Energy Efficient Scheduling of HPC-jobs on Virtualized Clusters Using Host and VM Dynamic Configuration

Ibrahim Takouna, Wesam Dawoud, and Christoph Meinel

Hasso Plattner Institute
University of Potsdam
Potsdam, Germany
emails:{ibrahim.takouna, wesam.dawoud, christoph.meinel}@hpi.uni-potsdam.de

ABSTRACT
Energy efficient resource management has become a significant concern in virtualized data centers to reduce operational costs and extend systems' lifetime. The opportunity of reducing energy can be achieved by using Dynamic Voltage Frequency Scaling (DVFS) and hosts consolidation. However, energy management of emerging High Performance Computing (HPC) clouds that host CPU-intensive jobs is more challenging. In this work, we present an optimization solution to assuage the trade-offs between energy and acceptance ratio of jobs. To achieve this, we consider the current multicore processor architecture, which supports DVFS scheme. Furthermore, we tailored an energy model for multicore processor based on the number of active cores, the average running frequency, and memory. A power-aware local VM scheduler is also implemented at the host level. Importantly, we show the importance of including static power in the energy model. Finally, we compared our approach with pure DVFS and DVFS with live migration. The results show that our approach outperforms the other approaches in terms of energy, SLA violation percentage, system utilization, and number of finished jobs. As a future work, we will implement this in a heterogeneous cluster and will consider the cost of turning off and on the hosts.

1. INTRODUCTION
Green Computing has become an essential requisite for current HPC data centers that consume a huge amount of energy as much as a city does. This necessity is due to several factors. First, the limited power budget might prevent the capability to add new physical machine into a data center. However, there is a rapid increase in the power consumption, which is needed to run cloud data centers. For example, Hamilton [1] has reported that Amazon’s datacenters are facing a highly increased power demand where the servers consume 59% of the total power supply. Second, the operational cost rises with the increase in energy price as the energy is the dominating factor in the operational cost of data centers [2]. The U.S. Environmental Protection Agency (EPA) has reported that the energy consumption of the data centers located in U.S. is 61 billion kilowatt-hours in 2006, which costs $4.5 billion [3]. Finally, the reliability of the system might be undermined by the increase of the chip temperature that causes system failure [4]. The current multicore processors require high-power densities, which cause overheating [5]. However, these processors utilize DVFS, which can be an efficient technology to control power consumption at host level [6]. DVFS scheme has been exploited by many works to achieve energy saving [7]-[12].

Based on the previous mentioned reasons, energy-aware resource management for HPC data centers is highly crucial, but realizing energy saving is not a trivial procedure due to the trade-offs between performance and energy. In a study of 5,000 of Google hosts, they have found that the average CPU utilization for most of the hosts is between 10% and 50%[13]. Thus, there is a chance to achieve energy saving by virtual machines (VMs) consolidation using virtualization technologies, which enable live migration and dynamic configuration of VMs. Furthermore, virtualization allows agile management and guarantees performance isolation where a VM is a container of a job. Thus, consolidation of VMs into few numbers of physical hosts and turning off unused hosts can achieve energy savings. Currently, virtualized platforms are used not only for web or commercial transaction applications but also in grids and HPC clouds that host HPC applications. Virtualization technology are in continues improvement to relax the concerns regarding application performance due to virtualization overhead and resource contention. For example, to enhance the I/O performance in virtualized environments. Huang et al. [14] have implemented an efficient access to network resources so that the HPC applications can communicate at near-native performance. Thus, it is possible for HPC applications to use virtualized platforms without causing significant performance degradation.

Much research has proposed power management techniques at two levels: host level and cluster level. At host level, processor’s power is managed using DVFS, which adjusts the dynamic power portion of the total power. However, this achieves a limited amount of energy savings because of the bounded portion of the dynamic power. To realize more energy savings, algorithms exploiting live migration and consolidation can be implemented at cluster level. This allows adjusting the amount of idle power by keeping few hosts with...
high utilization and turning off unused ones. Nevertheless, a job is still allocated in pre-configured VM, and then this VM is scheduled into the suitable host in the cluster by a central scheduler [15]. In this paper, we propose a dynamic configuration of VM to minimize energy by exploring the number of core in the multicore processor.

![Diagram](image)

Figure 1: Energy vs. job acceptance

To show the motivation of this work, we adapted the example in [16] illustrating the trade-offs between the energy and job acceptance at the host level. Figure 1 represents an example of a host with three VMs scheduled with different policies. In this example, we compute the energy using the formula $E = RF^2 \cdot t$ where $RF$ is the relative frequency $RF = f_i / f_{max}$. First, Figure 1-(a) shows that when the host works at full speed, it finishes the three jobs within 8 seconds. In this case, any job arrives at the 8th second will be accepted and scheduled into this host instead of activating a new host. However, the amount of energy to finish these jobs is 8WS (i.e., 8 Watts.Seconds) using the former energy formula $8 = (1) \cdot 8$. Second, Figure 1-(b) depicts that running the same jobs using the half of the host capacity. As a result, the execution time of the jobs increases, but the amount of energy decreases. In this case, the energy is 3.12WS $((0.5)^2 \cdot 8 = 3.12)$, which is less than the amount of energy when the host runs at highest speed as in Figure 1-(a). However, when VM4 arrives at the 6th second requiring 1500MIPS, it will be rejected because the host has not enough MIPS. Similarly, when VM5 with 2000MIPS arrives at the 10th second, it will be also rejected by this host. In this case, the arriving jobs will be scheduled in a new host, which causes an increase of the total energy to finish these jobs. Finally, Figure 1-(c) shows that there is an optimal operating point, which is not running at the highest speed or at the half speed. This Figure depicts that a big chance of accepting VM4 and VM5 if the host increase the operating speed to 71% of the maximum speed. The energy in this case is 4.22WS $((0.71)^2 \cdot 8 = 4.22)$. Thus, we need an optimization algorithm to find this optimal operating point whenever a new job arrives or a job exists. By this solution, we minimize the total energy of the cluster and improve acceptance ratio, which is the main goal of this paper. To achieve this goal, we implement an energy-aware scheduling algorithm for HPC cluster where virtual machines are dynamically provided and configured for executing jobs. The jobs are heterogeneous in terms of MIPS and memory demand, number of processor, and the execution time. A host in this cluster consists of multicore processor (i.e., 6 cores or processing element), and an efficient power-aware multicore scheduler is also presented. Furthermore, we use a power model, combining number of active cores, average running frequency, and memory. Finally, we investigate using migration as a solution to achieve cluster energy optimization and compare it with our proposed approach. The rest of the paper is structured as follows. The next section presents the system model including the energy and job models. Section 3 describes Energy-aware Cluster Scheduling consisting of local and global energy optimization. Section 4 presents the simulation and the experiments settings. The related work is discussed in Section 5. Finally, the conclusions and future work are presented in Section 6.

## 2. SYSTEM MODELS

This section presents a multicore energy model, a job model, and jobs’ statistical analysis. However, we consider homogeneous hosts that have identical CPU and memory capacity.

### 2.1 Multicore Energy Model

Power of a host consists of static and dynamic power as represented in Equation (1). On the other hand, energy is the total amount of power that the host consumes during time $t_a$ as shown in Equation (2). For example, if a host consumes 200 watts and runs for 1 hour, the total energy for that hour is 200 Wh. Generally, the dynamic power of the host is computed by $c.V_d^2.f$. Nevertheless, this can be an accurate value if all cores run at the same frequency and voltage level. Actually, the current multicore processor architecture enables the cores to run at different levels of frequency. Experimentally, we have found the following two observations with a processor consisting of 4 cores. First, the processor consumed 25 watts when it was utilized 25% of the system capacity while only 1 core was active. Second, the processor consumed 19 watts when the same system also has the same utilization (i.e., 25%), but in this case the 4 cores were active. The explanation of this is that in the first case, the core was running at the full speed meanwhile in the second case the 4 cores runs at only 25% of the speed (i.e., frequency). Thus, without considering the number of active cores the utilization value does not reflect an accurate estimation of the power consumption. Hence, we use a power model of multicore processor that has been proposed in [17].

The power model of multicore processor consisting of number of active cores $f$ and average operating frequency $n_v$ in each host as shown in Equation (3). Importantly, the constant $a$ should be calibrated based on the processor model because the constant value depends on the number of cores and operating frequency values (i.e., minimum and maximum frequency). Hence, we calibrated the value of constant $a$ to fit with the power model of Intel Xeon X5675 processor used in the simulation.

$$P_{host} = P_{static} + P_{dynamic}$$

$$E_{host} = (P_{static} + P_{dynamic}) \cdot t_a$$

The dynamic power portion $P_{dynamic}$ is divided between the processor and memory. The memory power includes background power and operational power [18]. First, background power is the total power consumed in the following
power state: active (CKEH), fast powerdown (CKEL) and self-refresh (SR). Second, operational power is modeled by determining power per bandwidth unit, based on energy per operation, and multiplied by the consumed bandwidth. The background and the operational power are presented in Equation (4). However, we assumed memory power is proportional to memory utilization [19]. Finally, the total consumed energy by a host is the summation of CPU power multiplied by $t_{cpu}$ and memory power multiplied by $t_{mem}$. In this work, $t_{cpu}$ and $t_{mem}$ are identical.

$$P_{cpu} = a(f^2 + f + n_c)$$  \hspace{1cm} (3)

$$P_{mem} = P_{sr} + P_{CKEL} + P_{CKEN} + [(P_{BW,r} \times RBW + P_{BW,w} \times WBW)] $$ \hspace{1cm} (4)

$$E_{dynamic} = P_{cpu} \times t_{cpu} + P_{mem} \times t_{mem}$$ \hspace{1cm} (5)

### 2.2 Static Power Influence

In this section, we discuss the influence of the static power portion in energy optimization. There have been many research papers discussed the static power, but they have not considered it during problem formulation [24]. The authors have concluded that the static power is proportional to dynamic power; however, after reduction the problem formula, they actually have removed the static power from dynamic power. Thus, the total host power is proportional to the dynamic power curve but shifted by the static power value. However, the total host energy is not proportional to the dynamic portion of host energy as shown in Figure 2. Using this fact, the most efficient energy to run this job is at the relative frequency 0.6, allowing the job to finish after 3.33 seconds.

$$E_{host} = (0.25 \times \text{max}(f) + 0.75 \times \text{max}(f) \times (RF^2)) \times t_a$$  \hspace{1cm} (6)

Figure 2 shows a job which finishes within 2 seconds at highest frequency settings. In this example, we assume that the static power and dynamic power is 25%, and 75% of the host maximum power, respectively as shown in Equation (6). The host’s maximum power is 222 watts. As the execution time of a job is inverse proportional to frequency, reducing the frequency increases the execution time of the job. Thus, the job takes 20 seconds at lowest frequency. Therefore, to find the most efficient energy to execute this job, we should consider that the total host power is proportional to the dynamic power. Thus, the total host power is the dynamic power curve but shifted by the static power value. However, the total host energy is not proportional to the dynamic portion of host energy as shown in Figure 2. Using this fact, the most efficient energy to run this job is at the relative frequency 0.6, allowing the job to finish after 3.33 seconds.

### 2.3 Job Model

In this paper, we consider jobs that are multithreaded of CPU-intensive applications. These jobs are independent with no communication among each other. A job is represented by five parameters: $t_{ar}$, $t_{ex}$, proc, mips, and mem. These parameters symbolize the arrival time, the execution time, number of required processor (i.e., number of virtual CPUs), MIPS, and the required memory size, respectively.

![Figure 2: The relationship between energy and execution time.](image)

### 3. ENERGY-AWARE CLUSTER SCHEDULING

Typically, a cluster system, consisting of multiple hosts with multicore processors, has a central resource controller, namely energy-aware resource manager as shown in Figure 3. The submitted job is encapsulated into a VM, and each host consolidates multiple VMs. Each host has its own independent vCPU queue and scheduler. Whenever a job arrives to the cluster system, the energy-aware resource manager controls a job admission and placement based on information from hosts in the system. The job admission and placement in the system are discussed in [20] and depicted in Figure 3.

Algorithm 1 has two main functions $\text{isSchedulable}(\text{host, vm}_j)$ and $\text{estimateHostEnergy}(\text{host, vm}_j)$, which are executed by the host. $\text{isSchedulable}(\text{host, vm}_j)$ function assures the capability of the host to schedule $vm_j$, so the host schedule $vm_j$ only if there is enough available MIPS and Memory. Using $\text{estimateHostEnergy}(\text{host, vm}_j)$ function, each host returns the estimated amount of energy that can be consumed if $vm_j$ is scheduled on that host. Finally, the energy-aware resource manager selects the host with the lowest energy to schedule $vm_j$.

We present two significant issues regarding multicore processor architecture. First, the host estimates the power

![Figure 3: System architecture overview.](image)
consumption based on the number of active cores and the average frequency as presented in Equation (3). Second, it is more efficient to exploit the existing number of core instead of just scheduling VMs on few cores and turning off unused ones. In fact, the power consumption of a core increases dramatically at high frequency as illustrated in Figure 4. Thus, whenever there is a free core, vCPUs should be scheduled in the queue of a free core instead of increasing the frequency of the busy core as presented in Algorithm 2.

![Figure 4: Multicore processor power model.](image)

**Algorithm 1: Energy-aware FindHostForVm**

**Input:** clusterHostList, vmj  **Output:** selectedHost

1. foreach host in clusterHostList do
2. if (isSchedulable(host, vmj) == true) then
3. hostEnergy ← estimateHostEnergy(host, vmj)
4. if (hostEnergy < minHostEnergy) then
5. selectedHost ← host
6. minHostEnergy ← hostEnergy

7. return selectedHost

**Function** estimateHostEnergy(host, vmj)

1. power ← host.getPower(getUsedPesNumber(),
2. *getAvarageUsedMips(),
3. getUsedMemory())*vmj.estimatedruntime()

**return energy**

Finally, as energy-aware resource manager is responsible for VMs migration, it executes migration algorithm periodically to specify the VMs placement to keep the cluster at most efficient energy state. In this paper, we use migration algorithm with single threshold presented in [22]. The algorithm triggers VM migration when a host utilization exceeds a pre-specified threshold value. The migration of a VM causes performance degradation, which equals to 10% of the VM’s capacity. In fact, we enabled migration only to compare our approach with the approaches that combine DVFS and live migration [22][23].

### 3.1 Power-Aware Host-VM Configuration

After selecting the most energy-efficient host as we presented in the previous section, we perform further local optimization exploiting its number of cores. This is performed by Power-Aware Host-VM configuration module (PAHVM). So, when a job arrives with requested MIPS and number of cores as requirements, PAHVM searches for the optimal operating frequency of the host and VM configuration (i.e., number of vCPUs) as well. Importantly, PAHVM does not modify number of vCPUs for VMs that have requested 1 core assuming that the job is not multithreaded. The proposed optimization problem was implemented using IBM ILOG CPLEX Studio, which implements efficient algorithms to search for an optimal configuration solution. The problem was always solved in less than 1 second using a computer with Pentium 2.6GHz. Finally, Host-VM Configuration manager applies these configurations to the host and its VMs after calculating suitable weight values for each VM. Then, the provisioned resources within the host will be proportionally divided according to the weight setting of each VM. The weight is determined according to the workload demand of each job.

### 3.2 Dynamic Optimization Configuration

Multithreaded jobs can exploit number of cores in a processor to reduce execution time. A VM optimization problem that we consider is to determine the most efficient energy Host-VM configuration to handle throughput demand. To realize fine-grained energy-performance optimization, the optimization problem could output heterogeneous settings of cores’ frequency. For instance, a VM has two vCPUs. This first one could be mapped to physical core that runs on frequency f1, and the other one could be mapped on another physical core, running on frequency f2.

We introduce the following notation to formulate our optimization problem. We assume that a host has C cores. These cores are homogeneous cores in terms of clock frequency. Each core c runs on frequency f. The frequency level f is in Fc. The host schedules multiple jobs J. Each job j ∈ J is encapsulated into a VM. The parameter Dj represents the workload demand that guarantees job j completion before the deadline. The binary decision matrix is defined by Sc,j to denote whether the core c has been selected to run on frequency f to handle the total workload of jobs J. The matrix rows represent the number of cores meanwhile columns represent the frequency levels. For instance, S1,1 = 1 means that core 1 has been selected to run on frequency level 1. In fact, frequency level 1 gives the slowest clock frequency and the lowest performance. The provisioned capacity CAPc,f is the capacity of the host when it runs
on core $c$ with frequency $f$. These settings also will provide an amount of power $P_{c,f}$. Thus, the optimization problem is represented by the following mixed integer program (MIP):

$$\text{Minimize :}$$

$$\sum_{c \in C} \sum_{f \in F_c} S_{c,f} \cdot P_{c,f} \cdot t_a$$  \quad (7)$$

$$\text{Subject to :}$$

$$\sum_{j \in J} D_j \leq \sum_{c \in C} \sum_{f \in F_c} \text{CAP}_{c,f} \cdot S_{c,f} \quad \forall c \in C, \forall f \in F_c$$  \quad (8)$$

$$\sum_{f \in F_c} S_{c,f} \leq 1 \quad \forall c \in C$$  \quad (9)$$

$$t_a \leq \min_{j=1}^{J} \{\text{ert}_j\}$$  \quad (10)$$

$vCPU_j \leq n \quad \forall j \in J$  \quad (11)$$

The objective function given by Equation (7) is to find a Host-VM configuration that minimizes energy consumption of the host. As observed experimentally via the model of power in Figure 4, the power consumed by a host grows linearly with the number of cores at the same given CPU frequency. Equation (8) is the constraint that avoids the solutions at which the demand of all running jobs $J$ exceeds the total capacity of the host at a specific configuration. Equation (9) guarantees that only one frequency $f \in F_c$ is assigned to a given set of cores $c$. Furthermore, these configurations guarantee that the execution of the hosted jobs is completed with the minimal possible energy consumption. Equation (10) guarantees that a job can finish on time or before where $\text{ert}_j$ is the estimated remaining execution time of job $j$. Finally, Equation (11) specifies the maximum number of vCPU of VMs. The parameter $n$ either can be specified by the use or can be equivalent to the number of logical cores in the host. For example, if $n$ is chosen to be 4 for a specific host, then all multithreaded VMs will be configured with 4 vCPUs. This issue will be discussed in the experiment section. However, more detail about the implementation can be found in our work [21]. The optimization problem is solved whenever a new job enters or leaves the host to maintain the host at efficient energy as presented in Figure 2.

4. SIMULATION AND EVALUATION

In this section, we present statistical analysis for the jobs trace that is used in the simulation using CloudSim toolkit [22]. Then, we discuss the results of the proposed optimization solution comparing them against other solutions.

4.1 Jobs Statistical Analysis

We utilize jobs trace of LHC Computing (LCG)[25]. The LCG testbed, which is used for high-energy physics data processing, currently has approximately 180 active sites with a total number of 24,515 CPUs. Actually, the trace has neither parallel jobs nor required memory size; however, we randomized the number of required vCPU and the amount of required memory to process data as shown in Figure 5. Figure 5-(a) shows that 25 % of the jobs are single thread with one vCPU and the other 75% are multithreaded with different number of vCPU. Figure 6 depicts the required MIPS for each vCPU of the jobs. Finally, Figure 7 shows the arrival rate of the first 1650 jobs in LCG trace. It shows the variation of the number of submitted jobs every minute.

4.2 Experimental Setup

To evaluate our approach, it was hard to find a real large-scale infrastructure that we could deploy our solution and evaluate the results. Hence, we conducted the simulation using CloudSim toolkit, which simulates IaaS cloud and is widely accepted in cloud community. Furthermore, it allows
simulating dynamic workloads and implementing energy-aware resource management algorithms. We used version 3.0 of CloudSim, which is the latest version at the time of writing this paper. We simulated a cluster with 100 hosts; the host is a ProLiant DL360 G7 server with (Intel Xeon X5675 processor) (6 Core, 3 GHz, 12 MB L3 Cache, 16 GB). We considered the measured power consumption values that were provided by the SPECpower benchmark [26]. The power consumption values were used to find the suitable constant $a$ in Equation for the Intel Xeon X5675 processor power model. Then, we used the multicore power consumption model where the power was estimated by number of active cores, the average frequency, and memory utilization. The processor has Voltage range 0.75V-1.35V. Accordingly, we assumed that the processor has 7 frequency levels. The maximum level is 3GHz and the minimum level is 1.8GHz. Using HP documentation for Sizing Mainframe workloads for Intel Xeon-based Platforms, we considered that each core has maximum performance of 3500 MIPS for CPU-intensive workloads [27].

Table 1: Host characteristics: Frequency, its corresponding maximum capacity by MIPS, and Power consumption.

<table>
<thead>
<tr>
<th>Levels</th>
<th>Freq.(GHz)</th>
<th>Voltage</th>
<th>MIPS</th>
</tr>
</thead>
<tbody>
<tr>
<td>P0</td>
<td>1.8</td>
<td>0.75</td>
<td>3000</td>
</tr>
<tr>
<td>P1</td>
<td>2</td>
<td>0.85</td>
<td>6000</td>
</tr>
<tr>
<td>P2</td>
<td>2.2</td>
<td>0.95</td>
<td>9000</td>
</tr>
<tr>
<td>P3</td>
<td>2.4</td>
<td>1.05</td>
<td>12000</td>
</tr>
<tr>
<td>P4</td>
<td>2.6</td>
<td>1.15</td>
<td>15000</td>
</tr>
<tr>
<td>P5</td>
<td>2.8</td>
<td>1.25</td>
<td>18000</td>
</tr>
<tr>
<td>P6</td>
<td>3</td>
<td>1.35</td>
<td>21000</td>
</tr>
</tbody>
</table>

To accurately simulate the real operation of processors, we keep the cores running in discrete fashion where each core runs on a specific performance level. For example, if the total jobs' demand is 2500MIPS, the processor will run at 3000MIPS, which is the first performance level in the used processor (i.e., P0) as presented in Table 1.

4.3 Experiment Results

We carried out five different experimental setups as presented in Table 2. First, using only DVFS approach, the exact demand of MIPS by job $j$ was allocated to the VMs. Thus, the maximum number of used hosts to execute the 1650 jobs was 48 hosts with average CPU utilization 49.97%. The total energy to finish these jobs was 35.06KWh with 1.22% SLA violation. Second, we combined DVFS with migration to increase hosts utilization. Unfortunately, the situation became worse where the average hosts’ utilization was below the half of the host capacity. Furthermore, number of hosts increased to 59 hosts, consumed 44.27kWh with high SLA violation percentage. The reason behind the increase of SLA violation percentage was VMs migration. The total number of migrations were 75 migrations. As we mentioned earlier, a migrated VM loses 10% of its capacity until it completely migrates to the destination host. Similarity, Huang et al. [28] measured the cost of a migrating VM runs NPB benchmark suite. They found that the migrated job’s completion time increased 5 seconds. Consequently, if we assume that one migration process is imitated at each time, the total delay of the total execution time of jobs is 375 seconds (i.e., 0.1 hour). Furthermore, as shown in Table 2, VM migration increased number of active host as well. Third, DVFS-OHVO is one of the settings of our proposed optimization solution; however, it just configures the host without modifying the VMs configuration. We conducted this experiment to compare it with the fourth and fifth experimental setups. However, DVFS-OHVO approach used more hosts compared to pure DVFS. Fourth, DVFS-OHVO-C4 is another setting of our proposed optimization solution, but in this case it optimizes both of host and its VMs, where the maximum number of vCPU for a VM is 4. DVFS-OHVO-C4 consumed less energy although it increased the number of used hosts. DVFS-OHVO-C4 induces less SLA violation compared to pure DVFS and DVFS-OHVO. Importantly, DVFS-OHVO-C4 did not modify number of vCPU for VMs with a job requesting a single core. Finally, DVFS-OHVO-C6 setting was slimmer to DVFS-OHVO-C4 except that the number of maximum number of vCPU for a VM was set to 6.

It is clear from Table 2 that the online optimization for host and VMs has the best minimal energy in spite of the increased number of the used hosts compared to the pure DVFS approach. Importantly, the average CPU utilization is also increased while using DVFS-OHVO, DVFS-OHVO-C4, and DVFS-OHVO-C6 configuration respectively. Finally, the best minimal energy and the lowest SLA violation is DVFS-OHVO-C4, which ran fewer number of hosts compared to DVFS-OHVO-C6.

Figure 8 depicts the empirical cumulative function (eCDF), showing the number of active hosts during the simulation of the five experimental setups. First, as presented in Table 2, our approach used more hosts compared to the other approaches, however, Figure 8 shows that our approach used this number of hosts just for a short time. All approaches use less than 15 hosts for 92% of the simulation time. For this number of hosts just for a short time. All approaches, however, Figure 8 shows that our approach used more hosts compared to pure DVFS-OHVO approach. Fourth, DVFS-OHVO-C4 did not modify number of vCPU compared to pure DVFS and DVFS-OHVO. Importantly, DVFS-OHVO-C4 did not modify number of vCPU for VMs with a job requesting a single core. Finally, DVFS-OHVO-C6 setting was slimmer to DVFS-OHVO-C4 except that the number of maximum number of vCPU for a VM was set to 6.

Figure 8: Empirical distribution function with number of active servers.
4.4 Jobs Inter-arrival Influence

After presenting a general comparison between our proposed approach and the others approaches, we studied the influence of the inter-arrival time and energy. Figure 9 shows the normalized energy for executing jobs the different values of inter-arrival time for these jobs. The energy values were normalized to pure DVFS at inter-arrival time of 2 seconds. Unfortunately, DVFS-MIG consumed more energy and resulted in high SLA violation percentage compared to the others approaches. As mentioned earlier, migration degrades the performance of a VM during migration process and increases the execution time of the job. Pure DVFS showed almost the same performance with different inter-arrival times. When the inter-arrival time was smaller for the same number of jobs, the amount of energy required to execute them was larger than the amount of energy required when the inter-arrival time was larger. The reason was that hosts run to their highest capacity to cope with the high jobs arrival rate. Finally, our proposed approach achieved the best performance and the lowest energy. Our approach was capable to adapt with variant arrival rate including high and very low arrival rate. Importantly, the efficient exploitation of the number of physical cores assisted to achieve the lowest energy. Thus, by using our approach that is dynamically configure host and VMs, we assure the trade-offs between the energy and the acceptance ratio.

5. RELATED WORK

Starting with research concerning data centers, Zhu et al [29] have proposed three individual controllers, where each operates at a different time scale: hours-to-days, minutes, and seconds. These controllers assist to considerate VMs and cannot be generalized. The problem of dynamic consolidation to achieve multiple resource optimization. However, the proposed algorithms do not adhered to SLAs. Gandhi et al. [34] have investigated the problem of allocating an available power budget among host, while minimizing the mean response time. To investigate the effect of different factors on the mean response time, a queuing theoretic model has been implemented, which allowed the prediction of the mean response time as a function of the power-to-frequency relationship, arrival rate, and peak power budget. The model was used to determine the optimal power allocation for every configuration of the above factors. Minimizing energy consumption of multi-tier web applications in virtualized heterogeneous systems have been investigated in [36]. The authors have implemented a multidimensional bin-packing algorithm for the workload consolidation to achieve multiple resource optimization. However, their algorithm is based on an observation, which is the energy consumption per transaction results in a “U” shaped curve. Thus, the proposed solution is application specific and cannot be generalized. The problem of dynamic consolidation of VMs running a multi-tier web-application using live migration, while meeting SLA requirements, have investigated in [37][38]. The SLA requirements were modeled
6. CONCLUSIONS AND FUTURE WORK

In this paper, we presented an online dynamic optimization scheduling solution for HPC-jobs in a virtualized cluster. Our approach is based on dynamic configuration of host and its VMs to achieve the minimum energy necessary to execute a certain number of jobs. This optimization problem is solved at a host level at each arrival or completion of a job. Importantly, the time to solve the optimization problem was less than a second; therefore it is not considered as a high overhead. Using our approach, dynamic configuration of host and VMs, increased the host utilization, guaranteed jobs requirements with less SLA violation percentage, and minimize energy. To evaluate our approach against other approaches, we conducted several experiments with different configurations. The results showed that our approach outperformed both pure DVFS and DVFS with migration. Moreover, our experiments showed that using migration to achieve energy savings is not always a feasible and effective solution. The frequent migration of VMs degrades the system performance, so the decision of migration should be taken with awareness of its cost and the system dynamism. As a future work, we will implement this in a heterogeneous cluster and consider the overhead of turning off and on the hosts.

7. REFERENCES


[18] H. David, C. Fallin, E. Gorbakov, U. R. Hanebutte, and


