Understanding the Impact of vCPU Scheduling on DVFS-based Power Management in Virtualized Cloud Environment

Ming Liu, Chao Li, and Tao Li
Intelligent Design of Efficient Architectures Laboratory (IDEAL)
Department of Electrical and Computer Engineering, University of Florida
{mingliu, chaol}@ufl.edu, taoli@ece.ufl.edu

Abstract – Virtualized platform has emerged as a prominent environment for cloud computing, especially in today's power-constrained data centers. However, due to a lack of coordination between runtime power management and a virtual CPU (vCPU) scheduler, existing virtualized cloud platform is far from efficient. First, current frequency control mechanism is unable to satisfy the fast-changing vCPU frequency requirement imposed by vCPU scheduler, which we refer to as *demand imbalance* problem. In addition, newly created vCPUs, if scheduled solely based on fairness, can cause inefficient frequency rise and drop on an unmatched physical core, which we refer to as *utilization mismatch* problem. In both cases, the system incurs degraded power efficiency and sub-optimal workload performance.

In this study we perform a comprehensive analysis on the interplay between vCPU scheduling and processor-centric power control in virtualized cloud environment. Using representative workloads from CloudSuite and real server deployment, we examine the energy/performance implications of frequency scaling and vCPU scheduling on both single-VM and multi-VM cloud host. We show that existing virtualized platform has the potential to improve energy efficiency and workload performance by 32% and 25%, respectively, if vCPUs are balanced and appropriately scheduled. We also show that dirty page rate, virtual block device processing rate, virtual network packets arrival rate, and network I/O buffer availability are important efficiency indicators for energy-efficient virtualized cloud system design.

Keywords – virtualization; DVFS; cloud workload; evaluation

I. INTRODUCTION

Virtualized system represents a massive platform change in the cloud computing era, providing energy efficiency and cost savings that outperform bare metal system significantly [1]. It has been shown that nearly 70% of U.S. companies have adopted virtualization for their cloud services since 2012 [2]. Over 60% of the data center operators will consolidate their workloads (encapsulated in virtual machines) if computing demand increases, rather than add additional server resources to existing data center facilities [3].

When scaling out virtual platforms to handle the proliferation of cloud applications, challenge arises due to the power budget constraints in modern data centers, especially in those who are under-provisioned. Conventionally, server consolidation solves this problem by combining workloads from separate machines into a smaller set of systems. As VM density continues to increase in data center, the impact of workload consolidation on power saving diminishes over time. Recognizing this trend, many recent papers start to look at joint optimization of VM scheduling and processor-centric power control (e.g., DVFS) to further reduce energy consumption and free up power capacity [19, 34, 5, 6, 26].

Unfortunately, to date, a thorough analysis of the interplay between virtual CPU (vCPU) scheduling and DVFS-based runtime power management is still lacking. Existing proposals on power-efficient virtualized platforms mainly focuses on three design layers: (i) cluster-level load balancing through workload live migration [19, 34] techniques, (ii) server-level power budgeting using DVFS [5, 6], and (iii) system-level energy management by adjusting the computing resource assigned to each VM [26, 35]. While there have been papers discussing cross-layer power management on virtualized platforms [22, 26, 27], none of the existing proposals have ever explored the performance overhead and energy inefficiency issue of DVFS in the context of vCPU scheduling. In addition, prior studies mainly focus on traditional workloads with little consideration about virtualized cloud applications. Given that emerging cloud VMs have much more diverse communication and interaction behaviors (request partitioning and collecting, massive data caching, data scanning, heartbeat checking, etc.), it is crucial to understand the control effectiveness of DVFS-based power management in virtualized cloud environment.

There are two issues that might result in degraded performance and efficiency when using CPU frequency scaling blindly on a virtualized cloud platform. First, not all vCPUs in the execution queue of a physical core can receive desired frequency assignments. We term this issue as *demand imbalance* problem. Taking Xen credit scheduler for example, the default time slice (credit) for vCPU is 30ms [10] and the minimum sampling interval of frequency adjustment is 10ms [9]. The frequency of a physical core can only be adjusted one level at a time for current governors (e.g., *Ondemand*, *Conservative*). When a high-loading (e.g., 100%) vCPU is scheduled on a physical core that previously runs a low-loading (e.g., 20%) vCPU, the frequency of the physical core can be increased by at most three levels – which is still lower than the frequency target of the high-loading vCPU. Second, a newly inserted vCPU, if mapped to a physical core of far different loading, can often disturb the established stable frequency level and cause inefficient frequency adjustments. We term this problem as *utilization mismatch.* For example, it happens when mapping a latency-sensitive Apache Web server VM to a data store VM, or mapping a media streaming VM to a compute-intensive VM. The problem becomes even worse if the new vCPU obtains higher weights com-

pared to other vCPUs. Conventional credit-based scheduler is designed to achieve fairness and responsiveness, but might impact the power control effectiveness of virtualized cloud workloads adversely.

In this study we examine the efficiency of DVFS-based power management in virtualized cloud environment. Specifically, we investigate the energy/performance implications of dynamic frequency scaling and vCPU scheduling on both single-VM and multi-VMs cloud hosts. In the single-VM scenario, we vary the upper limit of CPU utilization in Xen's credit-based CPU scheduler and estimate its impact on energy efficiency. We explore optimization opportunities with detailed VM execution profiling data such as dirty page number, network activities (send/receive), and VBD (virtual block device) read/write. For multi-VM cloud hosts, we evaluate both homogeneous and heterogeneous cloud VMs and analyzed the *demand imbalance* issue and *utilization mismatch* issue in detail.

To the best of our knowledge, this paper presents the first work on the mutual impact between vCPU scheduling and DVFS based runtime power management for emerging cloud workloads on virtualized platforms.

This paper makes the following contributions:

- We build a comprehensive evaluation framework based on Xen and use representative cloud workloads from CloudSuite [8]. Our framework features a VM behavior analysis wrapper that enables detailed profiling. We classify cloud VM workloads based on their logic functions and grouped them into four categories: *computation VM*, *data serving VM*, *network I/O VM* and *intermediate transfer VM*. We evaluate various deployment scenarios.
- We investigate the implication of vCPU utilization capping in single-VM cloud host. We identify an *uncertainty issue* in runtime power control in virtualized cloud environment: although increasing utilization typically means higher VM energy consumption, it might improve energy efficiency. We show that dirty page rate, VBD local queue processing rate, virtual network packets arrival rate, and network I/O buffer availability are good efficiency indicators for system optimization.
- We identify and characterize a *demand imbalance* issue for multi-VM cloud hosts. For heterogeneous workload consolidation, it increases energy consumption by 13% but lowers performance by 18% on co-located VMs. For homogeneous workload consolidation, this problem also happens due to asynchronous workload execution phases, leading to 29% extra energy consumption with 16% performance degradation. We show that CPU utilization based vCPU grouping and separating, and workload synchronization could help improve energy efficiency.
- We identify and characterize a *utilization mismatch* issue for multi-VM cloud hosts. We show that by matching vCPU to an appropriate processor core, one can improve energy efficiency by up to 32% and 9% for heterogeneous and homogeneous workload consolidation, respectively. In the meantime, one can improve system performance by up to 25% and 20% for heterogeneous and homogeneous workload consolidation, respectively.

The rest of this paper is organized as follows: Section 2 introduces the background. Section 3 describes our cloud workloads classification and experimental setup. Section 4 presents our evaluation results and analyses. Section 5 discusses related work and Section 6 concludes this paper.

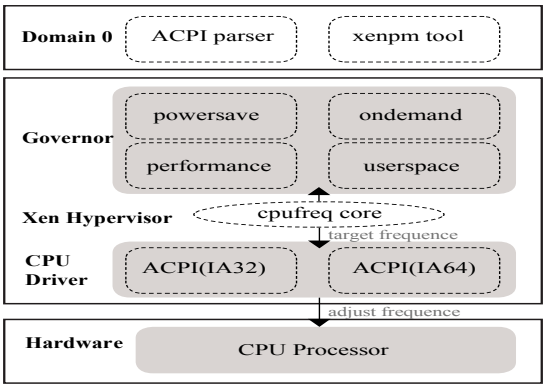


Figure 1: The Hypervisor-based *cpufreq*

II. BACKGROUND

A. DVFS-Based Runtime Power Management in Xen

Dynamics voltage and frequency scaling (DVFS) is the major runtime power saving mechanism for current multi-core processors. Today, most operating systems and hypervisors have incorporated related modules to support DVFS based runtime power management. For example, Xen supports CPU P-states (*cpufreq*) based on the Enhanced Intel SpeedStep Technology [11] or AMD PowerNow! Technology [12]. The *cpufreq* driver in virtualized environment periodically measures CPU utilization and adjusts frequency based on the status of CPU governors. Xen has two *cpufreq* implementations: one is Domain 0 based, which implements the *cpufreq* logic in Domain 0; and the other one is hypervisor based, which implements the *cpufreq* logic in hypervisor.

The Domain 0 based *cpufreq* implementation handles frequency adjustment requests through Domain 0 kernel code with *XENPF_change_freq* and *XENPF_getidletime* hypercalls to obtain system status and change the CPU frequency. In contrast, the hypervisor-based method is more complicated, which involves three layers of coordination. As shown in Figure 1, the hardware layer contains the processor with DVFS capability. The Domain 0 layer includes two components: 1) an ACPI parser that passes the frequency adjustment information to the hypervisor *cpufreq* core via a ACPI table, and 2) a user-level control tool called *xenpm*. The hypervisor layer has three components: *cpufreq* core, *cpufreq* governor and *cpufreq* CPU driver. The *cpufreq* core specifies the overall logic and the *cpufreq* CPU driver issues the frequency adjustment command to the processor. The *cpufreq* governors play a major role in setting the target fre-

quency based on system conditions. There are four major governors. *Performance* and *Powersave* governors set the highest and the lowest frequency statically. *Userspace* prescribes frequency based on user requirement. *Ondemand* governor samples the physical core utilization periodically and dynamically adjusts CPU frequency one level upper or lower when the utilization is above or below a specified threshold. We use the hypervisor-based *cpufreq* (xenpm module) with an *Ondemand* governor in this work.

B. Xen vCPU Credit Scheduler

Current vCPU schedulers in the hypervisor, such as VMware's proportional share algorithm [13] and Xen credit algorithm [10], are designed to share CPU resources (fairness), maximize CPU utilization (throughput) and satisfy latency-sensitive requests (responsiveness). Taking Xen credit scheduler as an example, the algorithm maintains a local queue of active vCPU for each physical core, sorted by priority. The scheduler assigns credits (time slice) for future execution. There are two different vCPU priorities: *over* or *under*, representing whether the vCPU has exceeded its credit share or not during the accounting period.

Every vCPU consumes credits when running. The hypervisor implements a system-wide accounting thread to compute vCPU credits. Positive credits imply *under* priority and negative credits means *over* priority. If a vCPU blocks, wakes up, or completes its time slice, the scheduler will choose the next one from the head of the local queue. When a new vCPU is inserted, it is placed after all active vCPUs with the same priority of one physical core. A CPU can also preemptively load one vCPU from other CPUs' queue to balance CPU resources system-wide when it becomes idle. The CPU scheduler also allows customized time slices, proportional share of CPU (weight), and utilization capping.

Traditional priority-based round-robin credit schedulers are fit to achieve fairness among VMs, maximize throughput, and accelerate request processing rate. However, they overlook the inefficiency and overhead of DVFS runtime power control, which motivates our study.

III. Methodology and Cloud Platform

A. Cloud Workload Classification

We use representative cloud workloads from CloudSuite [8] and classify them as either *symmetric* or *asymmetric*. For the *symmetric* workloads, all servers play an equal role in data processing. In contrast, for the asymmetric workloads, servers take different responsibilities (e.g. front-end request processing, task partition, back-end data serving, etc.).

The symmetric configuration is popular in data store or data caching systems and has three features: 1) data sets are split across all the servers; 2) each node can process a client's request independently based on its own fraction of data; 3) there is little coordination or synchronization among different servers. The symmetric workloads can be further divided into two different types based on communication patterns: *cooperative* and *independent*. In the *cooperative* type, such as Cassandra NoSQL distributed storage [14], the whole system is transparent to the client and each node stores a fraction of data along with its hash values. In this case, each node can receive client requests but they might be further routed to the corresponding nodes that have the data. In the *independent* communication scenario, such as a media streaming server, no communication exists between nodes and each node can directly reply to client's requests. In the CloudSuite benchmark, *Data Caching*, *Data Serving*, *Graph Analytics*, and *Media Streaming* all belong to this type.

TABLE I. Workload Description and Configurations

Benchmark	Description	Setup
Data Analytics (***Asymmetric***)	Run Bayesian classification algorithm on Wikipedia data set in Hadoop cluster	1 master VM and 2 slave VMs
Data Caching (***Symmetric)***	Run Memcached data cache server with twitter data set	4 data caching VMs
Data Serving (***Symmetric***)	Run Yahoo! Cloud Serving Benchmark (YCSB) dataset on Cassandra database	4 Cassandra data store VMs
Graph Analytics (**Symmetric**)	Run machine and data mining algorithms for graph analytics	4 standalone computing VMs
Media Streaming (***Symmetric***)	Run simulated clients with Faban driver on Darwin Streaming server	3 Darwin streaming VMs
Software Testing (***Asymmetric***)	Run Klee SAT Solver parallel symbolic engine	1 load VM and 3 worker VMs
Web Search (***Asymmetric***)	Run simulated clients with Faban driver to benchmark Nutch search engine	1 frontend VM and 2 web search VMs
Web Serving (***Asymmetric***)	Run simulated clients with Faban driver to benchmark social event calendar with Cloudstone [17]	1 frontend VM and 1 backend VM

The asymmetric configuration usually assigns one server (or multiple) as a coordinator, which is responsible for job partitioning or tasks balancing. The other servers just act as computation or data storing resources. In terms of how client requests are processed, we observe that nearly all workloads in asymmetric configuration maintain a "master-slave" cooperation style. For example, Hadoop [15] splits and assigns data sets through key/value pairs on master nodes and executes tasks on slave nodes. For load-balancer applications, load node receives clients' requests, splits complex problems into sub-tasks, and then balances sub-tasks among all the workers. In a Web processing scenario, a Web front-end node usually distributes client requests across other servers, collects and sorts temporary responses from Web back-end servers, gathers results then replies to clients. In CloudSuite, *Data Analytics*, *Software Testing*, *Web Search* and *Web Serving* are all asymmetric configurations.

In Table 1 we summarize the evaluated workload configurations. We consolidate these cloud workloads into virtualized environment. Each VM is referred to as a node in the benchmark. Based on the above classification and cloud VM logic function analyses, we further group all these VMs into four categories: *computation VM*, *data serving VM*, *network I/O VM*, and *intermediate transfer VM*.

The *computation VM* focuses on job (task) execution on any available computing resources. The slave node of *Data Analytics*, analysis node in *Graph Analytics*, and worker

node of *Software Testing* all belong to this kind. The *data serving VM* is mainly used for storing data, such as Cassandra node in *Data Serving*, cache node in *Data Caching*, and Web slave node in *Web Serving*. The *network I/O VM* handles client requests and sends network packages, which differs from *data serving VM* in that it has lots of network I/O requests instead of disk I/O requests. The *network I/O VM* is important in cloud workload as it is frequently used in media streaming application. These three types of VMs are usually used for nodes in symmetric organization.

The *intermediate transfer* VM is one of the most widely deployed schemes in clouds, especially for asymmetric configurations. It is responsible for client request analyzing, assigning tasks to slave (worker) nodes, collecting requests and replying to clients. It usually combines characteristics of *data serving VM* with *network I/O VM*. Most master nodes and Web front-end nodes belong to this type. For asymmetric cloud workloads, there must be one *intermediate transfer VM* with other type VMs. The impact of vCPU scheduling on each kind of VM will be discussed in Section IV.

B. Experimental Setup

We perform experiments on servers that use Intel Core™ i7-2600K processors, 32GB RAM, and 500GB hard drives. The processor has 4 physical cores with hyper-threading. It also supports 16 performance states with 1600MHz minimum frequency and 3801MHz maximum frequency. We use Xen-based virtualization to create virtual machines for cloud workloads. All the experiments are carried out on Xen-4.1.2 hypervisor with kernel 2.6.32.40 for Domain 0. Each evaluated VM is HVM type with 2 VCPUs and 2 G memories.

We choose the credit-based scheduler and *Ondemand* governor as vCPU scheduling and DVFS runtime power management. In the single-VM scenario, we study the effect of vCPU scheduling on each kind of VM's DVFS power control. For multi-VM cloud hosts, we analyze *demand imbalance* (detailed in Section IV-B) and *utilization mismatch* (detailed in Section IV-C) issues using both homogeneous and heterogeneous VM consolidation.

For each VM in the single-VM experiment, we characterize detailed virtual machine behaviors to explore efficiency indicators, including CPU utilization, CPU frequency, dirty page number per second, network packets send/receive per second, and VBD (virtual block device) sectors read/write per second. The CPU frequency is collected using *xenpm* tool. Other metrics including CPU utilization, network packets send/receive per second and VBD sectors read/write per second are collected through *xentop* tool. To count dirty page number per second, we develop a method to record dirty page number with log dirty mode. We integrate all instruments into a single wrapper (as shown in Figure 2) to generate dynamic profile of VM behavior under *Ondemand* governor. To achieve accurate frequency analyses, we use *xm vcpu-pin* method to pin each vCPU to a specific physical CPU. We use WattsUp [16] meter to measure the power consumption of our physical host.

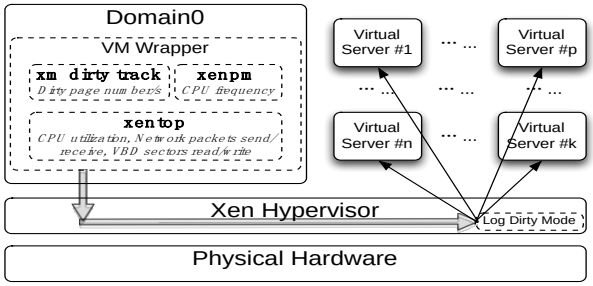


Figure 2: VM Behavior Evaluation Wrapper

IV. RESULTS AND ANALYSES

In this section, we first evaluate the impact of vCPU scheduling on single VM in virtualized cloud. We investigate how vCPU scheduling (especially CPU utilization capping) affects four types of representative cloud VM in terms of power, performance, and energy. We then identify and characterize the *demand imbalance* and *utilization mismatch* issues for multi-VM cloud hosts.

A. Single-VM Cloud Host

For single-VM scenario, we carefully evaluate four types of VM: 1) worker VM (*computation VM*) from the *Software Testing* benchmark; 2) data store VM (*data server VM*) from the *Data Analytics* benchmark; 3) Darwin streaming VM (*network I/O VM*) from the *Media Streaming* benchmark; and 4) Web front-end VM (*intermediate transfer VM*) from the *Web Search* benchmark. Figure 3 shows how vCPU capping affects the performance, power and efficiency of the evaluated VMs. We adjust the maximum CPU utilization by varying the vCPU cap value (the upper limit of the provided CPU utilization) through Xen credit scheduler. The command is: *xm sched-credit*.

The first interesting observation is that the vCPU utilization capping affects the energy efficiency of different VMs in a different way. According to the *Ondemand* DVFS policy implemented in the Linux kernel, processor frequency is dynamically changed depending on current usage. Therefore, CPU frequency often increases when utilization goes up. As shown in Figure 3-(a), although a higher CPU frequency typically results in increased power demand, the system energy consumption does not necessarily increase significantly. Here the *computation VM* yields a relatively flat normalized energy efficiency curve when utilization varies, indicating better energy proportionality. In contrast, other VMs are less energy-proportional to CPU utilization. For example, the energy consumption of *network I/O VM* and *data serving VM* increases significantly as we decrease their utilization caps. This is because they incur a great deal of un-core power demand which does not change much when we vary the CPU frequency. For *data serving VMs*, there are massive memory and disk operations due to the NoSQL database requests (such as column read and dataset update). For *network I/O VMs* (such as VMs that run *Media Streaming* workloads), storing and supplying data for clients

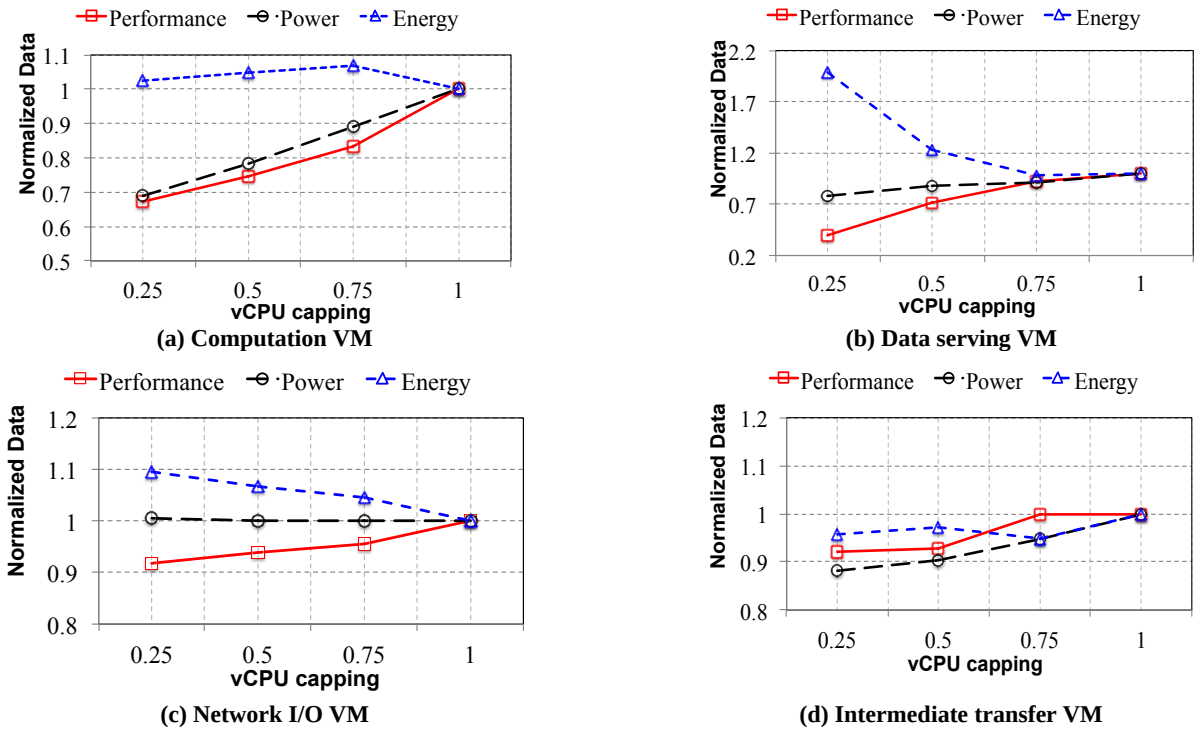


Figure 3: Performance, power, and energy of four VMs under different vCPU caps. All the experimental data are normalized to 100% vCPU cap in each VM. We use throughput or 1/execution time to represent performance. Power is quantified for the physical host and is averaged during the execution. Energy is computed based on performance and power.

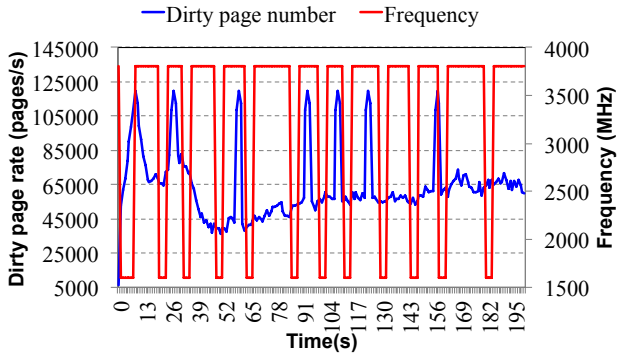
requires both disk and network I/O requests. The *intermediate transfer VM* combines the characteristics of both *data serving VM* and *networking I/O VM*. As shown in Figure 3-(d), its energy consumption fluctuates as we increase the utilization cap from 25% to 100%. The observed energy saving at highest CPU utilization is less than 5%.

Capping utilization of vCPUs could decrease the performance of all types of cloud VMs. In Figures 3(a)–3(d), the VM throughput is always positively correlated with CPU utilization. We observe that a higher CPU utilization can increase performance significantly only if the VMs are computation-intensive (such as the code analysis task in worker VM and the request partitioning/collecting in data store VM). For both *network I/O VM* and *intermediate transfer VM*, increasing vCPU cap from 25% to 100% results in less than 9% performance improvement. This is because these two types of workload have low computation demand but incur intensive I/O interrupts due to disk read/write and packets send/receive.

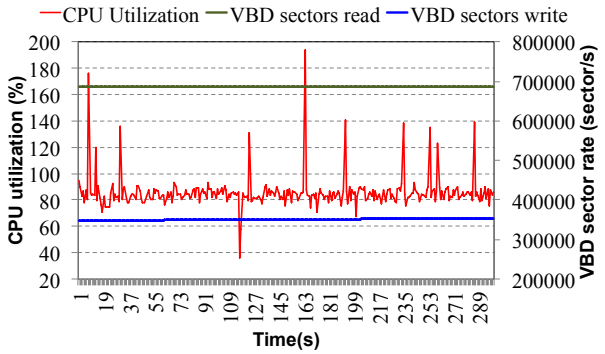
Figure 3 also indicates that the most energy-efficient operating statue (i.e., the most energy-efficient VM utilization cap) can be nondeterministic. During runtime, cloud workloads affect both CPU utilization and frequency adjustment in an unpredictable way, resulting in very complex system-hardware interaction. To fully understand the operational efficiency of VM scheduling in a DVFS-enabled cloud environment, we monitor detailed virtual machine runtime behaviors. We set unlimited vCPU utilization in each evaluated virtual machine and use *Ondemand* DVFS governor. Our VM characterization considers dirty page rate, network packet send/receive rate, and VBD sector read/write rate. Figure 4 reports our experimental results.

Similar to traditional architectural DVFS power control schemes [28], *computation VMs* incur inefficient frequency adjustment primarily due to memory operations. In this study we evaluate memory access behavior with dirty page rate at the hypervisor/OS level. For characterization purpose, we collect dirty page number of each virtual machine every second. As we can see from Figure 4-(a), high dirty page rate results in CPU frequency droop when CPU utilization is below the threshold. If application generates irregular dirty page ratio, the frequency jitter will happen, such as 20s ~ 70s and 80s ~ 160s period in Figure 4-(a).

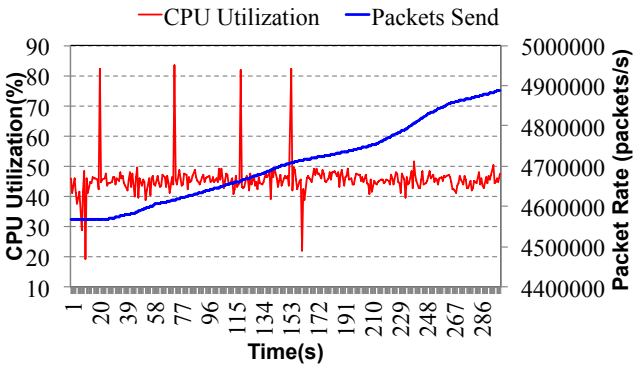
For *data serving VMs*, we monitor their virtual block device (VBD) operation factors. The VBD refers to an emulated block device (e.g., disk) for guest virtual machine. It serves as the interface between guest OS and physical block device. Figure 4-(b) shows the CPU utilization and VBD sector operation rate of Cassandra VM over 300s of execution duration. We observe that the VBD read/write rate is almost constant. However, the VBD read rate of a *data serving VM* (686406 sectors /s) can be 4.0X (169334 sectors/s) and 2.4X (302288 sectors/s) higher than that of *intermediate transfer VMs* and *network I/O VMs*. CPU utilization varies based on clients' requests and can randomly surpass the threshold of *Ondemand* governor, leading to an increased CPU frequency. In this case, lowering vCPU utilization would be more efficient here to wait for VBD processing.



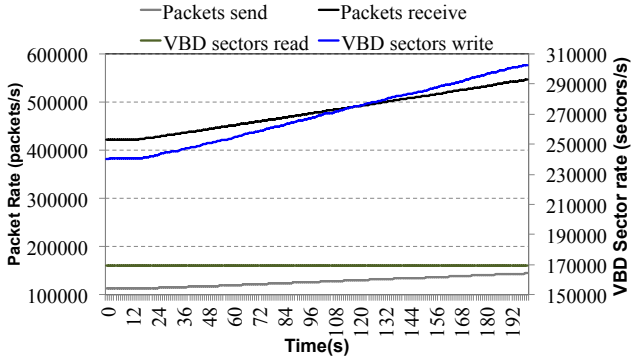
(a) Dirty page number and frequency variation of a **computation VM** in 200s.



(b) CPU utilization and VBD sector read/write variation of a **data serving** VM in 300s



(c) CPU utilization and packet send variation of a **network I/O VM** in 300s



(d) Packet send/receive and VBD read/write variation of an **intermediate transfer VM** in 200s

Figure 4: Detailed characterization of 4 differ VMs during execution with *Ondemand* governor under unlimited vCPU utilization

Figure 4-(c) presents the *network I/O VM*. As its name suggests, a media streaming VM has much higher network packet processing rate (4705394 packets/s) than that of an *intermediate transfer VM* (126900 packets/s). It frequently sends video back after receiving client's request. The CPU frequency stays around 1600MHz throughout execution and goes up occasionally. Specifically, the request analyses and data marshaling phases cause utilization to increase, while the video data read and packet prepare phases lower the utilization. Accelerating request parsing and packet marshaling by increasing vCPU utilization can improve the throughput of the system while maintaining low power consumption at the same time. It is useful when the clients' requests arrive frequently or the network buffer is ready.

Figure 4-(d) shows the VM characterization of *intermediate transfer VMs*. Even though VBD sector and network packets processing rate are not striking among these 4 VMs, it combines characteristics from *data serving VM* and *network I/O VM*. During 200s of execution time, VBD sector read/write has the same trend as *data serving VM* and network packet send/receive varies like *network I/O VM*.

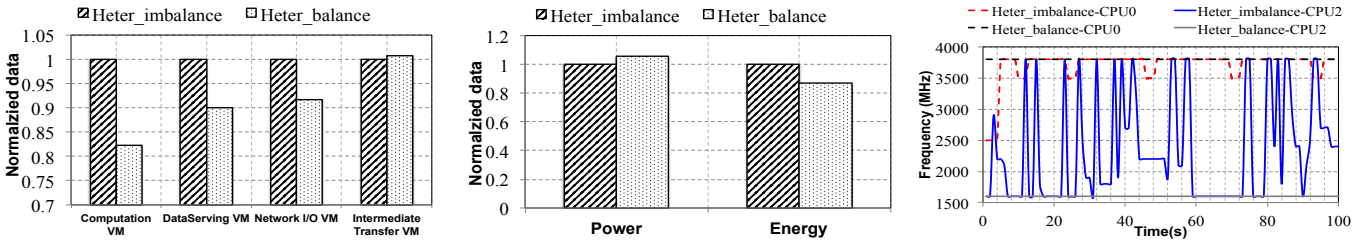
Summary: Virtual CPU (vCPU) scheduling affects DVFS power management effectiveness through setting different CPU utilization limits. Usually, the higher the utilization is, the better the performance will be. However, the impact of utilization tuning on VM energy consumption is uncertain. To save energy cost while maintain high performance, one should dynamically adjust the utilization capping of vCPU scheduling based on detailed VM behavior analyses. Our characterization shows that VM dirty page ratio, VBD local queue processing rate, virtual network packets arrival rate, and network buffer I/O availability may serve as good efficiency indicators for fine-grained vCPU utilization control.

This work mainly focuses on the impact of vCPU scheduling on DVFS-based power management. Further system design and software overhead optimization are out of the scope of this paper, and will be our future work.

B. Demand Imbalance Issue in Multi-VM Host

In this subsection, we focus on multi-VM hosts (VMs are co-located on a single server). Our key observation is that the server energy efficiency could decrease significantly when multiple vCPUs with different utilization requirements are co-located in the local queue of a single physical core. We refer to this problem as *demand imbalance*.

The *demand imbalance* issue happens because current frequency control mechanism is unable to follow the fast-changing vCPU frequency requirements imposed by vCPU scheduler. Existing schedulers typically execute these runnable vCPUs in a round-robin fashion until they finishes. The default vCPU switching interval is 30ms [10], while the minimal sampling rate of *Ondemand* governor is 10ms [9]. Although one can increase the switching interval for more effective DVFS control, it can significantly degrade the performance of some latency-sensitive virtual machines.



(a) Execution time **(b) Power and energy** **(c) Frequency variation in 100s**

Figure 5: Experimental results of heterogeneous workload consolidation under *demand imbalance*. Performance is evaluated through execution time. Energy is computed with the average power and the execution time of each computation VM. Performance, power and energy are normalized to *heter_imbalance* scenario. Frequency variation is selected from CPU 0 and CPU 2 in the first 100s.

To understand the *demand imbalance* issue, let's consider a system with 16 performance states (P-state, P0-P15). Assume that a 100% utilized vCPU ***A*** is newly scheduled on a physical core previous mapped by a vCPU ***B*** of 10% utilization (running at P3, for example). The physical core can increase its frequency by three levels at most, which is still much lower than the targeted frequency (i.e., the highest performance state P15). When ***A*** has used up its credits, ***B*** starts to execute again and the CPU frequency drop back to its original level. Consequently, ***A*** has to be running on a less efficient performance state. It can achieve its best performance-per-watt only if ***B*** is removed from the local queue. On the other hand, when ***B*** is scheduled on a core previously long mapped by ***A*** (with a steady high frequency level), ***B*** will always stay at a frequency that is much higher than required, leading to unnecessary energy consumption.

1) *Demand Imbalance with Heterogeneous Consolidation*

The *demand imbalance* scheduling problem is common in virtualized cloud environment, especially for heterogeneous workload consolidation. To demonstrate this, we compare two power management scenarios: heterogeneous consolidation with *demand imbalance* issue (*heter_imbalance*) and heterogeneous consolidation without *demand imbalance* issue (*heter_balance*). In both scenarios, we create 4 different types of multi-VM host: graph analysis VM (*computation VM*), Cassandra data store VM (*data serving VM*), media streaming VM (*network I/O VM*), and Web front-end VM (*intermediate transfer VM*). Each VM has 2 vCPUs and 4GB memory. Our detailed profiling data shows that graph analysis VM and Cassandra data store VM have similarly high CPU utilization, while media streaming VM and Web front VM have very low utilization.

In the *heter_imbalance* scenario, we map the vCPUs of *computation VM* and *network I/O VM* to two separate CPUs (CPU 0 and CPU 1), while *data serving VM* and *intermediate transfer VM* are consolidated on another group of two CPUs (CPU 2 and CPU 3). In the *heter_balance* scenario, we experiment with different VM consolidation scenarios; we consolidate the vCPUs of *computation VM* and *data serving VM* on CPU 0 and CPU 1. We deploy *network I/O VM* and *intermediate transfer VM* on CPU 2 and CPU 3. Hence, in the first scenario (i.e., *heter_imbalance*), each physical CPU runs two vCPUs of very different utilization. In the second scenario (i.e., *heter_balance*), however, each physical CPU runs two vCPUs of similar utilization.

Figure 5 presents our evaluation results with heterogeneous workloads. In terms of performance, except that *intermediate transfer VM* shows 0.7% degradation, all the other VMs' performance have been improved in the *heter_balance* scenario. The *computation VM* and *data serving VM* degrades 17.7% and 10.1% in *heter_imbalance* scenario because they can hardly run at their targeted frequency.

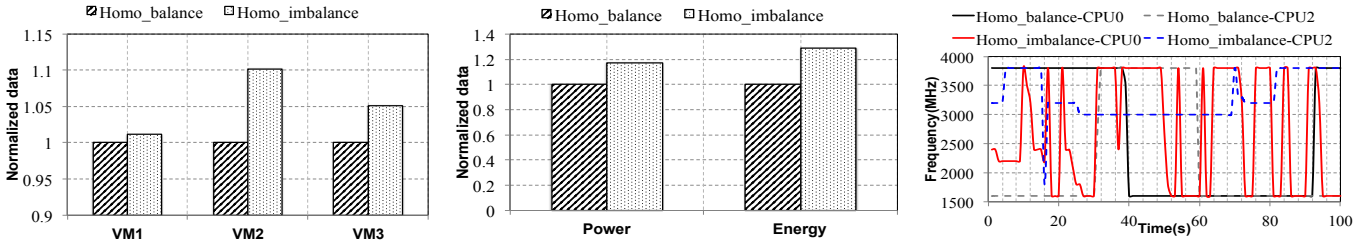
From the point of view of energy efficiency, balancing VM utilization can lead to very interesting results. As shown in Figure 5-(b), even though the balanced scenario shows 5.6% higher power demand, the total energy consumption is saved by 13.4% due to fast execution of *computation VM*.

Figure 5-(c) illustrates the frequency switching phenomenon in detail. In the *heter_imbalance* scenario, CPU 0 and CPU 2 frequently adjust their frequency. In contrast, in the *heter_balance* case, both CPU 0 and CPU 2 maintain constant frequency (3801 MHz and 1600 MHz, respectively). *Intermediate transfer VM* obtains slightly better performance since it executes in much higher frequency than expected. However, *network I/O VM* shows 8.3% performance improvement in *heter_balance* scenario due to performance interference, which means consolidating computation VM and network I/O VM together leads to more L2 cache miss. This has been discussed in [32, 33].

2) *Demand Imbalance with Homogeneous Consolidation*

Not only heterogeneous workload, but also homogeneous workload consolidation can suffer from *demand imbalance*. Even if the consolidated VMs have the same CPU utilization, their execution phases are not always synchronous. For example, consider one VM that has three periodical utilization phases of equal duration (e.g., 90%, 60%, and 20% utilization). When two VMs of this type are mapped to the same physical CPU, it is very likely that the second VM begins to execute when the first VM are in the third phase. In this case, the vCPUs of these two VMs are not coordinated very well, resulting in unnecessary frequency adjustment.

To explore the *demand imbalance* issue in homogeneous consolidation environment, we create three Cassandra data store VMs with 2 vCPUs and 2G memory. For each created VM, we map their vCPUs to separate physical CPUs (CPU 0 and CPU 1). We experiment with two scenarios. In the first scenario (*homo_balance*), we use synchronous vCPU scheduling. In the second scenario (*homo_imbalance*), we analyze asynchronous execution in vCPU scheduling by deferring the application start time.



(a) Execution time **(b) Power and energy** **(c) Frequency variation in 100s**

Figure 6: Experimental results of homogeneous workload consolidation *under demand imbalance*. Performance is measured through execution time. Energy is computed as the average energy at the time that all VM finishes. Performance, power and energy are normalized to *homo_balance* scenario. Frequency variation is selected from CPU 0 and CPU 2 in the first 100s.

Figure 6 shows our experimental results with homogeneous workloads. Obviously, *homo_balance* is much stable than *homo_imbalance* with fewer frequency jitters since synchronous execution allows different VMs to take advantage of each other for DVFS-based software power management. In contrast to *homo_imbalance* scenario, different VMs in the *homo_balance* scenario maintain similar execution phase. Compared to *homo_balance*, *homo_imbalance* increases the execution time by 1.1%, 10.2, and 5.1% for VM1-VM3, respectively. The measured average power also decreases by 17.2%, leading to 28.7% total energy savings.

Note that the above results are conservative since there are still a significant amount of overlaps among the evaluated VM execution phases. For cloud workloads, nearly all applications on the server side maintain one daemon process, waiting for client's requests and then starting related actions. Such execution pattern always leads to asynchronous execution in homogeneous workload environment.

Summary: There are two ways to improve vCPU scheduler performance and system energy efficiency: 1) grouping vCPUs based on their utilization and assigning different groups (with different utilization levels) on physical cores. 2) synchronizing the workload execution phases.

C. Utilization Mismatch Issue in Multi-VM Host

We further explore the inefficiency of existing vCPU scheduler when a new vCPU has been created and inserted in existing vCPU queue. We refer to this problem as the *utilization mismatch* in multi-VM host environment.

For current credit-based scheduler, when a new vCPU is brought into the system, the scheduler will place it into the local queue of one physical core either according to the user's preference or based on system's load balancing mechanism. However, this newly inserted vCPU may disturb the steady performance state (P-state) established by runtime DVFS power management. For example, when a vCPU of *computation VM* is assigned to a local queue with all vCPUs from *network I/O VM*, this vCPU can hardly achieve its targeted frequency level. Similarly, if a vCPU of *network I/O VM* comes into a runnable queue of all vCPUs from *computation VM*, it will always run at a higher frequency level, leading to unnecessary power consumption. Note that the *utilization mismatch* problem could eventually become a *demand imbalance* problem. However, the *utilization mismatch* problem is mainly concerned with vCPU provisioning and distribution efficiency.

1) Utilization Mismatch with Heterogeneous Consolidation

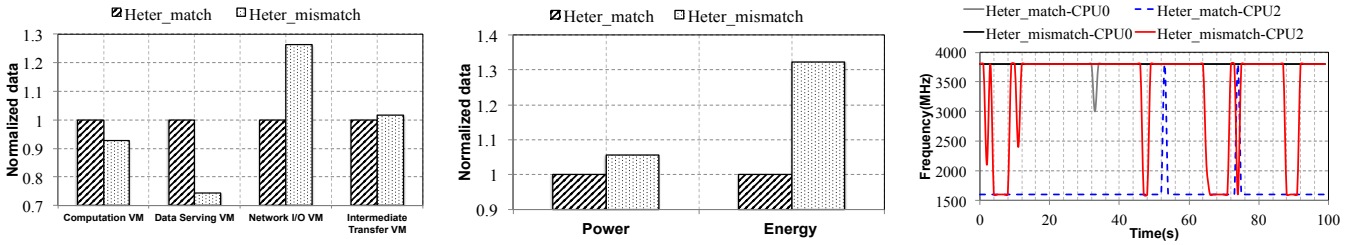
For heterogeneous workloads, we create 4 different types of VMs as before, software testing VM (*computation VM*), Cassandra data store VM (*data serving VM*), media streaming VM (*network I/O VM*), and Web front VM (*intermediate transfer VM*). Each VM has 2 vCPUs and 4G memory. We deploy *computation VM* and *data serving VM* on CPU 0 and CPU 1 while *network I/O VM* and *intermediate transfer VM* are mapped to CPU 2 and CPU 3. The former two have similar high utilization and the latter two stay at low utilization. In this experiment the newly created VM is a *computation VM* with 2 vCPU and 4G memory. In a utilization matched scenario (*heter_match*), we statically pin this VM to CPU 0 and CPU 1. In a utilization mismatched scenario (*heter_mismatch*), we map the new VM to CPU 2 and CPU 3.

We present our results in Figure 7. In the *heter_match* scenario, since we have assigned additional vCPUs to CPU 0 and CPU 1, the execution time of *computation VM* and *data serving VM* have been increased by 7.4% and 25.6%, respectively. Similarly, in the *heter_mismatch* scenario, the execution time of *network I/O VM* and *intermediate transfer VM* have been increased. Regarding the newly inserted VM, its execution is 94.7s in the matched case and 118.6s (25.2% longer) in the mismatched scenario. Therefore, avoiding utilization mismatch can help improve the performance of the newly inserted VMs significantly.

In terms of energy efficiency, the *heter_match* could reduce 5.6% power demand and 32.4% energy consumption, as shown in Figure 7-(b). This can be explained in Figure 7-(c). The frequency of CPU 0 in the matched case is not disturbed by the newly created vCPU, which still maintains at 3801 MHz as it is in the second one. However, the frequency of CPU 2 in *heter_mismatch* scenario rises since the newly inserted *computation VM* has higher utilization. We can see more frequency jitter in Figure 7-(c), where the CPU 2 in *heter_match* case only has two jitters due to the *intermediate transfer VM*.

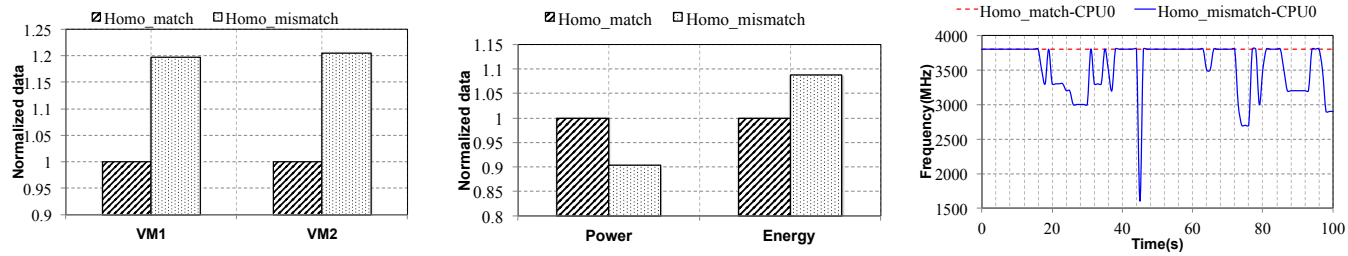
2) Utilization Mismatch with Homogeneous Consolidation

In homogeneous consolidation environment, we create 2 *data serving VMs* with 2 vCPUs and 4G memory. We insert a *computation VM* in the matched scenario (*homo_match*) and we insert an *intermediate transfer VM* in the mismatched scenario (*homo_mismatch*). The newly inserted VM in both cases have the same configuration. All the VMs are pinned to CPU 0 and CPU 1.



(a) Execution time **(b) Power and energy** **(c) Frequency variation in 100s**

Figure 7: Experimental results of heterogeneous workload consolidation with *utilization mismatch*. Performance is measured by execution time. Energy is computed with the average power and the time that all 4 VMs are finished execution. Performance, power and energy are normalized to *heter_match* scenario. Frequency variation is selected from CPU 0 and CPU 2 in the first 100s.



(a) Execution time **(b) Power and energy** **(c) Frequency variation in 100s**

Figure 8: Experimental results of homogeneous workload consolidation with *utilization mismatch*. Performance is measured by execution time. Energy is computed with the average power and the time that all 2 VMs are finished execution. Performance, power and energy are normalized to *homo_match* scenario. Frequency variation is selected from CPU 0 and CPU 2 in the first 100s.

As shown in Figure 8-(c), CPU 0 in *homo_mismatch* scenario varies a lot as the *intermediate transfer VM* has unstable CPU utilization compared to that in *homo_match* scenario where CPU 0 maintains high frequency. Thus, the execution time of two *data serving VMs* in the mismatch scenario increases by 19.7% and 20.5%, compared to a matched scenario. Although the physical host consumes 9.7% more power, its energy consumption decreases by 8.8% due to fast execution.

Summary: For a newly created VM, utilization mismatched scheduling not only affects the performance of collocated virtual machines, but can also degrades energy efficiency of the physical host. This could happen in both homogeneous and heterogeneous cloud environment. Without appropriate optimization, existing scheduling schemes can adversely affect DVFS-based power management. By re-mapping the unmatched vCPU to a different physical core, one could improve the overall system efficiency.

V. Related Work

Prior work on power management in virtualized environment mainly include cluster-level management, server-level and system-level management.

There have been many papers discussing power control of VMs across the data center. Nearly all approaches [4, 19, 21, 34] leverage live migration techniques for online scheduling. For example, vGreen [19] proposes a multi-tiered software system to manage VM scheduling at cluster-level to achieve energy efficiency. Its key idea is to explore the relation between architectural characteristics (IPC, memory behavior, etc.) of a VM and system performance. The distributed resource scheduler (DRS) from VMware [20] is a proprietary solution, which also takes power into consideration when placing and scheduling one VM. Recently, [21] proposes a monitoring infrastructure to measure both performance and QoS metrics for different types of workloads. Their scheme dynamically manages services and batch jobs in order to maximize throughput. Their solution scales well for resource management in heterogeneous workloads.

At the server level, prior studies mainly focus on power-aware scheduling using DVFS. These studies typically monitor virtual machine load and apply fine-grained performance control based on DVFS technique. For example, [5] proposes a feedback control framework, which takes processor load as input and dynamically decides the new frequency based on the historical trend. It also allocates a power credit for each guest OS, which is used to account power consumption for different CPU frequency. In contrast, [6] proactively monitors VM CPU load and dynamically scales processor frequency to save energy. In addition, a recent proposal [7] explores low-latency power states (S states) in enterprise servers and integrates them with an end-to-end power-aware virtualization management scheme.

At the system level, [22] leverages the power management decisions of the guest OS on virtual power states to determine local and global policies. It mainly relies on efficient power management policies in the guest OS and takes little VM characterization into consideration. The authors in [23] design a coordinated multi-level solution to drive VM placement and power management via power estimation and workload utilization. Stoess et al. [24] implements an effective infrastructure to account, distribute and control the power consumption in multi-layered software virtual environment. For HPC workload, the authors in [25] use VM characteristics such as cache footprint and working set to achieve

power-aware VM placement. Besides, the authors in [26] build power models based on VM resource usage to infer VM power consumption at runtime then they further work to budget power in virtualized data centers [27].

There are also extensive studies focusing on DVFS evaluation and optimization in the machine architecture level. For example, [28] and [29] put an emphasis on DVFS performance predictor. [30] and [31] explore workload execution phase detection. In this paper we focus on system-level frequency scaling in cloud environment.

Our paper distinguishes itself from prior work in two aspects. First, this paper provides the first detailed characterization on the interplay between vCPU scheduling and DVFS-based system power management. Second, while prior work mainly focuses on traditional applications, we put an emphasis on emerging scale-out virtualized cloud workload (including single- and multi- VMs hosts), which has much more complex execution behaviors.

VI. CONCLUSIONS

In this study we explore the efficiency of vCPU scheduling on system-level DVFS power management in virtualized cloud environment. We evaluate four types of representative cloud virtual machine and we examine the interplay between vCPU scheduling and dynamic frequency scaling on both single-VM and multi-VMs cloud hosts.

For single-VM hosts, vCPU scheduling affects DVFS power management through setting different CPU utilization caps. Although workload performance increases when utilization goes up, we observe that the energy efficiency of the physical host is actually workload-dependent.

For multi-VM hosts, we identify and characterize the *demand imbalance* and *utilization mismatch* issues which can cause inefficient or unnecessary frequency tuning activities. We show that existing platforms have the potential to improve energy efficiency and workload performance by 32% and 25%, respectively, if vCPUs are balanced and appropriately scheduled based on their utilization.

This work represents our first step toward designing a joint optimization scheme for improving both virtual machine's performance and physical host's energy efficiency. We believe this paper provides valuable insights for energy-efficient virtualization in the cloud environment.

VII. ACKNOWLEDGEMENTS

We thank the anonymous reviewers. This work is supported in part by NSF grants 1320100, 1117261, 0845721(CAREER), and by Microsoft Research Safe and Scalable Multi-core Computing Awards. Ming Liu is also supported by University of Florida Graduate Fellowship.

REFERENCES

[1] Server consolidation benefits, http://www.vmware.com/solutions/consolidation/consolidate.html
[2] Enterprise virtualization survey, http://usa.kaspersky.com/about-us/press-center/press-releases/survey-among-us-companies-adopting-it-virtualization-53-percent
[3] The 2012 Uptime Institute Data Center Industry Survey, The Uptime Insitute, 2012
[4] H. Esmaeilzadeh, E. Blem, R. St. Amant, K. Sankaralingam, and D. Burger, "Dark Silicon and the End of Multicore Scaling", in ISCA, 2011
[5] C. Wen, J. He, J. Zhang and X. Long, "PCFS: Power Credit Based Fair Scheduler Under DVFS for Multicore Virtualization Platform", in GREECOM-CPSCOM, 2010
[6] C. M. Kamga, G. S. Tran, and L. Broto, "Power-aware scheduler for virtualized systems", in the proceedings of 2nd International Workshop on Green Computing Middleware, 2011
[7] C. Isci, S. McIntosh, J. Kephart, R. Das, J. Hanson, S. Piper, R. Wolford, T. Brey, R. Kantner, A. Ng, J. Norris, A. Traore, and M. Frissora, "Agile, Efficient Virtualization Power Management with Low-latency Server Power States", in ISCA, 2013
[8] M. Ferdman, A. Adileh, O. Kocberber, S. Volos, M. Alisafaee, D. Jevdjic, C. Kaynak, A. D. popescu, A. Ailamaki, and B. Falsafi, "Clearing the Clouds: A Study of Emerging Scale-out workloads on Modern Hardware", in ASPLOS 2012
[9] Xen power management, http://wiki.xen.org/wiki/Xen_power_management
[10] Xen credit scheduler, http://wiki.xen.org/wiki/Credit_Scheduler
[11] Intel EIST technology, http://www.intel.com/cd/channel/reseller/asmona/eng/203838.htm
[12] AMD PowerNow!, http://en.wikipedia.org/wiki/PowerNow!
[13] VMware technical white paper, "The CPU Scheduler in VMware vSphere 5.1"
[14] The Apache Cassandra Project, http://cassandra.apache.org
[15] Hadoop, http://hadoop.apache.org
[16] Wattsup meter, http://wattsupmeters.com
[17] Will Sobel, et al., "Cloudstone: multi-platform, multi-language benchmark and measurement tools for web 2.0", in the 1st Workshop on Cloud Computing and Its Applications, October 2008
[18] CloudSuite Benchmark, http://parsa.epfl.ch/cloudsuite/cloudsuite.html
[19] G. Dhiman, G. Marchetti, and T. Rosing, "vGreen: a system for energy efficient computing in virtualized environment", in ISLPED, 2009
[20] http://www.vmware.com/products/vsphere/features-drs-dpm
[21] G. Dhiman, V. Kontorinis, R. Ayoub, L. Zhang, C. Sadler, D. Tullsen, and T. S. Rosing, "Themis: Energy Efficient Management of Workloads in Virtualized Data Centers", in Euro-Par, 2012
[22] R. Nathuji and K. Schwan, "VirtualPower: coordinated power management in virtualized enterprise systems", in SOSP, 2007
[23] R. Raghavendra, P. Ranganathan, V. Talwar, Z. Wang, and X. Zhu, "No "power" struggles: coordinated multi-level power management for the data center", in ASPLOS, 2008
[24] J. Stoess, C. Lang, and F. Bellosa, "Energy management for hypervisor-based virtual machines", in USENIX ATC, 2007
[25] A. Verma, P. Ahuja, and A. Neogi, "Power-aware dynamic placement of HPC applications", in ICS, 2008
[26] A. Kansal, F. Zhao, J. Liu, and A. A. Bhattacharya, "Virtual Machine Power Metering and Provisioning", in SoCC, 2010
[27] H. Lim, A. Kansal and J. Liu, "Power budgeting for virtualized data centers", in USENIX ATC, 2011
[28] R. Miftakhutdinov, E. Ebrahimi, and Y. N. Patt, "Predicting Performance Impact of DVFS for Realistic Memory Systems", in MICRO, 2012
[29] M. Moeng and R. Melhem, "Applying statistical machine learning to multicore voltage and frequency scaling", in CF (*Int. Conf. Computing Frontiers*), 2010
[30] C. Isci, G. Gontreras, and M. Maronosi, "Live, runtime phase monitoring and prediction on real systems with application to dynamic power management", in MICRO, 2006
[31] C. Isci and M. Martonosi, "Phase characterization for power: Evaluating control-flow-based and event-counter-based techniques", in HPCA, 2006
[32] S. Govindan, J. Liu, A. Kansal, and A. Sivasubramaniam, "Cuanta: Quantifying Effects of Shared On-chip Resource Interference for Consolidated Virtual Machines", in SOCC, 2011
[33] R. Nathuji, A. Kansal, and A. Ghaffarkhah, "Q-Clouds: Managing Performance Interference Effects for QoS-Aware Couds", in Eurosys, 2010
[34] I. Goiri, F. Julia, R. Nou, J. L. Berral, J. Guitart, and J. Torres, "Energy-aware Scheduling in Virtualized Datacenters", IEEE Cluster, 2012
[35] K, H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of virtual machines for real-time Cloud services", in Journal of Concurrency and Computation: Practice & Experience, Vol. 23, Issue 13, 2011