Traffic-Sensitive Live Migration of Virtual Machines

Umesh Deshpande* and, Kate Keahey†
* Binghamton University, Binghamton, NY
udeshpa1@binghamton.edu
† Argonne National Lab, Lemont, IL
keahey@mcs.anl.gov

***Abstract*—In this paper we address the problem of network contention between the migration traffic and the Virtual Machine (VM) application traffic for the live migration of co-located Virtual Machines. When VMs are migrated with pre-copy, they run at the source host during the migration. Therefore the VM applications with predominantly outbound traffic contend with the outgoing migration traffic at the source host. Similarly, during post-copy migration, the VMs run at the destination host. Therefore the VM applications with predominantly inbound traffic contend with the incoming migration traffic at the destination host. Such contention increases the total migration time of the VMs and degrades the performance of the VM application. Here, we propose a *traffic-sensitive* live VM migration technique to reduce the contention of migration traffic with the VM application traffic. It uses a combination of pre-copy and post-copy techniques for the migration of the co-located VMs (those located on the same source host), instead of relying on any single pre-determined technique for the migration of all the VMs. We base the selection of migration techniques on the VMs' network traffic profiles so that the direction of migration traffic complements the direction of the most VM application traffic. We have implemented a prototype of traffic-sensitive migration on the KVM/QEMU platform. In the evaluation, we compare traffic-sensitive migration against the approaches that use only pre-copy or only post-copy for VM migration. We show that our approach minimizes the network contention for migration, thus reducing the total migration time and the application degradation.**

***Keywords*-Virtual Machine, Live Migration, Traffic Sensitivity**

I. Introduction

Live migration of VMs is used in datacenters for quickly eliminating the hot-spots [37], [17], [6], [33], to free up resources to save power [9], [5], or to perform system maintenance. It can also be used to defragment a datacenter or to obtain resources with specific properties, for example, specific network topology critical for HPC applications [14], [15]. Pre-copy [7], [26] and post-copy [16], [18] are commonly used live VM migration techniques. Pre-copy provides low service downtime for the migration of VMs executing read-mostly workloads. It is used as a default migration technique by Xen [2], VMware [20], and KVM [25] hypervisors. Whereas, post-copy is known for its low network overhead and allows quick consolidation [17] or eviction [13] of VMs.

Live VM migration is a network intensive activity; it requires transfer of Gigabytes of VM memory state from the source to the destination host over the network. When the migrating VMs are running network-bound applications, the applications' traffic competes with the migration traffic for the Network Interface Cards (NICs) at the source and the destination hosts. Such a contention prolongs the VM migration, thus delaying the deprovisioning of resources occupied by the VM at the source host. Furthermore, prolonged contention of network flows also degrades the performance of the network-bound VM applications. When migrating VMs are part of a group of collaborating VMs, degradation of any single VM can result in the overall degradation of the application jointly executed by these VMs.

In this paper, we address the problem of network contention between the migration and VM application traffic for the VMs executing network bound applications. The widely used 1 Gigabit NICs provide 1 Gbps bandwidth in each direction. However, when the migration traffic and the VM application traffic have the same direction, both network flows contend for the available bandwidth. Whereas when the direction of the flows is opposite, they do not compete with each other. Hence the level of contention depends on the rate of VM traffic in each direction and the direction of the migration traffic. In pre-copy migration, the VMs run at the source host during their migration; therefore the migration traffic contends with the VMs that predominantly have outbound network flow. In contrast, in post-copy, the VMs run at the destination host during their migration; therefore the migration traffic contends with the VMs that predominantly have inbound network flow.

A common approach to reduce the contention in pre-copy is by having a limit on the VM's total migration time or the amount of data transferred. Pre-copy VM migration techniques are hard coded with terminating conditions [26], [7]. When pre-copy exceeds a pre-defined limit for total migration time or transfers more than a pre-defined amount of data, its execution is suspended at the source host, and its execution state is transferred to the destination host. Even though this approach reduces the network overhead of pre-copy, premature termination of pre-copy rounds can increase the VM's downtime. Another approach to minimize the contention is to limit the rate of transfer of the migration traffic. The lower rate of migration traffic offers a larger share of the available bandwidth to the VM application, thus reducing their degradation. However, this increases the VM's total migration time and is not suitable when quick deprovisioning of resources is required.

In this paper we propose an approach called *traffic-sensitive* live migration. In this approach, we reduce the network contention for the simultaneous migration of VMs running network-bound applications with predominantly unidirectional

traffic. VMs often have a skewed network traffic profile. That is, the rate of their network traffic in one direction is greater than the rate of traffic in the other direction. For instance, when the VM is responding to a remote client, where the response size is larger than the request size or when VM is checkpointing an application state to remote storage.

In traffic-sensitive migration, we propose to use both pre-copy and post-copy live VM migration techniques. We select a migration technique for each VM to reduce the contention at the source and the destination host. The motivation behind using two migration techniques is that pre-copy competes with the outbound VM application traffic at the source host, whereas post-copy competes with the inbound VM application traffic at the destination host. Therefore we select one migration technique over the other depending on the network profile of the application.

The key contributions of our work are as follows

- In *traffic-sensitive* migration, we present a mechanism to calculate the network contention between the migration and the VM application traffic. Our approach uses a combination of pre-copy and post-copy for the migration of co-located VMs to reduce the contention. We monitor the network traffic on each host to generate a per-VM network traffic profile. Then we weigh every combination of pre-copy and post-copy for all the VMs and select the one that yields the least traffic contention.
- We describe a prototype implementation of traffic-sensitive migration on the KVM/QEMU [25] platform.
- We evaluate traffic-sensitive migration on an 18-node cluster, each connected to a switch with a 1 Gbps Ethernet interconnect. We compare the traffic-sensitive approach against the traditional approach that uses only pre-copy or only post-copy for migration.

The rest of this paper is organized as follows. Section II provides the background on traditional pre-copy and post-copy techniques. Section III experimentally demonstrates how the contention of migration traffic with the VM application traffic increases the total migration time of VM and degrades the application performance. Section IV describes the design of traffic-sensitive migration. Section V describes its implementation on the KVM/QEMU platform. Section VI compares the performance of the traffic-sensitive approach against pre-copy and post-copy. Section VII describes related research. Section VIII presents our conclusions and the direction of our future research.

II. Background

In this section, we describe pre-copy and post-copy live VM migration techniques and the effect of network contention on them.

A. Pre-copy VM Migration

Live VM migration primarily involves transfer of VM's CPU, memory and I/O state from the source host to the destination host. In pre-copy live VM migration the source host transfers the VM's memory state to the destination host in iterations. The transfer is performed live; that is, while the VM is still running at the source host. In the first iteration the source host transfers the entire memory of the VM to the destination, whereas in the subsequent iterations only the pages that are modified by the running VM are transferred. When the writable working set (WWS) of the VM has been identified, the VM is suspended at the source host and its CPU state, and WWS is transferred to the destination host. The time during which the VM remains inactive is known as *downtime* of the VM. The duration of the downtime depends on the size of the WWS. The more aggressive a VM is in dirtying its memory state, the longer it takes for pre-copy to identify the VM's WWS, which increases its total migration time. Moreover, when faced with network contention, pre-copy takes longer to converge on the VM's WWS.

B. Post-copy VM Migration

In post-copy VM migration, VM is suspended immediately upon beginning the migration. Its CPU state is transferred to the destination host, while its memory state is still residing at the source host. The VM now running at the destination host, faults upon the pages that have not yet been received by the destination host. The faulting pages are requested from the source host. Simultaneously, the source host actively transfers the remaining VM memory state to the destination host. The downtime of post-copy is minimal since it transfers only the VM's execution state during the downtime. However, it takes longer for VM to retrieve enough memory state from the source host to become responsive. Since, post-copy transfers each VM page over the network only once, it provides lower total migration time than does pre-copy for write-intensive applications. Again, however, the network contention with the migration traffic increases the VM's total migration time and slows the retrieval of faulted pages from the source host, thus degrading the performance of VM applications.

III. Demonstrating the Problem

In this section, we demonstrate the effect of network contention on the performance of migration and network-bound VM application through an experiment. We deploy two VMs on two hosts, each having 5 GB of memory and 2 vCPUs. The hosts have 16GB of memory and 8 physical CPUs. First VM executes a Netperf [3] client, which sends a TCP stream to another VM running a Netperf server. We migrate both VMs simultaneously while the test is in progress and measure the effect of the migration on the performance of Netperf. We compare the following configurations for the migration of both VMs.

1) When both VMs are migrated simultaneously with pre-copy.
2) When both VMs are migrated simultaneously with post-copy.

For the migration of multiple VMs, the total migration time is defined as the time from the start of their simultaneous migration to the completion of the last migration. Table I compares the total migration time and the amount of data

	Pre-copy	Post-copy	Pre-copy Idle-VM	Without Migration
Total Migration Time (seconds)	79.2 (79.2, 47.2)	92.1 (45.8, 92.1)	47.5	-
Amount of Data Transferred (MB)	10280	10277	10270	-
Netperf Performance (Mbps)	690.47	660.05	-	940.1

TABLE I
COMPARISON OF THE PERFORMANCE FOR THE MIGRATION OF TWO VMS FROM TWO HOSTS WITH PRE-COPY AND POST-COPY. THE INDIVIDUAL TOTAL MIGRATION TIME FOR EACH VM IS INCLUDED IN THE PARENTHESES IN THE FOLLOWING FORMAT : (VM1, VM2).

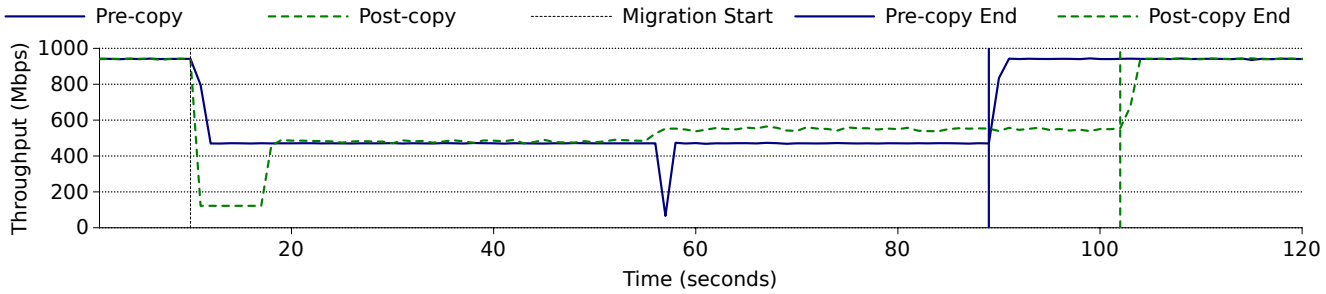

Fig. 1. Comparison of Netperf throughput for the migration of two VMs using pre-copy and post-copy. The first VM executes a Netperf client, and the second VM executes a Netperf server.

transferred for the migration of two idle VMs with that of two busy VMs running Netperf. When both VMs are migrated with pre-copy, the migration traffic for the VM running the Netperf client competes with the outgoing Netperf traffic. Therefore the total migration time of the VM increases compared with that of the other VM for which the direction of the Netperf traffic complements the direction of the migration traffic. Similarly, when both VMs are migrated with post-copy the VM running the Netperf server has a higher total migration time compared with that of the other VM because of the incoming network traffic contention at the destination host. Note that the performance of Netperf also suffers with pre-copy and post-copy by 27% and 30%, respectively, compared with its performance without migration.

Because of the small writable working set of the VM, the amount of data transferred with all three approaches is almost equal. Since pre-copy re-transfers the dirtied VM pages, pre-copy transfers more data than post-copy. Figure 1 shows the throughput of Netperf during VMs' migration. When we migrate both VMs with pre-copy, for the VM running Netperf server, the direction of the Netperf traffic complements the direction of the migration traffic. Therefore it completes migration earlier (at the 57 seconds mark) than the VM running the Netperf client. However, due to contention of traffic for the VM running the Netperf client, the throughput of Netperf remains low until it completes the migration. With post-copy, the performance of Netperf drops to 150Mbps right after starting the migration. During this time VM requests its minimal working set from the source host to become responsive. Once the VM becomes responsive its performance recovers to about 500 Mbps; however it remains lower than the maximum performance due to the contention of incoming application traffic at the VM running the Netperf server with the migration traffic.

The experiment demonstrates that the contention of the VM application traffic with the migration traffic increases the total migration time of the VMs and degrades the performance of the network-bound VM application.

IV. DESIGN

In this section, we present the design of the traffic-sensitive migration; we present our approach for the selection of migration techniques for co-located VMs and we describe the method to calculate the possible network contention at the source and destination hosts.

We make the following assumptions in the design. First, we address the network contention at the migration endpoints, that is, the source and the destination hosts. The core network switches also experience additional load during the migration. Although traffic-sensitive migration relieves the load on the core switches due to reduction in the amount of data transferred during migration, reducing the core network congestion is not the primary focus of our work. Second, we use 1 Gbps NICs that provide full bi-directional bandwidth. The 1 Gbps NICs are widely used in datacenters [32], which often become a bottleneck especially during network intensive operations, such as VM migration. Even though our approach can benefit high bandwidth interconnects, such as 10 GbE NICs currently adopted in many datacenters, it is only a question of degree. Third, our prototype considers the migration of VMs between same source and destination hosts. In a datacenter environment, the VMs that communicate with each other [35], [23] or complement each other's services [37] are often hosted on the same machine. Such VMs are migrated together between the same source and destination hosts in order to preserve the locality of VMs.

A. Selection of VM Migration Techniques for Co-located VMs

To calculate the network contention, we consider the rate of outbound VM traffic at the source host and the rate of inbound VM traffic at the destination host. The outgoing traffic from all the VMs migrated by using pre-copy contends with the migration at the source host whereas the incoming traffic

towards all the VMs migrated using post-copy contends with the migration at the destination host. Since the performance of migration depends upon the slower of the two endpoints, the maximum of the traffic contending at the source and the destination host decides the severity of contention. We periodically measure the incoming and outgoing traffic for each VM before the migration. The monitoring allows traffic-sensitive migration to estimate the contending traffic during the migration. Then we consider all the combinations of pre-copy and post-copy for all the VMs migrating from the same host, and we measure the possible contention with each combination. The combination of migration techniques that yields the lowest traffic contention is selected for migration. For pre-copy migration, reduction in the network contention at the source host allows it to quickly converge on its WWS, which reduces the amount of data transferred during the migration. The reduction in the network traffic also reduces the load on datacenter switches during the migration.

When some of the co-located VMs are migrated by using pre-copy while others by using post-copy, the traffic between the co-located VMs can also contend with the migration traffic. The VMs migrated with post-copy immediately switch their execution state to the destination host. They communicate with the VMs at the source host that are migrating with pre-copy. When the traffic between them is in the same direction as that of migration, it competes with the migration traffic. We address this problem by accounting for the traffic from the VMs migrated with pre-copy toward the VM migrated with post-copy. Since this traffic is outgoing for the source host and incoming for the destination host, we account it for both the contention at the source and destination hosts.

B. Calculating the Network Contention

In this section we describe the model used by traffic-sensitive migration to calculate the network traffic contention for the migration of co-located VMs. We define the contending traffic as any network traffic that competes with the migration traffic for available bandwidth at the source or the destination host. On a machine hosting VMs this traffic consists mainly of incoming and outgoing traffic of the network-bound VM application. VMs that cooperate and communicate with each other to jointly provide a certain service, such as multi-tier applications, often show a traffic pattern that persists for a long time [19]. Even though the rate of the VM network traffic may suddenly vary during the migration, such changes may not be entirely predictable. Therefore our model uses the profile of network traffic monitored just before the migration in order to calculate the possible contention at the source and the destination hosts.

To calculate the level of contention, we individually consider each source host and calculate the contention resulting from all the VMs running on that host. We calculate the network contention by considering both pre-copy and post-copy techniques for every VM on the same host. Figure 2 shows the equations used for calculating the contention. Equation (1) calculates the possible contention at the source host with a given combination of migration techniques for all VMs from the same host. The contention is calculated by adding rate of all the outgoing traffic from the VMs migrated with pre-copy. Similarly, Equation (2) gives the contention at the destination host, which is calculated by adding the rate of incoming traffic towards the VMs migrated with post-copy. Other background outgoing traffic at the source and incoming traffic at the destination is also accounted toward the contention at the respective hosts. All the traffic from the VMs migrated pre-copy towards the VMs migrated with post-copy is added in both the source and the destination contentions, whereas all the traffic between two VMs migrating with pre-copy or two VMs migrating with post-copy is disregarded. Then we consider the maximum of the traffic at the source and the destination as the contending traffic for a given combination. The motivation behind considering the maximum of the two is that the endpoint with the least available bandwidth becomes a bottleneck for the migration and slows the migration, thus delaying the deprovisioning of the resources at the source host. Once the network contention for every combination of migration techniques is calculated, the combination yielding the least contention is selected for migration.

The time complexity of considering every combination of pre-copy and post-copy for n number of co-located VMs is $O(2^n)$. Even though our current algorithm yields the best combination for up to 20 VMs within 0.2 seconds, it may not be suitable for large number of VMs. In such cases, we can use an approach that considers only a limited number of combinations and select the best among them that yields the lowest contention. For instance, when n is large, for the first $n - 20$ VMs, we can select a VM migration technique individually according to their predominant traffic direction, whereas for the remaining 20 VMs, we can compare different combinations of migration techniques to reduce the overall contention. Various industry surveys, however, show that this level of consolidation is not common. The average consolidation ratio in datacenters is around 6 VMs per host and rarely exceeds 15 [34], [8].

C. Disk I/O Intensive VMs

Cloud providers allow customers to select the storage type for their VM instances. The customers can choose local storage, which does not require network access or they can choose a shared storage connected with a dedicated high-bandwidth interconnect to meet high I/O requirement. In such cases, the migration traffic does not contend with the disk I/O traffic. However, when the network card is shared by the I/O traffic of the shared storage and the migration traffic, the contention between them can also adversely impact the performance of migration. Although here we focus on the migration network intensive VMs, traffic-sensitive migration can potentially be employed to reduce the contention between disk I/O and migration traffic. In traffic-sensitive migration, we can detect the inbound and outbound per-VM disk I/O traffic rates and account them to select a migration technique for each VM using the above approach.

$$\text{Source Contention} = \sum_{i=1}^{n} \text{Rate of outgoing traffic in Mbps for } VM_i, \text{ if migrated with pre-copy} + \text{ Outgoing background traffic} \quad (1)$$

$$\text{Destination Contention} = \sum_{i=1}^{n} \text{Rate of incoming traffic in Mbps for } VM_i, \text{ if migrated with post-copy} + \text{ Incoming background traffic} \quad (2)$$

$$\text{Contending traffic} = \text{Max (Source contention, Destination contention)} \quad (3)$$

Fig. 2. Calculating the network contention using the network traffic information of the co-located VMs.

V. Implementation

Here we present the implementation of the traffic-sensitive virtual cluster migration on the KVM/QEMU platform. Our implementation consists of the following components.

1) *Migration manager* is responsible for performing the VM migration. A migration manager at the source host establishes a TCP connection with the migration manager at the destination host and transfers the VM state over the connection. It coordinates with the central server to begin the migration.
2) *Traffic monitoring module* at the source host periodically monitors the VM traffic with the help of iptables and updates the central server.
3) *Central server* calculates the network contention and selects a migration technique for each VM.

1) Migration Manager: On the KVM/QEMU platform each VM is created as a user process. A part of process's address space is used as a physical memory for the VM. For the migration of the VM, the KVM/QEMU process spawns a thread, referred to as the *migration manager*. Since the migration manager is created by the KVM/QEMU process, it can access the VM's memory. During the migration, the migration manager at the source host establishes a TCP connection with the migration manager running at the destination host and transfers the VM's memory and execution state over the connection. The KVM/QEMU provides a command console to configure a migration technique for the VM.

Traffic-sensitive migration is implemented on the KVM/QEMU platform. We use the pre-copy migration provided in the code-base of the KVM/QEMU, whereas we use the publicly available post-copy migration code from Yabusame [18]. We modify the migration manager to implement traffic-sensitive migration. In traffic-sensitive migration, upon beginning the migration, the migration manager establishes a TCP connection with the central server. The central server instructs each VM to begin the migration with the instructed migration technique; either pre-copy or post-copy. The central server select a migration technique for each VM depending on the selected combination of migration techniques that yields the least contention.

2) Traffic Monitoring Module: Traffic monitoring module monitors the traffic for the VMs configured by using a bridged network. The bridged network allows all VMs to become a part of the same subnet as the host and makes them appear as normal hosts to the rest of the network. A bridge is a link layer device that directs the traffic between networks based on

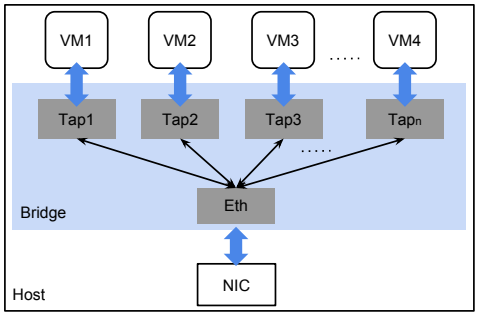

Fig. 3. Virtual networking in KVM/QEMU through bridging.

the MAC addresses on the packets. Within a host, a software bridge emulates the hardware bridge and allows the packets to be directed to and from the VMs. The VMs are connected to the bridge through network tap (TAP) devices, which act as network channels and provide networking capability to the VMs. The TAP devices receive networking packets from the host network stack and pass them to the VMs. Figure 3 shows how VMs communicate with the external network using TAP devices.

We monitor the incoming and outgoing network traffic passing through the TAP device. Every 3 seconds, the module for each VM updates the central server with the rate of incoming and outgoing traffic. Our evaluation indicates that such periodic monitoring of network traffic has negligible overhead and does not adversely impact VMs' performance. The traffic monitoring is performed at the source host for all VMs using iptables. iptables is a program that allows the system administrator to configure a firewall. It maintains a tables of IP packet filter rules in the Linux kernel. Each table consists of chains and each chain consists of a list of rules. A rules defines an action to be performed on the packet that matches the specified criteria. This action is referred to as the *target*. For instance an action can be a jump to a different chain in the same table or dropping the packet. When the packet does not match, the next rule in the chain is considered. The iptables rules continuously monitor the incoming and outgoing packets from each VM.

Figure 4 shows the rules used for monitoring the traffic. First two instructions create two chains, IN and OUT. The third instruction defines a jump to the target chain OUT when the outgoing packets directed from a TAP device (tap1) to the NIC (eth0) are encountered. Similarly, the fourth instruction defines a jump to the target IN when incoming packets (eth0 to tap1) are encountered. The instruction 5 shows an example of a rule to monitor the traffic between co-located VMs (tap1

```
iptables -N IN                                                                    (1)
iptables -N OUT                                                                   (2)
iptables -I FORWARD -m physdev --physdev-in tap1 -m physdev --physdev-out eth0 -j OUT   (3)
iptables -I FORWARD -m physdev --physdev-in eth0 -m physdev --physdev-out tap1 -j IN    (4)
iptables -I FORWARD -m physdev --physdev-in tap1 -m physdev --physdev-out tap2 -j OUT   (5)
iptables -L -vnx                                                                  (6)
```

Fig. 4. iptables rules used to capture the incoming and outgoing packet to and from the VMs.

to tap2). The chains, IN and OUT are empty, that is they do not contain any rules. Therefore packets continue further without any further filtering. The sixth instruction extracts the information about the number of packets encountered. We use this information to monitor the rate of incoming and outgoing traffic for each VM.

To monitor the traffic, we use timer of the KVM/QEMU. We register a timer handler routine, which is called every 3 seconds by the KVM/QEMU process. The handler extracts the rate of incoming and outgoing traffic using iptables instructions described above and calculates an exponential moving average. This average is sent to the central server over a TCP connection.

3) Central Server: Figure 5 shows the communication between the KVM/QEMU process, the migration manager and the central server for traffic-sensitive migration of VMs. The KVM/QEMU process on each source host establishes a TCP connection with the central server to periodically update it with the corresponding VM's network traffic information. The central server keeps track of the per VM information. When the administrator initiates the migration of the VMs, each VM's migration manager establishes a TCP connection with the central server and waits for the instruction from the central server to begin the migration. The central server uses the network traffic information to select the best possible combination of migration techniques for each host to minimize the network contention. With a given combination, each VM is migrated either using pre-copy or post-copy migration. Then the central server simultaneously instructs all the migration managers to initiate the migration with a given migration technique. The central server can run on any server running VMs, and it need not be a dedicated server. The network overhead of communication between the network monitoring modules of VMs and the central server is negligible. The central server receives only 18 bytes per-VM every 3 seconds.

VI. Evaluation

In this section we present an evaluation of the traffic-sensitive migration by comparing it against pre-copy only and post-copy only approaches. Our testbed consists of hosts, each equipped with 8 CPUs and 16 GB of physical memory. Each host is connected to a top of the rack switch with a 1 Gbps Ethernet link. The VM images are stored using network attached storage, thus their migration is not required.

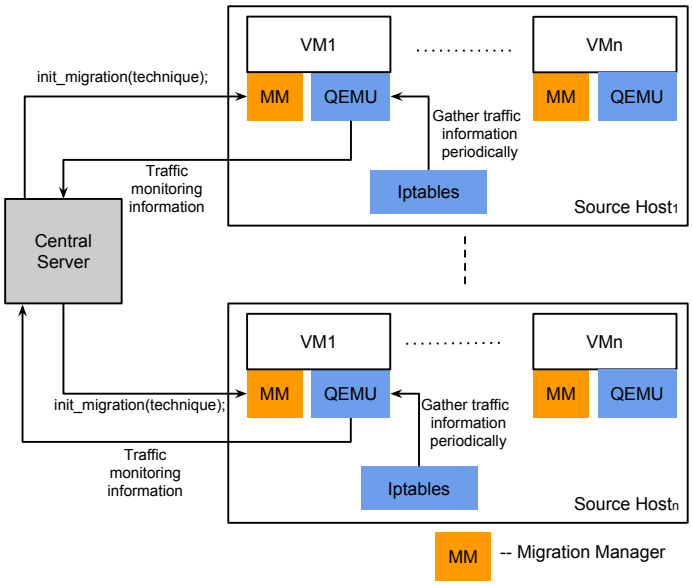

Fig. 5. Communication between central server and QEMU for traffic-sensitive migration of a VM

A. Effect of Network Contention on the Migration of a Single VM

In this section we evaluate the effect of the direction of VM traffic on the total migration time and the performance of the VM application by migrating a single VM. The VM is configured with 5 GB of memory and 2 vCPUs. It executes a Yahoo! Cloud Serving Benchmark (YCSB) [4], a benchmarking system to evaluate the performance of key-value stores. The YCSB client running inside the VM queries a REmote DIctionary Server (Redis) [28], an in-memory key-value store, running on an external host. We load the Redis server with 5GB of dataset and submit read and insert queries from the YCSB client. We migrate the VM while the test is in progress, and we measure the total migration time and the application's performance over 1.5 million requests. We use both pre-copy and post-copy techniques for migration and compare their performance.

1) Read Operations: The read queries from the YCSB are smaller in size compared to their response (request:response == 1:29). Therefore VM experiences more incoming traffic from the Redis server compared with the outgoing traffic toward it. Figure 6 shows the total migration time for the migration of the VM with an increasing rate of read queries. For pre-copy migration, the migration traffic and the read

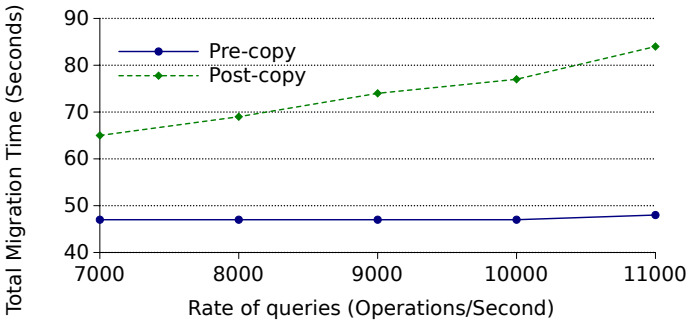

Fig. 6. Comparison of total migration time for pre-copy and post-copy migration of a 5 GB VM running a YCSB client. YCSB client queries an external dataset with read requests.

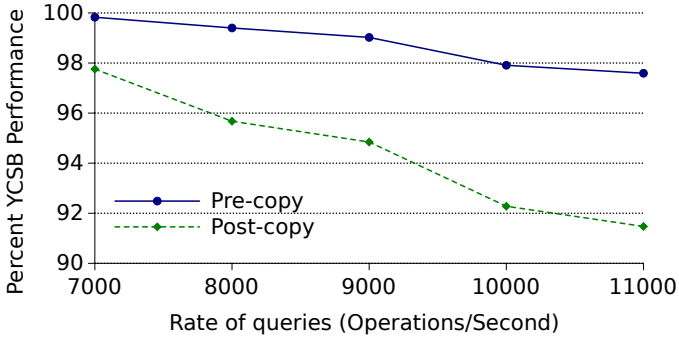

Fig. 7. Comparison of percent YCSB performance for pre-copy and post-copy migration of a 5 GB VM running a YCSB client. YCSB client queries an external dataset with read requests.

response traffic have opposite directions, and both flows can use full available bandwidth in each direction. Therefore the total migration time remains constant for an increasing number of requests. However, with post-copy, the migration traffic contends with the read response traffic, and the total migration time of the VM increases. With 11000 read requests/second, the total migration time is 75% higher than that with pre-copy.

Figure 7 shows the corresponding performance of YCSB in terms of the percentage of the target rate of requests. With increasing rate of requests, the performance of YCSB decreases with both pre-copy and post-copy. But the degradation is higher with post-copy. For 11000 requests/second, it can deliver only 91% of the performance.

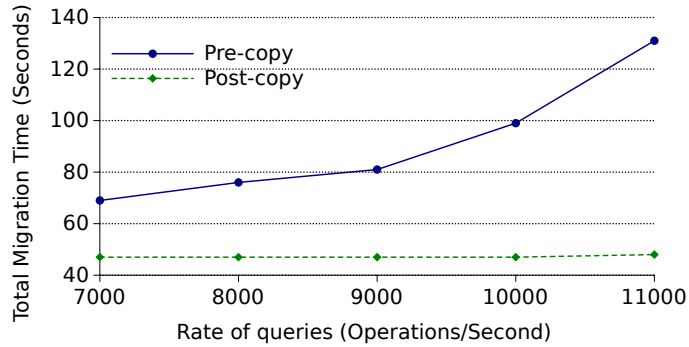

Fig. 8. Comparison of total migration time for pre-copy and post-copy migration of a 5 GB VM running a YCSB client. YCSB client queries an external dataset with insert requests.

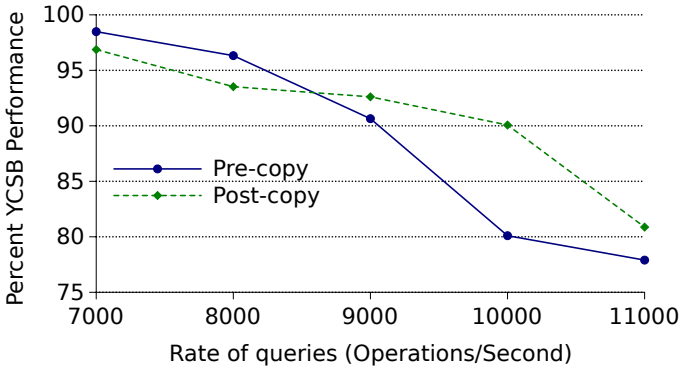

Fig. 9. Comparison of the percentage of the YCSB performance for pre-copy and post-copy migration of a 5 GB VM running a YCSB client. The YCSB client queries an external dataset with insert requests.

2) Insert Operations: Here we submit insert requests to the Redis server from the YCSB client executing inside the VM. For insert requests, the request size is higher than that of the response. Therefore with pre-copy the migration traffic contends with the application traffic whereas with post-copy the direction of the migration traffic complements the direction of the application traffic. Figure 8 shows the total migration time of the VM with an increasing rate of insert requests. It can be seen that the total migration time with post-copy remains constant whereas it increases with pre-copy. For 11000 insert requests/second the total migration time with pre-copy is 180% more than that with post-copy. Figure 9 shows the performance of the application. With a lower rate of requests (less than 9000 operations/second), the application performs worse with post-copy than with pre-copy. This is because with post-copy the VM executing at the destination host has to retrieve its pages from the source host on encountering a page fault. The delay in retrieval of pages over the network degrades the performance of the VM application. However, as the rate of requests increases the performance of YCSB degrades quickly with pre-copy, whereas post-copy yields better performance.

These results demonstrate that a VM migration technique that contends with the VM application traffic not only degrades the application performance but also increases the total migration time of the VM. On the other hand, selecting a migration technique so that the direction of the application traffic complements the direction of the migration traffic, reduces the contention and yields a lower total migration time and application degradation.

B. Migration of Multiple Communicating VMs

Recall that in Section III we demonstrated the effect of network contention on the performance of migration and the VMs running network-bound application. In this section we show that traffic-sensitive migration reduces the network contention for the migration of multiple communicating VMs by selecting a VM migration technique according to each VM's traffic profile. This reduces the total migration time of the VMs and the adverse impact of migration on the VM applications. We use the same setup; that is, we migrate two 5 GB VMs from the two source hosts to the two destination

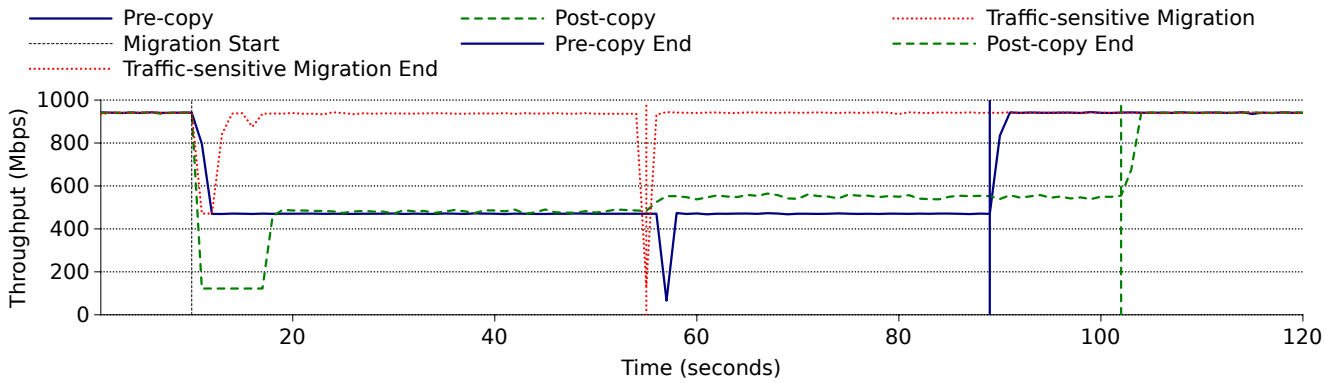

Fig. 10. Throughput of Netperf during the migration of two communicating VMs executing Netperf client and server with pre-copy, post-copy and traffic-sensitive migration.

hosts. The first VM executes a Netperf client while the second VM runs a Netperf server. We migrate both VMs while the TCP stream test is in progress with pre-copy and post-copy. The traffic-sensitive migration selects pre-copy for the Netperf server with incoming traffic and selects post-copy for the Netperf client with outgoing traffic. We compare the total migration time and the amount of data transferred with traffic-sensitive migration with those of pre-copy and post-copy. As we observed in Section III, with pre-copy and post-copy the outgoing and incoming Netperf traffic contends with the migration traffic, respectively. Therefore for the VM with same direction of traffic as the migration, the total migration increases. In contrast, with traffic-sensitive migration, selection of a migration technique for each VM always complements the direction of its application traffic and therefore has the lowest total migration time.

Figure 10 shows the throughput of Netperf during VMs' migration. When we migrate both VMs with pre-copy or post-copy, the performance of Netperf degrades during the migration due to the network contention. With traffic-sensitive migration the throughput of Netperf degrades at the beginning of the migration due to transfer of the second VM's (running Netperf server) CPU context to the destination. However performance recovers quickly as the VM retrieves its working set from the source host. Also note that without traffic contention, the VM becomes responsive sooner than with post-copy. Then onwards the Netperf client and server can communicate with each other at maximum throughput through the migration. At 55 seconds the first VM (running the Netperf client) experiences a downtime phase leading to slight degradation before recovering again to the maximum performance.

Table II compares the performance of migration and application with all three techniques. On average Netperf shows 5% throughput degradation with our approach as opposed to 27% and 30% with pre-copy and post-copy, respectively. Also the total migration time with traffic-sensitive migration time is 42% and 49% lower than with pre-copy and post-copy respectively.

C. Migration of Multiple VMs from Multiple Hosts

In this section we migrate 16 VMs from 8 hosts, i.e. 2 VMs per host. Each VM is configured with 2 GB of memory and 2 vCPUs. We run Redis servers in 12 VMs each containing a 1.5 GB in-memory dataset. The remaining 4 VMs run the YCSB client to query the dataset, each querying 6 VMs. The queries include read, insert, and update operations. Therefore each VM has a predominant incoming or outgoing network flow. The setup represents a scenario when the VMs that communicate and co-operate to jointly provide certain service are migrated simultaneously.

With pre-copy and post-copy, we migrate all 16 VMs using respective migration technique, whereas with traffic-sensitive migration the migration technique for each VM is selected based on its traffic profile. We simultaneously migrate all 16 VMs while the test is in progress using pre-copy, post-copy, and traffic-sensitive migration. Table III shows the performance of the migration.

1) Total Migration Time: The total migration time of a VM depends upon the amount of data transferred and the available bandwidth. The total migration time for the migration of multiple VMs is the time measured from the start of the migration of the first VM to the end of the migration of the last VM. Whereas the average migration time is the average of individual total migration times calculated over all the migrating VMs. When all VMs are migrated with pre-copy the total migration time of VMs increases due to retransmission of dirtied pages and contention of the migration traffic with the outgoing VM traffic. When all the VMs are migrated with post-copy the average and total migration times increase due to contention of the incoming VM traffic with the migration traffic. The increase in the total migration time with post-copy is in spite of transferring each VM page only once. In contrast, traffic-sensitive VM migration technique reduces the traffic contention by selecting a suitable migration technique for each VM. Thus, it has the lowest average and total migration times among the three techniques.

2) Amount of Data Transferred: Table III shows the aggregate amount of data transferred for the migration of 16 VMs. It can be observed that pre-copy transfers more data than other two techniques. As noted previously, with pre-

	Pre-copy	Post-copy	Traffic-sensitive Migration
Total Migration Time (seconds)	79.1	92.1	48.2
Amount of Data Transferred (MB)	10280	10277	10278
Netperf Performance (Mbps)	690.47	660.05	894.65

TABLE II
COMPARISON OF THE PERFORMANCE OF MIGRATION WHEN THE COMBINATION OF PRE-COPY AND POST-COPY IS USED IN TRAFFIC-SENSITIVE MIGRATION AGAINST THE PERFORMANCE OF PRE-COPY AND POST-COPY.

	Without Migration	Pre-copy	Post-copy	Traffic-sensitive Migration
Average Migration Time (seconds)	-	50.56	60.48	37.79
Total Migration Time (seconds)	-	74.5	139	57.75
Amount of Data Transferred (GB)	-	50.90	30.18	34.07
YCSB Performance (Operations / second)	4802	3875	4161	4126

TABLE III
PERFORMANCE COMPARISON FOR THE MIGRATION OF VMS FROM 8 HOSTS, WITH 2 VMS PER HOST. OUT OF 16 VMS, 4 VMS RUN YCSB CLIENTS WHEREAS 8 VMS RUN THE REDIS SERVERS.

copy the traffic contention increases the total migration time and more pages are dirtied. Therefore it transfers significantly more data than post-copy. In contrast, post-copy transfers each VM page only once, therefore it has the lowest amount of data transferred. With traffic-sensitive migration, the VMs migrated with pre-copy face less network traffic contention than with the pre-copy (for all VMs) approach. Therefore VMs can quickly converge on their WWS. Therefore traffic-sensitive migration only slightly increases the amount of data transferred as compared to post-copy.

3) Application Degradation: We measure the performance of the application by calculating the average performance of all the YCSB workload clients. The application performs the best with post-copy among the three techniques. Post-copy transfers the lowest amount of data; therefore the application suffers the least. With pre-copy, the retransmission of dirtied pages adversely impacts the performance of the migration, yielding the lowest performance. Traffic-sensitive migration performs slightly worse than post-copy since more data is transferred for the migration. However, due to significant reduction in the amount of data transferred the application performs notably better than with pre-copy.

VII. RELATED WORK

In this section we review the literature that optimize the migration of a single VM or multiple VMs by reducing the network overhead.

Post-copy [16], [18] transfers each VM pages only once over the network, thus reducing the total migration time and network traffic overhead for write-intensive VM applications compared with pre-copy [7], [26]. Content-optimizations such as compression [21], deduplication [40], [36], and differential compression [31] are used to reduce the amount of data transferred for VM migration. Jo et al. [22] avoid the transfer of VM memory pages that are identical to the disk blocks from network attached storage. Shrinker [30], [29], Gang Migration [11], [10], [12] optimize live migration of multiple VMs over datacenter and wide area networks. The above approaches reduce the network traffic for live VM migration and hence reduce the network contention. However, they rely on a single VM migration technique, which may not be best suited for a given VM traffic profile. Moreover, since traffic-sensitive migration uses both pre-copy and post-copy, any content optimization for these migration techniques is orthogonal to the approach and may further reduce the network contention.

VMFlock [24] performs non-live migration of VM images over wide area networks. Redundancy-aware virtual disk mobility [27] also migrates VM images between datacenters. It uses a peer-to-peer approach to gather the identical VM image blocks from multiple data centers. Adaptive live migration over WAN [39], presents a fractional hybrid pre-copy approach for memory and storage over WAN. It transfers only a fraction of the memory and storage during the pre-copy phase whereas the remaining state is demand-paged. The fraction is adjusted to quickly restore the migrating VM's performance back to its original level. The above approaches focus on the migration of VMs over WAN, where migration of storage is a main concern. For intra-datacenter migration of VMs over LAN often the storage is shared between the source and the destination hosts.

VCT [1] performs synchronized migration of a HPC virtual clusters running data-driven applications by reducing the amount of data transferred for their migration using compression. However, the main focus of this work is non-live migration of VMs and their disk images, when the VM memory and disk states are suspended to a file and resumed at a later time. Ye et al. [38] perform live migration of virtual clusters. They compare the performance of different migration strategies for virtual cluster, including concurrent, sequential, unidirectional, bi-directional migration of virtual clusters. However, they do not focus on the contention of migration and VM application traffic. Further, they only consider the simultaneous migration of VMs using pre-copy. In contrast, our work employs both pre-copy and post-copy migration to minimize the mutual adverse impact of VM application traffic and migration traffic.

To the best of our knowledge traffic-sensitive migration is the first approach to take into account the traffic direction of the VM applications in order to to reduce the network contention for the migration of co-located VMs.

VIII. CONCLUSIONS AND FUTURE WORK

In this paper, we presented a traffic-sensitive approach for migration of co-located VMs that predominantly have

unidirectional network traffic. Our approach reduces the contention of VM application traffic with the migration traffic by selecting a suitable migration technique for each VM. The selection is based on each VM's network traffic profile. We have implemented a prototype of traffic-sensitive migration in the KVM/QEMU environment. We demonstrate through evaluation that by reducing the network traffic contention for migration, traffic-sensitive migration reduces the total migration time of VMs and minimizes the adverse impact of migration on the performance of the VM application.

We plan to extend the traffic-sensitive migration in the following way. Currently, traffic-sensitive migration addresses the problem of the migration of VMs from the same source host to the same destination host. In the future, we plan to extend traffic-sensitive migration for migration of VMs from the same source host to different destination hosts. Second, we plan to account for the traffic at the destination hosts to select a suitable destination host for each VM to minimize the network contention.

ACKNOWLEDGEMENTS

We thank Pierre Riteau for his help in setting up the testbed and for his valuable suggestions that helped improve this paper. This material is based in part on work supported in part by the U.S. Department of Energy, Office of Science, under contract DE-AC02-06CH11357.

REFERENCES

[1] Paolo Anedda, Simone Leo, Simone Manca, Massimo Gaggero, and Gianluigi Zanetti. Suspending, migrating and resuming hpc virtual clusters. In *Future Generation Computer Systems*, volume 26 - 8, 2010.

[2] P. Barham, B. Dragovic, K. Fraser, S. Hand, T. Harris, A. Ho, R. Neugebauer, I. Pratt, and A. Warfield. Xen and the art of virtualization. *SIGOPS Operating Systems Review*, 37(5):164–177, 2003.

[3] Network Performance Benchmark. http://www.netperf.org/netperf.

[4] Yahoo! Cloud Serving Benchmark. http://labs.yahoo.com/news/yahoo-cloud-serving-benchmark.

[5] N. Bila, E. de Lara, K. Joshi, H. A. Lagar-Cavilla, M. Hiltunen, and M. Satyanarayanan. Jettison: Efficient idle desktop consolidation with partial VM migration. In *Eurosys*, April 2012.

[6] N. Bobroff, A. Kochut, and K. Beaty. Dynamic placement of virtual machines for managing sla violations. In *Proc. of Integrated Network Management*, page 119–128, May 2007.

[7] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live migration of virtual machines. In *Network System Design and Implementation*, May 2005.

[8] VMWare: Server Consolidation and Containment. http://www.vmware.com/pdf/server_consolidation.pdf.

[9] T. Das, P. Padala, V. Padmanabhan, R. Ramjee, and K. G. Shin. LiteGreen: Saving energy in networked desktops using virtualization. In *USENIX Annual Technical Conference*, 2010.

[10] U. Deshpande, U. Kulkarni, and K. Gopalan. Inter-rack live migration of multiple virtual machines. In *Virtualization Technologies in Distributed Computing*, June 2012.

[11] U. Deshpande, B. Schlinker, E. Adler, and K. Gopalan. Gang migration of virtual machines using cluster-wide deduplication. In *International Symposium on Cluster, Cloud and Grid Computing*, May 2013.

[12] U. Deshpande, X. Wang, and K. Gopalan. Live gang migration of virtual machines. In *High Performance Distributed Computing*, June 2011.

[13] U. Deshpande, Y. You, D. Chan, N. Bila, and K. Gopalan. Fast server deprovisioning through scatter-gather live migration of virtual machine. In *IEEE International Conference on Cloud Computing*, July 2014.

[14] Pei Fan, Zhenbang Chen, Ji Wang, Zibin Zheng, and Michael R. Lyu. Topology-aware deployment of scientific applications in cloud computing. In *Proc. of International Conference on Cloud Computing*, 2012.

[15] Abhishek Gupta, Dejan Milojicic, and Susanne M. Balle. Hpc-aware vm plancement in infrastructure clouds. In *Proc. of International Conference on Cloud Engineering*, 2013.

[16] M. R. Hines, U. Deshpande, and K. Gopalan. Post-copy live migration of virtual machines. *SIGOPS Operating System Review*, 43(3):14–26, 2009.

[17] T. Hirofuchi, H. Nakada, S. Itoh, and S. Sekiguchi. Reactive consolidation of virtual machines enabled by postcopy live migration. In *Virtualization Technologies in Distributed Computing*, June 2011.

[18] T. Hirofuchi and I. Yamahata. Yabusame: Postcopy Live Migration for Qemu/KVM. In *KVM Forum 2011, Vancouver, Canada*, August 2011.

[19] Liting Hu, Karsten Schawan, Ajay Gulati, Junjie Zhang, and Chengwei Wang. Net-cohort: detecting and managing vm ensembles in virtualized data centers. In *Proc. of International Conference on Autonomic computing*, 2012.

[20] VMware Inc. http://www.vmware.com.

[21] H. Jin, L. Deng, S. Wu, X. Shi, and X. Pan. Live virtual machine migration with adaptive memory compression. In *Cluster Computing and Workshops*, August 2009.

[22] C. Jo, E. Gustafsson, J. Son, and B. Egger. Efficient live migration of virtual machines using shared storage. In *Virtual Execution Environments*, March 2013.

[23] K. Kim, C. Kim, S-I. Jung, H Shin, and J-S. Kim. Inter-domain socket communications supporting high performance and full binary compatibility on xen. In *Proc. of Virtual Execution Environments*, March 2008.

[24] S. A. Kiswany, D. Subhraveti, P. Sarkar, and M. Ripeanu. Vmflock: Virtual machine co-migration for the cloud. In *High Performance Distributed Computing*, June 2011.

[25] A. Kivity, Y. Kamay, D. Laor, U. Lublin, and A. Liguori. KVM: the linux virtual machine monitor. In *Linux Symposium*, June 2007.

[26] M. Nelson, B. H Lim, and G. Hutchins. Fast transparent migration for virtual machines. In *USENIX Annual Technical Conference*, 2005.

[27] C. Peng, M. Kim, Z. Zhang, , and H. Lei. Vdn: Virtual machine image distribution network for cloud data centers. In *International Conference on Computer Communications*, March 2012.

[28] Redis. http://redis.io.

[29] P. Riteau, C. Morin, and T. Priol. Shrinker: Efficient wide area live virtual machine migration using distributed content-based addressing. In *http://hal.inria.fr/inria-00454727/en/*, February 2009.

[30] P. Riteau, C. Morin, and T. Priol. Shrinker: Improving live migration of virtual clusters over WANs with distributed data deduplication and content-based addressing. In *EURO-PAR*, September 2011.

[31] Petter Svrd, Benoit Hudzia, Johan Tordsson, and Erik Elmroth. Evaluation of delta compression techniques for efficient live migration of large virtual machines. In *Proc. of International Conference on Virtual Execution Environments*, 2011.

[32] Arista: Upgrading the Data Center to 10 Gigabit Ethernet. http://www.arista.com/assets/data/pdf/10gige_whitepaper.pdf.

[33] A. Verma, P. Ahuja, and A. Neogi. pMapper: power and migration cost aware application placement in virtualized systems. In *Middleware'08*.

[34] V-Index: virtualization industry quarterly survey. http://www.veeam.com/news/veeam-launches-v-index-to-measure-virtualization-penetration-rate.html.

[35] Jian Wang, Kwame-Lante Wright, and Kartik Gopalan. XenLoop: a transparent high performance inter-VM network loopback. In *Proc. of High Performance Distributed Computing, Boston, MA, USA*, pages 109–118, 2008.

[36] T. Wood, K. K. Ramakrishnan, P. Shenoy, and J. van der Merwe. Cloudnet: Dynamic pooling of cloud resources by live wan migration of virtual machines. In *Virtual Execution Environments*, March 2011.

[37] T. Wood, P. Shenoy, A. Venkataramani, and M. Yousif. Sandpiper: Black-box and gray-box resource management for virtual machines. *The International Journal of Computer and Telecommunications Networking*, 53(17), December 2009.

[38] Kejiang Ye, Xiaohong Jiang, Ran Ma, and Fengxi Yan. Vc-migration: Live migration of virtual clusters in the cloud. In *Proc. of International Conference on Grid Computing*, 2012.

[39] W. Zhang, K. T. Lam, and C. Wang. Adaptive live vm migration over a wan: Modeling and implementation. In *IEEE International Conference on Cloud Computing*, July 2014.

[40] X. Zhang, Z. Huo, J. Ma, and D. Meng. Exploiting data deduplication to accelerate live virtual machine migration. In *Proc. of International Conference on Cluster Computing*, September 2010.