Multiple Virtual Machine Live Migration in Federated Cloud Systems

Walter Cerroni
Department of Electrical, Electronic and Information Engineering - University of Bologna
via Venezia 52, 47521 Cesena - Italy
E-mail: walter.cerroni@unibo.it

***Abstract*—The idea of computing utility incarnated by the cloud paradigm is gaining a lot of success, both for entertainment and business applications. The consequent increasing demand of computing, storage and communication resources within data centers is fostering new forms of infrastructure sharing such as cloud federations, which can take advantage of virtualization technologies and, in particular, of multiple virtual machine live migration techniques. Such a scenario requires a quantitative characterization of the performance of the inter-data center network infrastructure underlying the cloud federation. In this paper we propose an analytical model useful to dimension network capacity in order to achieve some given performance level in a federate cloud, assuming some simple multiple virtual machine live migration strategies.**

***Index Terms*—Cloud Computing; Inter-Cloud Communication; Virtualization; Virtual Machine Live Migration.**

I. Introduction

Software applications based on the cloud computing paradigm have become very popular in the last few years, both for entertainment and business purposes, producing a plethora of new services — including entire virtual IT infrastructures — that are commonly considered part of "the Cloud" [1]. Such an idea of the cloud as a sort of computing utility has become a reality owing to the recent advances in Data Center (DC) technologies. However, in order to cope with the exponentially increasing number of cloud service subscribers — especially mobile cloud users — more advanced networking infrastructures and technologies are expected to be deployed for both intra-DC and inter-DC communications [2].

Data center processing power over-provisioning may not always be the right answer, as increasing the size of a DC can become a very expensive and energy demanding operation. For this reason, the emerging *federated cloud computing* model is based on the idea of a smart sharing of the workload across the DC resources of multiple cloud providers following some kind of mutual agreement [3]. Before cloud federations become current practice, several issues still remain to be solved, among which the correct design of the inter-DC interconnection network by means of a suitable planning of the communication infrastructure to achieve the required level of quality of service (QoS) [4].

The use of Virtual Machines (VMs) to implement end-user services is one of the key enablers of cloud federations. In fact, decoupling service instances from the underlying processing and storage hardware allows to flexibly deploy any application on any server within any DC independently of the specific operating system used. One of the main advantages is that a VM can be instantiated, cloned, migrated, rolled-back to a previous state without expensive hardware interventions. This is particularly useful in a cloud federation, where VMs can be easily moved from one DC to another as long as hypervisor compatibility is guaranteed. Live VM migration is an additional feature that allows to move services from one server/DC to another with minimal disruption to end-user service availability [5].

Moving VMs to/from different DCs requires to maintain network state consistency. Emerging technologies, such as Software Defined Networking (SDN) [6], offer new opportunities to seamlessly migrate live virtualized environments together with their current network states [7]. This is particularly useful when considering groups of correlated VMs that must be live migrated together while maintaining reciprocal connectivity. In fact, many multi-tier applications are often executed across multiple VMs (e.g., database front-end and back-end components) and the relative services are available to the end-user only when all VMs in the group are active and connected to each other.

While the issue of single VM live migration has been extensively studied in literature and successfully solved and optimized in commercially available hypervisors, the case of multiple VM migration is still to be investigated in detail. Some studies have been carried out to understand the implications of live migrating a group of VMs together with the virtual network interconnecting them [8], whereas other works focused on some optimization aspects [9], [10]. However, to the best of the author's knowledge, a quantitative analysis of the performance of multiple VM migration within a cloud federation is still missing.

In this paper we propose an analytical model that could represent a useful design tool to dimension the inter-DC network capacity in order to achieve some given performance level in a federate cloud, assuming some simple multi-VM live migration strategies. Section II describes the federated cloud network scenario considered by the model. Then, Section III introduces the main parameters of interest for multiple VM live migration, whereas the proposed Markovian model of the inter-DC network is presented in Section IV. After discussing some numerical results in Section V, conclusions are drawn in Section VI.

978-1-4799-3088-3/14/$31.00 ©2014 IEEE

II. Federated Cloud Network Scenario

The general federated cloud network scenario assumed here consists of a number of DCs remotely interconnected by a full mesh of guaranteed-bandwidth network pipes. For instance, such network pipes could be implemented as MPLS Label Switched Paths (LSPs) or as lightpaths established between the edge nodes of an inter-DC optical network. Those LSPs or lightpaths are assumed to be established according to a long-term network resource planning strategy (e.g., by means of well-known routing and wavelength assignment techniques [11]). It is reasonable to assume that some QoS requirements (e.g., minimum bandwidth) must be guaranteed within a cloud federation network, to be able to control the performance of the inter-cloud communications.

Let us consider the case of a set of correlated VMs, currently running on a given DC, that must be migrated, e.g. for load balancing, energy saving, server consolidation or DC maintenance reasons. This situation generates a *migration request* towards the federated cloud management system, which is in charge of finding a suitable DC where the VMs can be hosted. In principle, any DC within the cloud federation is suitable to host the set of VMs to be migrated. However, it would be more realistic to assume that only a subset of the DCs are actually able to receive the workload of the set of VMs to be migrated. This can be true for a number of reasons: for instance, not all DCs may have the availability of the specific computing or storage resources required by the given VM set; or maybe the specific kind of services provided by the VMs has some latency requirement that cannot be satisfied if they are migrated to a DC located too far from their final users; also, not all DCs in a cloud federation are equivalent in terms of energy savings or maintenance schedules; last but not least, load balancing reasons may force to choose some DCs instead of others.

Therefore, in this work we assume that the generic request z of migrating a group of M_z VMs can be satisfied by a specific set of resources C_z, which are available in a subset of the $n+1$ DCs that constitute the cloud federation. In general, different requests may need different resource sets. We assume that the m_z resource set instances $C_z^1, C_z^2, \ldots, C_z^{m_z}$ available in the cloud federation are randomly distributed over the n remote DCs, which are reachable from the source DC via as many established network pipes $L_1, L_2, \ldots, L_n$. Any of the m_z resource set instances is equivalent for hosting the VMs, according to the general *anycast* service model.

Within the federated cloud management system, each migration request z is associated to the resource set C_z by means of an anycast address, that is then translated into:

- the set of network pipes between the source DC and any possible location of C_z instances;
- the amount of network pipe capacity b_z needed to migrate the M_z VMs, as requested by z.

Then the management system is in charge of finding the location of the most suitable instance of C_z, namely C_z^x. We assume that the choice is made based on the availability of the required capacity b_z towards the location of C_z^x. More specifically, let B_i be the total amount of capacity guaranteed on network pipe L_i, $i = 1, 2, \ldots, n$, and $B_{\mathrm{a},i}$ the capacity currently available on L_i, i.e. the network pipe capacity not used by other data transfers between the source DC and the i-th remote DC. Any network pipe towards a remote DC hosting at least one of the m_z resource set instances and such that $B_{\mathrm{a},i} \geq b_z$ is equivalent for the purpose of our model and can be used to migrate the M_z VMs. A migration request z is blocked when there is not enough capacity available between the source DC and any remote DC where the C_z resource set instances are located.

The purpose of the model described in this paper is to evaluate the performance of the aforementioned network scenario and to provide a useful design tool for network pipe dimensioning, considering the specific nature of VM migration traffic. As a first approximation and in order to keep our model simple enough to be tractable, we assume the following hypotheses:

- **H.1** each request z needs to migrate the same number $M_z = M$ of VMs;
- **H.2** each multi-VM migration request is given the same amount of guaranteed capacity $b_z = b$ to be executed;
- **H.3** each network pipe provides the same total amount of capacity $B_i = B$;
- **H.4** each DC has the capacity of hosting up to k resource set instances;
- **H.5** each migration request is allowed to choose among the same number $m_z = m$ of resource set instances, which are uniformly distributed over the n remote DCs (considering the general case when multiple instances of the same resource set could be available in the same DC).

The migration request blocking model is obtained in two steps. First, we derive an exact formulation of the multiple VM migration time, so that we can compute how long each migration request consumes the given amount of capacity (section III). Then, we use the previous result to characterize the service time of a simple Markov chain that approximates the current state of the network pipes and allows to compute the migration request blocking probability (section IV).

III. Modeling Multiple VM Migration

The main advantage of live migration, i.e. of moving the VM from one hosting server to another while it is still running, is that the current state of VM kernel and running processes is maintained and the migration procedure has minimum impact on the end-user service availability. It also reduces the risk of inconsistencies due to duplicate VM instances running simultaneously on both the source and destination DCs.

Of course, any live migration procedure must ensure consistency of the VM state before and after the transfer to the new hosting server, especially when a group of correlated VMs must be migrated together. In particular, consistency must be maintained for memory, storage and network states. In this work we focus mainly on the memory migration issue. Storage

synchronization, if needed, must be completed before the live migration starts, so it is not taken into account here, although it may cause the presence of non-negligible background traffic. As for the network state consistency, the emerging SDN paradigm allows the adoption of different solutions to migrate virtual networks from one DC to another [8].

The most typical memory-oriented live migration technique is the so-called *pre-copy* strategy [5], which is currently adopted by many virtualization systems such as Xen and KVM [12]. The migration starts by transferring a first snapshot of the whole VM memory, while the VM is still running. During such transfer time, some memory pages can be modified by the running processes. Therefore, the memory migration enters an *iterative push phase*, where "dirty" memory pages (i.e., pages modified during a given transfer) are transmitted again during the next round, until the total size of dirty pages is below a given threshold or a maximum number of iterations is reached. After that, the *stop-and-copy phase* takes place: the VM is suspended at the source host and the remaining dirty pages are copied to the destination. Finally, during the *resume phase* the VM is brought back on-line at the destination host with consistent memory and network state.[1]

The two key performance parameters that are typically considered in the single VM migration process are the so-called *downtime* (T_{down}) and *total migration time* (T_{mig}). The former is defined as the amount of time the VM is paused during the transfer and it measures the impact of the migration on the end-user's perceived quality of the offered service. Keeping the downtime as small as possible helps to make the migration process look transparent to the end-user, even for time-critical services such as audio/video streaming and on-line gaming. On the other hand, the total migration time is also very important because it measures the impact of the migration process on the whole cloud federation infrastructure: in fact, the network pipe capacity consumed for transferring the VM as well as the computing resources reserved by the VM in both source and destination DCs are busy during the whole migration phase and cannot be used to perform other tasks.

A simple model used to evaluate T_{down} and T_{mig} in case of a single VM migrated with the pre-copy strategy was proposed in [13]. Based on this model, a generalized extension to the case of multiple VM migration was presented in [14]. In the latter case, the definition of the performance parameters depends on how the multiple VMs are scheduled for the live transfer and their mutual interactions when providing the service to the end-user. Here we briefly recall the multiple VM migration model formulation under the following assumptions:

- **A.1** the M VMs to be migrated as per request z are allotted the same amount of memory V_z;
- **A.2** the applications running on the VMs show the same constant memory page dirtying rate D;
- **A.3** all VMs have the same memory page size P;
- **A.4** the bit rate R_j used to transfer the j-th VM in the requested set is constant during the whole migration process; this is the amount of network pipe capacity b dedicated to transfer VM j, i.e., $R_j \leq b \ \forall j = 1, \ldots, M$.

We are aware that assumptions **A.1** and **A.2** may not be completely true in a real-word scenario, since the memory profile of each VM strongly depends on the specific applications that are being executed. However, these assumptions allow to simplify the equations and to capture the macroscopic performance aspects of multiple VM live migration. A more refined model is left for future work.

Let $T_{i,j}^{(z)}$ be the time needed to complete the i-th iteration in the push phase of VM j migration as per request z. From the general equations derived in [14] that describe the iterative migration process of each VM $j = 1, \ldots, M$, we can write the time needed to migrate the j-th VM and the number of iterations required as:

$$T_{\mathrm{mig},j}^{(z)} = \sum_{i=0}^{n_j} T_{i,j}^{(z)} = \frac{V_z}{R_j} \frac{1-\gamma_j^{n_j^{(z)}+1}}{1-\gamma_j} \quad (1)$$

$$n_j^{(z)} = \min\left\{ \left\lceil \log_{\gamma_j} V_{\mathrm{th}}/V_z \right\rceil, n_{\max} \right\} \quad (2)$$

where $\gamma_j = (P\,D)/R_j < 1$, since the pre-copy migration algorithm is sustainable as long as the average memory dirtying rate is smaller than the transfer rate.

The computation of the total migration time and downtime of a set of VMs strictly depends on how and when the VMs are scheduled to be migrated. In the following we consider two simple cases: (i) when the M VMs are transferred one at a time (*sequential migration*); (ii) when all the M VMs are transferred simultaneously (*parallel migration*). A useful parameter is the ratio between the dirtying rate and the network pipe bit rate $\gamma = (P\,D)/b$.

When the M VMs are migrated one at a time, each transfer is performed at full network pipe capacity, i.e. $R_j = b, \ \forall j$. In this case, $\gamma_j = \gamma$, $n_j^{(z)} = n_{\mathrm{s}}^{(z)} \ \forall j$, and the sequential migration time of the whole VM set z is given by

$$T_{\mathrm{s-mig}}^{(z)} = \sum_{j=1}^{M} T_{\mathrm{mig},j}^{(z)} = M\,\frac{V_z}{b}\,\frac{1-\gamma^{n_{\mathrm{s}}^{(z)}+1}}{1-\gamma} \quad (3)$$

The downtime of the whole VM set starts when the first VM is stopped at the source host (i.e., when the last iteration of the first VM begins) and ends when the last VM is resumed at the destination host. If T_{res} is the fixed time required to resume a VM at the destination host, the sequential migration downtime can be computed as

$$T_{\mathrm{s-down}}^{(z)} = \frac{V_z}{b}\,\gamma^{n_{\mathrm{s}}^{(z)}} + (M-1)\,\frac{V_z}{b}\,\frac{1-\gamma^{n_{\mathrm{s}}^{(z)}+1}}{1-\gamma} + T_{\mathrm{res}} \quad (4)$$

When the Mz VMs are migrated simultaneously, each one of them is transferred at a bit rate that depends on how the network pipe capacity is shared among the on-going connections. Assuming an equal share of the channel capacity and

[1]Network state can be either migrated during the resume phase or cloned at the beginning of the push phase, as suggested by [8]. The former solution should be more straightforward, whereas the latter seems to be more efficient in terms of latency and network load.

considering that all the VMs in set z have the same memory profile, all VMs start and end their migration at the same instants. Therefore, there are always M simultaneous transfers and the transfer rate seen by each VM is $R_j = b/M,\ \forall j$. In this case, $\gamma_j = M\gamma$, $n_j^{(z)} = n_{\mathrm{p}}^{(z)}\ \forall j$, and the parallel migration time of the whole VM set is equivalent to the migration time of any single VM, given by

$$T_{\mathrm{p-mig}}^{(z)} = T_{\mathrm{mig},j}^{(z)} = M\,\frac{V_z}{b}\,\frac{1-(M\gamma)^{n_{\mathrm{p}}^{(z)}+1}}{1-M\gamma} \tag{5}$$

The parallel migration downtime of the whole VM set corresponds to the last iteration (stop-and-copy phase) of any single VM and is given by

$$T_{\mathrm{p-down}}^{(z)} = M\,\frac{V_z}{b}\,(M\gamma)^{n_{\mathrm{p}}^{(z)}} + T_{\mathrm{res}} \tag{6}$$

As proved in [14], $T_{\mathrm{p-mig}}^{(z)} \geq T_{\mathrm{s-mig}}^{(z)}$ and $T_{\mathrm{p-down}}^{(z)} \leq T_{\mathrm{s-down}}^{(z)}$, meaning that, while parallel migration is better than sequential migration in terms of end-user service provisioning because the downtime is smaller, sequential migration shows a smaller total transfer time and thus is better than parallel migration in terms of communication and computing resource usage and transmission overhead.

IV. Markovian Model of Inter-DC Network

According to the federated cloud network scenario described in section II, migration request z is refused when other ongoing transfers consume all the capacity on network pipes reaching the remote DCs where the m instances of resource set C_z are located. In order to compute the migration request blocking probability, we need to determine which states of the inter-DC network are such that all the resource set instances are located within unreachable DCs. To this purpose, we can represent the state of the network as the number r of ongoing VM group migrations originated by a given source DC.

Being B the total capacity of each of the n network pipes connecting the source DC to the remote ones and b the capacity consumed by each group migration, the maximum number of simultaneous transfers that each network pipe is able to carry is $h = \lfloor B/b \rfloor$, and the total number of possible simultaneous transfers originating from the the source DC is nh. Therefore the states of the network as seen by the source DC are $r = 0, 1, \ldots, nh$.

If we assume that the VM group migration requests follow a Poisson arrival process, a Markov chain describing the network state evolution can be defined using the following parameters:[2]

- λ: request arrival rate;
- μ: service rate;
- $A_0 = \lambda/\mu$: load offered to the inter-DC network as seen by the source DC.

The service rate is given by the reciprocal of the average group migration time, as computed in section III. Therefore, we can define two possible values of μ, depending on the multi-VM migration strategy adopted (either sequential or parallel), as

$$\mu_{\mathrm{s}} = \frac{1}{E\left[T_{\mathrm{s-mig}}^{(z)}\right]} \qquad \text{or} \qquad \mu_{\mathrm{p}} = \frac{1}{E\left[T_{\mathrm{p-mig}}^{(z)}\right]}$$

where the migration time must be averaged over all migration requests z according to the statistical distribution of V_z.

The inter-DC network, as seen by the source DC network, is therefore modeled as a loss system where the number of servers is equal to nh and migration request blocking events may occur in a generic state r, depending on the requested resource set instance locations. In fact, the transition to state $r+1$ occurs only when at least one of the m requested resource set instances is placed in a DC reachable via a network pipe i with available capacity $B_{\mathrm{a},i} \geq b$, otherwise the request is blocked. Let $P_{b|r}$ be the blocking probability in state r. Transitions from state r to state $r+1$ occur with rate $\lambda_r = (1 - P_{b|r})\,\lambda$, whereas transitions from state r to state $r-1$ occur with rate $r\mu$.

To obtain $P_{b|r}$, first we need to analyze the combinatorial behavior of the anycast approach. Considering hypotheses **H.4** and **H.5** from section II, when the number of fully occupied network pipes is ℓ, the probability of choosing the m instances among the ℓk unreachable ones out of the total nk possible choices is given by

$$p(m|\ell) = \prod_{i=0}^{m-1} \frac{\ell k - i}{nk - i} \qquad \ell = 1, \ldots, n \tag{7}$$

Then, to correctly use formula (7), we have to consider all the possible sub-states of state r, i.e. all the possible ways the r ongoing migrations can be distributed over the n remote DCs. This means we must find all the possible partitions of r into up to $\min\{n, r\}$ positive terms not greater than h. In principle, a more complex Markov chain including all sub-states of each state r should be solved in order to obtain the exact sub-state probabilities. However, this approach may become impractical, as the number of sub-states quickly becomes very large. Therefore, we decided to approximate the sub-state probabilities considering only the "forward" evolution of the state, i.e. by recursively computing the sub-state probabilities of state r from those of state $r-1$.

As an example, consider the case when $n = 3$, $h = 3$ and $r = 0, 1, \ldots, 9$. Figure 1 shows the possible sub-states of the first 6 states and the probabilities of moving from a sub-state to another when a new request arrives, based on the location of the chosen instance. When the system is in sub-state (1), a new request is always accepted and the probability that the next sub-state is (2) is the probability of choosing a resource set instance located in one out of three DCs, i.e. $1/3$. Therefore, the sub-state probabilities, given state $r = 2$, are $s(2|2) = 1/3$ and $s(1,1|2) = 2/3$. Then, when moving to state $r = 3$, the sub-state probabilities are $s(3|3) = 1/3\,s(2|2) = 1/9$, $s(2,1|3) = 2/3\,s(2|2) + 2/3\,s(1,1|2) = 2/3$ and $s(1,1,1|3) = 1/3\,s(1,1|2) = 2/9$. If we are in sub-state (3), one of the

[2]Since this Markov chain, representing a system without a queue, can be classified as a loss system, the average service time is the only parameter affecting the blocking performance [15]. Therefore, the model is valid for any service time distribution with a finite mean.

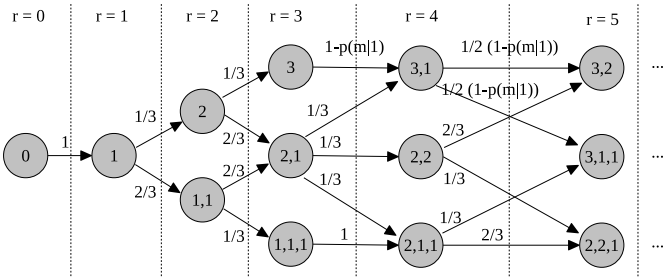

Fig. 1. Possible sub-states of states $r = 0, \ldots, 5$ and related "forward" transition probabilities, for the case $n = 3$ and $h = 3$.

remote DC has a full network pipe and a new request is blocked with probability $p(m|1)$, otherwise we move forward to sub-state $(3, 1)$. We must take into account the chance of a missing transition when we compute the sub-state probabilities for state $r = 4$, by normalizing to the sum of all the possible sub-state transitions, e.g.

$$s(3,1|4) = \frac{(1 - p(m|1))\, s(3|3) + 1/3\, s(2,1|3)}{(1 - p(m|1))\, s(3|3) + s(2,1|3) + s(1,1,1|3)}$$

We can compute the blocking probability in state r by averaging $p(m|\ell)$ over all sub-states such that at least one DC has a full network pipe, e.g. for $r = 6$:

$$P_{b|6} = p(m|2)\, s(3,3|6) + p(m|1)\, s(3,2,1|6)$$

It is possible to generalize the above computation in a recursive form that is easy to compute. We omit the resulting formulation here due to space limits, and will include it in an extended future work.

Then by solving the Markov chain of states r, we obtain the following steady-state probabilities:

$$P_1 = P_0 A_0, \qquad P_r = P_0 \prod_{\ell=1}^{r-1} \left(1 - P_{b|\ell}\right) \frac{A_0^r}{r!} \;\; 2 \le r \le nh$$

Finally, the total request blocking probability can be obtained by adding the contributions from each state, resulting in

$$P_b = \sum_{r=1}^{nh} P_{b|r}\, P_r \tag{8}$$

V. Numerical Results

In this section we present some numerical results obtained with the federated cloud network model introduced above. Unless explicitly mentioned, the charts included in this section show the performance trends as a function of the migration request arrival rate when the model parameters are assigned the reference values reported in Table I. The VM memory size V_z follows a bimodal distribution: $V_z = V_0$ with probability 0.75 and $V_z = 4\,V_0$ with probability 0.25.

Figures 2 and 3 show the migration request blocking probability as a function of the arrival rate λ, for different values of the total network pipe capacity B, respectively assuming a sequential and a parallel migration strategy. The curves have been obtained by applying the proposed model, whereas the points correspond to measurements obtained with an event-based simulator. The model, although approximate, shows a quite good match with simulation results, with a slight overestimation of the blocking probability.

TABLE I
Reference values for model parameters

$M = 8$	$P = 4\,\text{KB}$	$B = 4\,\text{Gbps}$
$b = 1\,\text{Gbps}$	$V_{\text{th}} = 100\,\text{MB}$	$n = 5$
$V_0 = 1\,\text{GB}$	$n_{\max} = 8$	$m = 3$
$D = 2500\,\text{pps}$	$T_{\text{res}} = 100\,\text{ms}$	$k = 8$

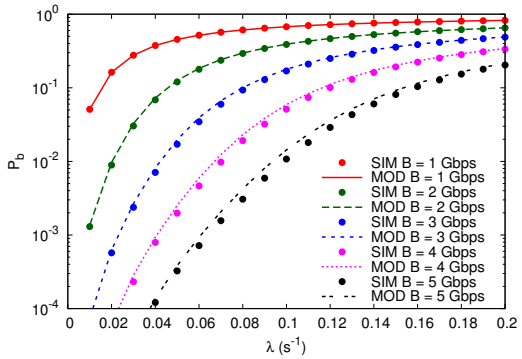

Fig. 2. Migration request blocking probability as a function of the arrival rate, for different values of the total network pipe capacity and assuming a sequential migration strategy.

The results clearly show that parallel migration has a more detrimental effect on the inter-DC network performance than sequential migration. This is due to the higher migration time experienced by VMs transferred simultaneously, and the root cause is the higher number of iterations needed because of the reduced transfer bit-rate — due to network pipe capacity sharing — while the memory dirtying rate remains the same [14]. The worse performance of parallel migration from the cloud federation management perspective is the price to pay

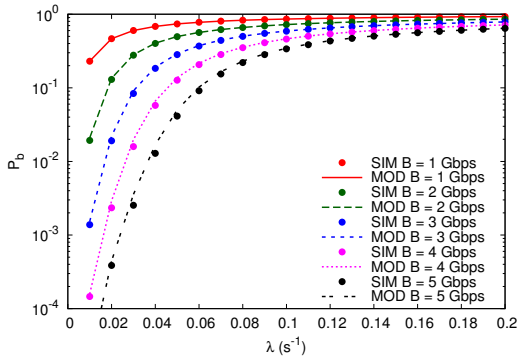

Fig. 3. Migration request blocking probability as a function of the arrival rate, for different values of the total network pipe capacity and assuming a parallel migration strategy.

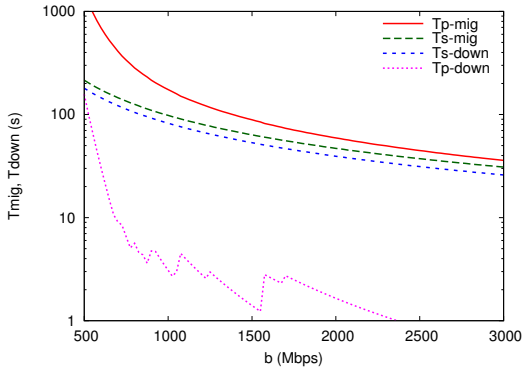

Fig. 4. Total transfer time and downtime of both sequential and parallel multi-VM migration strategies as a function of the network pipe slice capacity reserved to each transfer. $B = 3\,\mathrm{Gbps}$.

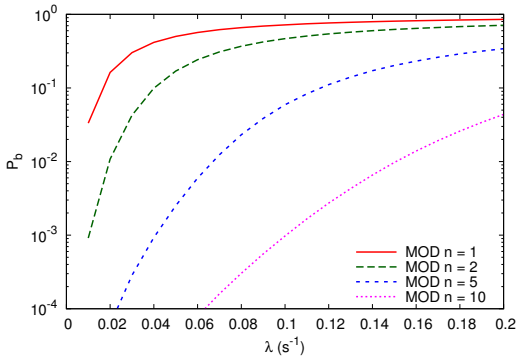

Fig. 5. Migration request blocking probability as a function of the arrival rate, for different values of the number of remote DCs and assuming a sequential migration strategy.

to keep the total downtime as small as possible, as shown in Fig. 4. In this sense, the proposed model allows: (i) to quantify the existing trade-off between sequential migration, which tends to reduce the network resource occupancy, and parallel migration, which is better for end-user's perceived quality; and (ii) to properly dimension the network pipe capacity according to both perspectives.

Another important aspect that the proposed model allows to quantify is the impact of the size of the cloud federation. To this purpose, Fig. 5 shows how the request blocking rate can be reduced by increasing the number of DCs that are interconnected by the federated cloud network. Obviously, the performance improvement is a direct consequence of the higher number of resources available when n increases, as there are more DCs capable of hosting the VMs to be migrated. However, increasing the size of the federated cloud network may have a significant infrastructure cost: the proposed model can also help in finding a good cost/performance trade-off, given that a complementary cost model is devised. Figure 5 refers to the sequential migration case only, but similar trends are obtained for the parallel migration case as well, although the absolute values of P_b are higher.

Many other aspects can be quantitatively analyzed with the proposed model, but they are omitted here due to the limited space available.

VI. Conclusion

In this paper we presented a model to evaluate the performance of an inter-DC network for cloud federations, assuming the network load is caused by live migration of a group of VMs cooperating to provide the end-user with a given service. After characterizing the VM group migration time for the two alternatives of sequential and parallel migration strategies, we evaluated how the former one has a less detrimental effect on network performance, although parallel migration results in a much smaller service downtime. The model can be used to properly dimension the inter-DC network capacity trading-off resource usage and service availability. Although some hypotheses used to derive the model may not be completely realistic, the obtained results give an interesting insight to the macroscopic performance of a federated cloud network. We plan to release some of these hypotheses in a future work and derive a more general formulation, taking into account also real DC traffic profiles.

References

[1] C. N. Höfer and G. Karagiannis, "Cloud computing services: taxonomy and comparison," *Journal of Internet Services and Applications*, Springer, Vol. 2, No. 2, pp. 81-94, September 2011.

[2] M. Chen, H. Jin, Y. Wen, V. C. M. Leung, "Enabling Technologies for Future Data Center Networking: A Primer," *IEEE Network*, Vol. 27, No. 4, pp. 8-15, July/August 2013.

[3] B. Rochwerger et al., "Reservoir - When One Cloud Is Not Enough," *IEEE Computer*, Vol. 44, No. 3, pp. 44-51, March 2011.

[4] BTI Systems, *Data Center Interconnect*, accessed February 2014. http://www.btisystems.com/solutions/data-center-interconnect.aspx

[5] C. Clark, et al., "Live Migration of Virtual Machines," *Proc. of the 2nd USENIX Symposium on Networked Systems Design & Implementation (NSDI)*, Boston, MA, May 2005.

[6] The Open Networking Foundation, "Software-Defined Networking: The New Norm for Networks," *ONF White Paper*, April 2012.

[7] R. Jain, S. Paul, "Network Virtualization and Software Defined Networking for Cloud Computing: A Survey," *IEEE Communications Magazine*, Vol. 51, No. 11, pp. 24-31, November 2013.

[8] E. Keller, S. Ghorbani, M. Caesar, J. Rexford, "Live Migration of an Entire Network (and its Hosts)," *Proc. of the 11th ACM Workshop on Hot Topics in Networks, (HotNets-XI)*, Redmond, WA, October 2012.

[9] K. Ye, X. Jiang, R. Ma, F. Yan, "VC-Migration: Live Migration of Virtual Clusters in the Cloud," *Proc. of 13th ACM/IEEE International Conference on Grid Computing (GRID 2012)*, Beijing, China, September 2012.

[10] U. Deshpande, X. Wang, K. Gopalan, "Live Gang Migration of Virtual Machines," *Proc. of the 20th International Symposium on High Performance Distributed Computing (HPDC '11)*, San Jose, CA, June 2011.

[11] B. Mukherjee, *Optical WDM Networks*, Springer, 2006.

[12] K. Z. Ibrahim, S. Hofmeyr, C. Iancu, E. Roman, "Optimized Pre-Copy Live Migration for Memory Intensive Applications," *Proc. of the International Conference of High Performance Computing, Networking, Storage and Analysis (SC)*, Seattle, WA, November 2011.

[13] H. Liu, H. Jin, C.-Z. Xu, X. Liao, "Performance and Energy Modeling for Live Migration of Virtual Machines", *Cluster Computing*, Springer, Vol. 16, No. 2, pp. 249-264, June 2013.

[14] F. Callegati, W. Cerroni, "Live Migration of Virtualized Edge Networks: Analytical Modeling and Performance Evaluation," *Proc. of IEEE Workshop on Software Defined Networks for Future Networks and Services (SDN4FNS 2013)*, Trento, Italy, November 2013.

[15] F. P. Kelly, "Loss networks", *The Annals of Applied Probability*, Vol. 1, No. 3, pp. 319-378, 1991