration duration and downtime can be controlled by adjusting two parameters: provisioned bandwidth, R, and number of iterations, n. The results also show that λ is more important in determining migration performance than provisioned bandwidth alone. In fact, migration performance can be better optimized with a fixed value instead of using a fixed bandwidth for all VM migrations (as in prior solutions). Similar observation is made in [9]. To further illustrate, let us first evaluate a simple example strategy which provisions a fixed amount of bandwidth for all migrations. Our goal is to show how ineffective this strategy can be in certain scenarios. Let the fixed bandwidth used in this strategy be 100 Mbps. Migrating a VM with dirtying rate of 250 Mbps using this strategy (making $\lambda = 250/100 = 2.5$) will result in a very high migration downtime. This is because the VM memory will be dirtied faster than it can be transmitted using the provisioned bandwidth, causing the iterative phase to not make any significant progress and the last stop-and-copy phase to have higher duration. On the other hand, this strategy will perform well when migrating a VM with dirtying rate 10 Mbps ($\lambda = 10/100 = 0.1$). But the provisioned bandwidth is unnecessarily high and will be underutilized. So, this strategy not only performs bad for the pathological case of $\lambda > 1.0$ but also for other scenarios. This is further supported by the numerical results in the next few sections.

Now, as opposed to the previous strategy, we aim to maintain a fixed λ value for all migrations, say 0.4. Provisioned bandwidth for migrating the VM with dirtying rate 250 Mbps is $R = 250/0.4 = 625$ Mbps, and for VM with dirtying rate 10 Mbps, its $R = 10/0.4 = 25$ Mbps. Migration duration and downtime for the first VM is significantly lower with this strategy compared to the fixed-bandwidth strategy. For the second VM, while these values are similar for both strategies (higher bandwidth does not notably reduce migration duration and downtime for the VM with low dirtying rate), network resource consumption, as defined in Eq. (4), is significantly lower with this strategy. For both migrations, this strategy is able to achieve reasonable migration performance when the previous strategy fails to do so. Therefore, we can say that λ is a more suitable control parameter for migrations than bandwidth. However, note that identifying an optimal λ value is a challenging task as it involves optimising metrics defined in Eqs. (2), (3), and (4). In this study, using the above analytical models and numerical experiments and observations, we investigate strategies to determine optimum λ value(s) for all migrations. We also investigate the optimum value for number of pre-copy iterations (n).

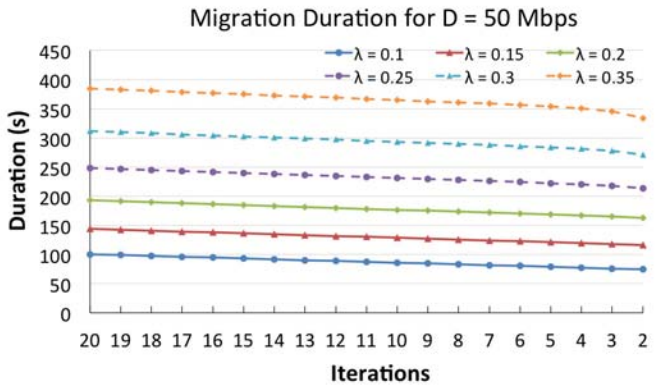


Fig. 4: Migration duration for different iterations and λ.

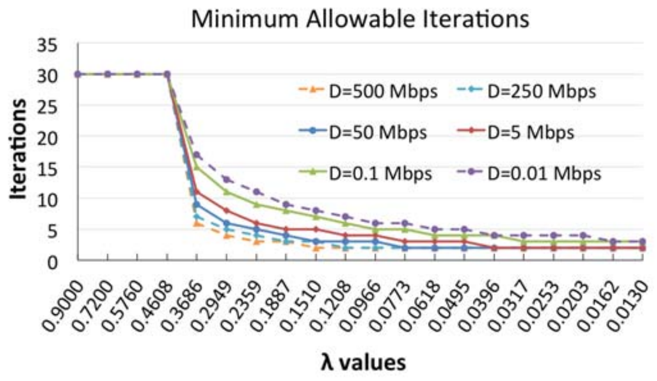


Fig. 5: Minimum required iterations for different D and λ with downtime constraints.

IV. Bandwidth–Provisioning Strategy

To determine optimal λ and n for VM migrations, we first present results from numerical experiments using models established in the previous section. We then observe patterns of these results to devise a strategy. This strategy will receive VM migration details as input, and determine λ and n values. From the λ value, we can calculate required migration bandwidth, $R = D/\lambda$. As noted earlier, a VM migration request may contain a desired downtime constraint. The values of R and n for those requests must ensure that the experienced downtime respects this constraint. In addition, the R and n values from this strategy should lead to a balance among all three performance metrics, namely migration duration, downtime, and network resource consumption. In some cases, we also consider optimizing only a subset of these metrics.

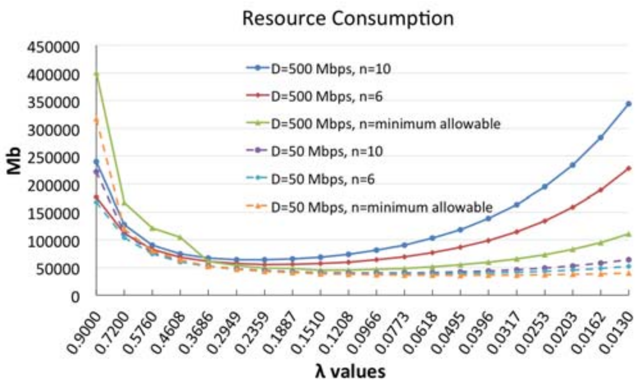
A. Preliminary Observations

1) Influence of n on T_{mig}: Given Eqs. (2) and (3), Fig. 4 shows total migration duration for different values of λ and n [9]. The y-axis shows total migration duration in seconds and the x-axis represents the number of iterations. Total migration duration decreases steadily with decreasing iterations for all λ values. This is because performing more iterations results in longer duration. Therefore, we observe that, *if migration duration is to be minimized for a VM, it is favorable to perform as few iterations as possible, and provision as high bandwidth (lower λ) as possible*. However, if there is any constraint on downtime, it will impose a lower limit on the number of iterations to meet this constraint. This is also observed in [13], where the authors suggest performing the minimum number of iterations that ensures downtime within allowable limit.

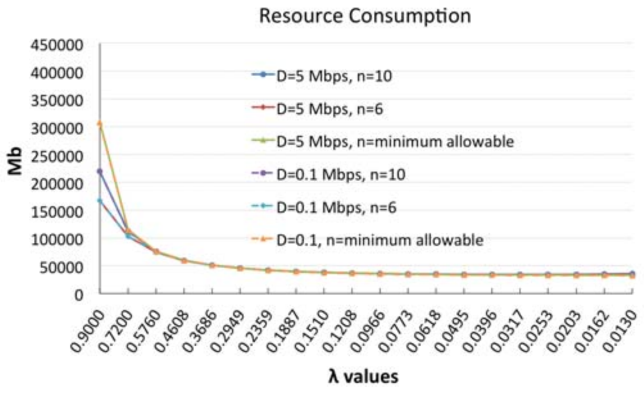
2) Influence of n on t_{down}: Figure 5 shows minimum required number of iterations that achieves a given downtime constraint for varying λ and dirtying rates. We consider migration downtime constraint values that are uniformly distributed between 1s to 1.2s with $\tau \approx 0.7$s and $t_g \approx 0.5$s.[1] The x-axis shows a range of λ values starting from 0.9. As in [9] and [13], we restrict maximum number of iterations to 30, when necessary, to stop the iterative phase. For higher values of λ, minimum required number of iterations is much larger than 30, and in some cases may never be achievable. Those cases are stopped after 30 iterations, as shown in the figure. On the other hand, with decreasing values of λ, minimum number of required iterations decreases rapidly. However, after a certain point, the minimum number of iterations does not change significantly any more. From these results, we gain another observation: *a lower λ value is desirable to reduce minimum required number of iterations; but if λ value is low enough, the required number of iterations does not change drastically, and there is no significant advantage in further decreasing λ.*

3) Influence of n and λ on W_{mig}: Figure 6 shows resource consumption, W_{mig}, in Megabits (Mb), for decreasing λ values. Minimum required number of iterations is performed for these migrations to ensure a downtime constraint (uniform between 1 sec and 1.2 sec). We see that resource consumption reduces fast with decreasing λ, especially in case of higher λ values. We noted similar results for migration duration in Fig. 3(a). For both high and low dirtying rates, Fig. 6 shows that W_{mig} converges fast as we decrease λ values, and beyond a threshold (≈0.4), it does not decrease as quickly. For higher dirtying rate, Fig. 6(a) shows that resource consumption starts to increase if λ values are decreased beyond a certain

[1]Note that t_g and τ values depend on network roundtrip-time (RTT). Assuming few hundred msec as a reasonable RTT for the example network in Fig. 1 and another few hundred msec for the processing time at the source and destination hosts, τ and t_g values remain below one second. But the value of τ will generally be higher than t_g as each migration iteration involves more processing than the final re-configuration. Throughout our experiments, we consider τ between 0.5 to 0.75 and $t_g \approx 0.5$. These values are conservative and for illustration purpose only. Different values can be used for evaluation.



(a) Higher dirtying rate.



(b) Lower dirtying rate.

Fig. 6: Resource consumption with exponential decrease of λ.

threshold. This is because a lower λ value translates to higher bandwidth. Although higher migration bandwidth causes faster transmission of VM memory, resource consumption, W_{mig}, increases due to slower decrease of T_{mig}, as shown in Fig. 3. This trend is not similar for lower dirtying rate, as seen in Fig. 6(b), in which cases, resource consumption does not change significantly beyond a threshold λ value. λ values beyond which W_{mig} starts to increase in Fig. 6(a) and remains unchanged in Fig. 6(b) are both close to 0.1 to 0.15. Note that these observed trends are due to the non-linear relationship presented in Eqs. (2) to (4).

From the above observations, we surmise that *migration performance metrics are optimized for most migrations if λ values are within a range*. These experiments show that this range is somewhere from 0.4 to 0.1. To generalize, let us assume that VM migrations can be optimized if λ values are constrained within a range, say λ_H to λ_L. This range can be estimated to be $\lambda_H = 0.4$ to $\lambda_L = 0.1$ from the results presented above. We make one more general observation from the results: *higher λ values lead to good performance results in case of higher dirtying rate and lower λ values perform well enough for lower dirtying rates*. This is because, not only lower λ values translate to high enough provisioned bandwidth for low dirtying rates, but higher λ values translate to high migration bandwidth for high dirtying rates as well. This also avoids the extreme cases of unnecessarily high bandwidth (if low λ value was used for high dirtying rate) or insufficient low bandwidth (if high λ value was used for low dirtying rate). Therefore, any strategy that aims to identify λ values should ensure higher λ value for high dirtying rate and lower λ value for low dirtying rate. However, this presents the problem of identifying whether a given dirtying rate is high or low. As a solution, we assume that, for a given cloud system, it is possible to determine the lowest and highest possible values of VM dirtying rates a priori. Let us denote the lower limit of dirtying rate by D_L and higher limit by D_H.

Identifying the above two range of values is a perquisite and an important aspect of our bandwidth-provisioning strategy described next. While observed range of λ values remains similar across our experiments with different cloud networks, we note that it may not be the case for all systems. For a different cloud system, this range of λ values can be determined through similar experiments and observations, if necessary. And the range of dirtying rates can be identified from the VMs and the application characteristics hosted at the VMs. Additionally, as this strategy will work with a flexible range of D values, this range can be estimated easily. By way of example, the value of D_L can be measured from the memory-dirtying rate of an idle server running only the necessary services while not serving any load. On the other hand, value of D_H can be measured from the memory-dirtying rate of the same server with a high simulated load. Another example method might be to use the limit of CPU to memory bandwidth as D_H and the CPU to memory bandwidth usage for an idle server as D_L.

B. *Proposed Algorithm: λ-range Mapping*

From the above considerations, we now present a strategy that accepts a VM's migration request information and identifies a bandwidth and iteration count for the pre-copy phase. A VM migration request consists of memory size of the VM, V_M; dirtying rate, D; inter-iteration delay, τ; and an optional downtime constraint, $\tilde{t}_{down}$. If no value of the constraint is provided, a default value, $\bar{t}_{down}$, is assumed. To determine an appropriate bandwidth, we first identify a certain λ value for the given migration request with dirtying rate D, and determine the required bandwidth from the relationship of λ, D, and R, namely $R = D/\lambda$. To identify λ, we uniformly map the range of dirtying-rate values, i.e., D_L to D_H, to the range of λ values, namely from λ_L to λ_H, such that a value close to λ_H is identified for a VM with dirtying rate close to D_H, and a value close to λ_L is identified for a VM with dirtying rate close to D_L. This can be done using the equation:

$$\lambda = \left[\lambda_L + \frac{(\lambda_H - \lambda_L)(D - D_L)}{D_H - D_L}\right]$$

This ensures that, to migrate a VM with high dirtying rate, a high value of λ is provisioned, and for a VM with low dirtying rate, a low value of λ is used. We previously noted this as necessary in our observations. We call this strategy λ-range mapping. Now, for a given VM with dirtying rate D, this strategy determines the bandwidth as:

$$R = \frac{D}{\lambda} = D \times \left[\lambda_L + \frac{(\lambda_H - \lambda_L)(D - D_L)}{D_H - D_L}\right]^{-1} \quad (5)$$

Also, migrations must perform a minimum required number of iterations. If a downtime constraint is provided for the migration request, the minimum required number of iterations can be determined from Eq. (3). If no constraint is provided, a default downtime constraint can be used to determine the minimum required iteration. We take this approach to simplify our strategy, but other approaches can be used as well (for example, a fixed number of iterations can be used). Algorithm 1 shows the pseudo code of our proposed strategy.

Algorithm 1 λ-range mapping

```
1: Define $D_L, D_H, \lambda_L, \lambda_H$                    ▷ Global constants.
2: Define $t_g$, default downtime constraint, $\bar{t}_{down}$
3: procedure MIGRATIONPARAMETERS ($V_M, D, \tau, \tilde{t}_{down}$)
4:     // $\tilde{t}_{down}$ is downtime constraint.
5:     if $\tilde{t}_{down} \leq 0$ then ▷ Negative, if no value is specified.
6:         $\tilde{t}_{down} \leftarrow \bar{t}_{down}$
7:     end if
8:     $\lambda \leftarrow \left[\lambda_L + \frac{(\lambda_H-\lambda_L)(D-D_L)}{D_H-D_L}\right]$
9:     $R \leftarrow D/\lambda$
10:    $n \leftarrow$ MIGRATIONITERATIONS($\tilde{t}_{down}, V_M, R, \lambda, \tau$)
11:    Return $(R, n)$
12: end procedure
13: procedure MIGRATIONITERATIONS ($t_{down}, V_M, R, \lambda, \tau$)
14:    Solve Eq. (3) for $n$ with $t_{down}, V_M, R, \lambda, \tau, t_g$
15:    Round $n$
16:    Return $n$
17: end procedure
```

V. Application in Cloud: Numerical Examples

Now, we apply λ-range mapping in sample cloud systems to quantitatively validate this approach. For comparison, we consider three other bandwidth-provisioning strategies: minimum-bandwidth, maximum-bandwidth, and Δ-increment.

- The minimum-bandwidth scheme assigns bandwidth larger than the dirtying rate by a predefined amount [4], say δ. This is to ensure that the migration bandwidth, R, is greater than memory-dirtying rate, D. As noted earlier, otherwise the iterative phase will not make any progress. This also ensures $\lambda < 1$. Typical value of δ used is 50 Mbps [4].
- The maximum-bandwidth scheme provides a maximum predefined amount of bandwidth for all migrations. For our experiments, this value is 10 Gbps, as in [9].
- The Δ-increment strategy takes a similar approach to our previous work [12]. For a VM migration with dirtying rate D, we identify bandwidth R from a series of values $(D + \Delta, D + 2\Delta, D + 3\Delta, ...)$, and a predefined Δ, such that the following condition holds:

$$(T_{mig}(R) - T_{mig}(R + \Delta))/T_{mig}(R) - \frac{\Delta}{R} \leq \epsilon$$

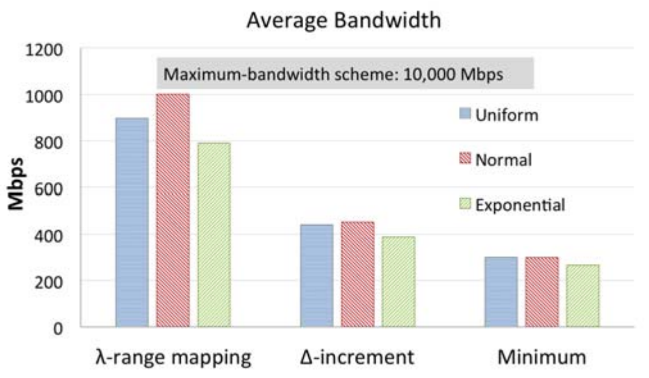
where ϵ is a predefined threshold.

Each of these strategies serves a purpose for comparing performance metrics against our proposed λ-range mapping technique. For example, the minimum-bandwidth scheme resembles the current operation technique, where low bandwidth is provided for most VM migrations [4]. Maximum-bandwidth scheme provides an indication towards an ideal solution where networks have almost unlimited bandwidth. Note that this strategy should predictably outperform all other strategies. However, as noted earlier, because bandwidth is an expensive resource, using maximum bandwidth is not a practical solution. But this strategy provides a benchmark for comparing against other strategies. The Δ-increment strategy resembles finding a local minima in Fig. 3(a). The smaller the ϵ value is, closer it can get to the optimum. Here, we verify our approach in multiple sample cloud systems and compare results against these approaches.

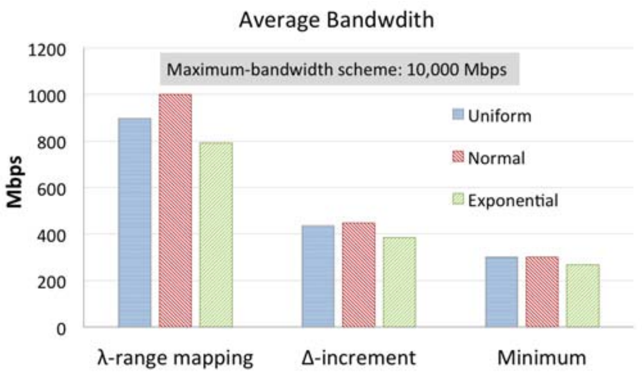
As cloud infrastructures, we consider two US-wide national networks. The first is shown in Fig. 1 with 11 datacenters, and the other is the NSF network [11] with 8 datacenters. The first network has 24 backbone nodes while the second has 14 backbone nodes. Each link has capacity of 100 Gbps. A datacenter can host thousands of VMs, and we simulate total number of VM migration requests on the order of 50-75 thousand. Dirtying rates for VMs are assigned values within the range D_H to D_L. Each of these values is picked using three different probability distributions: uniform, normal, and negative exponential. All these distributions have mean value of $(D_H + D_L)/2$. VM memory sizes are uniformly distributed between 4 GB ((32768 Mb) to 8 GB (65536 Mb). Range of λ considered is $\lambda_L = 0.1$ to $\lambda_L = 0.4$, and range of dirtying rates considered is $D_L = 0.1$ Mbps to $D_H = 500$ Mbps. Downtime constraint is uniformly distributed in the range 0.5 sec to 1.5 sec. Other values considered are $\Delta = 25$ Mbps, $\delta = 50$ Mbps, $t_g = 0.5$ sec, $\tau = 0.75$, $\epsilon = 0.1$, and maximum number of iterations = 20. We provision migration bandwidth as soon a request arrives, following a Poisson distribution. Note that some of the values used here are different from previous experiments to further validate our approach.

We use these experimental settings with the bandwidth-provisioning strategies listed earlier. For each of these experiments, we calculate four metrics: average provisioned bandwidth, average number of iterations, average migration duration, and average network resource consumption. Each experiment is performed several times to ensure variation in the probabilistic distribution of memory-dirtying rates and downtime constraints. Each of the metrics is then averaged first over the number of experiments and then again over number of VM migrations performed. We present these results in Figs. 7 through 10.

Figure 7 shows average provisioned bandwidth for the three different provisioning strategies, excluding results for maximum-bandwidth strategy, as for maximum-bandwidth strategy, provisioned bandwidth is always 10 Gbps. On average, λ-range mapping provisions bandwidth around 800 Mbps to 1 Gbps, which is only 10% of the maximum-bandwidth strategy (a 90% reduction). Δ-increment provisions around 400 Mbps, and minimum-bandwidth strategy provisions only around 300 Mbps. Average bandwidth provisioned by the

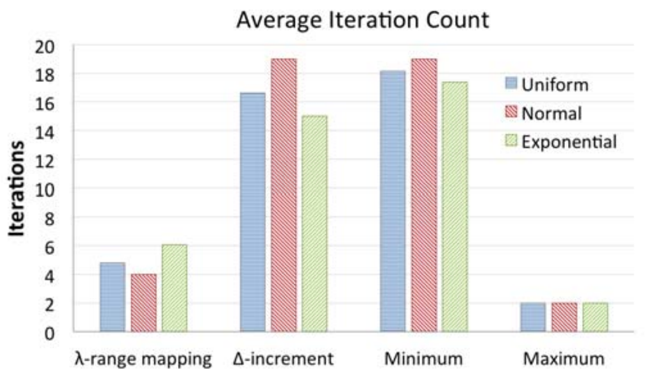


(a) For cloud network 1.

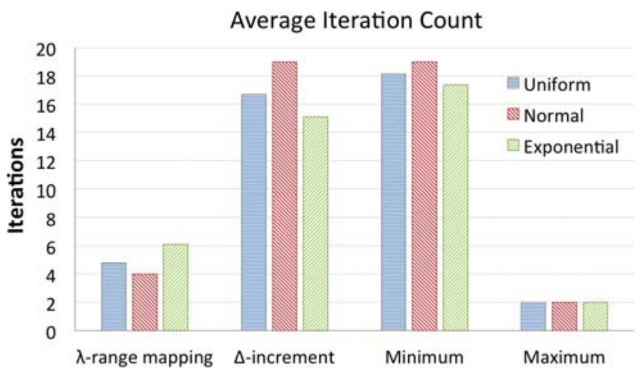


(b) For NSF network.

Fig. 7: Average migration bandwidth for different provisioning strategies.



(a) For cloud network 1.



(b) For NSF network.

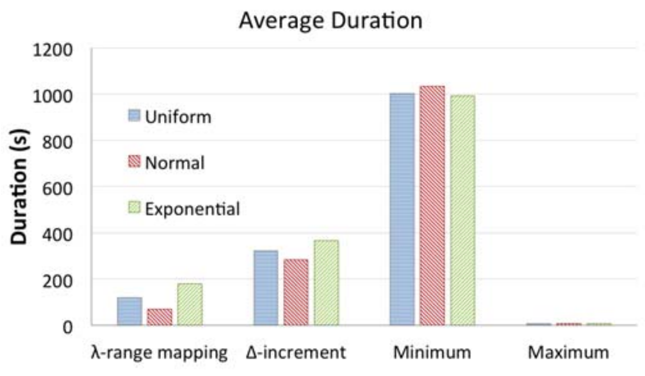
Fig. 8: Average number of iterations for different provisioning strategies.

minimum-bandwidth strategy has this range because the mean of dirtying rates is close to 250 Mbps. We see a similar trend in the NSF network as well. Note that, as our bandwidth strategy does not depend on network configuration, provisioned bandwidths in the two networks are very similar. We see the effect of network topology and configuration when we consider resource consumption.

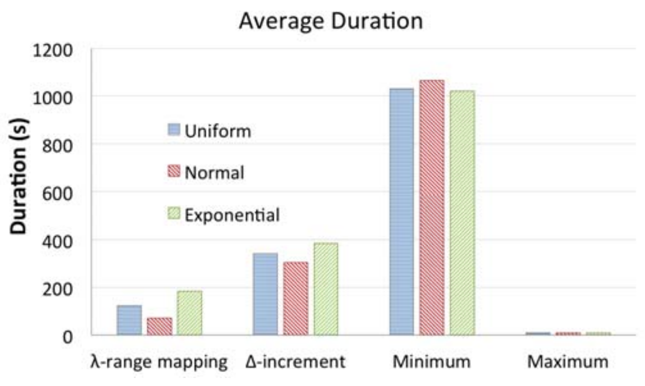
Figures 8 and 9 present average iteration counts and average migration duration. Average iteration count is minimum for maximum-bandwidth scheme because high bandwidth ensures that the downtime constraint can be achieved in few iterations, but for minimum-bandwidth scheme, it is close to the maximum value (note that we use maximum number of iterations as 20 here). Similar trend is seen for Δ-increment scheme as the bandwidth provisioned remains close to the minimum-bandwidth scheme. A significant change is seen in λ-range mapping scheme. While average iteration count of our approach is not as low as the maximum-bandwidth scheme, it is very close and significantly lower than minimum-bandwidth and Δ-increment schemes. This, in turn, explains the variation in the average migration duration in Fig. 9. Again, the maximum-bandwidth scheme has very low migration duration, while in minimum-bandwidth strategy, it is very high. λ-range mapping has reasonably low migration duration which is significantly lower than minimum-bandwidth scheme. Δ-increment scheme shows similar result as λ-range mapping. This again is due to the fact that migration duration decreases exponentially with decreasing λ, as seen in Fig. 3(a).

Finally, we compare average resource consumption per migration (in Megabits) in Fig. 10. Resource consumption for minimum-bandwidth scheme is significantly higher than any other scheme. Results of average resource consumption and average migration duration together demonstrate that minimum-bandwidth provisioning is not very beneficial for VM migrations. While Δ-increment shows significantly lower resource consumption than minimum-bandwidth scheme, it is still higher than in λ-range mapping and maximum-bandwidth schemes. In most cases, the λ-range mapping consumes less resources than the maximum-bandwidth scheme. This difference is around 5-10% in most cases. Note that the absolute resource consumption is different for different networks due to the differences in distances between source and destination hosts and number of hops in different networks.

The results shown above demonstrate the relatively strong aspects of λ-range mapping compared to the other approaches. This bandwidth-provisioning scheme finds a reasonable balance among provisioned migration bandwidth, migration duration, iteration counts, and resource consumption. Bandwidth provisioned using our scheme is 90% less than the maximum-bandwidth scheme. While migration duration for minimum-

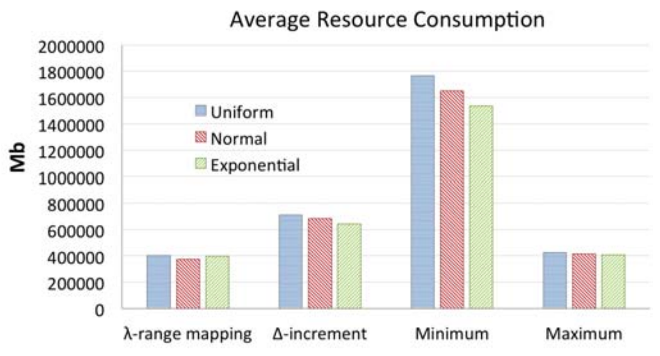


(a) For cloud network 1.

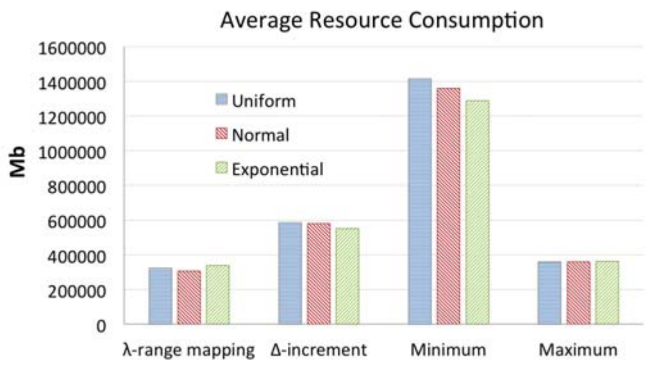


(b) For NSF network.

Fig. 9: Average migration duration for different provisioning strategies.



(a) For cloud network 1.



(b) For NSF network.

Fig. 10: Average resource consumption per migration for different provisioning strategies.

bandwidth and Δ-increment schemes are incomparably higher than the maximum-bandwidth scheme, our scheme produces comparable migration duration while using significantly low bandwidth. On the other hand, while our approach uses slightly higher bandwidth than minimum-bandwidth and the Δ-increment schemes, iteration count and migration duration are significantly lower than these schemes. Most importantly, our scheme exhibits best network resource-consumption behavior. Using an average bandwidth of only $1/10^{\text{th}}$ of the maximum-bandwidth scheme, our approach not only achieves comparable resource consumption, but in most cases it reduces resource consumption by 5-10%. This result is consistent in different networks with different number of nodes, links, datacenters, and migration requests.

As noted earlier, the λ-range values used in our approach can be re-evaluated for a particular cloud provider, if required, and the new values can be easily plugged in. While we used a wide range of VM memory-dirtying rates, this range can also be easily modified. Depending on VM and application characteristics of a cloud provider, different range of dirtying rates can be used. Furthermore, the range of λ can be modified to favor some performance metrics over others. For example, if only migration duration is important and high network resource consumption can be tolerated, smaller values of λ_H and λ_L can be assigned in our scheme. As seen in Fig. 3(b), this will ensure lower migration duration. Similarly, for other performance metrics, λ range can be appropriately changed. **We also note here that the limits of λ and D used in our experiments does not depend on the cloud systems significantly. On the other hand, while the inter-iteration delay (τ) may depend on a cloud network, the wide range of value used in our experiments ensures applicability to a variety of cloud systems.** As a result, our bandwidth-provisioning strategy is adaptable to any cloud environments.

VI. Conclusion

In virtualized cloud systems, migration of VMs over a WAN is a very useful technique for cloud providers. However, performance degradation for migrations over a WAN is much more critical than in migrations over a LAN. In this study, we first proposed a model to quantify migration duration and downtime for a WAN. Through evaluation over a range of migration parameters, we showed that our proposed model can produce results similar to published data. Based on these models and based on observation of results, we proposed a strategy, called λ-range mapping, to determine appropriate bandwidth and pre-copy iteration count to optimize migration over a WAN with respect to migration duration, downtime, and network resource consumption. We then applied this strategy to multiple cloud systems and compared it with standard approaches. Results demonstrated that our proposed bandwidth-provisioning scheme finds a compromise among several performance metrics while using low bandwidth. Also, the flexibility of our approach allows adaptability by cloud providers to achieve any desired migration performance goals.

REFERENCES

[1] M. Armbrust, A. Fox, R. Griffith, A. D. Joseph, R. Katz, A. Konwinski, G. Lee, D. Patterson, A. Rabkin, I. Stoica, and M. Zaharia, "A view of cloud computing," *Communications of the ACM*, vol. 53, no. 4, pp. 50–58, Apr. 2010.

[2] M. F. Bari, R. Boutaba, R. Esteves, L. Z. Granville, M. Podlesny, M. G. Rabbani, Q. Zhang, and M. F. Zhani, "Data center network virtualization: A survey," *IEEE Communications Surveys & Tutorials*, vol. 15, no. 2, pp. 909–928, 2013.

[3] S. Peng, R. Nejabati, and D. Simeonidou, "Role of optical network virtualization in cloud computing," *IEEE/OSA Journal of Optical Communications and Networking*, vol. 5, no. 10, pp. A162–A170, 2013.

[4] C. Clark, K. Fraser, S. Hand, J. G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live migration of virtual machines," in *Proc., USENIX Symposium on Networked Systems Design & Implementation*, vol. 2, 2005, pp. 273–286.

[5] VMware and F5 Techincal Brief, "Enabling Long Distance Live Migration with F5 and VMware vMotion," www.f5.com/pdf/white-papers/cloud-vmotion-f5-wp.pdf.

[6] U. Mandal, M. F. Habib, S. Zhang, B. Mukherjee, and M. Tornatore, "Greening the cloud using renewable-energy-aware service migration," *IEEE Network*, vol. 27, no. 6, pp. 36–43, Nov. 2013.

[7] T. Wood, K. Ramakrishnan, J. van der Merwe, P. Shenoy, and J. Hwang, "Cloudnet: Dynamic pooling of cloud resources by live WAN migration of virtual machines," *University of Massachusetts Technical Report UM-CS-2012-005*, 2012. [Online]. Available: http://faculty.cs.gwu.edu/˜timwood/papers/TR-cloudnet.pdf

[8] F. Travostino, P. Daspit, L. Gommans, C. Jog, C. de Laat, J. Mambretti, I. Monga, B. van Oudenaarde, S. Raghunath, and P. Y. Wang, "Seamless live migration of virtual machines over the MAN/WAN," *Future Generation Computer Systems*, vol. 22, no. 8, pp. 901–907, 2006.

[9] S. Akoush, R. Sohan, A. Rice, A. W. Moore, and A. Hopper, "Predicting the performance of virtual machine migration," in *Proc., Modeling, Analysis & Simulation of Computer and Telecommunication Systems (MASCOTS)*, 2010, pp. 37–46.

[10] M. Alicherry and T. V. Lakshman, "Optimizing data access latencies in cloud systems by intelligent virtual machine placement," in *Proc., IEEE INFOCOM*, 2013, pp. 647–655.

[11] U. Mandal, M. F. Habib, S. Zhang, M. Tornatore, and B. Mukherjee, "Bandwidth and routing assignment for virtual machine migration in photonic cloud networks," in *Proc., European Conference and Exhibition on Optical Communication (ECOC)*, Sept. 2013.

[12] U. Mandal, M. F. Habib, S. Zhang, M. Tornatore, and B. Mukherjee, "Heterogeneous bandwidth provisioning for virtual machine migration over SDN-enabled optical networks," in *Proc., OFC*, March 2014.

[13] V. Mann, A. Gupta, P. Dutta, A. Vishnoi, P. Bhattacharya, R. Poddar, and A. Iyer, "Remedy: Network-aware steady state VM management for data centers," *NETWORKING 2012*, pp. 190–204, 2012.

[14] H. Liu, C.-Z. Xu, H. Jin, J. Gong, and X. Liao, "Performance and energy modeling for live migration of virtual machines," in *Proc., High Performance Distributed Computing (HPDC)*, 2011, pp. 171–182.

[15] Y. Wu and M. Zhao, "Performance modeling of virtual machine live migration," in *Proc., International Conference on Cloud Computing (CLOUD)*, July 2011, pp. 492–499.

[16] W. Cerroni and F. Callegati, "Live migration of virtual network functions in cloud-based edge networks," in *Proc., International Conference on Communications (ICC)*, June 2014.

[17] D. Aikema, A. Mirtchovski, C. Kiddle, and R. Simmonds, "Green cloud VM migration: Power use analysis," in *Proc., International Green Computing Conference (IGCC)*, 2012.

Uttam Mandal is a Software Engineer at Cisco Systems Inc. in San Jose, California. He received his PhD and M.S. degrees from University of California, Davis and bachelor of engineering (B.E.) from Javadpur University, Kolkata, India, both in Computer Science. He authored several peer-reviewed papers in international journals and conference proceedings. His research has focused on designing resource-efficient cloud and content delivery networks and energy-efficient/green-energy-aware networks. Currently he is working in developing Network Functions Virtualization (NFV) management and organization system for a distributed cloud infrastructure. Previously, he worked at Anvato, building content (video) delivery solutions for leading content owners and broadcasters.

Pulak Chowdhury is an author of 30+ peer-reviewed scholarly papers in international journal and conference proceedings, mostly in resource-efficient, next-generation networking domain. He was an associate editor of IEEE Journal of Selected Areas in Communications (JSAC), Special Issue on Emerging Technologies in Communications. He has also served as a Technical Program Committee (TPC) member of international conferences and workshops. Dr. Chowdhury received the Ph.D. degree from University of California, Davis, the M.A.Sc. degree from McMaster University, Canada, and the B.Sc. Engineering degree from Bangladesh University of Engineering and Technology, Dhaka, Bangladesh, in 2011, 2005, and 2002, respectively. His research interests cover a variety of topics in optical networks, hybrid wireless-optical networks, and energy efficiency in next-generation networks. He is a founder of Ennetix inc., where he works on building network performance management solutions.

Massimo Tornatore is currently an Associate Professor in the Department of Electronics, Information and Bioengineering at Politecnico di Milano, Italy, where he received a PhD degree in Information Engineering in 2006. He also holds an appointment as adjunct associate professor in the Department of Computer Science at the University of California, Davis, where he served as a postdoc researcher between 2007 and 2009. He is author of more than 200 peer-reviewed conference and journal papers and his research interests include performance evaluation, optimization and design of communication networks (with an emphasis on the application of optical networking technologies), cloud computing, and energy-efficient networking. He is member of the editorial board of Springer journal "Photonic Network Communications". He is active member of the Technical Program Committee of various networking conferences such as INFOCOM, OFC, ICC, Globecom, etc. He was a co-recipient of seven best-paper awards from IEEE conferences.

Charles U. Martel received a B.S. from MIT in 1975 and a Ph.D. from Berkeley in 1980, both in Computer Science. Since 1980, he has been Professor at the UC Davis and was one of the founders of their Computer Science Department. He has been an Emeritus Professor since 2012. He has worked on design, analysis, and with an emphasis on applications of a broad range of combinatorial algorithms. His recent focus is on algorithms for computer networks. As a five time world bridge champion, and member of the American Contract Bridge League hall of fame, he also has an interest in computer bridge playing programs.

Biswanath Mukherjee is Distinguished Professor at University of California, Davis, where he has been a faculty member since 1987 and was Chairman of Computer Science during 1997-2000. He received the BTech degree from Indian Institute of Technology, Kharagpur (1980) and PhD from University of Washington, Seattle (1987). He was General Co-Chair of the IEEE/OSA Optical Fiber Communications (OFC) Conference 2011, Technical Program Co-Chair of OFC'2009, and Technical Program Chair of the IEEE INFOCOM'96 conference. He is Editor of Springer's Optical Networks Book Series. He has served on eight journal editorial boards, most notably IEEE/ACM Transactions on Networking and IEEE Network. In addition, he has Guest-Edited Special Issues of Proceedings of the IEEE, IEEE/OSA Journal of Lightwave Technology, IEEE Journal on Selected Areas in Communications, and IEEE Communications. To date, he has supervised 64 PhDs to completion and currently mentors 18 advisees, mainly PhD students. He is winner of the 2004 Distinguished Graduate Mentoring Award and the 2009 College of Engineering Outstanding Senior Faculty Award at UC Davis. He is co-winner of ten Best Paper Awards from various conferences, including Optical Networking Symposium Best Paper Awards at IEEE Globecom 2007 and 2008. He is author of the graduate-level textbook Optical WDM Networks (Springer, January 2006). He served a 5-year term on Board of Directors of IPLocks, a Silicon Valley startup company (acquired by Fortinet). He has served on Technical Advisory Board of several startup companies, including Teknovus (acquired by Broadcom). He is winner of the IEEE Communications Society's inaugural (2015) Outstanding Technical Achievement Award ”for pioneering work on shaping the optical networking area”. He is an IEEE Fellow.