Energy Efficiency of Cloud Virtual Machines: From Traffic Pattern and CPU Affinity Perspectives

Chi Xu, *Student Member, IEEE,* Ziyang Zhao, *Student Member, IEEE,* Haiyang Wang, *Member, IEEE,* Ryan Shea, *Student Member, IEEE,* and Jiangchuan Liu, *Senior Member, IEEE*

***Abstract*—Networking and machine virtualization play critical roles in the success of modern cloud computing. The energy consumption of physical machines has been carefully examined in the past, including the impact from network traffic. When it comes to *virtual machines* (VMs) in the cloud datacenters, the interplay between energy consumption and network traffic however becomes much more complicated. Through real world measurement on both Xen- and KVM-based platforms, we show that these state-of-the-art virtualization designs noticeably increase the demand of CPU resources when handling network transactions, generating excessive interrupt requests with ceaseless context switching, which in turn increases energy consumption. Even when a physical machine is in an idle state, its VM's network transactions will incur non-trivial energy consumption. More interestingly, the energy consumption varies significantly with *traffic allocation strategies* and *virtual CPU affinity conditions*, which was not seen in conventional physical machines. Looking closely into the virtualization architectures, we then pinpoint the root causes and examine that our measurement results can be extended for various network configurations. Moreover, we also provide initial solutions toward optimizing energy consumption in virtualized environments.**

***Index Terms*—Cloud computing, virtualization, networking, energy consumption, measurement.**

I. Introduction

Cloud computing is emerging as a promising paradigm that enables on-demand and elastic access to computing infrastructures. The rapid adoption of the cloud computing has made modern datacenters grow at a fast pace to support a wide spectrum of cloud systems. Unfortunately, the explosive expansion of large-scale datacenters greatly aggravates the power consumption. This has unavoidably restricted the sustainable growth of cloud services and posed unprecedented pressure to the cloud service providers. According to the U.S. Environment Protection Agency (EPA) congressional report on datacenter infrastructures [1], in year 2008 alone, datacenters consumed 61 billion kilowatt-hours (kWhs), representing approximately $4.5 billion cost and accounting for 1.5% of the total U.S. electricity use. A recent report [2] further indicates that such cloud-based enterprise users as Facebook used 153 million kWh of power in 2012 to host their main datacenter in Prineville, which is almost twice what they used in 2011.

The advances in modern inter-networking and machine virtualization technologies play critical roles in the success of cloud computing. Together they supply virtually unlimited resources from cloud datacenters for massive geo-distributed clients, as if each being in its dedicated and isolated space. The energy consumption of physical machines has been carefully examined in the past, including the impact from network traffic [1] [3] [4] [5] [6]. When it comes to *virtual machines* (VMs) in the cloud, the interplay between energy consumption and network traffic however becomes much more complicated. In particular, the inter-VM traffic can reside in different physical machines with their respective *network interface cards* (NICs), or share the same physical machine [7]. When multiple VMs share one physical NIC (pNIC), their traffic can interfere with each other, causing extra overhead [8]. To make the matter worse, the virtual machines can also dynamically migrate across physical machines, thereby changing the traffic pattern [9] [10]. It remains largely unexplored if such dynamic traffic will eventually affect the energy consumption in virtualized environments.

In this paper, we take our first step to understand the interplay between energy consumption and network traffic in representative virtualized environments through a measurement study. Conducted on the realworld platforms with synthetic workloads generated by the commercial software, our study reveals a series of unique energy consumption characteristics of the network traffic in this context. The key observations are as follows:

- State-of-the-art virtualization designs such as Xen and KVM noticeably increase the energy consumption when handling network transactions. This is mainly because the network packets will traverse through multiple layers in virtualized environments.
- Besides the traffic amount itself, energy consumption is proportional to the number of active VMs. Allowing more virtual machines simultaneously sending and receiving network traffic generates excessive interrupt requests, leading to ceaselessly context switching, which in turn increases energy consumption. Even when a physical machine is in an idle state, e.g., not involved in CPU-intensive jobs, its VM's network transactions still incur non-trivial energy consumption.
- More interestingly, even with identical number of active VMs and total amount of traffic on one specific physical machine, the energy consumption can vary significantly with different traffic allocation strategies and virtual CPU affinity conditions.
- Network parameters such as bridging schemes cause negligible energy consumption variations on delivering network traffic in virtualized environments.

The rest of this paper is organized as follows: In Section II, we investigate the state-of-the-art virtualization technologies, and use such classic environments as Xen and KVM to

explain how these virtualized systems are implemented to handle network traffic. To further understand their energy consumption issues, we conducted a comprehensive measurement study on the real world systems. Section III discusses the detailed configurations of our experiment design. Based on the measurement results, Section IV reveals the relationship between energy consumption and different network traffic patterns in the virtualized environments. Our follow-up investigations in Section V further indicate that some other factors as *virtual CPU affinity* can also affect VM's energy consumption. In Section VI, we provide further discussion, including the validation of our observations and the opportunities for further optimization on energy consumption. After that, Section VII summarizes the related works, and Section VIII concludes this paper.

II. Background on virtualization

In this section, we closely investigate popular virtualization solutions and take Xen and KVM as two examples to illustrate how these platforms handle network traffic.

A. Virtualization technologies

State-of-the-art virtualization solutions can be broadly divided into three categories, *Paravirtualization (PVM)*, *Hardware-assisted Virtualization (HVM)*, and *Container Virtualization* [11] [12].

PVM and HVM share the common feature that they generally introduce a *hypervisor* between multiple VMs and the underlying hardware to provide abstractions of physical resources. This design allows the VMs to share devices and ensures security as well as performance isolation. To name a few mature products in market, Xen is one representative of PVM, and KVM is one widely-known project of HVM. The difference is that PVM does not require virtualization extensions from the host CPU [13]; it only requires a specialized kernel that is ported to run natively above a hypervisor. In the PVM design, the guests are aware of the hypervisor and can run efficiently without emulation or virtual emulated hardware [14]. HVM, instead, allows guest operating system to run on a hypervisor without modification [15] [16]. An HVM hypervisor requires CPU virtualization extensions from the host CPU (e.g., Intel VT, AMD-V). The hypervisor traps and virtualizes the execution of certain sensitive, non-virtualizable instructions.

Container Virtualization [17], also known as operating system virtualization, is a light weight virtualization solution which creates multiple secure containers hosting different applications. The containers provide isolation in between different applications and guarantee the scalability and performance. Typical examples of container virtualization project include OpenVZ and Linux-VServer [18] [19]. It is known that container-based solutions incur minimal interference between two containers [20], and the extra energy overhead is almost negligible [21], which differs from that in PVM and HVM as we will show in this paper. They however lack flexibility in terms of OS selection. The users are usually limited to run a single operating system with multiple containers, e.g., users cannot run Linux and Windows together on the same physical machine. Therefore, container virtualization solutions are rarely adopted by large commercial cloud data centers.

Virtualization technologies are the building foundations of modern cloud computing. Amazon AWS and Rack-space Cloud, two major cloud platforms, are both based on customized Xen hypervisors; KVM hypervisors have been used in the Eucalyptus Cloud Service and the Ubuntu Enterprise Cloud, another two representative cloud platforms. Therefore, we focus on Xen and KVM in this paper to evaluate their energy consumption with different network traffic patterns.

B. Case study: Network architecture in Xen

A closer look into the network architecture in Xen is given in Fig. 1, which lists the key steps involved in delivering packets to a Xen-based VM. In this figure, Domain-0 is the initial domain started by the Xen hypervisor up on boot. It runs the Xen management toolstack, and has special privileges. Domain-U refers to a set of unprivileged VMs, being provided to tenants. In Domain-U, the *netfront module* is designed to manage the network traffic from/to Domain-0. This module and its counterpart, *netback module* (in Domain-0), are a pair of inter-linked drivers, bridging the network communications between different domains. For example, upon recipient of a packet in Xen, the physical NIC will deliver the packet to the physical device driver in Domain-0 [22]. Once the packet arrives at the bridge though the protocol stack, it will be transferred through the netback module to the netfront module. In particular, the netback module will allocate resources to process the packet and notify the netfront module in Domain-U. Finally, the netfront module receives the packet and passes it to the guest's network layer.

C. Case study: Network architecture in KVM

A similar network architecture is adopted in the KVM environment, despite that, the Linux kernel takes most of the responsibilities of Domain-0 in Xen. The hypervisor runs in the kernel space and provides the core virtualization infrastructure. To reveal the details, Fig. 2 presents a state-of-the-art architecture of KVM, where the *vhost* driver [23] in Linux provides in-kernel virtual I/O device emulation. This architecture puts virtual I/O emulation codes into the Linux kernel, which enables the device emulation codes to directly call into kernel subsystems instead of performing system calls from the user space. Note that the vhost architecture does not emulate a complete virtual I/O PCI adapter. Instead, it restricts itself to virtual queue operations only. QEMU [11] is a generic and open source machine emulator and virtualizer, and can be used to help perform virtual I/O feature negotiation. This means a vhost driver is not a self-contained virtual I/O device implementation, but depends on the user space to handle the control plane, and the data plane is done in the kernel space. The vhost working thread waits for the virtual queue dumping and then handles buffers that have been placed in the virtual queue. The vhost

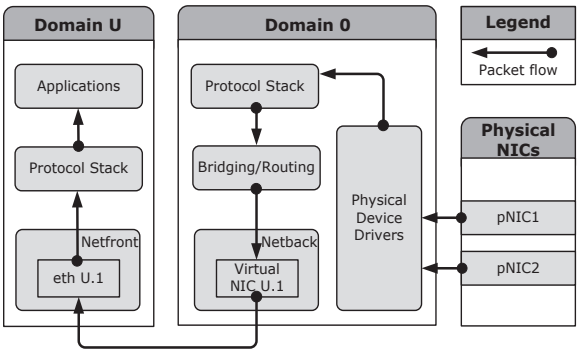


Fig. 1: Xen Network Architecture

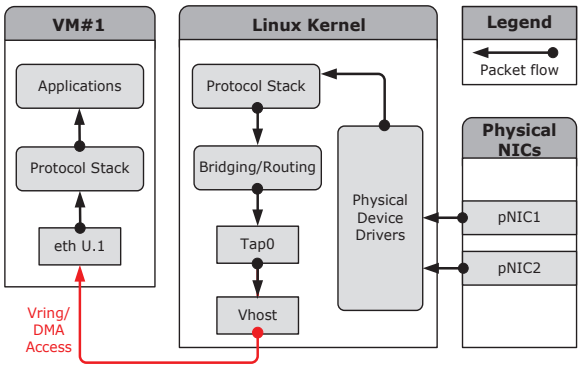


Fig. 2: KVM Network Architecture

architecture however is not tied to KVM in any way. It is a user space interface and has no dependency on the built-in KVM kernel module.

From the design of the vhost networking architecture, it is clear that the network packets have to traverse through multiple layers, and thus the extra overhead is inevitable. Although much effort has been put on reducing latency and improving latency and greater network throughput in virtualized environments, the trade-off between network performance and efficiency remains an open problem to answer. To our best knowledge, there is no perfect solution to achieve the best of the both. In our following experiments, we will evaluate such overhead in both Xen and KVM environments from the angle of energy consumption.

III. Methodology

To reveal the interplay between network traffic and power consumption in Xen and KVM, it is required to monitor the power usage of the physical datacenters. Unfortunately, we can hardly perform such experiments on the public clouds because the cloud providers do not reveal such system-level information to the general public. In this context, we configured multiple VMs on our local cloud testbed, and the detailed measurement setup is presented as follows.

A. Measurement Platform

Our testbed includes a typical midrange server equipped with an Intel's core i5 2400 3.09GHz quad core CPU, 8GB 1333MHz DDR3 RAM. We have two Broadcom network interface cards attached to the PCI-E bus, each of which has a maximum throughput of 1000Mbit/sec. The reason why we choose Intel's i5 as the CPU is that the Intel's x86 architecture is dominating the CPU market for a long period of time, and most major cloud computing providers like Amazon base their virtual machine implementation on the x86 architecture. The core i5 also well reflects Intel's effort toward energy-efficient CPU design [24]. Since we focus on the relationship between network traffic and power consumption, we have configured a second machine to work as the other end of the network traffic. These two machines are directly connected by a 1000Mbit/sec Linksys SD2005 SOHO switch.

We configured the Xen 4.3 Paravitualization hypervisor on our testbed machine. We set the number of accessible virtual CPUs to be two and the amount of RAM to be 2048MB for each virtual machine. Also, We compiled KVM version 1.2.0 from the official Debian source repository and deployed on the testbed machine. Similarly, each virtual machine was given full access to both of the two virtual processor cores as well as the 2048MB of the total memory. The disk interface was configured as a flat file on the physical host's file system.

B. Measurement Tools

Energy-efficiency has been an important consideration in the new generation of CPU design, and Intel has introduced the Running Average Power Limit (RAPL) hardware counters [25] in the Sandy Bridge line of processors. These accurate and versatile hardware counters allow users to extract the record of their CPU power consumption. We measured the power consumption using RAPL counters when we ran all the network-related experiments.

Unfortunately, the RAPL counters only record the power consumption of CPU, and thus we need other measurement tools to evaluate the overall power consumption of the physical machine. In general, there are modules other than CPU that consume considerable amount of power, namely, cooling fans, hard drivers, network interface cards. To measure the power spent on these parts, we wired a digital multi-meter (Mastech MAS-345) into the AC input power line of our machine. We obtained the data by collecting samples every second throughout our experiment from the power meter.

To examine the interplay between network traffic and power consumption, we used the benchmark tool *Iperf* and real world application to generate the network traffic. Iperf is a widely used configurable network benchmark, which allows users to generate network traffic and then gauge the performance of network flows, in terms of throughput, round trip time (RTT) and jitter. Also we can use *Iperf* to tune the traffic load on different VMs. We further set up Apache web servers and used the Apache benchmark suite to generate the downloading traffic on VMs. To provide further resource information, we captured the virtual CPU utilization in Xen using *Xentop*, which is a standard resource monitoring tool integrated in the Xen distribution. In KVM, we used the Linux hardware performance analysis tool *Perf* to collect system level statistics, such as CPU cycles consumed and context switching information, which can reveal more details on how

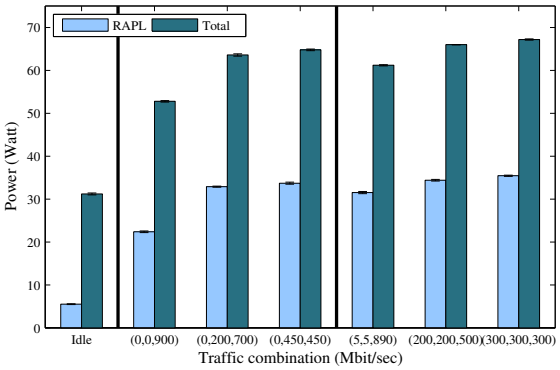


Fig. 3: Xen sending scenario

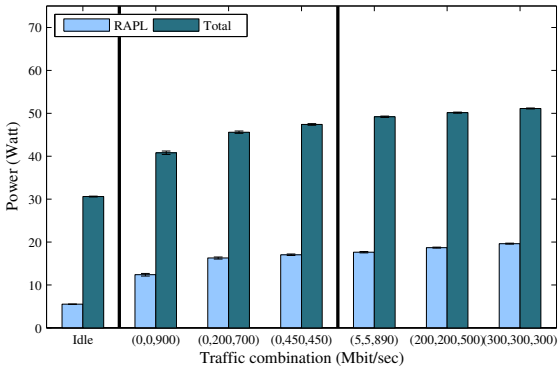


Fig. 4: KVM sending scenario

TABLE I: Xen CPU Profiling

VM #1	VM #2	VM #3	Dm-0 CPU Util (%)	Dm-U CPU Util (%)
0	0	900	55	51
0	200	700	81	73
300	300	700	95	94

TABLE II: KVM CPU Profiling

VM #1	VM #2	VM #3	CPU cycles	Context switching
0	0	900	1.46G	57.80K
0	200	700	2.48G	60.71K
300	300	700	3.25G	103.73K

much effort for CPU to deliver the network traffic. Both of our CPU benchmarks are set to use the lowest Linux scheduler priority by using the *NICE* command. The command ensures that the network task Iperf will be given priority to run on the virtual CPU. By doing this, we ensure that inside the VM, the benchmark process does not take CPU cycles needed to process our network traffic. To avoid randomness in our data, we ran each experiment five times and calculated the average and their standard deviation. The standard deviation is shown in final results as error bars in our figures.

IV. Energy consumption from network traffic

In this section, we will clarify the extra energy overheads in the virtualized environments and further explore the possible relationship between energy consumption and traffic patterns.

A. Extra energy overhead

We first set up three VMs on the physical machine in both Xen and KVM environments. To quantify the extra energy overhead caused by network traffic, we fix the cap of total traffic amount on this physical machine to be 900 Mbit/sec, which is corresponding to the maximum capability of a single NIC. The fixed cap ensures that the energy variation observed is not due to the change of total traffic amount, e.g., generating 800 Mbit/sec traffic surely consumes more power than 100 Mbit/sec traffic. After that, by tuning the traffic amount on different VMs, we use both RAPL and the AC power meter to collect the energy consumption readings of the physical machine. In this experiment, we define the traffic combination as a *3-tuple* across these three VMs. For example, $(100, 100, 700)$ denotes the three co-located VMs sending or receiving at 100 Mbit/sec, 100 Mbit/sec, 700 Mbit/sec, respectively. Value 0 means no traffic is assigned to this VM.

Observation 1: State-of-the-art virtualization designs such as Xen and KVM noticeably increase the energy consumption when handling network transactions.

As we can see from Fig. 3 and Fig. 4, when the physical machine is in an idle state, the RAPL reading is 5.50W in the Xen environment and 5.14W in the KVM environment. Meanwhile, the power meter captures that the effective current is 0.261A and 0.252A in the Xen and the KVM environments, respectively. The effective powers are then 31.3W and 30.24W, respectively. When we allocate 900 Mbit/sec traffic on the physical machine, we observe a significant increase (17.3W in Xen and 7.2W in KVM) on the CPU energy consumption. Meanwhile, by comparing the difference between the RAPL readings and the total energy consumption, we find that the energy consumption on other components, such as RAM, disk, NIC, and cooling fans, are relatively stable with negligible variations.

The main reason behind the extra overhead is that the design of hypervisor introduces extra layers between the physical devices and the destination applications. The detailed CPU profiling information is shown in Table I. It is easy to see that the physical NIC generates excessive interrupt requests to handling the network traffic. At the same time, the hypervisor also needs to schedule Domain-0 and Domain-U to run and share the physical cores. All these operations greatly increase the CPU utilization in both Domain 0 and Domain-U, consuming non-trivial energy. We can observe that the CPU utilization is 55% in Domain-0 and 51% in Domain-U when dealing with 900 Mbit/sec network traffic. We show the corresponding CPU cycles and context switching information in Table II when performing the same experiments in the KVM environment and similar observations can also be made.

Observation 2: Total number of active VMs affects the energy consumption in virtualized environments.

In both figures, we can observe that the power consumption has relatively noticeable spikes from case $(0, 0, 900)$ (one active VM) to case $(0, 200, 700)$ (two active VMs). Similar

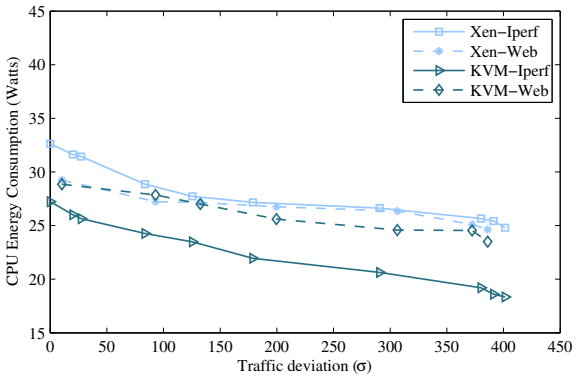


Fig. 5: CPU energy with 5 active VMs

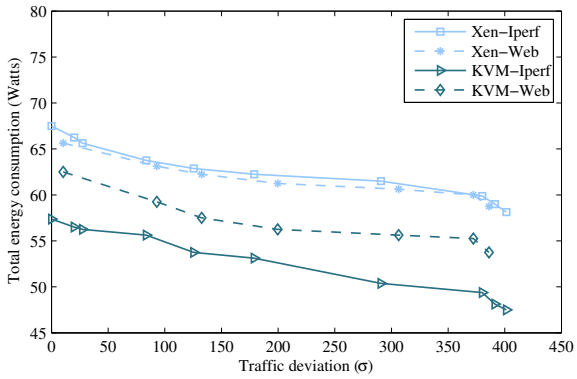


Fig. 6: Total energy with 5 active VMs

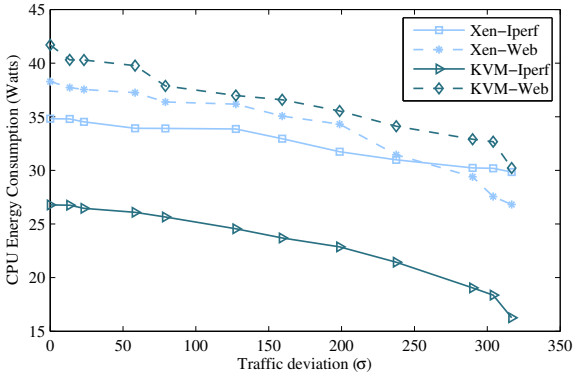


Fig. 7: CPU energy with 8 active VMs

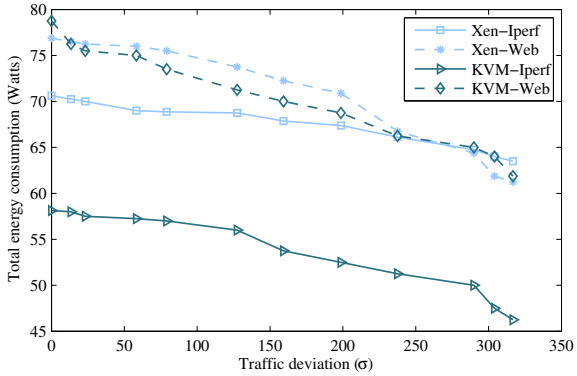


Fig. 8: Total energy with 8 active VMs

spikes also exist when we compare case $(0, 200, 700)$ (two active VMs) and case $(300, 300, 300)$ (three active VMs). For example, in the Xen environment, the RAPL readings in these three cases are 22.4W, 32.9W and 35.4W, respectively. The effective power are 52.9W, 63.84W, and 67.32W, respectively in these three cases. In Fig. 4, the same observation can also be made in the KVM environment. This result is relatively intuitive because the extra VMs will introduce more network interrupt requests and incur even more context switching. As we can see from Table I and Table II, an extra VM will approximately consume 1G CPU cycles in one second and largely increase the CPU utilization in both Domain-0 and Domain-U. This will unavoidably lead to higher energy consumption. It is also worth noting that the energy consumption can still be quite different with identical numbers of active VMs in Fig. 3 and Fig. 4. In detail, the energy consumption varies by $11.3\% - 12.6\%$ when there are three active VMs in the system. e.g., case $(5, 5, 850)$, case $(200, 200, 500)$ and case $(300, 300, 300)$. It is thus interesting to see whether different *traffic combinations* also affect the energy consumption.

B. Impacts of Traffic combinations

To understand the relationship between energy consumption and different traffic combinations, we extended our experiments and enable 5 and 8 active VMs in the system. The traffic combination is therefore a *5-tuple* or *8-tuple* under the new configurations, respectively. For the sake of clarity, we also define *Traffic Deviation*, σ, to capture the skewness of traffic distribution across different VMs.

Definition 1. *Traffic Deviation σ is defined as follows:*

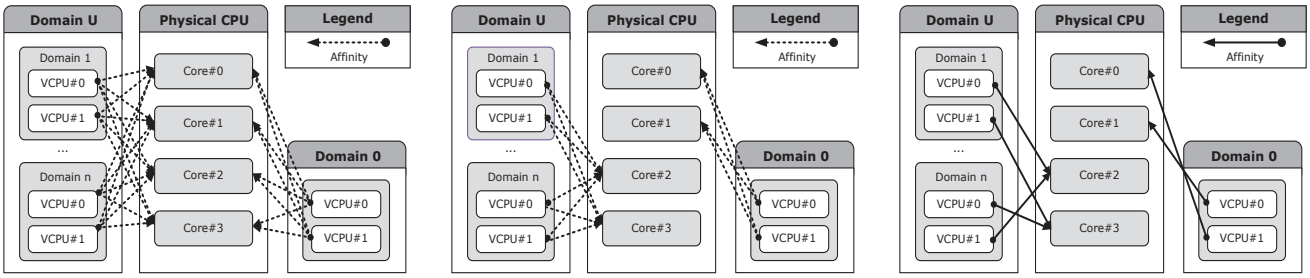
$$\sigma = \sqrt{\frac{1}{n}\sum_{i=1}^{n}(T(i) - \overline{T})^2} \tag{1}$$

where n is the number of the co-located VMs involved in network transactions, $T(i)$ denotes the traffic load on the ith VM. $\overline{T}$ refers to the average traffic load.

Traffic deviation σ can serve as an inverse indicator on the interference in between the traffic on VMs. Note that when we achieve a perfect balance across the VMs, the value of σ will be 0. On the other hand, the increasing skewness will lead to a larger σ. For example, σ can reach 400 when the traffic combination is $(0.5, 0.5, 0.5, 0.5, 898)$.

Observation 3: Even with identical number of active VMs and total amount of traffic on the physical machine, the energy consumption can vary significantly with different traffic allocation strategies.

In Fig. 5 and Fig. 6, we investigate the correlation in between the traffic deviation σ and the energy consumption. We can see that all the curves are monotonically decreasing. This indicates that the load balanced traffic ($\sigma = 0$) will unfortunately introduce very high energy consumption across a fixed number of active VMs. Meanwhile, by carefully adjusting the skewness of traffic distribution, we can achieve up to 31.6% energy saving on each physical machine. This



(a) Random setup (b) Separated setup (c) Pinned setup

Fig. 9: Different CPU affinity conditions

TABLE III: CPU Profiling under Different Traffic Combinations

VM #1	VM #2	VM #3	VM #4	VM #5	Traffic deviation	Xen Dm-0 Util (%)	Xen Dm-U Util (%)	KVM CPU cycles	KVM context switching
180	180	180	180	180	0	83	100	2.71G	221.01K
150	150	200	200	200	27.39	80	95	2.53G	203.19K
100	100	200	200	300	83.67	78.6	94	2.38G	186.7K
50	50	200	300	300	125.50	70.6	90	2.18G	154.73K
50	50	50	50	700	290.69	57	83	1.74G	124.28K
10	10	10	10	860	380.13	48.9	80	1.38G	91.44K
0.5	0.5	0.5	0.5	898	401.37	41	75	1.29G	81.91K

is a significant improvement in the context of datacenter environment.

It is also worth noting that the energy consumption is decreasing very fast when σ is between 0 to 100. The slope will then decrease when σ goes larger than 100. Based on this observation, it will be easy to achieve reasonable energy saving by slightly adjust the traffic assignments (slightly increase σ) in real-world cloud systems. To further verify our conclusion, we also applied such real-world applications as Apache web server, in both Xen and KVM to validate our observations. In this context, We use Apache benchmark suite to actively tuning the number of connections and the traffic load on each VM. The results are also presented in Fig. 5 and 6, which remain consistent with the previous Iperf experiments. We further present the experiment results of 8 active VMs in Fig. 7 and Fig. 8, which show similar results as those in the smaller-scale experiments and hence is the observation: increasing skewness of the traffic leads to energy saving.

Based on the above observations, it is thus reasonable to believe that the observed energy consumption variations in these experiments are due to the cross-VM interference. The design of typical PVM and HVM platforms introduces virtual switches to handle the network traffic [22] [26]. A virtual switch, which is implemented by software, has the same functionality as a physical switch. On one side of the virtual switch are ports that connect to virtual NICs, and on the other side are connections to physical NICs on the servers. When the VM sends out the network traffic, the packets inside the VM are first handled by the VM's kernel process, then a software interrupt will be send to the hypervisor process. The hypervisor process is scheduled to copy the packet from the VM's memory space into the host's memory space. The hypervisor process then notifies the host kernel process to collect the packet and eventually the virtual switch gathers all packets from the virtual NIC's back-end and sends to the physical NICs.

The fully balanced traffic makes the hypervisor to serve VMs more frequently, since the virtual switch need to collect the packets from different VMs more often. Once receiving the software interrupt, the hypervisor process is scheduled to run on one specific CPU core to handle the traffic, which meanwhile causes such interference that the original VM running on this core is blocked and re-scheduled. Such a great number of context switches put extra overload on the CPU. During our experiments, we also noticed that when a certain VM is sending at a lower traffic rate, the virtual switch handles the traffic from that VM with a non-work-conserving policy. Therefore, when the traffic deviation is large, the server benefits from such a design principle, reducing the energy overhead on CPU.

As an additional reference, Table III presents the traffic combination information, the CPU utilization in Xen, as well as the profiling captured in KVM. In the Xen environment, we can see that the increased CPU utilization of Domain-U sum is 25%, while the major increase, a growth of 42% can be witnessed in Domain-0. As we have discussed, Domain-0 controls the physical NIC drivers and the virtual switches, as well as handling the I/O operations. Therefore, the overhead generated in Domain-0 greatly contributes to the total CPU utilization increase and unavoidably leads to higher energy consumption. Similarly in KVM, an extra of 1.42G cycles are consumed by the cross-VM interference. The context switching overhead is almost tripled.

V. Energy Variation from Virtual CPU affinity

To further validate our conjecture of the cross-VM interference, we examined another factor in this section, that is, the *virtual CPU affinity*.

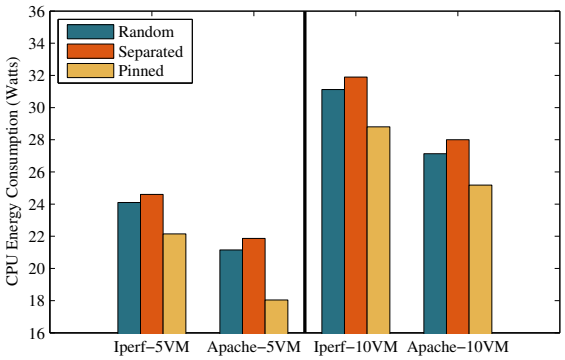


Fig. 10: CPU energy consumption in Xen

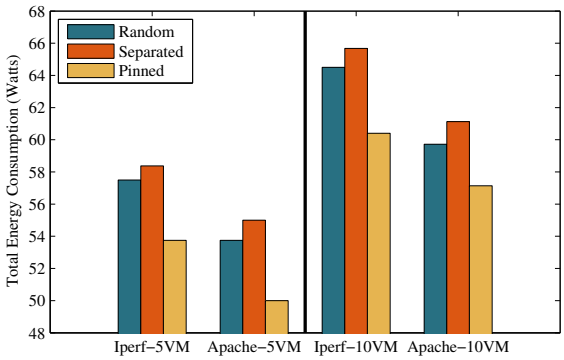


Fig. 11: Total energy consumption in Xen

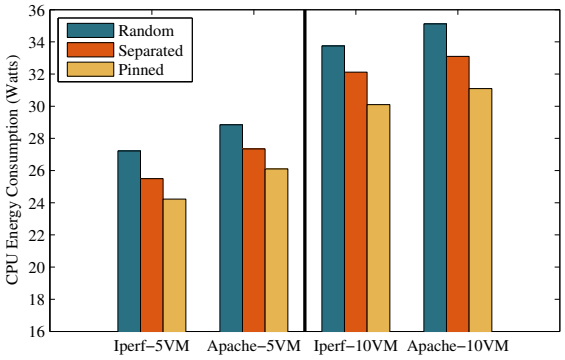


Fig. 12: CPU energy consumption in KVM

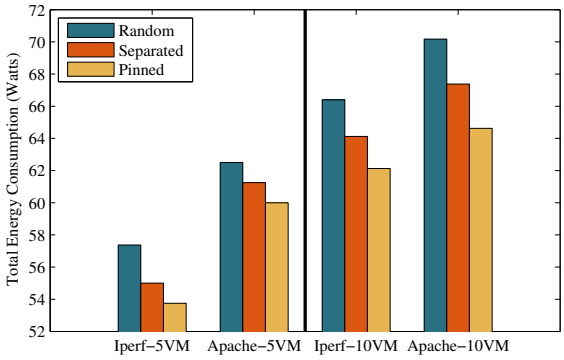


Fig. 13: Total energy consumption in KVM

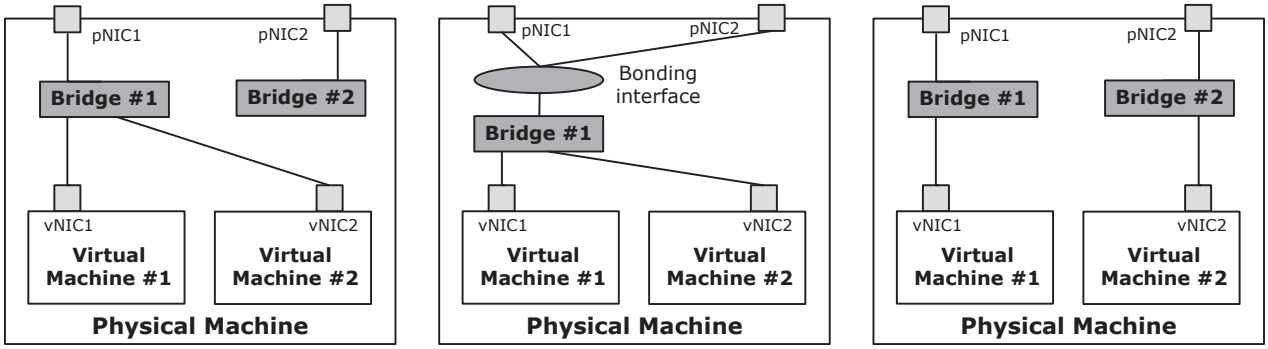
It is known that virtual CPU affinity, or *vCPU pinning* enables the binding and unbinding of a VM to a CPU core or a set of CPU cores, so that the VM will be scheduled to execute only on the designated CPU core or CPU cores rather than an arbitrary CPU core. Each VM waiting to be scheduled in the scheduling queue has a tag indicating its kin CPU. At the time of resource reallocation, each VM is allocated to its kin CPU core in preference to others. Scheduling one VM to execute on the specific CPU core can result in an efficient use of process by reducing performance-degrading situations such as cache misses. As a matter of fact, virtual CPU affinity can also have impact on the energy consumption variation. As a concrete example, in Fig. 9, we provide three common virtual CPU affinity conditions in a machine with 4-core CPU in the Xen environment. Fig. 9 (a) is the default setup adopted in Xen, that is, each virtual CPU can run on any physical CPU cores. Therefore, we define it as the *random setup*. This setup has advantage that it achieves higher utilization, while the problem is this setting can cause a bottleneck that Domain-0 might occasionally have to compete with Domain-U in terms of scheduling. According to previous research [27], Fig. 9 (b) is a setup resembling what Amazon EC2 small instance uses. The Domain-0 can run on two CPU cores exclusively, and other domains will share the remaining two CPU cores. We define it as the *separated setup* in this paper. Unlike the Amazon EC2 configurations, we did not set cap on the vCPU running time. In fact, researchers observe that setting cap will certainly have impairments on networking performance in [7] [28]. To avoid this problem, we only consider pinning the virtual CPU to physical CPU core and allowing unconstrained usage. Note this second configuration simply resolves the contention in between Domain-0 and Domain-U, while the utilization could be lower than the previous configuration. Fig. 9 (c) is a case that each virtual CPU is pinned on one specific core. We define it as the *pinned setup* in this paper. This configuration also guarantees that no contention exists between Domain-0 and Domain-U. To carry on our experiments, we used the built-in command $vcpu\text{-}set$ in Xen to adjust the virtual CPU affinity. Since KVM does not utilize the concept of Domain-0, we adjusted CPU affinity of each VM using the $Taskset$ tools in the KVM environment. Accordingly, hypervisor-related processes can be scheduled on any available cores, and we tested the situation where each VM has only one virtual CPU.

TABLE IV: KVM Profiling

	CPU Cycles	Context Switching
Random	2.71G	221.01K
Separated	2.59G	210.95K
Pinned	2.3G	180.2K

Observation 4: The energy consumption varies with virtual CPU affinity. Pinning virtual CPU can reduce the CPU energy overhead. The pinning strategy can also lead to unbalanced core utilization, causing energy hot spots.

We present our measurement results in the Xen environment in Fig. 10 and Fig. 11. The experiment details are as follows. For the experiments carried out on five VMs, the first test



(a) Sharing virtual bridge (b) Utilizing bonding interface (c) Using two virtual bridges

Fig. 14: Different network configurations

scenario is that we let five colocated VMs send the same amount of traffic (180 Mbit/sec), and the second one is that we generated download tasks on Apache web server, with each VM sending files at 180 Mbit/sec. Meanwhile, we adjusted the underlying virtual CPU affinity to be exactly the same as the three cases shown in Fig. 9. We also carried out the experiments on ten colocated VMs, with each one sending network traffic at 90Mbit/sec. By comparing the energy consumption of pinned setup with others in Fig. 10 and Fig. 11, we can see that the CPU energy consumption can be reduced from 21.87W to 18.04W (shown in the second bar set in Fig. 10), that is, a 17.5% energy reduction. Basically, the pinned setup can always achieve a lower energy overhead since each virtual CPU is constrained to run on specific physical cores, which essentially reduces cache miss ratio and inter-core scheduling.

We also have an interesting observation that separated setup adopted by Amazon [27] increases the energy usage under these scenarios. The key reason is that, two cores (core#0 and core#1) are taken over by Domain-0 in this setup, which leaves the other two cores (core#2 and core#3) shared by the five VMs. In this case, the network traffic consumes large amount of CPU cycles, and the VMs are frequently scheduled in and out. This in turn makes core#2 and core#3 become hot spots. Although there are enough time slices left on core#0 and core#1, those VMs have to compete with each other on core#2 and core#3. We performed similar experiments in the KVM environment using random, separated and pinned setups, but only allocate one virtual CPU for each VM.[1] The experiment results are shown in Fig. 12 and Fig. 13. The consumed CPU cycles and context switching are shown in Table IV. These data in Table IV are captured when the five colocated VMs sending traffic via *Iperf*. In this case, since each VM only has one virtual CPU, and hypervisor related processes can be scheduled on any available cores, we can observe that the energy consumption is monotonically reducing with decreasingly freedom on the virtual CPU scheduling. The implication of our experiments is that cloud providers could adaptively set virtual CPU affinity instead of using static strategies to obtain a better trade-off.

VI. Further Discussion

A. In-depth Validation

Besides the traffic pattern and the CPU affinity, we also examined other factors that have the potential to affect the energy consumption. In this section, we provide the further discussions on different *network bridging schemes*.

It is known that the machines in cloud data centers are often embedded with multiple NICs. For instance, a typical mid-range IBM server is attached with one or two physical NICs, while the high-end servers could have even more NICs, e.g., IBM's BladeCenter E series can have fourteen NICs equipped. Meanwhile, the VMs that are co-located in these data centers can also support multiple virtual NIC configurations. For example, Amazons Virtual Private Cloud (VPC) service allows the cloud users to dynamically attach/detach multiple virtual NICs on their VMs. Fig. 14 shows a basic example of possible relationships between physical and virtual NICs. We can see that the co-located VMs can share one virtual bridge in Fig. 14 (a). Moreover, as shown in Fig. 14 (b), it is also possible to bind two (or more) physical NICs together to achieve link aggregation. Such bonding interface is widely applied to enable fail-over guarantee as well as load balancing for the cloud platforms [29]. Besides, they can also be attached to different virtual bridges, which is shown in Fig. 14 (c).

While different network bridging schemes are seemingly complicating the internal path of traffic, our measurements indicate that it can only cause negligible energy consumption variance. Fig. 15, 16, 17 compare the CPU utilizations when two co-located VMs are configured using different network bridging schemes. In this experiment, the VMs generated exactly same amount of traffic[2]. As we can see in Fig. 15, for Domain-U, the different network configurations hardly have influence on the CPU utilization, with the variation range under 3%. In Fig. 16, however, we find that the bonding interface case increases the CPU utilization by up to 5%

[1]In the KVM environment, random, separated and pinned setup implies that each virtual CPU can be scheduled on 4, 2, 1 physical cores, respectively.

[2]Based on our measurements in Section IV B, this configuration can maximize the traffic interference.

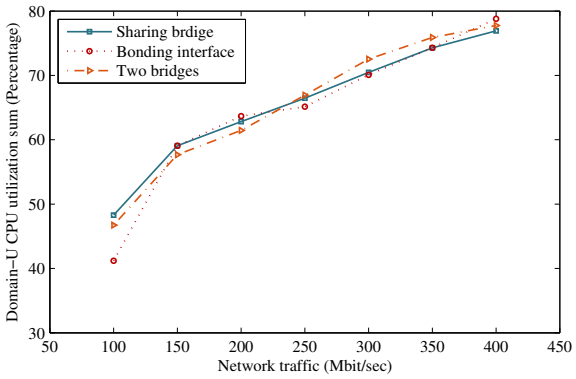


Fig. 15: Domain-U CPU utilization sum

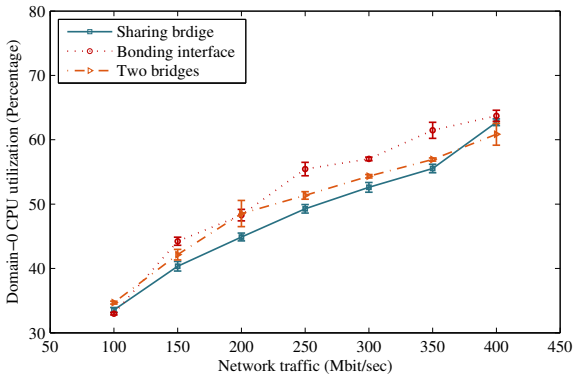


Fig. 16: Domain-0 CPU utilization

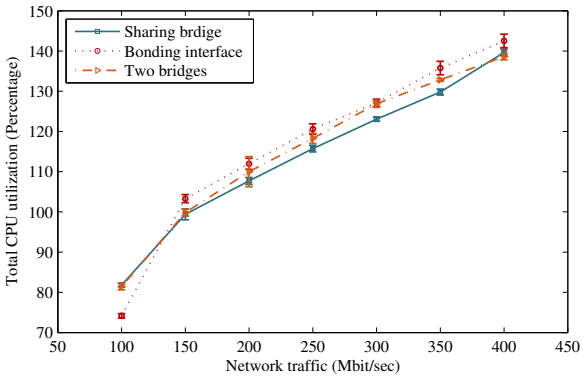


Fig. 17: Total CPU utilization sum

compared with the sharing bridge case, also we notice that when we use two virtual bridges, the CPU utilization also has a slight increase. To examine the overall impact, We present Fig. 17, which shows different bridging schemes have very limited impact on CPU. In particular, it can only introduce up to 8% difference on the CPU utilization, which will only lead to a 0.4W energy consumption difference.

B. Opportunities

Hardware manufacturers have noticed the high overhead for handling network traffic in virtualized environments, and have created devices that greatly alleviate the virtual NIC overhead. Two popular solutions are Virtual Machine Device Queues (VMDQ) and Single Root I/O Virtualization (SR-IOV) [22]. Both of them work by duplicating components of the NIC to give a VM more direct access to the hardware. Note that in the current VMDQ architecture, the software switch and the hypervisor process are still involved in copying packets in between the host's memory space and the VM's memory space. Therefore, the interference observed in this paper still exists, and there remains much room for further improvement. For SR-IOV, the hypervisor only handles interrupts generated by SR-IOV enabled NICs, which will greatly alleviate the extra overhead on CPUs, thereby mitigating the impact that we have observed. Although such sophisticated hardware are still expensive and with scalability and security issues, with advanced hardware technologies in the future, hardware-assisted solutions are promising to achieve better energy efficiency for cloud virtualization.

On the other hand, we can have solutions developed from the perspective of network traffic itself. For instance, based on the detailed log information on the cloud VM, cloud providers could reshape the traffic combination on each physical machine and increase the traffic deviation. A concrete example is to delay certain low-priority traffic (e.g., data sycronization traffic). Although this requires cross-layer design for cloud virtualization, the idea of application-aware is already one of the development trends for many enterprise cloud service providers as IBM [30]. Note we may not need a perfect algorithm to optimally skew the workload. Our measurement indicates that a slight increase on traffic deviation can already achieve reasonable energy savings.

There have been other pioneer studies on mitigating the overhead of network transactions in virtualized clouds by offloading work from the VM space to the host space. An example is vSnoop [31], which offloads the TCP acknowledgement function from the VM to the hypervisor. vPro [32] further offloads the TCP congestion control function to the hypervisor inside the host space. Despite the increased TCP throughput, such offloading approaches migrate only the control plane of the TCP protocol, with the data plane remaining inside the VM. As such, the data transmission still involves complicated cooperation between a VM, its hypervisor and the host kernel. Meanwhile, the modifications to the existing TCP protocol further limit the flexibility to adapt to different application scenarios. There is still much to explore to effectively offload the data plane to the host space.

Based on our observations on virtual CPU affinity, cloud service providers could also adaptively reconsider virtual CPU affinity to obtain a better tradeoff between energy efficiency and utilization of the underlying hardware. Although pinning VM to exclusively run on certain physical core decreases the utilization of the infrastructure, under certain circumstances, it is a feasible solution to reduce the energy overhead by 10-20%. This problem is not trivial and should be carefully investigated through classic resource provisioning and VM migration approaches.

VII. Related work

Nowadays, the rising popularity of cloud-based system deployment has attracted an increasing number of studies to investigate the VM co-location interference. Oh *et al.* [33] illustrated the co-location interference among the cloud VMs. The authors conducted real-world experiments to show the effects of performance interference when VMs with different characteristics are consolidated in a single server. With the consolidation of different workloads, Zhu *et al.* [34] further quantified the co-location interference of two VMs by the correlation of their resource utilization. To avoid the VM co-location interference, some other approaches, such as resources isolation, are widely suggested to reduce the VM-level overhead. For example, the cache and memory bandwidth interference can be alleviated by planning on compensating the impacted VMs for reserved CPU resource [35]. The study from Shieh *et al.* [36] summarized the VM isolation techniques on the network resource, such as installing multiple network adapters, implementing static bandwidth allocation for VMs in the Domain-0, or in network adapters and switches.

There have also been significant studies on the power consumption of virtualized cloud systems. Jin *et al.* [37] measured how virtualization techniques have influence on power consumption. They concluded that high throughput and low energy consumption cannot be achieved at the same time. Huang *et al.* [38] further evaluated the power consumption caused by VM migration. The authors showed that it is better to adopt the strategy of consolidation to achieve a less power overhead. In [39], Kansal *et al.* present a solution for VM power metering, named *Joulemeter*. The authors discussed different power consumption models to infer the power consumption from resource usage at runtime. To address the power consumption issues, Kusic *et al.* [40] implemented and validated a dynamic resource provisioning framework for virtualized server environments. A recently published paper from Shea *et al.* [21] showed that energy consumption caused by network transactions can be reduced by using adaptive packet buffering.

Our work was motivated by these studies; yet we explore the relationship between the energy consumption and different network traffic patterns. Our experiment reveals a series of new factors that will affect the total energy consumption in the virtualized environments, providing the opportunities to enable seamless and light-weight power saving solutions to the cloud computing platforms.

VIII. Conclusion and future work

In this paper, we systematically examine the power consumption brought by network traffic in virtualized cloud environment. We conducted systematic experiments to measure the CPU power usage as well as the power consumption of the whole machine. We revealed a series of detailed factors in virtualized environments that can have impact on energy consumption. The results showed that such state-of the-art virtualization designs noticeably increase the demand of CPU resources when handling network traffic. Besides on the network load itself, the traffic combinations and CPU affinity on the VMs also play key roles in deciding the power consumption. We analyzed the root cause and confirmed that the measurement results can be extended to different network configurations.

It is worth noting that our investigation is not limited to pinpoint the problems in the cloud environment, it also shed new lights to better mitigate the power consumption issues. For example, our cloud service providers can reshape the traffic combination on each physical machine (decrease the traffic deviation σ) to reduce the electric bill. They may also consider to further optimize the CPU affinity or even delaying certain traffic requests to avoid the interference among co-located VMs. Based on these observations, our future work will focus on the detailed design of energy-aware virtual machine enhancement and traffic optimization strategies.

References

[1] Brown R. "Report to congress on server and data center energy efficiency: Public law 109-431." *Lawrence Berkeley National Laboratory,* 2008.

[2] "Facebook sustainability report." https://www.facebook.com/green/app_439663542812831.

[3] Chase J S, Anderson D C, Thakar P N, et al. "Managing energy and server resources in hosting centers." *ACM SIGOPS Operating Systems Review,* 2001, 35(5): 103-116.

[4] Elnozahy M, Kistler M, Rajamony R. "Energy-efficient server clusters." *Power-Aware Computer Systems,* Springer Berlin Heidelberg, 2003: 179-197.

[5] Elnozahy M, Kistler M, Rajamony R. "Energy conservation policies for web servers." in *Proc. of USENIX Symposium on Internet Technologies and Systems-Volume 4,* 2003.

[6] Bianchini R, Rajamony R. "Power and energy management for server systems." *Computer,* 2004, 37(11): 68-76.

[7] Wang G, Ng T S E. "The impact of virtualization on network performance of amazon ec2 data center." in *Proc. of IEEE INFOCOM*, 2010.

[8] Mei Y, Liu L, Pu X, et al. "Performance measurements and analysis of network i/o applications in virtualized cloud." in *Proc. of IEEE CLOUD*, 2010.

[9] Clark C, Fraser K, Hand S, et al. "Live migration of virtual machines." in *Proc. of USENIX NSDI*, 2005.

[10] Zheng K, Wang X, Li L, et al. "Joint power optimization of data center network and servers with correlation analysis." in *Proc. of IEEE INFOCOM*, 2014.

[11] Chiueh S N T, Brook S. "A survey on virtualization technologies." *RPE Report*, 2005: 1-42.

[12] Sahoo J, Mohapatra S, Lath R. "Virtualization: A survey on concepts, taxonomy and associated security issues." in *Proc. of IEEE ICCNT*, 2010.

[13] Barham P, Dragovic B, Fraser K, et al. "Xen and the art of virtualization." *ACM SIGOPS Operating Systems Review*, 2003, 37(5): 164-177.

[14] Whitaker A, Shaw M, Gribble S D. "Scale and performance in the Denali isolation kernel." *ACM SIGOPS Operating Systems Review*, 2002, 36(SI): 195-209.

[15] Adams K, Agesen O. "A comparison of software and hardware techniques for x86 virtualization." *ACM Sigplan Notices*, 2006, 41(11): 2-13.

[16] Fisher-Ogden J. "Hardware support for efficient virtualization." *University of California, San Diego, Tech. Rep*, 2006.

[17] Soltesz S, Pötzl H, Fiuczynski M E, et al. "Container-based operating system virtualization: a scalable, high-performance alternative to hypervisors." *ACM SIGOPS Operating Systems Review*, 2007, 41(3): 275-287.

[18] Che J, Yu Y, Shi C, et al. "A synthetical performance evaluation of openvz, xen and kvm," in *Proc. of IEEE APSCC*, 2010.

[19] des Ligneris B. "Virtualization of linux based computers: the linux-vserver project." in *Proc. of IEEE HPCS*, 2005.

[20] Padala P, Zhu X, Wang Z, et al. "Performance evaluation of virtualization technologies for server consolidation." *HP Labs Tec. Report*, 2007.

[21] Shea R, Wang H, Liu J. "Power consumption of virtual machines with network transactions: Measurement and improvements." in *Proc. of IEEE INFOCOM*,2014.

[22] Shea R, Liu J. "Network interface virtualization: challenges and solutions." *IEEE Network*, 2012, 26(5): 28-34.

[23] "Vhost driver." http://www.linux-kvm.org/page/UsingVhost.

[24] Esmaeilzadeh H, Cao T, Xi Y, et al. "Looking back on the language and hardware revolutions: measured power, performance, and scaling," *ACM SIGARCH Computer Architecture News*, 2011, 39(1): 319-332.

[25] "Running average power limit." https://lwn.net/Articles/545745/.

[26] "Vmware infrastructure architecture overview." http://www.vmware.com/pdf/vi_architecture_wp.pdf.

[27] Xu Y, Musgrave Z, Noble B, et al. "Bobtail: avoiding long tails in the cloud," in *Proc. of USENIX NSDI*, 2013.

[28] Shea R, Wang F, Wang H, Liu J. "A deep investigation into network performance in virtual machine based cloud environments." in *Proc. of IEEE INFOCOM*, 2014.

[29] Hao F, Lakshman T, Mukherjee S, et al. "Secure cloud computing with a virtualized network infrastructure." in *Proc. of USENIX HotCloud*, 2010.

[30] Williams D, Zheng S, Zhang X, et al. "TideWatch: Fingerprinting the Cyclicality of Big Data Workloads." in *Proc. of IEEE INFOCOM*, 2014.

[31] Kangarlou A, Gamage S, Kompella R R, et al. "vSnoop: Improving tcp throughput in virtualized environments via acknowledgement offoad." in *Proc. of ACM/IEEE SC*, 2010.

[32] 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.

[33] Oh F Y K, Kim H S, Eom H, et al. "Enabling consolidation and scaling down to provide power management for cloud computing." in *Proc. of USENIX HotCloud*, 2011.

[34] Zhu Q, Zhu J, Agrawal G. "Power-aware consolidation of scientific workflows in virtualized environments." in *Proc. of ACM/IEEE SC*, 2010.

[35] Nathuji R, Kansal A, Ghaffarkhah A. "Q-clouds: Managing performance interference effects for qos-aware clouds." in *Proc. of ACM Eurosys*, 2010.

[36] Shieh A, Kandula S, Greenberg A, et al. "Seawall: performance isolation for cloud datacenter networks." in *Proc. of USENIX HotCloud*, 2010.

[37] Jin Y, Wen Y, Chen Q. "Energy efficiency and server virtualization in data centers: An empirical investigation." in *Proc. of Computer Communications Workshops (INFOCOM WKSHPS)*, 2012.

[38] Huang Q, Gao F, Wang R, et al. "Power consumption of virtual machine live migration in clouds." in *Proc. of IEEE CMC*, 2011.

[39] Kansal A, Zhao F, Liu J, et al. "Virtual machine power metering and provisioning." in *Proc. of ACM SoCC*, 2010.

[40] Kusic D, Kephart J O, Hanson J E, et al. "Power and performance management of virtualized computing environments via lookahead control." *Cluster computing*, 2009, 12(1): 1-15.