Delay Guaranteed Live Migration of Virtual Machines

Jiao Zhang, Fengyuan Ren and Chuang Lin
Dept. of Computer Science and Technology, Tsinghua University, Beijing, China
Tsinghua National Laboratory for Information Science and Technology, Beijing, China
Email: {zhangjiao08, renfy, clin}@csnet1.cs.tsinghua.edu.cn

***Abstract*—The proliferation of cloud services makes virtualization technology more important. One important feature of virtualization is live Virtual Machine (VM) migration, which can be employed to facilitate load balancing, fault management and server maintenance etc.. Two main metrics of evaluating a live VM migration mechanism are *total migration time* and *downtime*. The existing literature on live VM migration mainly focus on designing migration mechanisms to shorten these two metrics or making a tradeoff between them. Few of them can be applied to the applications with delay requirements, such as, delay-sensitive web services or a VM backup process that needs to be done in a specific time. This will not only negatively impact the user experiences, but also reduce the profit of cloud service providers. Besides, the frequently varied bandwidth required by the widely used pre-copy mechanism is difficult to be provided by current network technologies. In this work, we theoretically analyze how much bandwidth is required to guarantee the total migration time and downtime of a live VM migration. We first propose a deterministic-based model as a simple example, then assume that the dirtying frequency of each page obeys the bernoulli distribution. At last, we analyze the statistic features of the typical workload running in a VM and build a reciprocal-based workload model, and theoretically give the required bandwidth value to satisfy the performance metrics of a live VM migration. The experimental results demonstrate that the bandwidth obtained from the reciprocal-based model can guarantee the expected total migration time and downtime.**

***Index Terms*—Virtualization, Live Migration, Delay, Bandwidth**

I. Introduction

Virtualization has been firstly employed in IBM VM/370 [1], [2]. The technology can increase hardware utilization by allowing multiple isolated Operating System (OS) to run in the same host. *Migration* allows a running OS to be moved from one physical host to another. The combination of virtualization and migration, VM migration, is increasingly utilized in today's multi-tenant cloud data centers. VM migration brings lots of benefits. It facilitates to balance load and consolidate servers by migrating VMs out of overloaded or underloaded physical hosts. Also, once a server requires hardware maintenance or software update, the services running on it can be migrated to others [3], [4]. In the multi-tenant clouds, resource can be redistributed through live migration to allow more tenants [5], [6].

There are two main methods of migrating a VM: offline migration and online/live migration. If the offline migration is conducted, the services running on the migrated VM will terminate during the whole migration process, while live migration mechanisms can keep the services on the migrated VM alive during most of the migration process. The main advantages of live migration are that it not only endows the benefits of VM migration, but also imposes little interruption on the running services [7].

Two important metrics are used to evaluate the performance of a live VM migration mechanism [4]. 1) *Total migration time*, which is the duration of the whole process of a live migration. 2) *Downtime*, which is the interruption period of the services running in a migrated VM. The performance of live VM migration is mainly related to two factors: page dirtying rate and bandwidth for migration [8]. The page dirtying rate depends on the workload running in the migrated VM, which cannot be changed. Hence, allocating proper bandwidth is critical to improve the performance of live migration of VMs.

At present, the widely employed live migration algorithm is *pre-copy* mechanism proposed by C. Clark [4], in which a heuristic bandwidth allocation mechanism is designed. The main idea is that the bandwidth used for migration is the summation of the page dirtying rate plus a constant increment. Since the page dirtying rate varies as the dirty pages change and the constant bandwidth increment cannot adapt to dynamic migrating scenarios, the heuristic bandwidth allocation algorithm fails to provide guaranteed performance. Previous studies have demonstrated that the downtime varies considerably among applications with different memory usage patterns, ranging from 60 milliseconds when migrating a Quake game server [4] to 3 seconds in the case of High-Performance Computing benchmarks [9].

The uncontrollable total migration time and downtime could bring much penalty. For example, in multi-tenant clouds, sometimes the VMs allocated to the current tenants need to be migrated to make more efficient use of substrate resources and thus allow more tenants [5], [6]. Too large total migration time prolongs the wait time of a new tenant. Also, large downtime will affect the Service Level Agreement (SLA) of services and reduce the profit of the could service providers [10]. Furthermore, the required bandwidth changes in each iteration in the pre-copy mechanism. During the last several rounds, the bandwidth likely varies at the interval of no more than 1 second, which is so small that the current network technologies hardly ensure the required bandwidth.

To overcome the drawbacks of unstable performance of the

978-1-4799-3360-0/14/$31.00 ©2014 IEEE

978-14799-3360-0/14/$31.00 ©2014 IEEE

pre-copy live VM migration mechanism, the page dirtying rate pattern should be investigated and be taken into consideration to determine a constant bandwidth value during the pre-copy phase. Then some existing bandwidth guarantee mechanisms can be leveraged to reserve the bandwidth and thus provide guaranteed performance.

In this work, we theoretically analyze how much bandwidth should be provided to guarantee the total migration time and downtime of a live VM migration. First, a simple example is given under the assumption that the dirtying probability of all the pages is deterministic. Then the dirtying probability of all the pages is assumed to obey the bernoulli distribution [11]. At last, the statistic features of the typical workloads are analyzed and a reciprocal-based page dirtying model is proposed. The bandwidth for live VM migration with delay requirements are obtained for all the three models.

Our mechanism is evaluated using experiments. The results show that the bandwidth obtained from the reciprocal-based model can guarantee the delay requirements of both the total migration time and downtime, while the deterministic-based and bernoulli-based models are not proper.

The remainder of the paper is organized as follows. The following section describes the related work and motivation. The problem addressed in the work is formalized in Section III. The required bandwidth to guarantee the delay requirements of live VM migrations with the same or different page dirtying distribution functions is theoretically deduced in Section IV and Section V, respectively. In Section VI, we validate our model using experiments. At last, the paper is concluded in Section VII.

II. Related Work and Motivation

A. Related Work

1) Pre-Copy Migration: C. Clark *et al.* proposed a live VM migration mechanism, named *pre-copy* migration, in which the OS execution is not interrupted during most of the transfer [4]. In their algorithm, a tradeoff between the downtime and the total migration time is made through dynamic bandwidth allocation. In the first round, all the pages are transferred using the minimum bandwidth. The bandwidth used for subsequent rounds equals the summation of the dirtying rate of the previous round and a constant increment, (eg. 50Mbps in [4]). To avoid too much bandwidth is used for migration, a maximum bandwidth value is given. The transmission of the live migration employs TCP protocol.

The migrated VM stops its running applications, and the stop and copy phase starts if one of the four conditions satisfies: a) The transmission rate reaches up to the maximum bandwidth and the number of pages required to be sent in the next iteration is larger than that of the current iteration. b) The iteration number exceeds the maximum allowed number, e.g., 30. c) The number of dirty pages at the end of the current iteration is smaller than 50 pages. d) The number of the total sent pages exceeds the maximum allowed value.

The performance of the pre-copy based live migration algorithm is closely related to the three parameters: the minimum, maximum bandwidth and the bandwidth increment. Their default values are 100 Mbps, 500 Mbps and 50 Mbps, respectively. It is not clear how to change the three parameters in different network environments. Besides, there is no performance guarantee in terms of the downtime and total migration time.

2) Performance of the Pre-Copy Mechanism: In [3], the impact of the pre-copy live migration algorithms on applications running inside migrated VMs is evaluated. The experimental results show that in most cases, the overhead caused by migration is acceptable but cannot be disregarded, especially in the systems where service availability and responsiveness are governed by strict SLAs. However, the work does not present how to satisfy the SLAs of applications.

S. Akoush *et al.* show that link bandwidth and page dirtying rate are the major factors impacting migration behavior, and highlight the migration performance can vary considerably with different workload. They present two simulation models that can predict the performance of VM migration within 90% accuracy for both synthetic and real-world benchmarks [8]. However, how to employ the prediction results to guide the resource allocation and thus satisfy the requirements of services is not discussed.

D. Breitgand *et al.* proposed a model to quantify the tradeoff between minimizing the copy duration and maintaining an acceptable quality of service during the pre-copy phase [11]. The dirty frequency of all the pages are assumed to obey bernoulli distribution. In [15], F. Checconi *et al.* introduced a theoretical framework for estimating the variability of the duration of the overall migration time and downtime under the live migration policy proposed in [4]. The work shows an optimal ordering of pages under the assumption that the bandwidth is constant and the dirty frequency of each page follows the bernoulli distribution. Since the optimal order is difficult to be scheduled in practice, two heuristic migrating policies are proposed. However, no performance bound can be guaranteed by the proposed policies.

B. Motivation

The pre-copy algorithm works well at the premise that the network can provide frequently varied bandwidth. Figure 1 plots the bandwidth taken by the migration process without any background traffic. The experiment is conducted in a testbed which is comprised of 4 DELL OptiPlex 360 PCs with Intel 2.93 GHz dual-core CPU, 6 GB DRAM, and 300 GB hard disk, and a HP ProCurve 2910al Ethernet switch. The system is XEN-3.4.3 [12] with kernel 2.6.18.8-xen [13]. The migrated Domain-U VM has 400 MB memory and is compiling kernel during the migration process. We can see that without any background traffic, the variation of the transfer rate is similar to the results in [4], which starts from the minimum sending rate 100 Mbps, and then increases to the maximum sending rate 500 Mbps. After five iterations, the pre-copy phase terminates and the stop and copy phase begins. The total migration time is about 34 seconds and the downtime is 605 microseconds, which is acceptable. However,

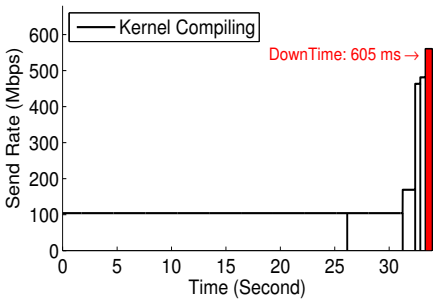

Fig. 1. Send rate variation during migrating a VM which is compiling kernel without any background traffic.

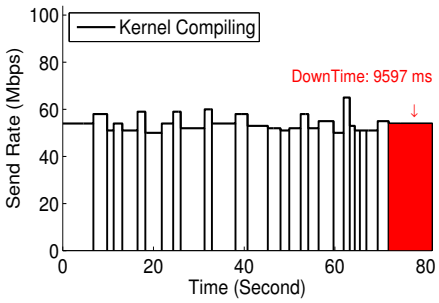

Fig. 2. Send rate variation during migrating a VM which is compiling kernel with 900 Mbps UDP background traffic.

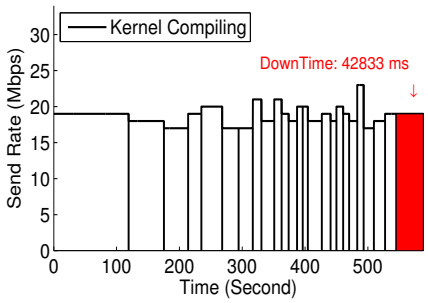

Fig. 3. Send rate variation during migrating a VM which is compiling kernel with 40 TCP flows background traffic.

the bandwidth changes quite frequently at the last several rounds. For example, the 4-th and 5-th iterations last only no more than half a second.

In data centers, except from the live migration traffic, there are many other kinds of traffic, which contends for bandwidth and thus influences the performance of live migration.

With UDP background traffic. For example, web search is an important service in data centers and attracts a lot of attention recently [16]–[20]. To achieve low latency, the web search traffic is given higher priority to obtain the required bandwidth [17], [18] or is scheduled first at the switches [20], [21]. It is possible that most of the link bandwidth is taken by the deadline-aware flows. To emulate this kind of situation, we let two servers generate 900 Mbps UDP background traffic and then migrate a VM that has 400 MB memory and is compiling kernel. Figure 2 plots the send rate values of each iteration. *The total migration time and downtime increases to 80 seconds and 9.6 seconds, respectively, which is quite large.*

Contending with other TCP flows. As the number of flows increases, the bandwidth taken by the live migration process becomes quite small even if all the flows fairly contend for the link bandwidth. In data centers, a rack typically has 40 servers. Figure 3 shows the send rate variation during the live migration with 40 TCP background flows. Since the live migration flow competes the bandwidth with the 40 TCP flows, the bandwidth taken by the live migration process is only about 20 Mbps. Thus, *the total migration time achieves up to more than 600 seconds, and the downtime increases to 58.8 seconds.*

The pre-copy mechanism assumes that the network can provide sufficient bandwidth for live migration of VMs. Otherwise, the total migration time and downtime will possibly become quite large as shown in Figures 2 and 3. However, it is not easy to provide the frequently varied bandwidth. The live migration results shown in both [4] and Figure 1 indicate that the bandwidth is changed every about 1 second during the last several rounds of the live migration. However, the network needs some time to reserve the new bandwidth value. IntServ [22] is designed to provide bandwidth guarantee services in Internet. RSVP protocol [23] is used to establish the switch state for the varied bandwidth and the switches utilize the packet scheduling mechanism, such as WFQ, priority-queue, to guarantee bandwidth. SecondNet employs priority-queue to provide bandwidth guarantee in data center networks [6]. From the experiments in SecondNet, we can infer that one variation of the reserved bandwidth needs about several seconds. Therefore, it is unpractical to provide the frequently changed bandwidth using the present network technologies.

Of course, the maximum bandwidth can be reserved during the whole migration process. However, how much the maximum bandwidth should be? A large value will cause much bandwidth wastage, while a small value will lead to long migration time. In [24], Y. Wang *et. al.* also found that the migration delay dramatically increases due to the bandwidth contention with other traffic. They suggest that the operators should provide separate bandwidth for the migration traffic. This method will not only incur bandwidth wastage but also brings management overhead. Therefore, it is necessary to determine a proper bandwidth value to satisfy the requirement of the total migration time and downtime before the migration starts. Correspondingly, for users, the SLA of services can be guaranteed. For network operators, the reservation of the bandwidth becomes practical and incurs less cost. For cloud service providers, loss of profit can be reduced. For example, Amazon EC2 service provider will pay the customers a credit card if the annual uptime percentage drops below 99.95% for the service year [10].

III. Problem Description

We only consider the pre-copy based live migration algorithm. The bandwidth during the pre-copy phase is constant. In the stop-and-copy phase, the bandwidth equals the maximum bandwidth determined by operators as stated in [4]. Next we first list the main denotations, then present the formalized problem.

A. Denotations

Let N be the total number of pages of the virtual machine. $N_i (i \geq 1)$ denotes the number of pages transferred in the i-th round of the pre-copy phase. Especially, $N_1 = N$. B_p represents the bandwidth provided by the network in the pre-copy phase in unit of pages/second. Let B_m be the maximum

bandwidth during the stop-and-copy phase. k is the total number of iterations during the pre-copy phase.

T_{tot} and T_{dwn} represents the expected total migration time and downtime of the live migration, respectively. $\tilde{T}_{tot}$ and $\tilde{T}_{dwn}$ represents the practical total migration time and downtime of the live migration, respectively. $F_i(t)$ stands for the distribution function of the interval between two write operations of page i.

B. Problem Formalization

The problem studied in this work can be formalized as given the maximum bandwidth B_m, minimizing the bandwidth during the pre-copy phase, B_p, to satisfy the total migration time T_{tot} and downtime T_{dwn}.

$$\min \quad B_p \tag{1}$$

$$s.t. \begin{cases} \tilde{T}_{tot} \leq T_{tot} \\ \tilde{T}_{dwn} \leq T_{dwn} \end{cases} \tag{2}$$

Based on the pre-copy live migration algorithm, the practical total migration time and downtime can be obtained

$$\begin{cases} \tilde{T}_{dwn} = \frac{N_{k+1}}{B_m} \\ \tilde{T}_{tot} = \frac{\sum_{i=1}^{k} N_i}{B_p} + \tilde{T}_{dwn} \end{cases} \tag{3}$$

We can see that the critical problem is to compute the number of pages in each round N_i. Next, we will discuss how to determine N_i in two cases.

IV. Dirtying Distribution Functions are the Same

Let $F(t)$ be the dirtying distribution function of one page, that is, $F(t) = \text{Prob.}\{\text{Dirtying interval} < t\}$. If all the pages have the same dirtying distribution function, then we have $F_1(t) = F_2(t) = ... = F_{N_0}(t) = F(t)$. Thus, the number of dirty pages $N_{i+1}(i \geq 1)$ has no relation to the transmission order. It can be computed from the dirtying distribution function and N_i.

Next, we will first assume $F(t)$ obeys the deterministic distribution, then illustrate how to calculate B_p if the dirtying distribution function follows the bernoulli distribution.

A. Deterministic Distribution Function

Let T be the parameter of the deterministic distribution function, that is,

$$F(t) = \begin{cases} 0 & t < T \\ 1 & t \geq T \end{cases} \tag{4}$$

Then the i-th transferred page will become dirty when the $(TB_p + i)$-th page is finished. Therefore, after N pages being transferred in the first round, TB_p pages are clean, and the others $N - TB_p$ become dirtied. At the next round, once a page is transferred, a page will get dirty. Thus we have

$$N_i = \begin{cases} N & i = 1 \\ N - TB_p & i > 1 \end{cases} \tag{5}$$

Based on the above analysis, we can get that the optimal round number is 1, and the pre-copy phase should stopped after transmitting TB_p pages. Under this strategy, the practical total migration time and downtime is

$$\begin{cases} \tilde{T}_{dwn} = \frac{N - TB_p}{B_m} \\ \tilde{T}_{tot} = \frac{TB_p}{B_p} + \tilde{T}_{dwn} \end{cases} \tag{6}$$

To satisfy the delay guarantee of T_{tot} and T_{dwn}, we have

$$\begin{cases} \frac{N - TB_p}{B_m} \leq T_{dwn} \\ T + \frac{N - TB_p}{B_m} \leq T_{tot} \end{cases} \tag{7}$$

Therefore, the bandwidth during the pre-copy phase is

$$B_p \geq \max\{\frac{N - T_{dwn}B_m}{T}, \frac{N - (T_{tot} - T)B_m}{T}\} \tag{8}$$

B. Bernoulli Distribution Function

Let p be the dirtying probability in each time interval $\frac{1}{B_p}$, that is, the time of transmitting a page. We can get the following theorem.

THEOREM 1: The expected number of pages transferred in the $(i+1) - th$ round is

$$N_{i+1} = \begin{cases} N & i = 0 \\ N - \frac{1 - (1-p)^{\sum_{j=1}^{i} N_j}}{p} & i \geq 1 \end{cases} \tag{9}$$

Proof: We prove this theorem using the inductive method.

1) When $i = 0$, the number of transferred pages in the first round is $N_1 = N$.

2) When $i > 0$, we first consider the second round. The probability that the first page transferred in the first round keeps clean at the end of the round is $(1-p)^{N_1 - 1}$, the second is $(1-p)^{N_1 - 2}$, etc. Hence, we can get that

$$N_2 = N - \sum_{j=1}^{N_1} (1-p)^{N_1 - j} = N - \frac{1 - (1-p)^{N_1}}{p} \tag{10}$$

3) Computing N_{i+1} under the assumption that $N_i = N - \frac{1 - (1-p)^{\sum_{j=1}^{i-1} N_j}}{p}, (i \geq 2)$. In the $(i+1) - th$ round, the pages needed to be transferred consists of two types:

- The pages transferred in the $i - th$ round become dirty again. The expected number is $\sum_{l=1}^{N_i} (1 - (1-p)^{N_i - l})$.
- The pages not transferred in the $i - th$ round become dirty. The expected number of dirty pages at the end of the $i - th$ round is $(N - N_i)(1 - (1-p)^{N_i})$.

Therefore, we can obtain that

$$\begin{aligned} N_{i+1} &= N_i - \sum_{l=1}^{N_i} (1-p)^{N_i - l} + (N - N_i)(1 - (1-p)^{N_i}) \\ &= N - \sum_{l=1}^{N_i} (1-p)^{N_i - l} - (N - N_i)(1-p)^{N_i} \\ &= N - \sum_{l=1}^{N_i} (1-p)^{N_i - l} - \frac{1 - (1-p)^{\sum_{j=1}^{i-1} N_j}}{p} (1-p)^{N_i} \\ &= N - \frac{1 - (1-p)^{\sum_{j=1}^{i} N_j}}{p} \end{aligned} \tag{11}$$

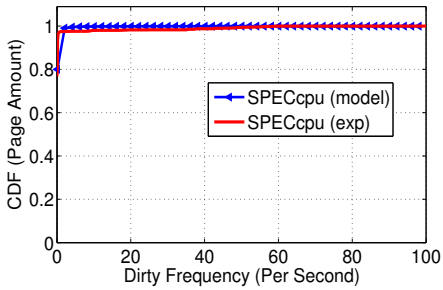

Fig. 4. CDF of SPECcpu

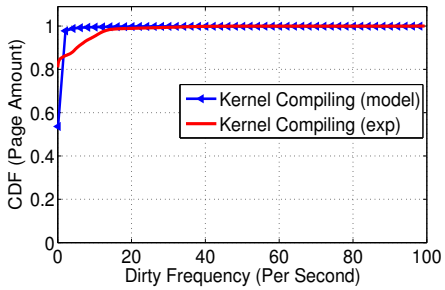

Fig. 5. CDF of kernel compiling

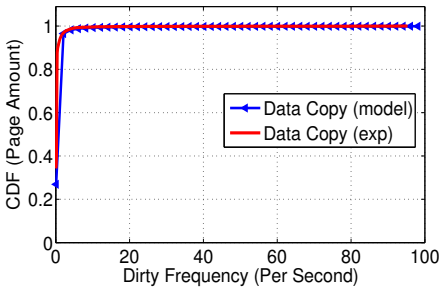

Fig. 6. CDF of data copy

Combining Eqs. (10)-(11), THEOREM 1 can be proved. ■

Let k be the number of rounds in the pre-copy phase. To guarantee T_{tot} and T_{dwn}, the following inequalities should be satisfied.

$$\begin{cases} N_{k+1} \leq T_{dwn} B_m \\ \sum_{j=1}^{k} N_j \leq (T_{tot} - T_{dwn}) B_p \end{cases} \tag{12}$$

According to *THEOREM 1*, we have

$$\begin{cases} N - \frac{1-(1-p)^{\sum_{j=1}^{k} N_j}}{p} \leq T_{dwn} B_m \\ \sum_{j=1}^{k} N_j \leq (T_{tot} - T_{dwn}) B_p \end{cases} \tag{13}$$

From Eq. (13), we have

$$\begin{aligned} & N - \frac{1-(1-p)^{\sum_{j=1}^{i} N_j}}{p} \\ \geq\ & N - \frac{1-(1-p)^{(T_{tot}-T_{dwn})B_p}}{p} \end{aligned} \tag{14}$$

Combining Eqs. (13) and (14), we obtain

$$N - \frac{1-(1-p)^{(T_{tot}-T_{dwn})B_p}}{p} \leq T_{dwn} B_m \tag{15}$$

Let θ be the probability that a page becomes dirty during a unit time. Then $(1-\theta)$ is the probability that a page keeps clean during a unit time. Therefore, during $\frac{1}{B_p}$ time units, the probability that a page keeps clean is $1 - p = (1-\theta)^{\frac{1}{B_p}}$. Combing with Eq. (15), we can get

$$\begin{aligned} & N - \frac{1-(1-p)^{(T_{tot}-T_{dwn})B_p}}{p} \\ =\ & N - \frac{1-(1-\theta)^{(T_{tot}-T_{dwn})}}{1-(1-\theta)^{\frac{1}{B_p}}} \\ \leq\ & T_{dwn} B_m \end{aligned} \tag{16}$$

From Eq. (16), B_p should be

$$B_p \geq \frac{ln(1-\theta)}{ln(1 - \frac{1-(1-\theta)^{(T_{tot}-T_{dwn})}}{N - T_{dwn} B_m})} \tag{17}$$

Given a specific scenario, N, T_{tot}, T_{dwn} and B_m are determined. The only parameter θ, i.e., the probability that a page becomes dirty per unit time, can be measured in practice.

V. Dirtying Distribution Functions are Different

In this situation, since the dirtying frequency of each page is different, the number of dirty pages at the end of a round is related to the transmission order. Various ordering functions lead to different $N_{i+1}(i > 0)$ and thus different bandwidth B_p is required to guarantee T_{tot} and T_{dwn}. Since the pre-copy mechanism does not take the transmitting order into consideration, we will compute the expected required bandwidth in this section.

A. Dirtying Distribution Function

If the dirtying probability of each node is different, then we need to consider the dirtying distribution of the whole memory. Three types of workloads are investigated using experiments to study the dirtying pattern of the memory pages.

- *SPECcpu2006* aims to evaluate the performance of systems on a known CPU-intensive workload, with emphasis on the system's processor, memory hierarchy and compiler [25]. The benchmark can represent the class of compute-intensive workload in data center networks.
- *Data Copy* represents the applications that need to transfer data. In data centers, communication between servers is quite frequent, and many servers need to transfer data to each other to cooperate or keep data consistent.
- *Kernel Compiling* is a representation of development workload. In our experiment, the Linux 2.6.28.10 kernel is compiled. The workload involves moderate disk I/O and frequent memory access.

Every 10 milliseconds, the function *xc_shadow_control()* with *XEN_DOMCTL_SHADOW_OP_CLEAN* mode is called to copy the dirty page bitmap to *to_send* and then clear the dirty bitmap. Then we can see which pages are dirtied during this interval. The total sampling time is 10 seconds.

Through investigating the dirtying frequency of all the pages, we found that the distribution of the dirtying pages' frequency can be approximately modeled as a function $G(x) = 1 - \frac{\beta}{x}, (x > 0)$, in which β is the parameter. Thus, the probability density function is $g(x) = G'(x) = \frac{\beta}{x^2}$. We can get the parameter of β using the least square method [26], that is, make the summation of the square of the difference between the model and the sampled data as small as possible. Figures 4-6 plot the CDF of the dirty frequency of pages when the VM

is running the SPECcpu benchmark, copying data or compiling kernel, respectively. We can see that $G(x)$ well characterizes the distribution of dirtying pages with different workloads. This model is referred to as *reciprocal-based model*.

B. Required Bandwidth

Denote $P(R,i)$ as the probability that page i transferred in round $(R-1)$ becomes dirty at the start of round R. Since the dirtying probability is related to the ID of the transferred page, applying the full probability formula, we can obtain $P(2,1)$, i.e., the probability that the first transferred page is dirty at the start of the second round

$$P(2,1) = \int_{x=x_{min}}^{\infty} g(x) \times P_d(x) dx \tag{18}$$

where $P_d(x)$ represents the probability that the page with dirtying frequency x becomes dirty again after transmission. The duration from the first page being transferred to the end of the first round is $RT_1 = \frac{N-1}{B_p}$. Thus, if the dirtying frequency of one page is larger than $\frac{1}{RT_1} = \frac{B_p}{N-1}$, the page will become dirty at the end of the first round. Thus, eq. (18) is

$$\begin{aligned} P(2,1) &= \int_{x=x_{min}}^{\frac{B_p}{N-1}} g(x) \times 0 dx + \int_{x=\frac{B_p}{N-1}}^{\infty} g(x) \times 1 dx \\ &= \int_{x=\frac{B_p}{N-1}}^{\infty} \frac{\beta}{x^2} dx = \frac{\beta(N-1)}{B_p} \end{aligned} \tag{19}$$

Similarly, the probability that the i-th transferred page becomes dirty at the end of the first round is

$$P(2,i) = \frac{\beta(N-i)}{B_p} \tag{20}$$

Therefore, we can obtain that at the start of the second round, the expected number of dirty pages is

$$N_2 = \sum_{i=1}^{N} P(2,i) = \frac{\beta N(N-1)}{2B_p} = \frac{\beta N_1(N_1-1)}{2B_p} \tag{21}$$

However, we cannot infer that the number of dirty pages needed to be transferred during the j-th round is $N_j = \frac{\beta N_{j-1}(N_{j-1}-1)}{2B_p}$. This is because the distribution of the pages' dirtying frequency will change after every round since the slowly dirtying pages are filtered out. Therefore, the proportion taken by the slowly dirtying pages gets smaller. As a result, the parameter β of the CDF function grows. Let β_j be the parameter of the CDF of the dirtying pages at the start of the j-th round ($\beta_1 = \beta$). Clearly, it is reversely proportional to N_j. Thus, we could let $\beta_j = \frac{\alpha}{N_j}$. N_j can be obtained as

$$\begin{aligned} N_j &= \frac{\alpha(N_{j-1}-1)}{2B_p} \\ &= \frac{\alpha^{j-1}(N-1)}{(2B_p)^{j-1}} - (\frac{\alpha}{2B_p})^{j-2} - (\frac{\alpha}{2B_p})^{j-3} - ... - \frac{\alpha}{2B_p} \end{aligned} \tag{22}$$

If a VM can be successfully migrated, then the number of dirty pages will decrease as the number of round increases. Thus, $\frac{\alpha}{2B_p}$ should be less than 1. We can ignore the power of $(\frac{\alpha}{2B_p})$ items as they are quite small compared with the hundreds of memory pages. Therefore, the number of transferred pages in the j-th round is $N_j = (N-1)(\frac{\alpha}{2B_p})^{j-1}$. Correspondingly, the number of the transferred pages at the pre-copy phase is

$$N_p = \sum_{j=1}^{k} N_j = (N-1)\frac{1-(\frac{\alpha}{2B_p})^k}{1-\frac{\alpha}{2B_p}} \tag{23}$$

Thus, to satisfy the delay requirement of the total migration phase and the downtime requirment, we have

$$\begin{cases} (N-1)\frac{1-(\frac{\alpha}{2B_p})^k}{1-\frac{\alpha}{2B_p}} \le (T_{tot} - T_{dwn})B_p \\ (N-1)(\frac{\alpha}{2B_p})^k \le T_{dwn} B_m \end{cases} \tag{24}$$

From the above second inequality, we can get that $k \le \frac{\log(\frac{T_d B_m}{N-1})}{\log(\frac{\alpha}{2B_p})}$. Let

$$k = \frac{\log(\frac{T_d B_m}{N-1})}{\log(\frac{\alpha}{2B_p})} \tag{25}$$

According to the first inequality in (24), we can obtain that

$$B_p \ge \frac{\alpha}{2} + \frac{N-1-T_{dwn}Bm}{T_{tot}-T_{dwn}} \tag{26}$$

Since $\beta_1 = \frac{\alpha}{N_1}$, we have $\alpha = \beta N$. β can be obtained through fitting the sampled dirtying frequency data. Then the value of B_p and k can be calculated.

VI. Experimental Evaluation

A. Experimental Setup

To validate that the computed bandwidth given specific delay requirements can indeed guarantee the delay, we conducted experiments using two Dell Servers connecting to a HP ProCurve 2910al Ethernet switch. The end hosts run XEN-3.4.3 [12] system with Linux kernel 2.6.18.8-xen [13].

The live migration command is launched at the user space. Once the `xm migrate` command is received by the VM monitor, it will notify the devices, check whether the memory is enough for migration, and establish a socket connection with the target physical host. After the source host receives the response from the target host, the VM begins to be migrated. Therefore, the page dirtying frequency can be sampled after the VM monitor receives the `xm migrate` command and before the VM migrating begins. In the following experiments, we sampled 10 seconds at the interval of 10 microseconds before one migration. To guarantee that the migrated VM can get the computed bandwidth, no background traffic is added.

The workload employed to evaluate the effectiveness of our model is the same as that in Section V-A. The memory size of the migrated VM is 400 MB. Since the number of iterations during the pre-copy phase is an integer, The top integral function `ceil` is used.

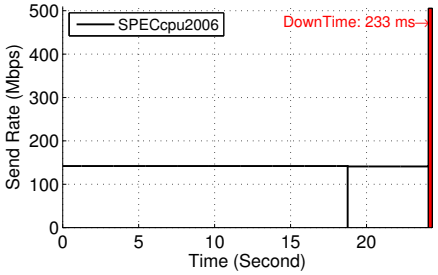

Fig. 7. Live migration process with $B_p = 131$ Mbps, $B_m = 500$ Mbps and $k = 2$. The delay requirement is $T_{tot} \leq 30$ s and $T_{dwn} \leq 0.6$ s.

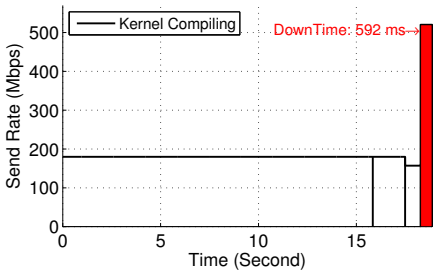

Fig. 8. Live migration process with $B_p = 166$ Mbps, $B_m = 500$ Mbps and $k = 3$. The delay requirement is $T_{tot} \leq 30$ s and $T_{dwn} \leq 0.6$ s.

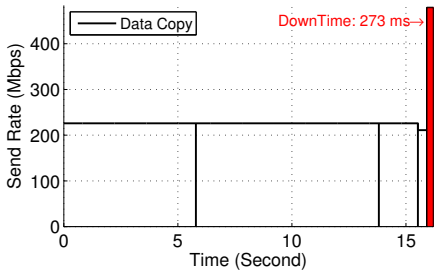

Fig. 9. Live migration process with $B_p = 208$ Mbps, $B_m = 500$ Mbps and $k = 4$. The delay requirement is $T_{tot} \leq 30$ s and $T_{dwn} \leq 0.6$ s.

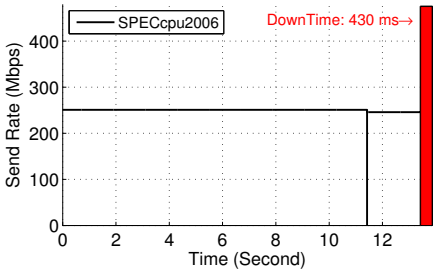

Fig. 10. Live migration process with $B_p = 240$ Mbps, $B_m = 500$ Mbps and $k = 2$. The delay requirement is $T_{tot} \leq 15$ s and $T_{dwn} \leq 0.4$ s.

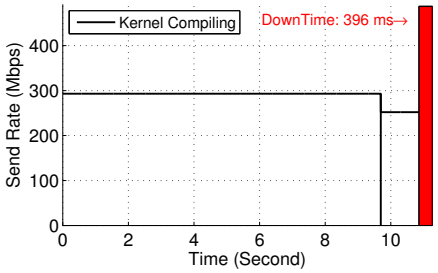

Fig. 11. Live migration process with $B_p = 280$ Mbps, $B_m = 500$ Mbps and $k = 2$. The delay requirement is $T_{tot} \leq 15$ s and $T_{dwn} \leq 0.4$ s.

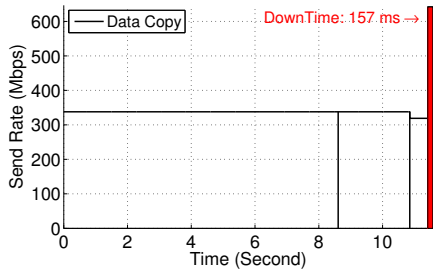

Fig. 12. Live migration process with $B_p = 323$ Mbps, $B_m = 500$ Mbps and $k = 3$. The delay requirement is $T_{tot} \leq 15$ s and $T_{dwn} \leq 0.4$ s.

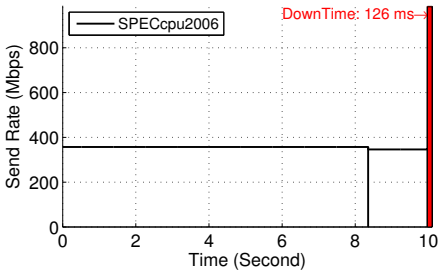

Fig. 13. Live migration process with $B_p = 341$ Mbps, $B_m = 900$ Mbps and $k = 2$. The delay requirement is $T_{tot} \leq 10$ s and $T_{dwn} \leq 0.3$ s.

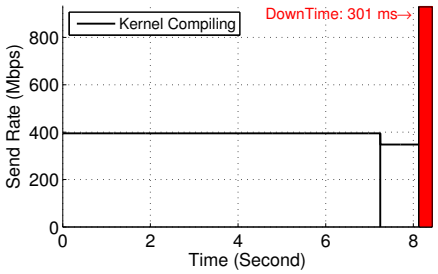

Fig. 14. Live migration process with $B_p = 377$ Mbps, $B_m = 900$ Mbps and $k = 2$. The delay requirement is $T_{tot} \leq 10$ s and $T_{dwn} \leq 0.3$ s.

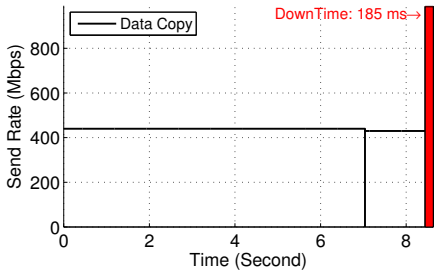

Fig. 15. Live migration process with $B_p = 420$ Mbps, $B_m = 900$ Mbps and $k = 3$. The delay requirement is $T_{tot} \leq 10$ s and $T_{dwn} \leq 0.3$ s.

B. Results

1) Model Validation: A series of experiments are conducted in our testbed to evaluate whether the reciprocal-based model can guarantee the delay requirements of a live VM migration in different scenarios.

First, given $T_{tot} = 30$ s, $T_{dwn} = 0.6$ s and $B_m = 500$ Mbps, and the migrated VM is running SPECcpu2006 benchmark, from eq. (25) and (26), the bandwidth during the pre-copy phase B_p is computed as 131 Mbps and the iteration number of the pre-copy phase k is 2. Figure 7 shows the evolution of the send rate of each iteration using the results of the model. We can see that the pre-copy phase lasts two iterations. The send rate of each iteration is about $B_p = 131$ Mbps. After that, the live migration enters into the stop and copy phase. The send rate is about B_m = 500 Mbps. The total migration process lasts about 25 seconds and the downtime lasts 0.233 seconds, which is smaller than T_{tot} and T_{dwn}, respectively. We then let the migrated VM compile kernel or copy data, the bandwidth during the pre-copy phase is 166 Mbps and 208 Mpbs, respectively, and the number of iterations is 3 and 4. The migration results shown in Figures 8 and 9 show that the total migration time and downtime can be guaranteed.

Next, we evaluate whether our model works well with smaller delay requirements. The values of T_{tot} and T_{dwn} decrease to 15 seconds and 0.4 seconds, respectively. B_m keeps the same value. Using the reciprocal-based model, we get that $B_p = 240$ Mbps and $k = 2$ with SPECcpu benchmark. B_p value becomes larger compared with that when $T_{tot} = 30$

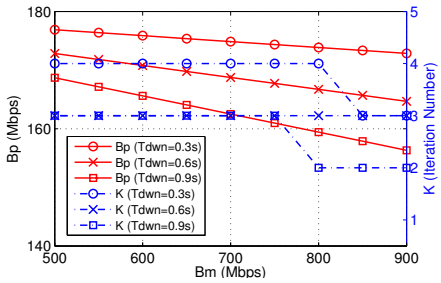

Fig. 16. Given $T_{tot} = 30$ seconds, the relationship between B_p, K, T_{dwn} and B_m.

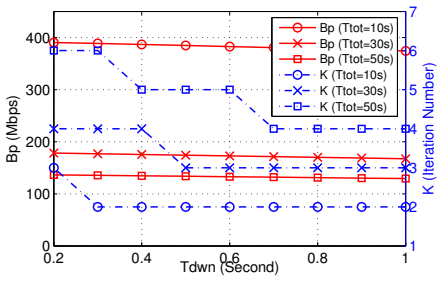

Fig. 17. Given $B_m = 500$ Mbps, the relationship between B_p, K, T_{dwn} and T_{tot}.

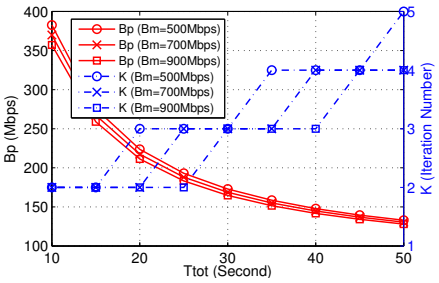

Fig. 18. Given $T_{dwn} = 0.6$ seconds, the relationship between B_p, K, B_m and T_{tot}.

seconds since the value of T_{tot} decreases. Figures 10-12 plot the live VM migration process with smaller delay requirements in different scenarios. The total migration time with the three kinds of workload is between 12 and 14 seconds, which are all smaller than $T_{tot} = 15$ seconds. The downtime of SPECcpu2006 is a little larger than T_{dwn}. The downtime in the other two scenarios is smaller than 0.4 seconds.

At last, we evaluate the effectiveness of our model when the maximum bandwidth, B_m, becomes larger. Figures 13-15 depict the send rate variation of live VM migration with $T_{tot} = 10$ seconds, $T_{dwn} = 0.3$ seconds and higher maximum bandwidth $B_m = 900$ Mbps. In the first scenario with SPECcpu2006 workload, the output of our reciprocal-based model is that $B_p = 341$ Mbps and $k = 2$. Compared with the result in Figure 10, the value of B_p increases. The curve in Figure 13 shows that the total migration time is about 10 seconds and the downtime is 0.126 seconds. The downtime is smaller than the delay requirement of T_{dwn}. When the migrated VM is compiling kernel or copying data, the total migration time is about 9 seconds and the downtime is quite close to or smaller than the downtime.

2) Model Comparison: The above results demonstrate that the reciprocal-based model can give proper B_p and k to guarantee the delay requirements of live migration of VMs. Due to lack of space, we could not present the experimental results of all the models. Next we will list the comparison of the three models in some scenarios. If the results of the deterministic-based model (eq. (8)) and bernoulli-based model (eq. (17) are close to the results of reciprocal-based model, then we can infer that they are also effective.

Table I shows the comparison of the three models. The maximum bandwidth $B_m = 500$ Mbps. Clearly, the other two models give too large bandwidth B_p with SPECcpu 2006 benchmark and kernel compiling workload, while a little small with data copy workload. In terms of the number of iterations k, it always equals 1 in the deterministic-based model, while the bernoulli-based model cannot tell the value of k. In practice, the pre-copy phase can be terminated either after the duration of $(T_{tot} - T_{dwn})$ or when the number of dirty pages is not larger than $T_{dwn}B_m$.

3) Parameters: In this subsection, we will investigate the relationship among T_{tot}, T_{dwn}, B_p, B_m and k with the reciprocal-based model. Only the results with the workload

TABLE I
MODEL COMPARISON (B_p (MBPS), K)

(T_{tot}, T_{dwn})	Workload	Deterministic	Bernoulli	Reciprocal
(30,0.6)	speccpu	(615,1)	(565,x)	(131,2)
	kernel	(2068,1)	(1903,x)	(166,3)
	data copy	(218,1)	(200,x)	(208,4)
(20,0.6)	speccpu	(615,1)	(565,x)	(147,2)
	kernel	(2068,1)	(1903,x)	(184,2)
	data copy	(218,1)	(200,x)	(227,2)
(30,0.3)	speccpu	(647,1)	(623,x)	(138,2)
	kernel	(2175,1)	(2097,x)	(173,3)
	data copy	(229,1)	(220,x)	(216,5)
(20,0.3)	speccpu	(647,1)	(623,x)	(188,2)
	kernel	(2175,1)	(2097,x)	(230,3)
	data copy	(229,1)	(220,x)	(267,3)

of kernel compiling is presented. The results with other workloads are similar.

Figure 16 plots the required bandwidth B_p and the number of iterations k during the pre-copy phase with different maximum bandwidth B_m, three different downtime T_{dwn} values and $T_{tot} = 30$ seconds. As the maximum bandwidth B_m increases, both the required bandwidth and the number of iterations decreases. Besides, given a determined B_m, B_p and k both become smaller with larger T_{dwn}. The trend of the curves is reasonable. Note that the required bandwidth is between 160 and 180 Mbps, which is not quite large. Preserving the constant bandwidth during the pre-copy phase is more practical than changing the bandwidth between 100 Mbps and 500 Mbps. The number of iterations during the pre-copy phase is between 2 and 4, which is quite small compared with the maximum allowed number of iterations, 30 rounds, in XEN.

Figure 17 plots the variation of B_p and k with different downtime T_{dwn} and three different T_{tot} values given $B_m = 500$ Mbps. When $T_{tot} = 10$ seconds, the calculated bandwidth B_p is about 390 Mbps. B_p slightly decreases as the downtime T_{dwn} increases since larger downtime indicates that more pages can be left at the end of the pre-copy phase. Thus, the pages can be transferred more slowly during the pre-copy phase. Besides, since T_{dwn} is quite small, it imposes little impact on the bandwidth during the pre-copy phase. For example, if T_{dwn} increases from 0.3 seconds to 0.4 seconds, the stop and copy phase can transfer $B_m * (0.4 - 0.3)$=50 Mbits

more traffic. If the pre-copy phase lasts 10 seconds and the dirtying frequency of the pages keeps the same, the value of B_p only decreases about $\frac{50 \text{ Mbps}}{10 \text{ seconds}} = 5$ Mbits, which is quite small. The number of k drops as T_{dwn} grows since the stop and copy phase can send more pages when T_{dwn} is larger.

Figure 18 depicts the variation of B_p and k with different total migration time T_{tot} and three different maximum bandwidth values. The downtime T_{dwn} is a constant value, 0.6 seconds. B_p declines dramatically with larger total migration time. This is because the memory size of the migrated VM is constant, larger total migration time indicates that the transfer rate B_p can be smaller. While B_m has little impact on B_p. The reason is similar to why T_{dwn} has little to do with B_p. The curves of the number of iterations during the pre-copy phase rise as the total migration time grows. This is because as T_{tot} becomes larger, B_p decreases. Correspondingly, one iteration lasts more time and more pages will become dirty again at the end of one iteration. Since the downtime is constant, more iterations are needed to ensure that the number of pages to be transferred during the stop and copy phase will not exceed $B_m \times T_{dwn}$.

VII. Conclusions

Pre-copy is a widely used live migration mechanism. It requires frequently varied transfer bandwidth, which brings great challenges to the network operators. Besides, the live migration mechanism cannot guarantee delay. In this work, we theoretically determine the proper bandwidth to guarantee the total migration time and downtime requirements of live migration of VMs. We first assume that the dirtying distribution function of all the pages follows the deterministic distribution to give a simple example, then the bandwidth is determined under the premise that the dirtying distribution function of all the pages obeys the bernoulli distribution. At last, we assume that the dirty frequency of each page is different and the CDF of the pages' dirtying frequency is a reciprocal function. The experimental results indicate that the reciprocal-based model well characterizes the dirtying frequency of the memory pages. Based on the results of the reciprocal-based model, the delay bound of live migration can be guaranteed. The relationship between different parameters are also analyzed.

Acknowledgement

The authors gratefully acknowledge the anonymous reviewers for their constructive comments. This work is supported in part by the National Basic Research Program of China (973 Program) under Grant No. 2014CB347800 and 2010CB328105, the National Natural Science Foundation of China (NSFC) under Grant No. 61225011, 60932003 and 61003226.

References

[1] P. H. Gum, "System/370 Extended Architecture: Facilities for Virtual Machines," *IBM Journal of Research and Development*, vol. 27, no. 6, pp. 530–544, 1983.

[2] L. Seawright and R. MacKinnon, "VM/370–A Study of Multiplicity and Usefulness," *IBM Systems Journal*, vol. 18, no. 1, pp. 4–17, 1979.

[3] W. Voorsluys, J. Broberg, S. Venugopal, and R. Buyya, "Cost of Virtual Machine Live Migration in Clouds: A Performance Evaluation," *Cloud Computing*, pp. 254–265, 2009.

[4] C. Clark, K. Fraser, S. Hand, J. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield, "Live Migration of Virtual Machines," in *USENIX NSDI*, 2005, pp. 273–286.

[5] Minlan Yu and Yung Yi and Jennifer Rexford and Mung Chiang, "Rethinking Virtual Network Embedding: Substrate Support for Path Splitting and Migration," *ACM SIGCOMM CCR*, vol. 38, no. 2, pp. 17–29, 2008.

[6] C. Guo, G. Lu, H. Wang, S. Yang, C. Kong, P. Sun, W. Wu, and Y. Zhang, "SecondNet: a Data Center Network Virtualization Architecture with Bandwidth Guarantees," in *ACM CoNEXT*, 2010, pp. 15–28.

[7] I. Pratt, K. Fraser, S. Hand, C. Limpach, A. Warfield, D. Magenheimer, J. Nakajima, and A. Mallick, "Xen 3.0 and the Art of Virtualization," in *Linux symposium*, 2005, pp. 65–77.

[8] S. Akoush, R. Sohan, A. Rice, A. Moore, and A. Hopper, "Predicting the Performance of Virtual Machine Migration," in *IEEE MASCOTS*, 2010, pp. 37–46.

[9] A. Nagarajan, F. Mueller, C. Engelmann, and S. Scott, "Proactive Fault Tolerance for HPC with Xen Virtualization," in *the 21st annual international conference on SuperComputing*, 2007, pp. 23–32.

[10] "Amazon EC2 Service Level Agreement." [Online]. Available: http://aws.amazon.com/ec2-sla/

[11] D. Breitgand, G. Kutiel, and D. Raz, "Cost-aware live migration of services in the cloud," in *ACM SYSTOR*, 2010.

[12] "Xen Hypervisor 3.4.3 Download." [Online]. Available: http://xen.org/download/index_3.4.3.html

[13] "linux-2.6.18-xen.hg." [Online]. Available: http://xenbits.xensource.com/linux-2.6.18-xen.hg

[14] N. Gude, T. Koponen, J. Pettit, B. Pfaff, M. Casado, N. McKeown, and S. Shenker, "NOX: Towards an Operating System for Networks," *ACM SIGCOMM CCR*, vol. 38, no. 3, pp. 105–110, 2008.

[15] F. Checconi, T. Cucinotta, and M. Stein, "Real-time Issues in Live Migration of Virtual Machines," in *Euro-Par 2009–Parallel Processing Workshops*. Springer, 2010, pp. 454–466.

[16] M. Alizadeh, A. Greenberg, D. Maltz, J. Padhye, P. Patel, B. Prabhakar, S. Sengupta, and M. Sridharan, "Data Center Tcp (DCTCP)," in *ACM SIGCOMM*, 2010.

[17] C. Wilson, H. Ballani, T. Karagiannis, and A. Rowstron, "Better Never than Late: Meeting Deadlines in Datacenter Networks," in *ACM SIGCOMM*, 2011, pp. 50–61.

[18] J. H. Balajee Vamanan and T. N. Vijaykumar, "Deadline-Aware Datacenter TCP (D2TCP)," in *ACM SIGCOMM*, 2012.

[19] P. M. David Zats, Tathagata Das and R. H. Katz, "DeTail: Reducing the Flow Completion Time Tail in Datacenter Networks," in *ACM SIGCOMM*, 2012.

[20] C.-Y. Hong, M. Caesar, and P. B. Godfrey, "Finishing Flows Quickly with Preemptive Scheduling," in *ACM SIGCOMM*, 2012.

[21] M. Alizadeh, S. Yang, S. Katti, N. McKeown, B. Prabhakar, and S. Shenker, "Deconstructing Datacenter Packet Transport," in *ACM Workshop on HotNet*, 2012, pp. 133–138.

[22] R. Braden, D. Clark, and S. Shenker, "RFC 1633 – Integrated Services in the Internet Architecture: an Overview," 1994. [Online]. Available: http://www.ietf.org/rfc/rfc1633.txt

[23] L. Zhang, S. Deering, D. Estrin, S. Shenker, and D. Zappala, "RSVP: A New Resource ReSerVation Protocol," *IEEE Network Magazine*, vol. 7, no. 5, 1993.

[24] Y. Wang, E. Keller, B. Biskeborn, J. van der Merwe, and J. Rexford, "Virtual Routers on the Move: Live Router Migration as a Network-Management Primitive," *ACM SIGCOMM CCR*, vol. 38, no. 4, pp. 231–242, 2008.

[25] "SPEC CPU2006." [Online]. Available: http://www.spec.org/cpu2006/

[26] S. J. Miller, "The Method of Least Squares," in *Brown University*, 2006.

[27] "OpenFlow." [Online]. Available: http://www.openflow.org/