2016 IEEE International Conference on Cluster Computing

Quick Eviction of Virtual Machines Through Proactive Snapshots

Dinuni Fernando*, Hardik Bagdi*, Yaohui Hu*, Ping Yang*, Kartik Gopalan*, Charles Kamhoua†, Kevin Kwiat†
* Computer Science Dept., Binghamton University, Binghamton, NY, USA
Email: {dferna15,hbagdi1,yhu15,pyang,kartik}@binghamton.edu
† Air Force Rome Research Laboratory, Rome, NY, USA Email: {charles.kamhoua.1, kevin.kwiat}@us.af.mil

***Abstract*—Live migration of Virtual Machines (VMs) is a key technique to quickly migrate workloads in response to events such as impending failure or load changes. Despite extensive research, state-of-the-art live migration approaches take a long time to migrate a VM, which in turn negatively impacts the application performance during migration. We present, Quick Eviction, a new approach to significantly speed up the eviction of a VM from the source host with low impact on VM's performance during migration. Before migration, Quick Eviction regularly snapshots the VM's memory to a destination or a failover node. During the actual migration, Quick Eviction has to transfer only a small amount of dirtied memory resulting in a very short time to completely evict the VM out of the source. Our experimental results show that Quick Eviction in the KVM/QEMU platform significantly reduces the eviction time.**

***Index Terms*—Live migration, virtual machines**

I. Introduction

Live migration of a virtual machine (VM) refers to the transfer of a running VM's state from one physical machine (the *source*) to another physical machine (the *destination*). Live VM migration helps to decouple applications running inside a VM from the hardware on which the applications execute, and hence is crucial for activities like failure resilience, load balancing, and server maintenance.

The two pre-dominant live migration techniques, namely pre-copy[1] and post-copy[4], have primarily focused on reducing *eviction time* and *downtime*. The eviction time refers to the time to evict a VM's state from the source machine [2, 3], whereas the downtime is the duration for which the VM is completely paused during migration. All existing live migration techniques have a key limitation that the eviction time is determined by the total memory size of the VM being migrated. High eviction time may increase the response time to impending failures or degrade the performance of applications running inside the VMs.

We present *Quick Eviction*, a technique for reducing the eviction time of a VM with low impact on its performance. The basic idea is to regularly transfer incremental snapshots of a VM to another machine over the network. Upon a triggering event, such as an imminent failure of the source machine, any residual memory of the VM that was updated since the last snapshot is evicted to the destination. The eviction time thus achieved is small fraction of the total memory, since the size of the final residual memory is generally a fraction of the total memory size of the VM.

Approved for Public Release; Distribution Unlimited : 88ABW-2016-2139 20160428

Quick Eviction is designed to satisfy the following requirements. First, Quick Eviction should significantly reduce the time to completely transfer the VM out of the source machine, while maintaining a low downtime during migration. Secondly, Quick Eviction should not excessively increase the network load due to snapshot operations. Thirdly, Quick Eviction should not significantly impact the performance of applications inside the VM. To achieve these goals, we develop a technique to dynamically vary the snapshot intervals based on the threshold rate of the pages dirtied by a VM during its execution, which lowers the network overhead and performance impact of snapshots.

We have implemented Quick Eviction in the KVM/QEMU virtualization platform. Our experimental results show that Quick Eviction speeds up live migration by a significant factor.

II. Design

Quick Eviction speeds up the transfer of a VM's memory state from the source by evicting the bulk of the VM's memory footprint in the background during VM's normal operation.

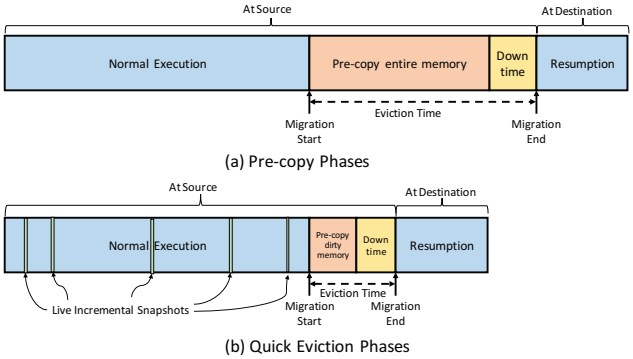

Fig. 1. Phases of live VM migration using (a) Pre-copy and (b) Quick Eviction.

Figure 1(b) demonstrates various phases of Quick Eviction algorithm. Prior to migration, the VM runs normally on the source machine. During this normal execution,

2168-9253/16 $31.00 © 2016 IEEE
DOI 10.1109/CLUSTER.2016.83

IEEE computer society

Authorized licensed use limited to: STATE UNIV NY BINGHAMTON. Downloaded on October 02,2024 at 11:52:59 UTC from IEEE Xplore. Restrictions apply.

the source machine regularly transfers live incremental snapshots of the VM's memory to the destination. In the very first live snapshot, all memory pages of the VM are transferred to the destination. In subsequent live snapshots, only the pages dirtied since the previous snapshot are transferred. Once the live migration is initiated, any pages dirtied since the last snapshot are transmitted to the destination using iterative transfers, as in pre-copy migration. During downtime, any remaining dirty pages, VCPU state, and I/O state are transferred to the destination to complete the eviction from source.

We use two techniques to reduce any impact of live snapshots on the network bandwidth and the performance of application running in VMs: *dynamic snapshot interval* and *transfer limit*. *Dynamic snapshot interval* dynamically spaces the live snapshots according to nature of the VM's workloads during normal operations. The next live snapshot operation is initiated once the dirty page count exceeds a threshold and a minimum time has passed since the last snapshot. *Transfer limit* reduces the network overhead during each live snapshot by bounding the number of pages that can be transferred during each snapshot operation.

There are several advantages of the above two mechanisms from the viewpoint of overhead reduction. First, applications that do not dirty memory very frequently pay very little penalty from dirty page tracking and memory transfers. Secondly, by limiting the number of pages transferred in each snapshot, we limit any potential interference of snapshot transfer with any network-bound applications. Thirdly, when the migration is initiated, the number of dirty pages is guaranteed to be lower than or equal to the threshold used for triggering the snapshot operation. Finally, if a VM dirties too many pages before a snapshot, then the cost of transmitting those dirtied pages is amortized over several subsequent snapshots rather than all at once in the immediate snapshot.

III. Preliminary Evaluation Results

This section compares the performance of Quick Eviction against the KVM pre-copy migration [1]. Our test environment consists of dual six-core 2.1GHz Intel Xeon machines with 128 GB memory connected through a Gigabit Ethernet switch with 1Gbps full-duplex ports. VMs in each experiment are configured with 1 vCPU and 8GB of memory with page size of 4KB unless specified otherwise. Virtual disks are accessed over the network from an NFS server.

Figures 2 compares the eviction time of an idle VM using Quick Eviction and pre-copy. The figure shows that the eviction time of Quick Eviction is very small (between $6ms$ and $246ms$). The downtime (not shown) of Quick Eviction and pre-copy are the same and constant around 5ms.

Figure 3 gives the eviction time when migrating a VM running write-intensive workloads, which repeatedly write random numbers to a large region of memory, using Quick Eviction and pre-copy, respectively. The figure shows that

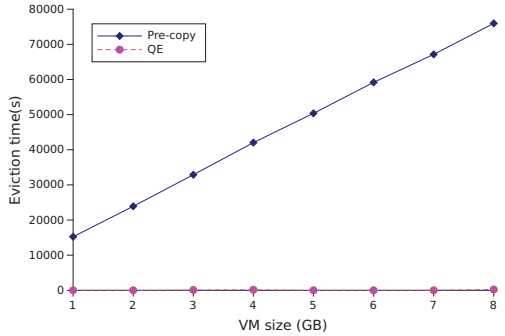

Fig. 2. Eviction time variation for different VM sizes (idle VM)

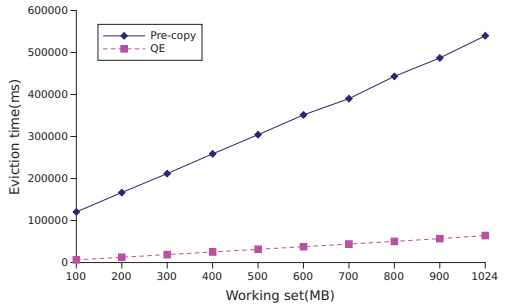

Fig. 3. Eviction time of a VM running a write-intensive application using Quick Eviction and pre-copy.

the eviction time for pre-copy increases at a much higher rate (by a factor of 8 to 18) than that of Quick Eviction.

IV. Conclusion

We presented Quick Eviction, a new approach to live migration of a VM that significantly reduces the eviction time of a VM from the source. Quick Eviction staggers the memory transfer throughout the lifetime of the VM in a controlled manner, yielding significantly low eviction times.

Acknowledgment: This work is supported in part by the Air Force Rome Research Laboratory and the National Science Foundation through grants 1320689 and 1527338.

References

[1] C. Clark, K. Fraser, S. Hand, J.G. Hansen, E. Jul, C. Limpach, I. Pratt, and A. Warfield. Live Migration of Virtual Machines. In *USENIX Symposium on Networked Systems Design and Implementation*, pages 273–286, 2005.

[2] U. Deshpande, Y. You, D. Chan, N. Bila, and K. Gopalan. Fast server deprovisioning through scatter-gather live migration of virtual machines. In *Proceedings of IEEE Cloud*, pages 376–383, 2014.

[3] Umesh Deshpande, Danny Chan, Steven Chan, Kartik Gopalan, and Nilton Bila. Scatter-gather live migration of virtual machines. *IEEE Transactions on Cloud Computing*, PP(99):1–1, 2015.

[4] Michael Hines and Kartik Gopalan. Post-copy based live virtual machine migration using adaptive pre-paging and dynamic self-ballooning. In *Virtual Execution Environments (VEE)*, pages 51–60, 2009.

Authorized licensed use limited to: STATE UNIV NY BINGHAMTON. Downloaded on October 02,2024 at 11:52:59 UTC from IEEE Xplore. Restrictions apply.