J Supercomput (2013) 66:1629–1655
DOI 10.1007/s11227-013-0956-1

# Analysis of virtual machine live-migration as a method for power-capping

**Jinkyu Jeong · Sung-Hun Kim · Hwanju Kim · Joonwon Lee · Euiseong Seo**

Published online: 11 May 2013

**Abstract** To reduce the construction cost of the power-supplying infrastructure in data centers and to increase the utilization of the existing one, many researchers have introduced software-based or hardware-based power-capping schemes. In servers with consolidated virtual machines, which can be easily found in cloud systems, exporting virtual machines to other light-loaded servers through live-migration is one of the key approaches to impose power-capping on servers. Up until now, most researchers who have tried to achieve power-capping through live-migration assumed that exporting a virtual machine instantly reduces the server power consumption. However, our analysis introduced in this paper reveals that the power consumption remains high or increases for a few seconds during a migration instance. This behavior contradicts the aim of power-capping, and may endanger the stability of servers. Based on this observation, we also propose and evaluate two power-suppressing live-migration schemes to resolve the power overshooting issue. Our evaluation shows that both approaches immediately limit the power consumption after live-migration is initiated.

J. Jeong · H. Kim
Department of Computer Science, Korea Advanced Institute of Science and Technology, Daejeon, Korea

J. Jeong
e-mail: jinkyu@calab.kaist.ac.kr

H. Kim
e-mail: hjukim@calab.kaist.ac.kr

S.-H. Kim · J. Lee · E. Seo (✉)
College of ICE, Sungkyunkwan University, Suwon, Korea
e-mail: euiseong@skku.edu

S.-H. Kim
e-mail: shkim@csl.skku.edu

J. Lee
e-mail: joonwon@skku.edu

**Keywords** Virtualization · Live-migration · Power-aware computing · Cloud computing · Distributed systems

## 1 Introduction

The power provisioning infrastructure in a data center is designed to endure the possible peak power requirement, which is determined by the sum of face-plate peak power values of all equipment inside the data center. For example, when twenty 500-Watt servers are connected to a power distribution unit (PDU), the PDU must be designed to provide more than 10 KWatts. However, the actual peak power of servers and other facilities is significantly lower than their face-plate numbers, and the average power consumption of servers is even lower than their actual peak power.

The power supply infrastructure comprises a large portion of the data center architecting cost. Consequently, the more underutilized a facility, the more expensive it becomes in terms of the total cost of ownership. For example, if a facility operates at 85 % of its peak capacity on average, then the cost of building the facility will still be higher than all electricity expenses for ten years of operation [9].

Most existing research efforts to improve server energy efficiency have been aimed at reducing variable expenses such as electricity cost. Thus, there has been little benefit with regard to the infrastructure architecting cost because the size of the power provisioning infrastructure is determined by the expected peak power, not by the actual average power consumption. Consequently, considering the building and managing cost of the data center power infrastructure, it would be desirable to accurately predict the actual peak power or to suppress the peak power below a predefined value so that the average utilization of the power infrastructure remains high.

If the peak power of servers is controllable, then data centers can adopt the oversubscription scheme, that is, the sum of the peak power of individual servers may be greater than the provisioned capacity [28]. Because a large portion of cutting-edge enterprise systems including cloud platforms are employing virtualization technology to secure flexibility in resource management, a few power-capping schemes, which suppress the power consumption of servers, racks, or PDUs (power distribution units) below a predefined value, have been introduced so far for the virtualized environment [4, 12].

Since virtualization enables fine-grained control of resource provisioning to virtual machines (VMs), controlling the power consumption of a server that runs VMs through a virtualization platform can be easily achieved. In addition, virtualization enables the live-migration technique, which migrates a VM running in a server to other servers if necessary. Combined with the oversubscription scheme, live-migration can effectively suppress the power consumption of a server while preserving availability by exporting VMs running in a server to other servers located in other racks or connected to other PDUs when the power consumption of the server exceeds the allowable threshold. Due to these benefits, many researchers believe that adopting virtualization will reduce the architecting cost of the power-supplying infrastructure in data centers [4, 9, 12, 24].

In many studies of power-capping in the virtualized environment, live-migration is commonly thought as a last resort to suppress power consumption. In such power-capping schemes, exporting power-hungry VMs from a server begins when the power consumption of the server hits the trip point, which is determined as the maximum allowable power consumption level, after all other possible means result in failure. Live-migration for power-capping is built upon a blindly optimistic assumption that exporting VMs from a server will instantly reduce the power consumption. However, live-migration is usually an expensive task in terms of processing overhead. It takes a few seconds to a few minutes to transmit the current VM state and its memory content to the destination node. Because these operations also consume additional power, the power consumption may even increase during the live-migration process. Although this additional power consumption due to live-migration may incapacitate the power-capping efficacy, to the best of our knowledge, no research about this issue has been conducted so far.

In this paper, we analyze the additional power consumption and performance overhead caused by live-migration through a series of experiments using profiling software and digital power meters. Because the amount and duration of the additional power consumption strongly depend on the migration schemes and characteristics of virtual machines, we investigate them under both pre-copy and post-copy live-migration schemes [15], which are two representative live-migration approaches. While the pre-copy scheme is employed by most virtualization solutions [3, 8, 33], the post-copy scheme is rarely used. Therefore, we implemented the post-copy live-migration scheme in the Xen virtualization platform based on an existing research prototype [23].

We take a close look at the live-migration operations from two view points. First, we investigate the power consumption changes during live-migration. A decision to export a VM is usually made when the power consumption of a server reaches its allowable ceiling. Therefore, an immediate power consumption drop is the desirable outcome from live-migration. However, our research reveals that the power consumption significantly rises during live-migration if uncontrolled. Albeit the live-migration is an infrequent event (60 attempts per 4 hours [39]), this characteristic may lead to catastrophic results in power-capped data centers [4]. Second, we analyze the performance degradation of the migrating VM as well as its colocated VMs due to the migration overhead. Although a VM can migrate anywhere according to power management policies, it and its colocated VMs must meet certain performance requirements or service-level agreements (SLAs).

In addition to this analysis, this paper also proposes two power-suppressing live-migration schemes that limit the additional power consumption during live-migration. The first approach restricts the scheduling of VMs related to live-migration. The other one lowers the clock speed of the server by using dynamic voltage and frequency scaling (DVFS) during live-migration. The proposed schemes are implemented in Xen and evaluated with the SPEC CPU2000 benchmark suite.

The rest of this paper is organized as follows: Section 2 explains the background knowledge and introduces related work about power-capping and live-migration. In Sect. 3, we explore the power consumption and performance overhead characteristics of both pre-copy and post-copy live-migration schemes with various workload configurations. Section 4 presents two power-suppressing live-migration techniques that combine the pre-copy live-migration scheme with DVFS and scheduling techniques, respectively. It also introduces implications from this research for designing novel power-capping-aware live migration schemes in the future. Finally, Section 5 concludes our research.

## 2 Related work

### 2.1 Power-capping and virtualization

The energy cost for a data center is determined not only by the amount of energy usage, but also by the data center power infrastructure (DCPI). The DCPI of a data center is usually designed to be able to supply more power than the expected peak power requirement of its servers. In comparison to applying power management schemes that temporally limit power consumption, downsizing the DCPI by structurally reducing the designated peak power is more beneficial [35]. Therefore, it is highly desirable to minimize the margin between the actual peak power and the designated peak power by accurately predicting or controlling the actual peak power.

While a server's name plate power is usually determined by summing the peak power of its internal components, the actual peak power of a server cannot reach 80 % of the name plate power [31]. When the power demand exceeds the capacity of a PDU, the fuse of the PDU disconnects the incoming power line to protect the PDU [12]. Therefore, when the aggregated power demand of servers connected to a PDU exceeds the capacity of the PDU, the power supply to all of the servers will be immediately shut down. In order to prevent such a catastrophic situation, the power infrastructure of a data center is designed to withstand the sum of the servers' name plate power because the actual peak power of the servers is unknown at the design stage [9].

Fan et al. [9] analyzed the power consumption patterns according to the workload characteristics in real warehouse-scale data centers. At the data center level, the measured peak power was only about 60 % of the predicted peak power, which was obtained from the name plate powers of the servers inside. In addition, they found that the occurrence frequency of the peak power is very rare. According to this observation, significant oversubscription of servers is achievable through eliminating the small number of peak power occurrences, which will result in a dramatic decrease in the measured overall peak power. The authors also proposed approaches to limit peak power such as mixing heterogeneous workloads in a single power plane or employing power-capping with DVFS.

Many server vendors introduced hardware-based power-capping solutions that maintain the power consumption of a server below a predefined threshold by monitoring power consumption with integrated power meters and controlling system states with firmware or integrated micro controllers. Some of them are already being sold commercially [14, 19]. Some of these approaches instantly lower the clock speed of processors through DVFS when the power consumption hits the power ceiling. This will provide rapid power-capping ability at the hardware level [42].

Much research on software-based power-capping has been conducted to limit the power consumption of a server [4, 5, 7, 10, 12, 21, 26, 27].

Lefurgy et al. [26, 27] proposed a power-capping scheme that uses a control-loop to manage DVFS according to the system power consumption changes. Gandhi et al. [10] also employed a control-loop to limit the power consumption. They used only the maximum and minimum processor performance levels and enforced idle time. Because this forced idleness achieves a higher effective frequency for a given power constraint than existing techniques, their approach improved the mean response time of server workloads by three times under the same power cap.

Cochran et al. [7] proposed a control technique designed to make the optimal DVFS and thread packing control decisions in order to maximize performance within a power budget. Their approach widened the power constraint range by the combined use of thread packing and DVFS, and saved a significant amount of energy.

Because virtualization is one of the essential technologies in cloud systems [2], many researchers attempted to predict, estimate, and control the power consumption of a VM in a server that consolidates multiple VMs [4, 5, 12, 21].

Kansal et al. [21] devised a scheme called *Joule Meter* that accurately estimates the energy consumption of a virtual machine. Joule Meter monitors the activities of a VM and measures the power consumption of the server with the power meter, which is integrated into the power supplying unit (PSU). These data are used as inputs to the power consumption model to calculate the amount of energy consumption by an individual VM. Such prediction or estimation schemes enable server oversubscription by predicting or estimating the actual peak power of a server when it consolidates multiple heterogeneous VMs.

Choi et al. [4, 5] predicted the peak power of a server running a mixture of multiple VMs with heterogeneous workloads when the power consumption statistics of each VM is identified beforehand. Since the suggested approach precisely forecasts the power consumption characteristics of a server, it helps the match-making decisions between VMs and servers while maintaining the server power-caps.

Govindan et al. [12] obtained the actual power consumption profiles of diverse VM workloads, and were able to pack twice as many servers into a PDU using the collected information in comparison to the configuration based on the name plate powers. Also, exploiting the observation that the peak power occurrences are exceptional, they oversubscribed additional 20 % more servers by ignoring the top 10 % high power consumption values. These ignored occurrences are assumed to be removed with various power-capping schemes.

Wang et al. [41] used feed-back control loops to control the power consumption not only of a single server, but also of a group of servers. The proposed group power-capping algorithm simply distributes the power budget of a server group to its servers proportionally to their power consumption during the previous time interval. In this scheme, the more power a server used in the previous interval, the more power budget it would get in the next interval. In addition to this unfairness, their approach assumed the server group configuration is flat while the power supplying structure, and thus server group configurations, in a real data center are hierarchical.

Much research with two different objectives, to reduce the average power consumption and the peak power consumption, has been conducted, and many of these research efforts use DVFS or device sleep states on different scales such as node-, rack-, cluster- or data center-levels. If two or many such approaches are used together, the outcomes would be different than expected due to the interferences among them. In order to resolve this issue, Raghavendra et al. [34] categorized the existing approaches and proposed a scheme that can effectively utilize their combinations. This scheme also takes into consideration the energy savings from the consolidation of VMs.

Nathuji et al. [32] proposed a VM-aware power budgeting scheme that manages the power caps of servers belonging to the same PDU in order to maximize the performance within the given power budget for the server group by considering both the relationship between power and performance levels and characteristics of heterogeneous servers.

## 2.2 Live-migration and power-capping

The VM live-migration originated from process migration, which was actively researched in the 1980s and 1990s. Process migration, which migrates a running process from a system to other systems without termination, was attempted to achieve load-balancing or improve availability. However, most of process migration efforts ended up with failures because exporting dependency on underlying file systems, resources and peripheral devices together with the processes is extremely difficult to accomplish [6]. However, these research results became the foundations of VM live-migration in the 2000s.

A live-migration instance usually takes a few seconds to a few minutes to finish, which hinders its application to commercial server systems. Among all procedures for live-migration, memory content transmission takes the longest time and thus most affects the migration performance [6]. A desirable live-migration technique should rapidly finish migration instances while minimizing the throughput degradation of the migrated VMs. Two representative approaches for transferring memory contents, the pre-copy and post-copy migration schemes, were introduced first for process migration and later adopted for VM migration. Both approaches are illustrated in Fig. 1.

A stop-and-copy migration scheme transmits all memory contents before the migrated VM resumes its execution in the destination node [22, 36]. Since all memory contents are in the destination node at the time of resuming execution, there is

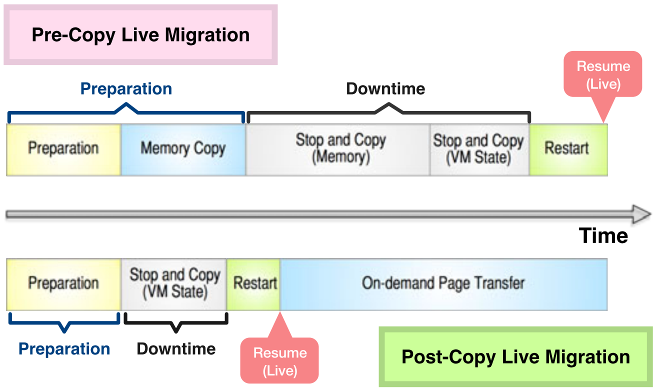


**Fig. 1** Time-series of migration activities in pre-copy and post-copy migration schemes

no throughput degradation after the VM resumes. However, because the VM to be migrated in the source node must suspend during the memory content transfer, the services provided by the VM have to be halted for that interval, which may take up to a few minutes [6].

To overcome this drawback, the pre-copy migration was suggested [6, 33]. This approach enables the VM in migration to suspend only for transferring a small number of modified pages after copying the entire memory region to the destination node beforehand. Accordingly, the VM seems to be alive during the migration. This scheme iteratively transfers modified pages to the destination node. Since the number of modified pages of a VM in a time interval is generally smaller than the number of pages transferring through network in that interval, after a few repetitive transfers of dirty pages, the number of remaining dirty pages becomes sufficiently small to suspend the VM and transfer the remaining dirty pages all at once. Although this approach significantly reduces the service interruption time during migration, it extends the time to finish a migration instance because a migration instance continues until the number of modified pages becomes the minimum threshold.

In the post-copy migration scheme, the hypervisor sends only the minimal and essential memory contents and information of a migrating VM to the destination and resumes the execution of the VM [15, 23]. A page fault occurs when the migrated VM accesses untransferred pages. In response to the page fault, the hypervisor in the destination node makes requests of the missing pages to the source node, and in turn the source node hypervisor transmits them to the destination. After securing the missing pages, the VM resumes its execution. The post-copy approach has significant benefit over its pre-copy counterpart. The post-copy scheme significantly reduces both the suspension time of the migrating VM and the latency between migration initiation and the context switch to the destination node. However, due to the expensive page fault handling operations, it may critically slow down the performance of the VM until all memory contents of the VM are transferred to the destination node through repetitive page faults. Hirofuchi et al. [16, 17] proposed a post-copy migration scheme with active pushing technique. By actively pushing the memory pages to the destination node after transmitting the context, the completion time of a migration instance becomes short and deterministic. Therefore, the active pushing post-copy migration has additional benefit over the pre-copy counterpart in terms of energy efficiency, especially when the migrating VM frequently performs write operations over sparse memory region.

In addition to these two representative schemes, Lie et al. [29] proposed the CR/TR-Motion (check-pointing/recovery and trace/replace). Once a live-migration instance is initiated, this scheme first transmits the minimal information about the migrating VM to the destination node and then periodically sends execution traces of the VM so that the destination node creates the target VM image and replays the execution traces in the created VM. When both VMs in the source and destination nodes synchronize their states, the migration finishes and the VM in the source node halts its execution. This scheme minimizes the service discontinuation due to live-migration. However, this scheme results in a heavy load on both source and destination nodes. In addition, when the performance of the source node is similar to or greater than the destination node, the migration overhead becomes unacceptably high.

Most virtualization platforms provide one of these VM live-migration schemes to improve the availability in system operation and flexibility in resource provision because live-migration enables dynamic load-balancing and nonstop servicing during sever shutdown. Also, live-migration can be used to reduce the power consumption. According to an empirical study by Sharifi et al. [38], approximately 25 % power savings can be obtained through changing the VM-to-server mappings by live-migration of consolidated VMs.

Specifically, many power-capping studies suggest the live-migration technique as the last resort for maintaining the power caps. Usually, power-capping schemes export power-hungry VMs to other nodes after trying all other avenues such as DVFS when power consumption hits the ceiling [5, 12, 32]. They assumed that the reduction in power consumption occurs right after releasing a VM. However, there have been no research efforts to identify whether live-migration is actually effective for power-capping.

The cases in which live-migration for power-capping occurs are mostly when power consumption unexpectedly rises beyond the threshold or when an abrupt change in power consumption occurs. In these cases, the power consumption must be lowered immediately. Thus, the live-migration schemes for power-capping must be able to reduce power consumption without delay. In addition, the performance impact on both the source and destination nodes as a result of live-migration should be minimized because a server usually runs multiple VMs for various services with independent SLAs.

However, a live-migration instance is a heavy task that transmits memory contents, which span over up to a few Gigabytes, to remote nodes in a short interval. Especially, the pre-copy migration scheme, which employs iterative transfers of modified pages, causes a significantly long migration time [6, 15, 29] during which notable additional power consumption occurs. However, this additional power consumption completely contradicts the purpose of live-migration for power-capping, which is immediate power consumption reduction. Voorsluys et al. [40] examined the performance impact of the VM live-migration. Lefèvre and Orgerie [25] measured power consumption change due to live-migration with a simple experiment. Huang et al. [18] examined the power consumption of live migration with different CPU utilization levels. There has been, however, no power-performance-correlated analysis of live-migration. Also, no live-migration schemes dedicated to power-capping have not been proposed yet to the best of our knowledge.

## 3 Analysis of live-migration

This section analyzes the power consumption and performance overhead of the post-copy and pre-copy live-migration approaches.

### 3.1 Experiment environment

The experimental system used for our analysis was configured with multiple Linux VMs running on the Xen hypervisor for the x86 architecture. The specifications of

**Table 1** Specifications of experimental system I

| Processor configuration (Intel Core i7-920) | | | |
|---|---|---|---|
| Num. of cores | 4 | L2 cache | 256 KB per core |
| Clock freq. | 2.67 GHz | L3 cache | 8 MB per processor |
| ISA | x86_64 | TDP | 130 W |
| 9 DVFS states | | Hyperthreading disabled | |
| **System configuration** | | | |
| Main memory | 4 GB | GPU | Nvidia 6600GT |
| Hypervisor | Xen 3.4.0 | Guest kernel | Linux 2.6.18.8 |

the experimental system are listed in Table 1. Although this system is equipped with a graphic processing unit (GPU), we believe that the GPU hardly affects the power consumption of the system because all VMs were installed to use the console mode, not the GUI mode. Thus, most of the dynamic power consumption is supposedly caused by processors, network interface cards (NICs) and disks. We set up two servers; one for the source node and the other for the destination node. These servers were directly connected to each other through 1 Gbps ethernet. All VM workloads including the VM to migrate run in the source node. The destination node has no active VMs and was set to be awaiting migration requests.

In order to analyze the power consumption changes according to the internal activities caused by migration, both power consumption and processor utilization must be measured concurrently. Power consumption was measured with a digital multimeter equipped with a data acquisition unit (DAQ) by tapping the A/C power line to the server and attaching the digital multimeter to it. We used a Yokogawa Digital Power Meter WT210, which can sample every 20 μs and generate an aggregated value every 100 ms. We used *xentop*, which is provided by the Xen package to display real-time information about VMs, to keep track of processor utilization changes of each VM in the system.

The pre-copy migration scheme is already implemented and deployed in Xen while the post-copy approach is not included by default. Therefore, we adopted the post-copy migration implementation for Xen from the Snowflock project [23] in our experimental systems.

Each VM used in our experiments is configured to be equipped with a virtual processor and 600 MB of memory. Since the experimental system has four physical cores, it can run four VMs concurrently. Thus, we ran up to four VMs at the same time in our experiments. The VM to migrate was set to place its root file system in a remote file server through the network file system (NFS). The root file systems of the other VMs were placed in the local hard disks of the experimental server.

Each VM ran one of four benchmark workloads, *twolf, bzip2, parser*, and *gap*, chosen from the SPEC CPU2000 benchmark suite. The four benchmark workloads were chosen based on study conducted by Clark et al. [6] so that they represent diverse memory write intensity characteristics. *twolf* is the least memory-intensive; its working set size is less than 2 MB most of the time [13]. *gap* is the most memory intensive workload and has large memory footprint (192 MB). It is especially write intensive [6, 13]. The other two workloads are less memory intensive than *gap*, but more than *twolf*.

Under both post-copy and pre-copy schemes, domain 0 handles most of the migration activities such as initiating migration procedures, managing required data structures and transferring memory contents. Consequently, the processor utilization due to domain 0 vastly increases during live-migration. We formed two workload groups, *CPU-saturated workload group* and *CPU-unsaturated workload group*, with the four benchmark programs to reflect the case in which the additional processor usage of domain 0 affects the performance of other VMs and the opposite case in which the additional processor utilization does not influence the execution of other VMs, respectively.

The CPU-saturated workload group consists of four concurrently running VMs and each of them runs one of the four benchmark programs. Because all processing cores of the system are continuously used by these four VMs, the additional processing load of domain 0 due to live-migration is expected to degrade the performance of the VMs in this group. However, the power consumption increases due to live-migration is expected to be insignificant because the processor would run at its full performance already before the migration instance begins.

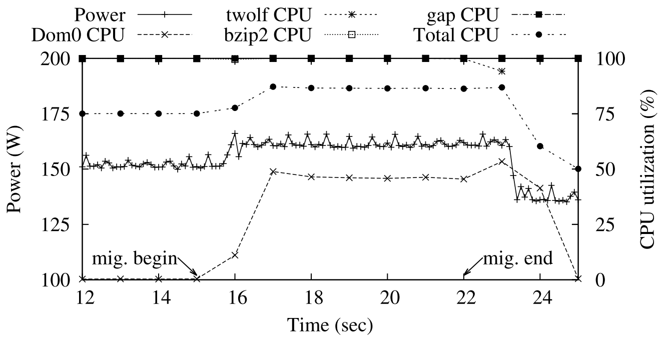
The CPU-unsaturated workload group comprises three concurrently running VMs. The VMs in this group run only *gap, bzip2 and twolf*, excluding *parser*. When running this group, domain 0 runs on a surplus core. Therefore, the additional processor load of domain 0 due to live-migration will not affect the performance of the other VMs. However, the system power consumption is expected to significantly change according to the processor utilization of domain 0.

Neither post-copy nor pre-copy are expected to decrease the power consumption when the number of VMs running after migration is considerably larger than the number of processor cores and all of the VMs are exhaustively using the processing resources. Therefore, live-migration for reducing power consumption should be initiated only when exporting VMs is expected to yield idle cycles. We focused only on such cases in our research.

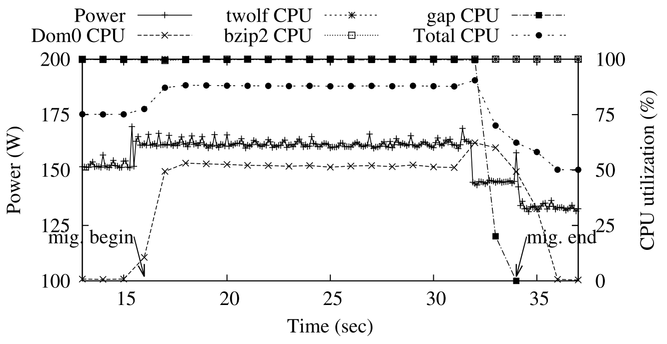
## 3.2 Pre-copy migration

Figure 2 shows the experiment results with the CPU-unsaturated workload group. Figure 2(a) is the result obtained when exporting *twolf*, and Fig. 2(b) is the result from exporting *gap*. The memory footprints of *twolf* and *gap* are approximately 3.5 MB and 200 MB, respectively. In both experiments, the migration instances were initiated 15 s after beginning the benchmark programs. The CPU utilization values in the graphs denote the per-core processor utilization rate. For example, 100 % reflects the case in which the VM dominates a core and runs continuously on it while 0 % means that the VM is idling.

The time to finish a migration instance vastly differs depending on the memory footprint of the migrating VM. The *twolf* VM took approximately 8 s to complete migration while the *gap* VM took 20 s. This difference stemmed primarily from the intensive memory access of the *gap* VM, which result in lots of the dirty page transfer iteration.



(a) Migrating twolf (CPU intensive)

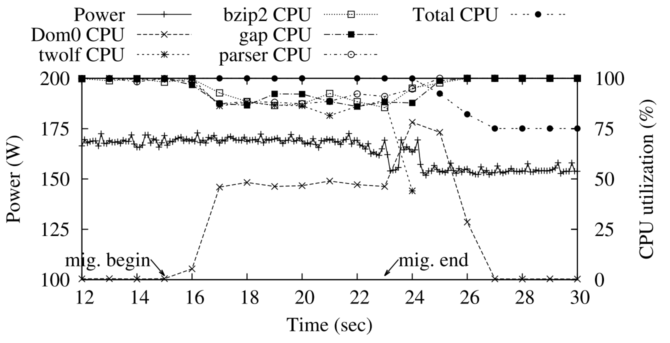


(b) Migrating gap (memory intensive)

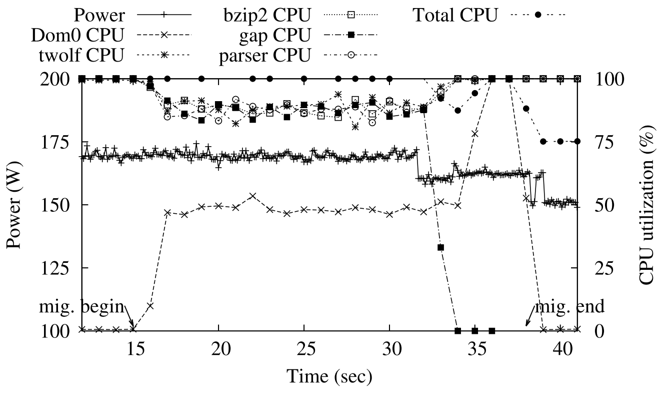
**Fig. 2** Power consumption and CPU utilization of pre-copy migration for the CPU-unsaturated workload (three VMs)

Domain 0 consumed approximately 50 % of the processor time during migration since the migration procedure executes in Domain 0 in Xen hypervisor. The main functions of the migration procedure are to map guest VM's memory pages into its own address space and invokes network send requests for each pages during the migration. Since domain 0 stalls frequently due to network I/O operations, it does not consume more processor time than that. This additional processor utilization resulted in total CPU utilization increase and additional power consumption. In the early phase of a migration instance, which is the first 2 s in our experiments, no actual data transfer occurs and only preparation for migration and negotiation with the destination node occurs. Consequently, the processor utilization and additional power consumption remain low in this interval. The processor utilization ramps up when the actual transfer begins. Following the increase in processor utilization, the power consumption of the system increases by roughly 12 W.

In this experiment, the throughput of the remaining VMs was not affected by the migration instance because domain 0 ran on a surplus core. Also, the migrating VM consumed almost 100 % of the processor time during migration. This observation indicates that the pre-copy migration scheme hardly affects the performance of the migrating VM and remaining VMs.



(a) Migrating twolf (CPU intensive)



(b) Migrating gap (memory intensive)

**Fig. 3** Power consumption and CPU utilization of pre-copy migration for the CPU-saturated workload (four VMs)

After the migration is complete and the migrated VM in the destination node resumes its execution, the original VM in the source node is supposed to stop its execution. However, the original VM used processor for 1 s after completing the migration for cleaning up and domain 0 actively ran for another 1 s after the termination of the original VM. Accordingly, the power consumption dropped not at the completion of migration, but at the termination of the original VM, and it dropped when domain 0 became idle. In our experiments, the power consumption finally decreased by 15 to 20 W in comparison to that before the migration began.

If this migration instance began to suppress the power consumption that were approaching the power cap, then the additional power consumption during the migration might drive the system into disaster. And the worse thing is that this threat can last for a significantly long time depending on the memory write accesses of the migrating VM. Therefore, the live migration instance for power-capping pushes the unsafe situation to an even more dangerous condition.

Figure 3 shows the experimental results for the CPU-saturated workload group. Right after the migration instances began, the CPU utilization of all VMs decreased. The processor utilization of domain 0 was slightly lower than that in the CPU-unsaturated workload group because domain 0 competes with other VMs for the processor resource. However, the time to finish migration of *twolf* or *gap* was not noticeably different in comparison to the migration time of the CPU-unsaturated workload group.

Because the processor had been fully utilized before the migration began, there was no additional power consumption induced by the migration instance [25]. The power consumption of the system decreased after the migration and cleaning up of the exported VM, and the extent of the power decrease was similar to that of the CPU-unsaturated workload group counterparts.

Although the pre-copy migration scheme did not bring significant additional power consumption, considering that the condition of live-migration is when the power consumption is close to the power cap and is threatening the reliability and safety of the system, we conclude that the pre-copy migration scheme is not adequate to impose power-capping due to its long delay in reducing power consumption.

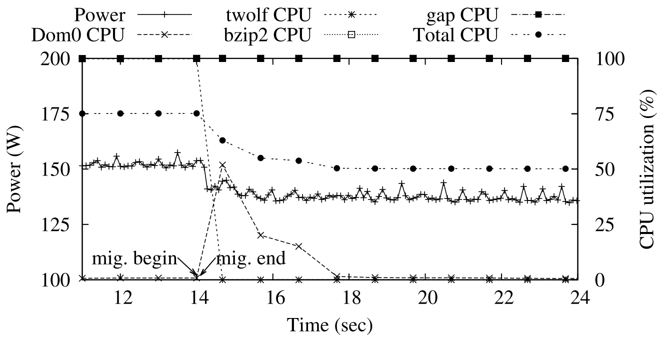
### 3.3 Post-copy migration

Since post-copy migration switches the context to the destination node right after transferring the essential information about the migrating VM, we expected that the power reduction would be remarkably faster than the pre-copy scheme. In the early stage of our research, based on this expectation, we believed that post-copy migration would be the appropriate candidate live-migration method for power-capping. In order to verify our expectation, we analyzed the power consumption and processor utilization changes simultaneously during post-copy migration like we did with pre-copy migration.

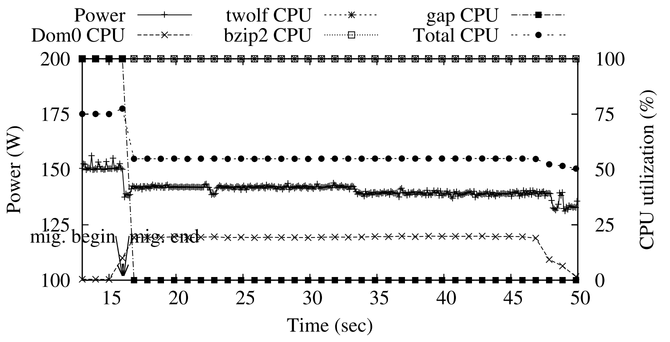
Figure 4 shows the time series of power consumption and CPU utilization during post-copy migration experiments with the CPU-unsaturated workload group. Regardless of whether the migrated VM was CPU-intensive or memory-intensive, the power consumption dramatically dropped approximately 0.12 s after the initiation of migration. At the point of the power consumption drop, the context was transferred to the VM in the destination node. However, we observed that domain 0 in the source node continuously utilized the CPU to deal with the page faults that occurred in the remote node.

In terms of performance overhead, the post-copy migration is definitely worse than pre-copy. The execution time of *twolf* was prolonged by 10 % by post-copy migration in comparison to pre-copy. The performance overhead worsened for the memory-intensive workload. The execution time for *gap* was 68.1 % longer with post-copy migration compared to pre-copy. Although post-copy migration immediately decreased the power consumption for the CPU-unsaturated workload group, the post-copy migration instances tend to last longer than the pre-copy counter parts. This property usually brings performance decrease to the remaining VMs. We will further investigate this issue in Sect. 4.

We found similar behavior in the experiments with the CPU-saturated workload group. After a migration instance began, the power consumption promptly dropped as shown in Fig. 5. The performance was similarly poor. Based on these observations, we conclude that post-copy migration quickly reduces power consumption regardless of whether or not processors are fully utilized.



(a) Migrating twolf (CPU intensive)



(b) Migrating gap (memory intensive)

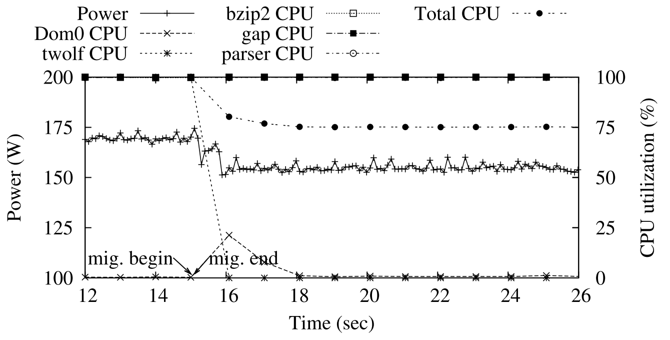
**Fig. 4** Power consumption and CPU utilization of post-copy migration for the CPU-unsaturated workload (three VMs)

## 3.4 Effects of underlying hardware

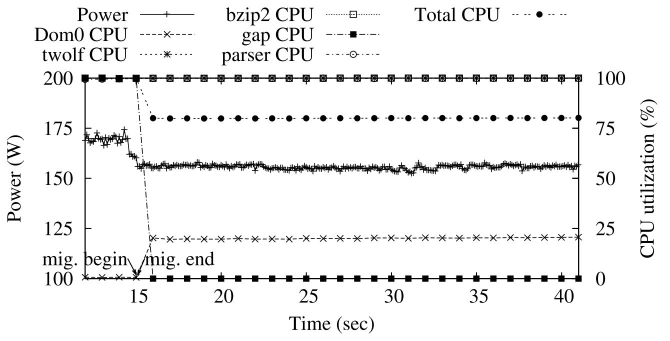
The overhead of VM live migration schemes can vary on the characteristics of the underlying hardware, and it may be also affected by the hypervisor. Accordingly, it is important to analyze the performance and power consumption characteristics on different configurations. We analyzed the power consumption and performance overhead of both post-copy and pre-copy live-migration schemes on KVM (kernel-based virtual machine) hypervisor running in a different hardware configuration. Table 2 shows the configuration of the server system used in the following experiments.

The pre-copy migration scheme is already implemented in KVM hypervisor while the post-copy scheme is not included by default. We applied a post-copy migration implementation for KVM, which is being developed by H. Takahiro et al. [16]. We note that the post-copy migration scheme used in this evaluation adapts *active pushing* unlinke the post-copy implementation of the Xen system. The active pushing actively sends (or pushes) a migrating VM's memory pages that are not transferred to the destination yet.

The primary advantage of this approach is network page fault reduction through actively prefetching the memory pages from the source node. The cost in return for this benefit is a large amount of intensive network transmission until all memory



(a) Migrating twolf (CPU intensive)



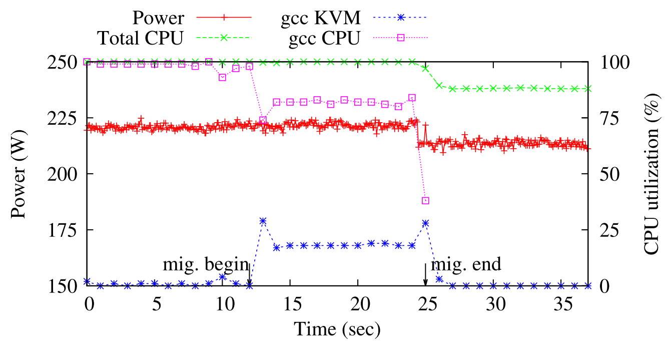
(b) Migrating gap (memory intensive)

**Fig. 5** Power consumption and CPU utilization of post-copy migration in CPU-saturated workload (4 VMs)

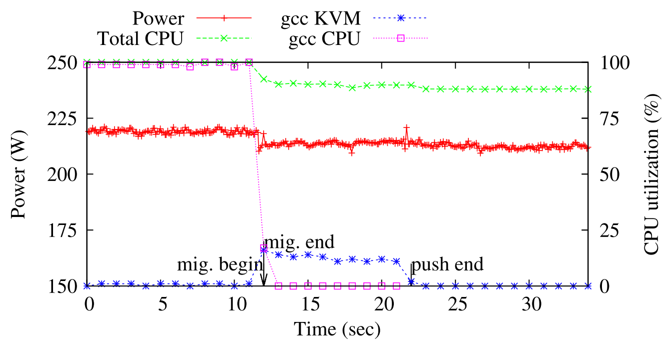
**Table 2** Specifications of experimental system II

| Processor configuration (two Intel Xeon E5620) | | | |
|---|---|---|---|
| Num. of cores | 4 | L2 cache | 256 KB per core |
| Clock freq. | 2.4 GHz | L3 cache | 12 MB per processor |
| ISA | x86_64 | TDP | 80 W |
| 7 DVFS states | | Hyperthreading disabled | |
| **System configuration (Dell PowerEdge R410)** | | | |
| Main memory | 16 GB | GPU | MGA G200eW |
| Hypervisor | Linux KVM 3.2.14-rc1 (QEMU 1.0.91) | Guest kernel | Linux 3.2.14 |

pages of the migrating VM are sent to the destination; this entails additional power consumption due to CPU computation for the network transfer in the source host. Both migration schemes for KVM provide the feature that limits the network bandwidth of the migrating VM.



(a) Pre-copy migration



(b) Post-copy migration

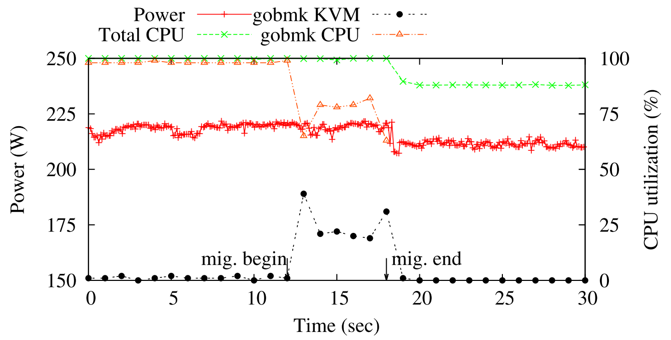
**Fig. 6** Power consumption and CPU utilization when migrating gcc (memory intensive) in CPU-saturated workload (8 VMs)

Each VM used in these experiments is configured to be equipped with a single virtual processor and 1 GB of memory. Since the system has eight physical cores, we varied the number of VMs from six to eight. Similarly to the previous experiments, each VM has NFS-based root file system to enable VM live-migration without storage migration.

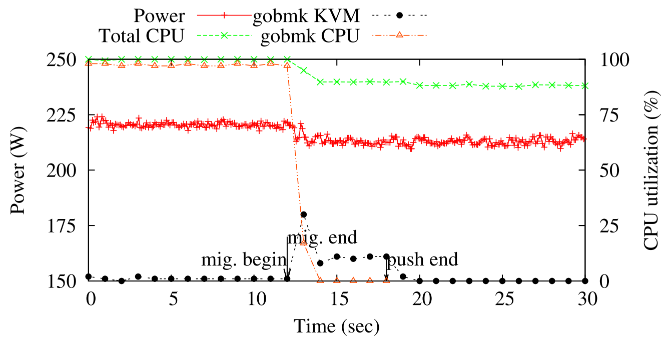
Each VM ran one of eight benchmark workloads, *gcc, gobmk, libquantum, sjeng, hmmer, bzip2, perbench,* and *perlbench*, which are chosen from the SPEC CPU2006 benchmark suite.

While multiple VMs (from six to eight) run concurrently, we migrated one of two VMs each of which runs *gcc and gobmk*. The gcc workload is the most memory-intensive one in SPEC CPU2006 integer applications while the gobmk workload is the most CPU-intensive [11]. Since VM live-migration schemes show different results depending on the memory access intensity, the two workloads are chosen as the targets of live-migration.

Figure 6 shows the power consumption and CPU consumption of the source node during the migration of the gcc workload. In this experiment, the CPU utilization was saturated by eight concurrently running VMs (including the gcc VM) from the beginning. Recall that additional power consumption of the pre-copy scheme is minimal



(a) Pre-copy migration



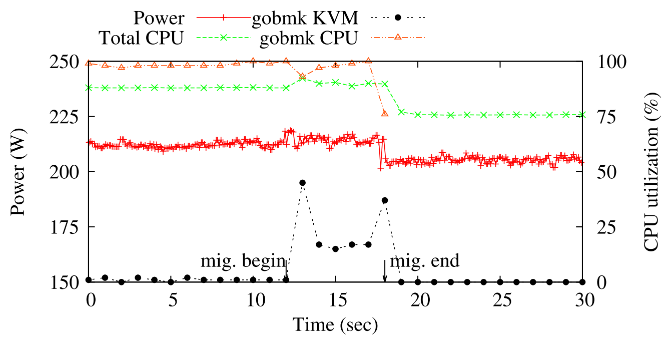
(b) Post-copy migration

**Fig. 7** Power consumption and CPU utilization when migrating gobmk (CPU intensive) in CPU-saturated workload (8 VMs)

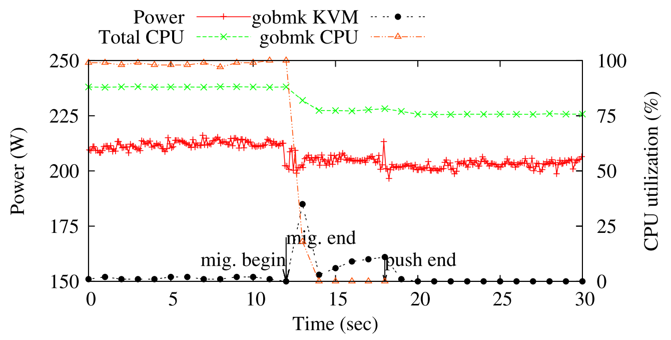
when the CPU is saturated in the PC-based experiments in Sect. 3.2. Similar behavior was observed as shown in Fig. 6(a). Until the migration ended the power consumption was not decreased because the CPU context of the migrating VM was running in the source host till the end of the migration.

When using the post-copy scheme, immediate power consumption decrease happened right after migration initiated. As shown in Fig. 6(b), the power consumption was immediately dropped by about six watts due to the migration. Since the post-copy scheme used in this experiment adapts active pushing, the background transfer of memory pages slightly increased the power consumption; The line *KVM* shows the KVM backend's CPU consumption including that for hardware virtualization, and VM migration, which is dominant in this experiment.

Figure 7 shows the experiment results similar to the one illustrated in Fig. 6 except that the workload of the migrating VM is gobmk (CPU-intensive), not gcc (memory-intensive). The memory access intensity significantly affects the total migration time under the pre-copy scheme. The total migration time of gobmk was 6 seconds whereas that of gcc was 15 seconds under the pre-copy scheme. Except the total migration time, power consumption behaviors of both experiments were similar. The post-copy scheme immediately dropped power consumption at the point of the



(a) Pre-copy migration



(b) Post-copy migration

**Fig. 8** Power consumption and CPU utilization when migrating gobmk (CPU intensive) in CPU-unsaturated workload (7 VMs)

migration initiation. The pre-copy scheme, however, did not show obtrusive power drop until the migration ended.

Under the CPU-unsaturated condition, the pre-copy scheme showed power consumption increase, which is consistent to the results in Sect. 3.2. Figure 8 shows the power consumption and CPU utilization while the gobmk workload was migrating with available CPU utilization. In Fig. 8(a), when migration begins, the KVM hypervisor for the VM in migration showed CPU utilization increase. This was the dominant source of causing increased power consumption of the pre-copy migration scheme. Note that until the migration ended, all seven VMs ran in the source host. Accordingly, additional computations for mapping a migrating VM's pages and sending them to the destination node consumed additional CPU time and power.

When the post-copy scheme was used, the power consumption instantaneously dropped by the migration since immediate CPU context transfer to a destination node decreased the number of CPU contexts in the source host as we have already identified. In Fig. 8(b), the power consumption decreased by approximately 8 watts. While active pushing was performing (between the arrows *mig. end* and *push end*), the CPU consumption of KVM slightly increased until all memory pages were transferred (the arrow *push end*). When the background transfers were completed, the

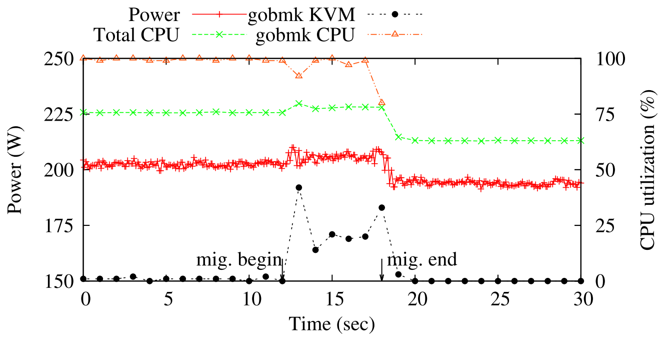


**Fig. 9** Power consumption and CPU utilization when migrating gobmk (CPU intensive) using pre-copy scheme

additional power consumption for the active pushing was also decreased by approximately 3 watts.

In order to examine the power consumption effect of the pre-copy migration under different CPU utilization levels, we varied the number of concurrently running VMs and conducted similar experiments. Figure 9 shows the power consumption during the migration of the gobmk workload when the number of running VMs is six and, thus, the CPU utilization level is lower than that of the experiment in Fig. 8(a).

When the CPU utilization level was low (about 75 % of CPU utilization), the power consumption of the source host was also low. The pre-copy scheme consumed additional power by about 5 watts during the migration. When the migration finishes, the source host's power was dropped by approximately 10 watts.

In comparison with Fig. 8(a), different CPU utilization level did not seriously change the power consumption behavior of the pre-copy migration scheme. Hence, the additional CPU context for the pre-copy migration was expected to be the main contributor of the power consumption increase.

## 4 Live-migration schemes for power-capping

### 4.1 Adaptation of existing schemes

Our analysis showed that the pre-copy live-migration scheme, which is being widely used in commercial systems, temporarily requires heavy processor utilization and, in turn, maintains or increases power consumption for a short time. On the contrary, while the post-copy live-migration scheme performs successfully at reducing power consumption through the instant context exportation, its impact on the throughput of the migrating VM is unacceptably large due to the expensive page fault handling cost.

The desired traits of the live-migration schemes for power-capping are instant reduction in power consumption and minimal impact on performance. On the contrary to the many existing research efforts that suggest live-migration as the last resort for power-capping, our analysis showed that neither the pre-copy nor the post-copy migration scheme is appropriate for that purpose. Consequently, we should devise a novel approach to the live-migration scheme for power-capping.

The performance degradation under post-copy migration is mainly caused by the page fault handling mechanism that induces frequent networking between the source and destination nodes. This drawback is inherent in the post-copy migration scheme and not easily avoidable. Therefore, before researching the novel power-capping-aware live-migration scheme from the scratch, we propose two workarounds for the pre-copy live-migration scheme in this section to limit its additional power consumption.

The first approach is to lower the processor clock speed at the time of migration initiation. The decreased power consumption due to the lowered clock speed is expected to offset the additional power consumption from migration activities. Some processor architectures do not allow cores integrated in the same processor to operate at different clock speeds [37]. Even though all cores in the same processor are allowed to operate at different clock speeds, they have to share the same voltage input. Thus, giving different clock speeds to cores leads to energy inefficiency, making all cores in a processor generally work at the same clock speed. Therefore, lowering the clock speed may affect the performance of other VMs in the source node. This approach itself cannot be a long-term solution of power capping. If lowering the clock speed continues, the hosted VMs may suffer from the performance degradation.

The second approach is to set the *CAP* value of the migrating VM. Under the *Credit scheduler*, which is the default VM scheduler in Xen, a *CAP* value is given to each VM. The *CAP* value of a VM determines the amount of maximum processor time allowed to the VM in a time interval [1]. The *CAP* of a VM is set to a value between 0 to 100, and the default value is 0, which means that the VM is allowed to use unlimited processor time. *CAP* values between 1 and 100 denote the limitation of CPU allocation for the VM in a percentage of processing time of a processing unit, or a core. Restrictive processor time provisioning schemes such as the *CAP* value are employed by most VM schedulers including the Credit scheduler. By setting the CAP value of the migrating VM not to consume additional processor time when it initiates the migration instance, this approach limits the additional power consumption. Jin et al. [20] claimed that the time to finish a pre-copy live-migration instance, and thus its resource usage, are nondeterministic. To resolve this flaw, similarly to our approach, they set the *CAP* value to control both the resource usage rate for migration instances and the time to finish them.

In Sect. 3, we revealed that the additional power consumption was caused by domain 0 for the memory content transfer. Therefore, setting the CAP value of domain 0 is desirable for suppressing the additional power consumption. However, because domain 0 not only handles migration, but also deals with I/O requests on behalf of other VMs, restricting the scheduling time of domain 0 may harm the performance of the overall system and the scheduling fairness among VMs. Thus, our approach sets the CAP of the migrating VM to a sufficiently small value so that the shortened scheduling time of the migrating VM compensates for the increased scheduling time of domain 0.

Generally, as shown in Fig. 10, the smaller the CAP of the migrating VM is, the more the processor utilization of other VMs including domain 0 use when the number of running VMs exceeds the number of available cores. However, determining the proper CAP value that is fair for the migrating VM as well as the other VMs is

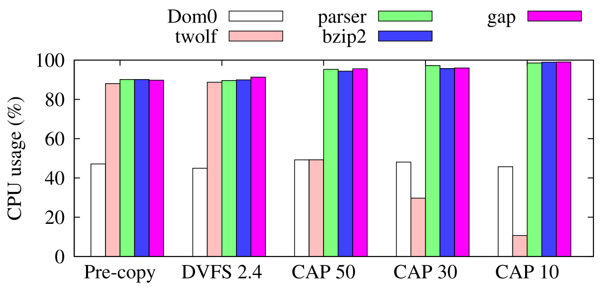
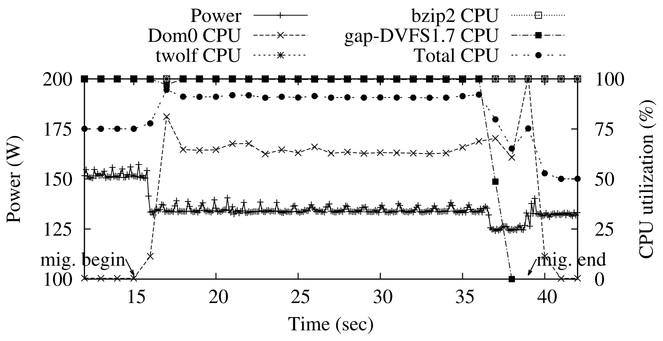
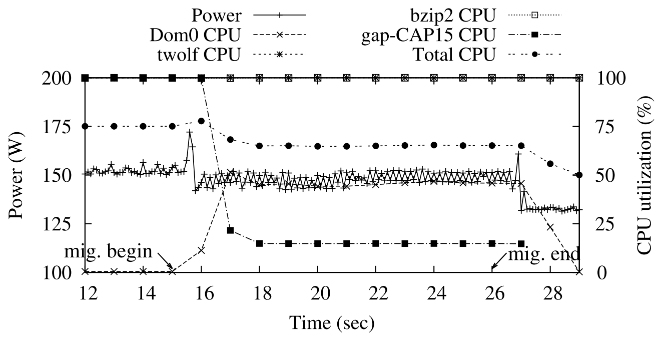


**Fig. 10** CPU usage of four VMs and domain 0 during pre-copy migration with different CAP values of the migrating VM (*twolf*)



(a) DVFS to 1.7GHz

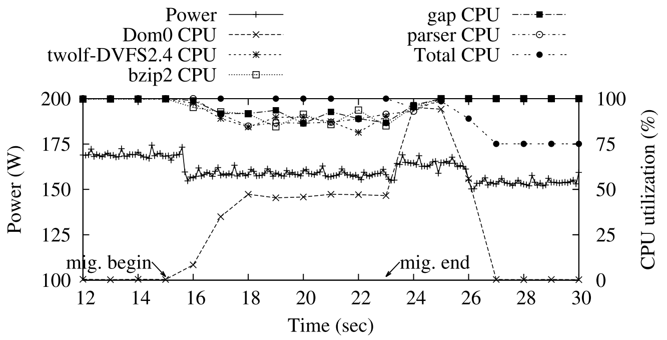


(b) CAP value to 15

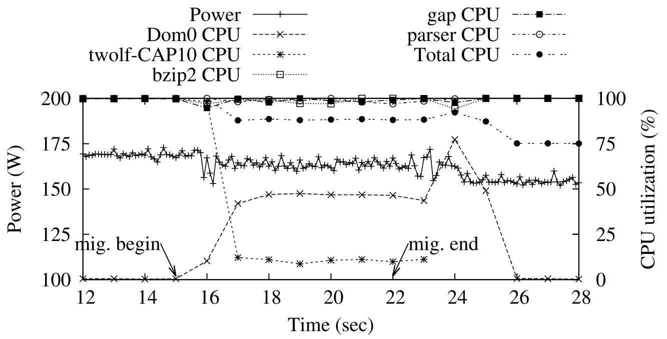
**Fig. 11** Power consumption and CPU utilization during migration of *gap* for the CPU-unsaturated workload (three VMs) under DVFS and CAP, respectively

difficult because obtaining the optimal CAP value requires an accurate scheduling time estimation model considering many parameters such as processor performance, memory access speed and so on. That is beyond the scope of our research. Thus, we used arbitrarily chosen CAP values in our experiments.

Figures 11(a) and 11(b) show the power consumption and CPU utilization changes when the processor clock speed was dropped to 1.7 GHz, which is the slowest available clock speed and approximately 64 % of the fastest, during migration and when the CAP of the migrating VM was set to 15, respectively. We used the CPU-unsaturated workload group for these experiments. Different from Fig. 2, the power



(a) DVFS to 2.4 GHz



(b) CAP value to 10

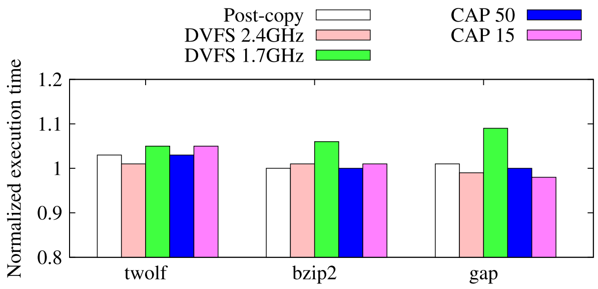
**Fig. 12** Power consumption and CPU utilization when migrating *twolf* for the CPU-saturated workload (four VMs)

consumption dropped immediately after the migration instances began under both approaches.

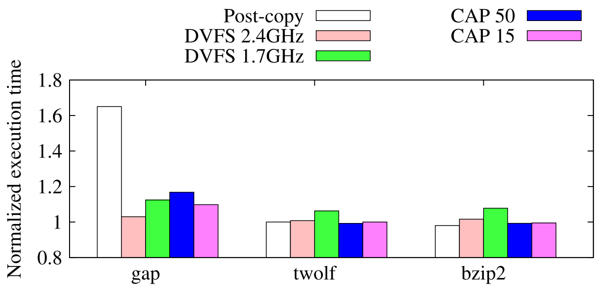
In terms of the effectiveness in reducing power consumption, the DVFS approach was significantly better than the CAP approach. Even when the CAP value was set to 15, a significantly low number, the power consumption reduction was less than that of the DVFS approach. As shown in Fig. 11(b), the CAP approach lowered the power consumption to 148.3 Watts, which is approximately 3 % less than the original 151.6 Watts, while using DVFS dropped the power consumption by approximately 10 % to 135.6 Watts. This is because using CAP only degrades the performance of the migrating VM whereas the DVFS approach slows down all VMs in the system. This difference is vividly observed with the CPU-saturated workload group.

Figure 12 shows the experimental results with the CPU-saturated workload group when using DVFS and CAP, respectively. We set the target clock speed to 2.4 GHz when using DVFS, which is faster than that in the experiments shown in Fig. 11, and also lowered the CAP value to 10. However, even with the smaller CAP value, we could not obtain any significant power consumption reduction from the CAP approach. The higher the processor utilization is, the more the DVFS approach saves energy. On the contrary, the effectiveness of the CAP approach decreases when the number of CPU-bound VMs is larger than the number of cores. Obviously, this comes

**Fig. 13** Normalized execution time of the CPU-unsaturated workload with post-copy, DVFS with pre-copy, and CAP value with pre-copy



(a) Migrating *twolf* (CPU intensive)



(b) Migrating *gap* (memory intensive)

at a cost. Because lowering the clock speed affects the performance of all VMs, the CAP approach should be preferred to the DVFS approach if the power consumption reduction from setting the CAP value is acceptable. The rate limiting scheme [6] enables accurate control of resource usage by domain 0. Applying the rate limiting scheme thus will allow higher CAP values for the migrating VMs under the same power cap.

As mentioned before, using CAP or, especially, DVFS affects the performance not only of the migrating VM, but also of the colocated VMs in the source node. We measured the prolonged execution time of the migrating VM and remaining VMs in the CPU-unsaturated workload due to the use of CAP or DVFS. For comparison, we measured the prolonged execution time by the post-copy migration. In this experiment, we migrated *twolf* and *gap*, which represent workloads with small and large memory footprints, respectively.

The experimental results in Fig. 13 are normalized to the execution times without any migration instances. The results show that the post-copy migration did not significantly prolong the execution time of both migrating VM and remaining VMs when the memory footprint of the migrating VM is small. Except the case in which the clock speed was set at 1.7 GHz, neither DVFS nor using CAP notably affected the workload throughput when the migrating VM had small memory footprint. However, when a migrating VM is highly memory-intensive, the execution time of the migrating VM was heavily prolonged under the post-copy migration scheme as explained in Sect. 3.3. The CAP approach slowed down the execution of the migrating VM only, and thus the other VMs' throughput did not noticeably change or sometimes even improved. On the contrary, the DVFS approach harmed the throughput of all VMs in the source node.

## 4.2 Implications for a novel approach

During the analysis illustrated in Sect. 3, we obtained implications for designing a novel power-capping-aware live migration schemes.

First, the post-copy migration scheme immediately and spontaneously decreases the power consumption of a source node. This scheme, however, significantly impacts the performance of the migrating VM when the workload running in the VM is memory-intensive. On the contrary, CPU-intensive VMs gain marginal performance penalty from the post-copy migration scheme. Optimization techniques, such as active pushing [16] and prepaging [15] may remedy the performance penalty of memory-intensive workloads.

Second, the pre-copy migration scheme shows a trivial power increase when the source node is already CPU-saturated. Otherwise, additional computation costs for mapping VM pages and sending them to a destination node increase the CPU utilization and power consumption of the source node. Considering the purpose of migration, the power increase due to migration is an opposite outcome. In terms of the performance of the migrating VM, the pre-copy scheme is better than the post-copy counterpart. However, when migrating a memory-intensive workload, the increased power consumption remains during the entire migration process, which is usually longer than that under the post-copy scheme.

Third, either lowering the clock frequency of a source node or adjusting the CPU allocation to a migrating VM can be combined with the pre-copy migration scheme in order to offset the additional power consumption by the migration.

The power consumption by the pre-copy migration can be quantified by how much CPU the migration procedure is allocated; the network link power consumption is usually deemed negligible [30]. However, the CPU utilization of the migration procedure is related to the allowed network bandwidth for a live migration. Thus, network bandwidth of the migrating VM can be used as a handy tool for controlling the power consumption of the migration procedure since there are various tools and methods that can limit the network performance of a VM.

Figure 14 shows the power consumption and the CPU utilization of the system when an idle VM is being migrated with varying the network bandwidth limit. As shown in the figure, the more the network bandwidth is allocated for the migration, the higher the CPU utilization is. Since the migration procedure is highly network-dependent, when the migrating VM uses up the allowed bandwidth quantum in a given time, then the migration process suspends until a new quantum is allocated. Accordingly, the CPU utilization is mostly proportional to the allowed network bandwidth for migration.

In addition, the relationship between utilization level and power consumption of most modern processors is linear [9, 18]. Therefore, the power consumption of the migration procedure can be quantified from the allowed network bandwidth for migration. Then, lowering CPU clock speed or limiting the CPU allocation to a VM in migration can offset the additional power consumption. Although the relationship between CPU utilization and network bandwidth limit varies depending on the system

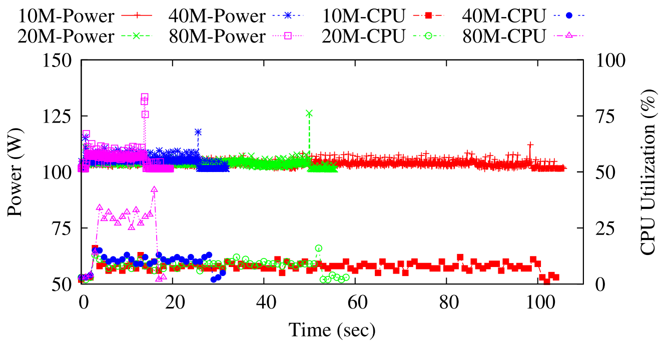


**Fig. 14** Power consumption and CPU utilization when migrating an idle VM with varying the allowed network bandwidth in KVM in the server system

configurations, it can be easily obtained through a series of experiments in advance to be used during the operation.

## 5 Conclusion and future work

Along with virtualization, server power-capping, which is expected to reduce the architecting cost of energy infrastructure in data centers, is drawing enormous attention from both academia and industry. Up until now, many power-capping schemes have been introduced and many of studies have employed live-migration as the last resort to maintain the power consumption of servers in cases of power overshooting.

This suggestion was built upon the belief that live-migration would immediately drop the power consumption and resolve the unsafe situation. However, in a series of experiments and analysis, we revealed that the current live-migration schemes bring significant additional power consumption to the source node or performance damage to the remaining VMs. This contradictory result demands a novel live-migration scheme that is suitable for the power-capping purpose.

In addition to this analysis, we proposed two workarounds for the existing pre-copy live-migration schemes that can instantly reduce the power consumption and minimize the performance impact on the remaining VMs. The two proposed schemes exploit DVFS and limited scheduling, respectively, to suppress power consumption and performance degradation. In our evaluation, even these simple and straightforward approaches successfully suppressed the power consumption increase due to live-migration, although they cannot accurately control or predict the power consumption changes and performance impact due to live-migration instances.

When VMs in a server show unpredictable power consumption characteristics or abruptly change their behavior, though lowering the clock speed of processors could be a temporary remedy, live-migration would fundamentally settle such cases by redistributing workloads over the server pool. Therefore, using live-migration in power-capping schemes is inevitable. However, the current power-capping schemes, even with our workarounds, are incomplete in terms of controlled power consumption suppression. Thus, we are devising a novel live-migration scheme from scratch that can precisely predict and control the power consumption and performance impact to fit the power-capping schemes.

**Acknowledgements** This research was supported by Next-Generation Information Computing Development Program (2012-0006423) and Basic Science Research Program (2012R1A1A2A10038823) through the National Research Foundation of Korea (NRF) funded by the Ministry of Education, Science, and Technology.

## References

1. Ackaouy E (2006) The Xen Credit CPU scheduler. In: Proceedings of 2006 Fall Xen Summit
2. Armbrust M, Fox A, Griffith R, Joseph AD, Katz RH, Konwinski A, Lee G, Patterson DA, Rabkin A, Stoica I, Zaharia M (2009) Above the clouds: a Berkeley view of cloud computing. Technical report UCB/EECS-2009-28, UC Berkeley
3. Barham P, Dragovic B, Fraser K, Hand S, Harris T, Ho A, Neugebauer R, Pratt I, Warfield A (2003) Xen and the art of virtualization. In: Proceedings of the nineteenth ACM symposium on operating systems principles
4. Choi J, Govindan S, Jeong J, Urgaonkar B, Sivasubramaniam A (2010) Power consumption prediction and power-aware packing in consolidated environments. IEEE Trans Comput 59:1640–1654
5. Choi J, Govindan S, Urgaonkar B, Sivasubramaniam A (2008) Profiling, prediction, and capping of power consumption in consolidated environments. In: IEEE international symposium on modeling, analysis and simulation of computers and telecommunication systems, MASCOTS 2008.
6. Clark C, Fraser K, Hand S, Hansen JG, Jul E, Limpach C, Pratt I, Warfield A (2005) Live migration of virtual machines. In: Proceedings of the 2nd conference on symposium on networked systems design & implementation, NSDI'05, Berkeley, CA, USA. USENIX Association, Berkeley, pp 273–286
7. Cochran R, Hankendi C, Coskun AK, Reda S (2011) Pack & cap: adaptive DVFS and thread packing under power caps. In: Proceedings of the 44th annual IEEE/ACM international symposium on microarchitecture
8. Das T, Padala P, Padmanabhan V, Ramjee R, Shin KG (2010) Litegreen: saving energy in networked desktops using virtualization. In: Proceedings of USENIX annual technical conference, Usenix ATC'10
9. Fan X, Weber W-D, Barroso LA (2007) Power provisioning for a warehouse-sized computer. In: Proceedings of the 34th annual international symposium on computer architecture, ISCA'07
10. Gandhi A, Harchol-Balter M, Das R, Kephart JO, Lefurgy C (2009) Power capping via forced idleness. In: Proceedings of workshop on energy-efficient design, WEED'09
11. Gove D (2007) Cpu2006 working set size. Comput Archit News 35(1):90–96
12. Govindan S, Choi J, Urgaonkar B, Sivasubramaniam A, Baldini A (2009) Statistical profiling-based techniques for effective power provisioning in data centers. In: Proceedings of the 4th ACM European conference on computer systems, EuroSys'09. ACM, New York, pp 317–330
13. Henning J (2001) SPEC CPU2000 memory footprint. http://www.spec.org/cpu2000/analysis/memory/
14. Hewlett-Packard Development Company, LP (2011) HP power capping and HP dynamic power capping for ProLiant servers, 2nd edn. Technology brief TC110107TB, Hewlett-Packard Development Company, LP
15. Hines MR, Gopalan K (2009) Post-copy based live virtual machine migration using adaptive prepaging and dynamic self-ballooning. In: Proceedings of the 2009 ACM SIGPLAN/SIGOPS international conference on virtual execution environments, VEE'09. ACM, New York, pp 51–60
16. Hirofuchi T, Nakada H, Itoh S, Sekiguchi S (2010) Enabling instantaneous relocation of virtual machines with a lightweight vmm extension. In: 10th IEEE/ACM international conference on cluster, cloud and grid computing, CCGrid'10, pp 73–83
17. Hirofuchi T, Nakada H, Itoh S, Sekiguchi S (2011) Making VM consolidation more energy-efficient by postcopy live migration. In: Proceedings of the second international conference on cloud computing, GRIDs, and virtualization
18. Huang Q, Gao F, Wang R, Qi Z (2011) Power consumption of virtual machine live migration in clouds. In: Third international conference on communications and mobile computing, CMC'11, pp 122–125

19. Jenkins D (2009) Intel intelligent power node manager. White paper 322705-002US, Intel Corporation
20. Jin H, Gao W, Wu S, Shi X, Wu X, Zhou F (2011) Optimizing the live migration of virtual machine by CPU scheduling. J Netw Comput Appl 34(4):1088–1096
21. Kansal A, Zhao F, Liu J, Kothari N, Bhattacharya AA (2010) Virtual machine power metering and provisioning. In: Proceedings of the 1st ACM symposium on cloud computing
22. Kozuch M, Satyanarayanan M (2002) Internet suspend/resume. In: Proceedings of the fourth IEEE workshop on mobile computing systems and applications
23. Lagar-Cavilla HA, Whitney JA, Scannell AM, Patchin P, Rumble SM, de Lara E, Brudno M, Satyanarayanan M (2009) Snowflock: rapid virtual machine cloning for cloud computing. In: Proceedings of the 4th ACM European conference on computer systems
24. Lee Y, Zomaya A (2012) Energy efficient utilization of resources in cloud computing systems. J Supercomput 60:268–280
25. Lefèvre L, Orgerie A-C (2010) Designing and evaluating an energy efficient cloud. J Supercomput 51:352–373
26. Lefurgy C, Wang X, Allen-Ware M (2007) Server-level power control. In: Proceedings of the fourth international conference on autonomic computing
27. Lefurgy C, Wang X, Allen-Ware M (2008) Power capping: a prelude to power shifting. Clust Comput 11:183–195
28. Lim H, Kansal A, Liu J (2011) Power budgeting for virtualized data centers. In: Proceedings of the 2011 USENIX conference on USENIX annual technical conference, USENIXATC'11. USENIX Association, Berkeley, p 5
29. Liu H, Jin H, Liao X, Hu L, Yu C (2009) Live migration of virtual machine based on full system trace and replay. In: Proceedings of the 18th ACM international symposium on high performance distributed computing
30. Liu H, Xu C-Z, Jin H, Gong J, Liao X (2011) Performance and energy modeling for live migration of virtual machines. In: Proceedings of the 20th international symposium on high performance distributed computing, HPDC'11. ACM, New York, pp 171–182
31. Mitchell-Jackson J, Koomey J, Nordman B, Blazek M (2003) Data center power requirements: measurements from silicon valley. Energy 28(8):837–850
32. Nathuji R, Schwan K (2008) VPM tokens: virtual machine-aware power budgeting in datacenters. In: Proceedings of the 17th international symposium on high performance distributed computing, HPDC'08, pp 119–128
33. Nelson M, Lim B-H, Hutchins G (2005) Fast transparent migration for virtual machines. In: Proceedings of Usenix annual technical conference, Usenix ATC'05
34. Raghavendra R, Ranganathan P, Talwar V, Wang Z, Zhu X (2008) No "power" struggles: coordinated multi-level power management for the data center. In: Proceedings of the 13th international conference on architectural support for programming languages and operating systems, ASPLOS XIII. ACM, New York, pp 48–59
35. Rasmussen N (2010) Implementing energy efficient data centers. APC white paper, no 114, rev 1
36. Sapuntzakis CP, Chandra R, Pfaff B, Chow J, Lam MS, Rosenblum M (2002) Optimizing the migration of virtual computers. Oper Syst Rev 36:377–390
37. Seo E, Jeong J, Park S, Lee J (2008) Energy efficient scheduling of real-time tasks on multicore processors. IEEE Trans Parallel Distrib Syst 19:1540–1552
38. Sharifi M, Salimi H, Najafzadeh M (2012) Power-efficient distributed scheduling of virtual machines using workload-aware consolidation techniques. J Supercomput 61:46–66
39. Verma A, Kumar G, Koller R, Sen A (2011) Cosmig: modeling the impact of reconfiguration in a cloud. In: Proceedings of the 2011 IEEE 19th annual international symposium on modelling, analysis, and simulation of computer and telecommunication systems, MASCOTS'11. IEEE Comput Soc, Washington, pp 3–11
40. Voorsluys W, Broberg J, Venugopal S, Buyya R (2009) Cost of virtual machine live migration in clouds: a performance evaluation. In: Proceedings of the 1st international conference on cloud computing
41. Wang Z, Zhu X, McCarthy C, Ranganathan P, Talwar V (2008) Feedback control algorithms for power management of servers. In: Proceedings of the third international workshop on feedback control implementation and design in computing systems and networks
42. Ware M, Rajamani K, Floyd M, Brock B, Rubio JC, Rawson F, Carter JB (2010) Architecting for power management: the IBM® Power7™ approach. In: Proceedings of the 16th international symposium on high performance computer architecture