MemNet: Enhancing Throughput and Energy Efficiency for Hybrid Workloads via Para-virtualized Memory Sharing

Chi Xu*, Xiaoqiang Ma†, Ryan Shea*, Haiyang Wang‡, and Jiangchuan Liu*

*Simon Fraser University, British Columbia, Canada
†Huazhong University of Science and Technology, Hubei, China
‡University of Minnesota at Duluth, Minnesota, USA
{chix,xma10,rws1}@sfu.ca, haiyang@d.umn.edu, jcliu@cs.sfu.ca

***Abstract*—Virtualization has become a building block for modern IT industry, and many datacenters are now highly virtualized. It is known that virtualization also introduces non-trivial overhead, which can cause severe *self-interference* inside a VM when CPU intensive tasks and bandwidth intensive tasks are co-located. Energy efficiency of the server can be affected as well. While such overhead is well-studied in application/protocol specific context, a more comprehensive solution is yet to be explored for general cloud services. In this paper, we present *MemNet*, a novel protocol-independent solution that enables para-virtualized memory sharing between host and guest VMs. This design successfully decouples I/O and computation operations and lifts the offered interface from the physical devices to high-level network services. Our real-world implementation on KVM indicates that, MemNet can achieve 27% and 70% gain in terms of computing and networking performance, respectively, by resolving the self-interference. It also provides 32% improvement in terms of energy efficiency.**

I. Introduction

State-of-the-art enterprise cloud services leverage virtualization technologies to achieve high resource utilization as well as performance isolation among co-located virtual machines (VMs). Through resource multiplexing, the cloud providers can effectively reduce their operational costs, which in turn enables flexible pricing strategies to attract cloud users. Despite the benefits, it is known that virtualization introduces non-trivial overhead, which often leads to longer and unstable job completion time for typical computation-intensive applications [1]. Meanwhile, the virtualized networking subsystem is also suffering from performance degradation with unpredictable latency and notably lower throughput [2]. To make the matter worse, a broad spectrum of cloud-based applications, e.g., video transcoding in content delivery, and compression/error check in cloud storage, incur both intensive network transmission and realtime computation. This further increases their *self-interference* [3]. In contrast to the well discussed *cross-VM interference* [4] [5] [6], self-interference happens within a VM when the network traffic handling process of the VM is starved, as other computation processes have used up the CPU resources allocated to this VM.

There have been many works focusing on improving the network performance in the cloud [8] [9] [10]. These pioneer studies mainly target specific networked applications or protocols (TCP in particular), and the self-interference has not been well addressed, either. Furthermore, the coexistence of CPU intensive and bandwidth intensive tasks also brings negative impacts on the CPU usage and the energy efficiency of a physical server. For example, CPU intensive tasks may experience longer finish time and the physical server consumes a greater amount of energy, since the ceaseless VM running state change under the scheduling policy, together with the context switching between tasks, increases the job finish time and the energy consumption of the underlying server. Such impact remains unexplored, and a solution is to be developed for general cloud services demanding both transmission and realtime data processing with VMs.

In this paper, we present *MemNet*, a novel architecture that explores a para-virtualized memory sharing mechanism between host and guest VMs. Instead of offloading certain protocol functions, *MemNet* shifts the general I/O intensive operations to the host and decouples I/O and computation workloads for the cloud VMs. The possible task self-interference is therefore minimized, enabling enhanced network performance and shorter job completion time, as well as improving the energy efficiency of the physical server. The design of MemNet is not confined to any specific protocols or applications, and raises the level of provision from physical device interfaces to high-level network services. This eliminates the cost of setting up complex network configurations and hence minimizes the encapsulation overhead on the data exchanging between host and its guest VMs. We have addressed the key design issues in MemNet architecture, including efficient intermediate data sharing and QoS-aware memory usage redistribution. Our prototype-based evaluation shows that MemNet can improve the network throughput by 70% by resolving the self-interference, with a moderate memory usage. Meanwhile, the experiments also indicate that MemNet can reduce the job completion time by 27%. As for the energy efficiency of the physical server, it also exhibits a 32% improvement.

2159-6190/16 $31.00 © 2016 IEEE
DOI 10.1109/CLOUD.2016.147

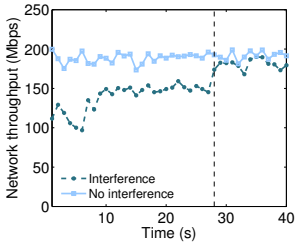

Fig. 1. Network interference

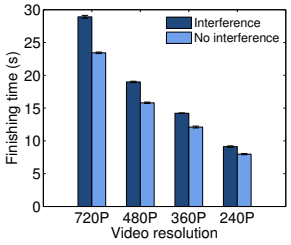

Fig. 2. Transcoding job

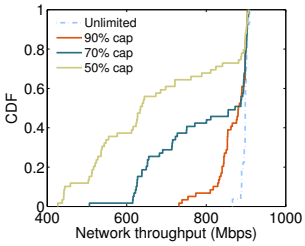

Fig. 3. Scheduling policy

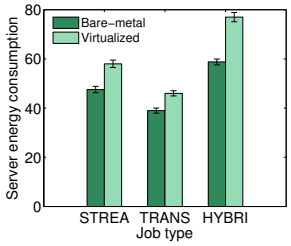

Fig. 4. Energy Comparison

II. Measurement-based Motivation

Previous research has confirmed the existence of self-interference inside a single VM, and quantified the degradation of the network performance with standard benchmark tools [3]. To further understand the self-interference from real-world applications, we have conducted measurements on two sets of typical hybrid workloads: 1) video transcoding and streaming, and 2) file compression and delivery. Despite the great effort towards highly modulized and layered design for modern cloud applications, such hybrid workloads involving both network transmission and realtime computation still widely exist in cloud environments, which can hardly be decoupled.

Our first experiment is conducted on a VM which serves as a video streaming server and also a transcoder with realworld application `LIVE555` and `FFmpeg`. We present our experiment results in Fig. 1. As the experiment begins, the total throughput on this VM is relatively stable around 190 Mbps when it is dedicated to handling the streaming traffic. Unfortunately, when we start to add concurrent transcoding workload to the VM, the total throughput becomes as low as 115 Mbps, not to mention the high variance. Before the transcoding finishes at 28s, the clients experience nearly 40% throughput degradation. On the other hand, for the concurrent transcoding workloads, the processing is also delayed by $10\% - 18\%$ for multiple resolution levels, which is shown in Fig. 2. Another set of experiments with file compression and delivery workloads shows similar impaired performance. To take a further step, we maintain a hard limit on the CPU usage of the VM. Such a *non-work conserving* scheduling policy has often been adopted in realworld commercial clouds, e.g., Amazon AWS, to achieve better isolation between VMs. The CDF (cumulative distribution function) of the VM's network throughput in Fig. 3 shows that, as we limit a lower cap on the CPU usage of the VM, the average throughput significantly decreases, together with a perpetual network instability. Our measurement results clearly indicate that adopting such a scheduling policy further deteriorates the network performance with more severe self-interference. In fact, the root cause of such self-interference is a combined effect of the network architecture design and the scheduling policy in virtualized environments. When hybrid workloads are introduced on the VM, the computation-intensive task can consume the entire quotas demanded by the network traffic, and vice versa. Furthermore, the ceaseless VM running state change, together with the context switching between the CPU intensive tasks and the network intensive tasks inside the VM, can further increase the energy consumption of the underlying server. As a concrete example, Fig. 4 shows that, when we run the stand-alone streaming job or the stand-alone transcoding job inside the VM, the energy consumption of the physical server increases by 22% and 18%, respectively, compared to running the stand-alone task inside the bare metal Linux system. However, when we run the hybrid workloads inside the VM, the energy consumption increases by 31%, also compared to the bare metal case. The results simply indicate that the context switching, brought by the hybrid workloads, consumes more power inside the VM than the bare metal system. Although progress is made on improving the scheduling policy [11] [3] to benefit the VM with heavy I/O workloads, the existence of the hybrid workloads within the VM remains and so is the self-interference. In this paper, we seek the opportunities for raising the provision level from physical device interfaces to high-level networking services. Such approach can be applied for decoupling the hybrid workloads and largely reducing the self-interference in cloud environments.

III. Design and Prototype Implementation

In this section, we present the design of our enhanced virtualization framework. To mitigate the self-interference, the principle of our system is to enable a communication expressway by explicitly providing a lightweight paravirtualized memory sharing interface. As shown in Fig. 5, this design consists of the following components:

- **Processing Module**: The processing module performs CPU intensive jobs inside the guest VM, with a complete isolation against networked communications.
- **Transmission Module**: The transmission module performs network intensive jobs, running in the host space, so as to eliminate the networked transmission of the large bulk data in between the VM and its host space.
- **Shared RAM Space**: The shared memory space stores the temporary intermediate data. It is in the virtual memory maintained by the host kernel.
- **Controller**: The controller is to coordinate the computation progress and the network transmission speed.
- **Profiler**: The profiler is to collect application performance indicators from the processing module, e.g., job response time and detailed logs.

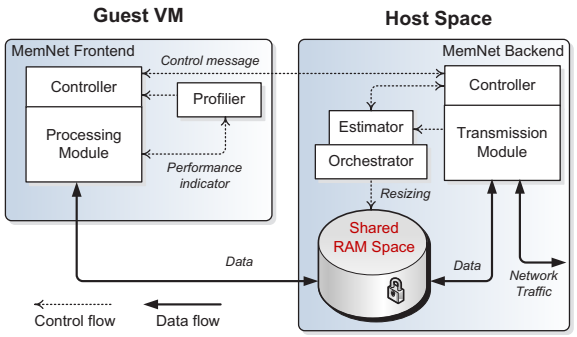

Fig. 5. MemNet architecture

- **Estimator**: The estimator collects information from the continuously-updated performance profiler. Based on the information, it also makes optimization decisions.
- **Orchestrator**: The orchestrator implements and executes estimator's optimization decisions in the system, i.e., the shared RAM space assignments of each VM. This is executed by the memory resizing mechanisms.

As presented in Fig. 5, our framework provides a paravirtualized memory sharing interface to allow the guest VM directly operate on the intermediate data in the shared RAM space. While the shared memory design is not a brand new idea to improve OS efficiency [13], our system takes the first step towards the utilization of this classic approach to decouple the self-interference in cloud VMs. In particular, we utilize the page cache space from operating systems. This space is in the virtual memory and maintained by the host kernel. Such a deployment allows VMs to access data via direct memory-to-memory copy without expensive I/O requests. To achieve better efficiency, our design also makes use of existing kernel interfaces, which further minimizes the impact on memory resource management. Moreover, instead of having a fixed amount of memory space for the exclusive use, this architecture also provides adaptive memory allocation and control to handle workload dynamics.

The design of such shared memory space is undoubtedly the key to enable efficient data transfers between a host and its guest VMs. Note that the rigid management of shared memory space, however, suffers from the inflexibility in the presence of highly skewed memory intensive workloads across co-located VMs. Therefore, the static management in virtualized cloud may result in either resource waste or VM performance degradation. Although almost all modern hypervisors implement memory overcommitment mechanisms such as ballooning, page sharing, and host swapping; they lack policies to coordinate these mechanisms in order to minimize performance degradation as memory space utilized by MemNet can be redistributed across VMs. Therein, a critical design issue in MemNet is then to smartly assign and adjust the shared memory space among different VMs. In the cloud context, the hybrid job types and the arrival rate on each VM are highly dynamic and hence is the efficiency

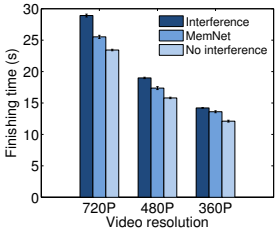

Fig. 6. Video Trancoding

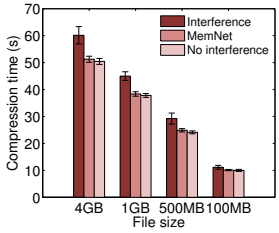

Fig. 7. File compression

of the MemNet space usage. As such, we propose an *online self-adaptive control scheme* to efficiently adjust the size of MemNet space and meet users' SLOs.

In our MemNet architecture, we combine queueing theory modeling and self-tuning of adaptive control together to provide a robust regulation scheme. The reason why we need such a design is twofold. First, we learn from a large cloud provider-Google's trace analysis [15] that typical job inter-arrival time actually exhibits an exponential distribution. Meanwhile, queueing models are also widely applied in the cloud context to provide simplification on the system that has a bottleneck stage [7] [16]. Second, the adaptive feedback loop can build the residual error model and the controller, based on realtime measurements of the system. It can reduce inaccuracies in the queueing model and handle sudden changes of VM workloads in a dynamic fashion. Together with the queueing model estimator, the proposed queueing model-integrated self-adaptive control scheme provides a better performance regulation under a wide range of workloads and conditions. We included further details of our MemNet design in our extended report [14].

We implemented a prototype of *MemNet* in real-world KVM environments. Similar to an in-memory file system, our implementation provides highly dynamic shared memory space between hosts and VMs. This space can be quickly mounted and lifted when needed. To efficiently create a paravirtualized interface on top of the RAM space, our implementation leverages the 9p-virtio (*Plan 9 folder sharing over VirtIO*) framework to provide a direct memory sharing on top of the native host and guest I/O transport [12]. The shared RAM space is ported as a local file system on the guest VM. Note that VirtIO PCI transport allows the memory space shared in such a way that both guest and host driven I/O operations can be zero-copy. This greatly improves the efficiency and makes our approach outperform standard network file systems.

IV. Performance Evaluation

Fig. 6 shows the job completion time when we ran transcoding workloads on MemNet. As we have discussed in Section II, when the transcoding workloads and the streaming workloads are overlapped (labeled as "Interference" in the figure), their interference greatly slow down the job completion on the cloud VMs. Based on the results in Fig 6, we can see that MemNet can successfully decouple such interference and greatly accelerate the processing of transcoding. To further

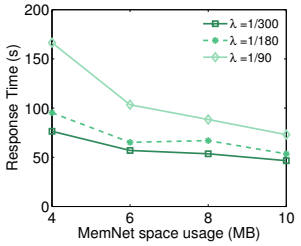

Fig. 8. Memory Usage

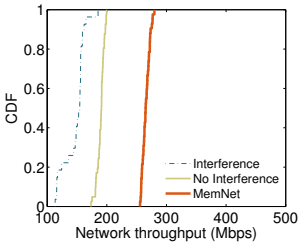

Fig. 9. Network Throughput

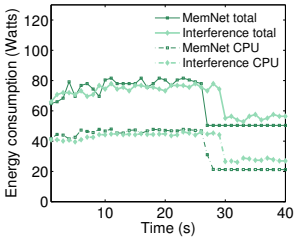

Fig. 10. Streaming and transcoding

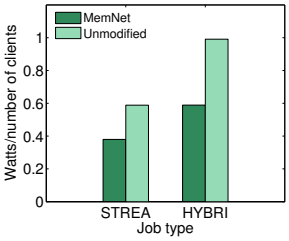

Fig. 11. Energy efficiency

extend our experiment to other applications, we also examined the interference between file compression and file delivery workloads. As we can see in Fig. 7, The proposed MemNet design achieves similar job completion time with the "No Interference" case. To demonstrate the effectiveness of the memory usage control scheme, we provide a memory usage comparison with different job arrival rates λ in Fig. 8. This figure shows that with the online memory usage control scheme, MemNet can operate with a moderate memory usage under dynamic workload stress. As for the networking performance, in Fig. 9 we present the CDF of the streaming throughput when the streaming traffic is colocated with the transcoding job. In this figure, "No interference" means a stand-alone streaming service inside the VM. By comparing the results between "Interference" and "No Interference", we can observe a 40% throughput degradation. When we enabled our MemNet architecture, the average throughput reached up to 260 Mbps, which is a 70% improvement compared with the "Interference" scenario. The throughput of MemNet even has a better performance than the "No interference" case, and this is because in this case the streaming module has a near bare-metal performance when it runs at the host space.

We also pinpoint the energy saving achieved by MemNet design. We closely measured the energy consumption while running the hybrid workloads. As shown in Fig. 10, after we initialized the transcoding job, the energy consumption of the "MemNet" case is only slightly higher than the "Interference" case, despite that the MemNet system was sending faster by 110 Mbps and has a better processing performance. In the figure, the transcoding job is completed 26s for the "MemNet" case, shorter than 29s for the "Interference" case. To make it more clear, we calculated *energy consumed per client* for both the streaming only case and the hybrid workload case to evaluate the actual energy efficiency. Here we assume that each client requires 2 Mbps bandwidth and divide the total energy consumption by the maximum number of clients. The results are shown in Fig. 11. In this figure, "STREA" denotes the streaming only case and "HYBRI" denotes the streaming and transcoding case. We can see MemNet can provide approximately 32% improvement in terms of total energy efficiency when handling the hybrid workloads.

V. Conclusion

In this paper, we closely examined the self-interference from real-world applications in virtualized environments. To jointly optimize the hybrid job performance and the energy efficiency of the physical server, we designed and developed *MemNet*, a novel protocol-independent solution that leverages the para-virtualized memory-sharing mechanism to directly share intermediate data between a host and its guest VM with minimum encapsulation overhead. We implemented a prototype of MemNet based on KVM and evaluated MemNet through realworld workloads, which indicates that such a design can largely improve the network throughput and accelerate the processing of computational jobs in the presence of the self-interference, as well as enhance the energy efficiency.

Acknowledgements

This publication was made possible by NPRP grant #[8-519-1-108] from the Qatar National Research Fund (a member of Qatar Foundation). The findings achieved herein are solely the responsibility of the authors.

References

[1] Younge A J, Henschel R, Brown J T, et al. "Analysis of virtualization technologies for high performance computing environments." in *Proc. of IEEE CLOUD*, 2011.
[2] Wang G, Ng T S E. "The impact of virtualization on network performance of Amazon EC2 data center." in *Proc. of IEEE INFOCOM*, 2010.
[3] Shea R, Wang F, Wang H, et al. "A deep investigation into network performance in virtual machine based cloud environments." in *Proc. of IEEE INFOCOM*, 2014.
[4] Nathuji R, Kansal A, Ghaffarkhah A. "Q-clouds: managing performance interference effects for QoS-aware clouds." in *Proc. of ACM Eurosys*, 2011.
[5] Ayodele A O, Rao J, Boult T E. "Performance Measurement and Interference Profiling in Multi-tenant Clouds." in *Proc. of IEEE CLOUD*, 2015.
[6] Maji A K, Mitra S, Bagchi S. "Ice: An integrated configuration engine for interference mitigation in cloud services." in *Proc. of IEEE ICAC*, 2015.
[7] Bruneo D, Lhoas A, Longo F, et al. "Modeling and Evaluation of Energy Policies in Green Clouds." *IEEE TPDS*, 2015, 26(11): 3052-3065.
[8] Gamage S, Xu C, Kompella R R, et al. "vPipe: Piped I/O Offloading for Efficient Data Movement in Virtualized Clouds." in *Proc. of ACM SoCC*, 2014.
[9] Gamage S, Kangarlou A, Kompella R R, et al. "Opportunistic Flooding to Improve TCP Transmit Performance in Virtualized Clouds." in *Proc. of ACM SoCC*, 2011.
[10] Gamage S, Kompella R R, Xu D, et al. "Protocol Responsibility Offloading to Improve TCP Throughput in Virtualized Environments." *ACM Transactions on Computer Systems (TOCS)*, 2013, 31(3): 7.
[11] Xu C, Gamage S, et al. "vSlicer: latency-aware virtual machine scheduling via differentiated-frequency CPU slicing." in *Proc. of ACM HPDC*, 2012.
[12] Jujjuri V, Van Hensbergen E, Liguori A, et al. "VirtFS-A virtualization aware File System passthrough." in *Proc. of Ottawa Linux Symposium (OLS)*, 2010.
[13] Hwang J, Ramakrishnan K K, Wood T. "NetVM: high performance and flexible networking using virtualization on commodity platforms." in *Proc. of USENIX NSDI*, 2014.
[14] Xu C, Ma X, Shea R, Wang H, Liu J. "Enhancing Throughput and Energy Efficiency for Hybrid Workloads through Para-virtualized Memory Sharing." *Technical Report*, 2016.
[15] Reiss C, Tumanov A, Ganger G R, et al. "Towards understanding heterogeneous clouds at scale: Google trace analysis." *Intel Science and Technology Center for Cloud Computing*, Tech. Rep, 2012: 84.
[16] Xu F, Liu F, Jin H, et al. "Managing performance overhead of virtual machines in cloud computing: A survey, state of the art, and future directions." *Proceedings of the IEEE*, 2014, 102(1): 11-31.