University of Castilla-La Mancha

A publication of the

Computing Systems Department

Power and Performance Optimization for Heterogeneous Clouds

by

Selome Kostentinos Tesfatsion[1], Julio Proaño[2], Luis Tomás[1],
Blanca Caminero[2], Carmen Carrión[2], Johan Tordsson[1]

Technical Report **#DIAB-16-03-1** June, 2016

[1]Department of Computing Science. Umeå University. SE-901 87 Umeå, Sweden
[2]Computing Systems Department. University of Castilla-La Mancha. 02071 Albacete, Spain

This work was partially supported by the Swedish Research Council (VR) for the project Cloud Control, the Spanish Government under Grant TIN2012-38341-C04-04 and the Ecuadorian Government under the SENESCYT Scholarships Project. We also thank Eddie Waldrob for his feedback regarding optimizer formulation.

DEPARTAMENTO DE SISTEMAS INFORMÁTICOS
ESCUELA SUPERIOR DE INGENIERÍA INFORMÁTICA
UNIVERSIDAD DE CASTILLA-LA MANCHA
Campus Universitario s/n
Albacete - 02071 - Spain
Phone +34.967.599200, Fax +34.967.599224

Power and Performance Optimization for Heterogeneous Clouds

[1] Selome Kostentinos Tesfatsion[b], [1]Julio Proaño[a], Luis Tomás[b], Blanca Caminero[a], Carmen Carrión[a], Johan Tordsson[b]

[a] *Computing Systems Department*
University of Castilla-La Mancha
Campus Universitario s/n
02071 Albacete, Spain

[b] *Department of Computing Science*
Umeå University
SE-901 87 Umeå, Sweden

1. Introduction

Current cloud computing infrastructure typically assumes a homogeneous collection of commodity hardware [1]. Some applications may have a high computational processing requirements that cannot be obtained from these commodity servers. These include numerical intensive applications, applications with high parallelism, and applications that exhibit near real-time performance requirements. One strategy for increasing processing efficiency to the cloud is to add heterogeneity; providing access to resources better suited to complex computation [2]. Heterogeneous systems integrate more than one processing unit with different performance and energy consumption characteristics. Unlike its homogeneous counterpart, a heterogeneous system address both throughput and efficiency for various workloads by matching resources to each application's needs and has a much higher potential in saving energy. However, exploiting this potential requires a well designed resource allocation system that maps heterogeneous resources to applications' requirement with minimum cost in performance and power. This is a nontrivial task.

Field-programmable gate array (FPGAs) offer a significant acceleration in execution over CPUs. They are also significantly more power-efficient, resulting in a computational efficiency improvement in the orders of magnitude over both CPUs and GPUs [3], [4]. As more and more workloads are being deployed in the cloud, it is appropriate to consider how to make FPGAs and their capabilities

Email addresses: selome@cs.umu.se ([1] Selome Kostentinos Tesfatsion), Julio.Proano@alu.uclm.es ([1]Julio Proaño), luis@cs.umu.se (Luis Tomás), mariablanca.caminero@uclm.es (Blanca Caminero), Carmen.Carrion@uclm.es (Carmen Carrión), tordsson@cs.umu.se (Johan Tordsson)

[1]Contributed equally to this work.

available in the cloud [5].

Virtualization technologies enable rapid provisioning of Virtual Machines (VMs) and thus allow cloud services to scale up and down resources allocated to them on-demand. This elasticity can be achieved horizontally where VMs are changed during a service's operation and vertically where capabilities of a running VM, typically in terms of CPU and RAM, are changed dynamically. Resource provisioning offered by Infrastructure as a Service (IaaS) providers, e.g. Amazon EC2 [6] is typically done through horizontal scaling of resources. Using a VM as scaling unit is generally coarse grained and can cause unnecessary over-provisioning [7]. Conversely, vertical elasticity allows fine-grained resource allocation and rapid resource enactment—resources may be allocated for as short as a few seconds. Vertical elasticity could thus enable efficient resource provisioning and power usage.

Autonomous resource provisioning in the cloud has been widely studied to guaranty system-wide performance, that is, to optimize data center resource management for pure performance [8]. Energy consumption has become a fundamental problem in data centers, raising issues to all energy-related costs, including capital, operating expenses, and environmental impacts [9]. It is important to minimize energy consumption to insure a sustainable future growth of cloud computing. However, minimizing power consumption may inherently result in performance degradations. It is thus essential to guarantee the application Quality of Service (QoS) while minimizing the power consumption.

Existing power management systems based on dynamic voltage frequency scaling (DVFS), including the linux on demand CPU governor [10] and most OS implementations, are changed based on CPU utilization. That is, CPU frequency is changed depending on its utilization. For example the on demand governor transitions to the next higher or lower p-state if the current CPU utilization crosses a threshold for a certain period of time. However, this approach can cause over or under-provisioning as it is oblivious to the observed performance of each service. We argue that to save power without service-level objective (SLO) violations, DVFS decisions should be based on application-level performance and server power usage.

In this paper, we propose a system for power- and performance-efficient resource management in heterogeneous clouds. The proposed approach maps applications' requirement to resources (CPU/FPGA) with minimum cost in performance and power. A technique that combines scheduling of FPGAs and optimized resource allocation technique for commodity servers is proposed. The scheduler makes efficient use of FPGAs by controlling the assignment of FPGA to computationally intensive applications. For the other concurrently running applications, the optimizer employs both

virtual machine resizing (CPU) and dynamic CPU frequency scaling to optimize server power and performance. In particular, the contributions of this work are:

- Design of performance and power models to determine the relationship between application-level performance (and server power) and resource allocation of the system (Section 3.1).
- An optimizer that minimizes power consumption while meeting performance targets. The optimizer allocates the right amount of resources for each application and combined with CPU frequency scaling it achieves this objective by selecting a configuration that yields the least power usage (Section 4.2).
- A scheduler that assign application to the FPGA. The scheduler makes efficient use of FPGA to increase performance and reduce power consumption (Section 4.1).
- An evaluation of the potential of the proposed approach using multiple cloud applications. The results demonstrate that our approaches achieves the lowest power consumption while meeting the performance targets (Section 5).

2. Architecture

Figure 1 shows an overview of the architecture of the heterogeneous cloud system for a server with an FPGA. The virtualized server hosts multiple applications and has an FPGA attached to it. Each application is deployed in a VM. The proposed system performs three main tasks; monitoring, modeling and decision making. For monitoring, the system extracts information from the server, and the VMs/applications using the *sensor*. The system model captures applications' performance and power consumption behavior using *performance* and *power models* respectively. Based on these models, the *controller* makes an optimal decision to minimize power consumption using two subcomponents: i) the *FPGA scheduler* that ensures efficient use of the FPGA and ii) the *optimizer* that combines CPU allocation and frequency scaling for the VMs running collocated applications. Finally, the *actuator* applies the new configuration. A high level function of each component of the system architecture is described as follows:

- The sensor periodically polls the performance of each application. We currently measure performance in terms of throughput for each application but the sensor can be extended to also handle

other performance metrics. The sensor also gathers the server power consumption. The performance and power usage are then sent to the the performance and power models respectively, for model re-computation.

- The performance model describes the relationship between an application's resource allocation and its performance. It automatically learns a model for this relationship. The details of this model is described in subsection 3.2.

- The power model describes a relationship between the server's resource allocation and its power usage (refer section 3.1). This model is also updated online. The updated parameters from the performance and power models are sent to the optimizer to determine the optimal configuration.

- The controller maps applications' requirement to different resources using the optimizer and the scheduler. The **optimizer** makes allocation decisions based on applications' performance targets and model parameters received from the power and performance models. It determines the configuration that minimizes power consumption and meets the performance requirements. The new configuration may result in changes in CPU frequency and/or number of cores allocation. The **scheduler** controls the assignment of FPGA to VMs that can make use of the accelerator. The selection of a VM depends on the algorithm used. The details of the scheduling are described in section 4.1.

- The actuator is used to i) attach the FPGA to the VM selected by the scheduler, and ii) actuate the configuration chosen by the optimizer, that is to change the CPU frequency and number of cores of the running VMs.

3. Models

In order to analyze the energy and performance efficiency of a particular configuration, it is first important to understand the relationship between resource allocation and power consumption, as well as between resource allocation and application's performance.

3.1. Power Model

The processor is one of the largest power consumers in today's servers, and the choice of CPU has a significant impact on the servers power consumption [11]. The power consumption of processors

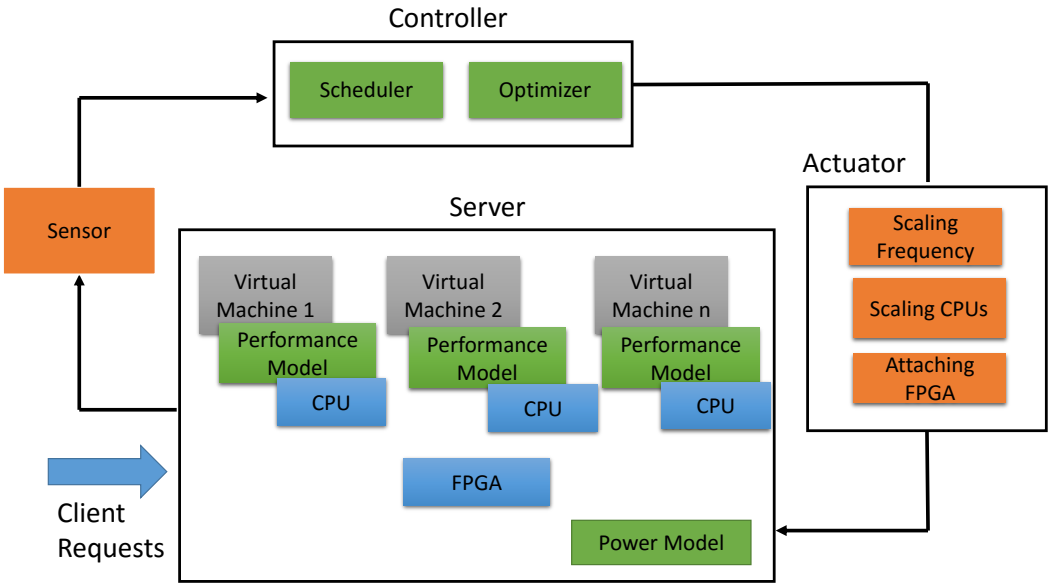


Figure 1: General Architecture of the proposed system

partly depends on the CPU frequency they are running. Servers also have non-neglectable idle energy consumption. The relationship between server resource and its actual power consumption is normally nonlinear due to the complexity of computer systems. Since nonlinear control can lead to unacceptable runtime overhead, one feasible approach to capturing the system behavior is to linearize the system model [11], [12]. Thus, we use the following linear model to approximate the quantitative relationship between power usage and resource (CPU and CPU frequency) of the server running K VMs:

$$W = \alpha \sum_{i=1}^{K} c_i f_i + W^{\texttt{static}}, \tag{1}$$

where c_i is the CPU assigned for VM_i, and f_i is the frequency of VM_i's CPU. $W^{\texttt{static}}$ represents static system power. We have not (explicitly) included the FPGA power usage in the power model because it has a low and fixed power usage. For the experiment we conducted, the measured power difference when fully utilization the FPGA is less than 3% compared to when not using it. The parameter α is recomputed online using the Recursive Least Squares (RLS) method [13] periodically The recursive nature of the ELS algorithm makes the time needed for this computation negligible, as the model is updated recursively instead of being computed from scratch every interval.

3.2. Performance model

The performance model is based on our previous work [14]. The throughput of application i depends on the number of cores assigned as well as the frequency of the VM. More precisely, we model the performance of application i as

$$P_i(c_i, f_i) = \alpha_i c_i f_i \tag{2}$$

where c_i and f_i are CPU allocation and CPU frequency of VM i. The coefficient α_i captures the relationship between the current performance and CPU allocation of VM i. Here again, the parameter α_i is updated periodically using the RLS method.

4. Controller design

As described in Section 2 the controller selects an optimal configuration through the use of the scheduler and the optimizer. First, the scheduler maximizes the use of the FPGA and iteratively attach it to the VMs that needs it the most according to some criteria. Then the optimizer performs the scaling of server CPU frequency and number of cores to all other VMs that are running on the server, keeping their needed performance while consuming as less energy as possible. Detailed description of each component given below.

4.1. Scheduler

The scheduler is in charge of making an efficient use of the available heterogeneous resources, i.e., the cores and the FPGA. This means the scheduler objective is to meet applications SLOs while at the same time consuming as less energy as possible. Due to the higher performance and higher energy efficiency of the FPGA, the scheduler must maximize FPGA utilization. While doing so, it also must consider which VM should use the FPGA at each point in time so that they can meet their performance requirements. Note that due to the PCIe express limitations over virtual environments, the FPGA cannot be shared for more that one virtual machine at the same time. Therefore, we use a strategy in which the FPGAs are shared by slots of time among virtual machines. In other words, the scheduler selects the best candidate VM and attach the FPGA to it.

The VM selection process is parameterizable and can be based on different criteria, such as throughput, application deadlines, round-robin, etc. In this paper we propose an energy-aware metric

Algorithm 1 FPGA Scheduling

Require: VM_{first}, $List_{VMs}$, $metric : throuhgput, deadline$, VM_i, i=1,2,...K

```
while True do
    if FPGA_status == Free then
        for each VMi, i=0,1,2,..N do
            calculate metric
            create List_VM
            cort List_VM, higher metric first
        end for
        pick VM_first from List_VM
        attach FPGA to VM_first
    end if
end while
```

that considers both the applications performance and their deadlines to make an estimation of how much computational capacity they would require to fulfill their requirements. Then, the application with higher computational requirements, therefore the one that would need more energy to complete on time, is selected and the FPGA is attached to it.

Algorithm 1 summarizes the FPGA scheduling process. First, the selected metric is calculated for each VM, in our case the computational capacity requirements. Next, the scheduler sorts all VMs based on that metric and assigns the FPGA to the first VM in the list. This process is repeated every time the FPGA becomes available. Note the scheduler must be aware of when the FPGA can be released and re-assigned to a different VM. A wrong synchronization could cause a misbehaving of the system. To achieve this goal the status of the FPGA is monitored periodically and it is released only when is "free" (not processing) otherwise the scheduler cannot assign it to another VM.

4.2. Optimizer

Once the scheduler has performed the mapping between VMs, cores and FPGA, the optimizer is in charge of the VMs' vertical scaling and CPU frequency adaptation to reduce the energy consumption while maintaining the required performance.

The optimization algorithm considers uniform CPU frequency for the full physical machine for every optimization decision, i.e. all VMs run at the same CPU frequency. To run the VMs just fast enough to meet their desired QoS, we use a resource allocation method that assigns cores even at the fractional level. The problem is to determine the most power-efficient configuration, i.e. number

Algorithm 2 Optimizer

Parameters: α, min_{cost}, $minc_i$,$optcore_i$ for VM_i, i=1,2,...K

for each $f_j \in \boldsymbol{List}_{freq}$ **do**
  $c_i = P_i^{\texttt{target}}/(\alpha_i f_j)$, i=1,2,...,K
  if $\sum_{i=1}^{K} c_i \leq CORES$ **then**
    $obj_{cost} = \sum_{i=1}^{K} \alpha f_j c_i + W^{\texttt{static}}$
    if $obj_{cost} \leq min_{cost}$ **then**
      $mincore_i = c_i$
      $frequency = j$
      $min_{cost} = obj_{cost}$
    end if
  end if
end for
if $mincore_i > 0$ **then**
  $optfreq = frequency$
  $optcore_i = mincore_i$
end if

of (fractional) cores for VMs and server CPU frequency, while meeting performance targets. This process is detailed at Algorithm 2.

The server comprises N cores and in total K VMs. To formulate the optimization problem, we let $\boldsymbol{A}_f$ be the set of clock frequencies available on the server. Let $P_i^{\texttt{target}}(t)$ be the minimum performance target of VM_i at time t , for $i = 1, 2..., K$. We let c_i denotes the number cores allocated to VM_i and define $\boldsymbol{c} = (c_1, c_2, ..., c_K)^T$.

Next we define the following sets of admissible configurations. Let $\mathcal{C} = \{\boldsymbol{c} \in \mathbb{R}^K | \sum_i^K c_i \leq N, c_i \geq 0 \forall i\}$ be the set of all possible core allocations for the K virtual machines. Let $\mathcal{U} = \{(\boldsymbol{c}, f) | \boldsymbol{c} \in \boldsymbol{C},$ and $, f \in \boldsymbol{A}_f\}$.

The task of minimizing the power consumption, at each time t while attaining at least the pre-specified performance target can now be formulated as the constrained optimization problem:

$$\begin{aligned} &\min_{(\boldsymbol{c},f)\in\mathcal{U}} W(\boldsymbol{c}, f) \\ &\text{subject to} \\ &\qquad P_i(c_i, f) \geq P_i^{\texttt{target}}(t) \;\; i = 1, 2, \ldots, K. \end{aligned} \tag{3}$$

5. Evaluation

In this section, we present experimental setup and results from a number of experiments that we conducted to evaluate our framework.

5.1. Hardware and Server Setup

The experiments were performed on an Intel Core I5 server with 4 cores and 14GB of Ram Memory. The server runs Ubuntu 14.04.1 with Linux kernel 3.13.0-44-generic. It is attached to a ML605 Evaluation Kit through Xilinx which contains a Virtex 6 (XC6VLX-240T-1FFG1156) FPGA [15]. A WattsUp PRO power meter is used for measuring power usages of server [16]. This device is a digital electronic system that utilizes a high-frequency sampling to measure voltage, current and true power (in Watts). The VMs are managed by qemu-KVM version 2.0.0 [17] hypervisor. The FPGA communicates with the VMs through PCIe bus and passthrough with VT-d Intel technology[18]. This technology allows a direct communication (VMs/FPGA) with less overhead and latency. Additionally, the RIFFA[19] framework is used for development the software in both CPU and FPGA.

5.2. Applications and Workload

In order to evaluate the proposed approach, we use two groups of applications: cloud applications that run on commodity hardware—in this case the RUBiS benchmark [20] and applications that can use FPGA—here represented by video Sobel convolution[21]. RUBiS is an eBay-like e-commerce application. RUBiS application is deployed with all of its components such as web servers and database servers inside one VM as is commonly done in practice [22]. The VM is based on Debian 6.0.6, initially configured with 4 cores. To emulate the users accessing the RUBiS, we use an open source tool, httpmon[3]. Dynamic variation in incoming load is emulated by changing the number of concurrent users at runtime. The RUBiS client runs on a machine with the same experimental hardware setup described in Section 5.1. For this application we use WIKI workload traces. A control interval of 15 seconds is used for the optimizer as this is short enough to adapt the underlying infrastructure more quickly to workload changes and long enough to observe the effects of the re-configuration [7].

The video application uses a Sobel operator to compute an approximation of the gradient of the image intensity function. The Sobel convolution is based on convolving an image with a small,

[3]https://github.com/cloud-control/httpmon

and integer-valued filter in the horizontal and vertical directions. The video we have used is a (.avi) file with a resolution of 720x384. The FPGA, is dynamically reconfigured with the application-specific instructions, in this case the Sobel convolution filter bitstream for processing a sequence of videos. We vary the size of videos of each each VM by assigning a random number chunks, each chunk containing 13639 frames. To allow sharing of the FPGA by slots of time among virtual machines, a scheduling decision is made whenever the FPGA completes processing a chunk of video.

The optimization algorithm written in C is implemented in-house and need less than 20 ms to execute with the current setup. The scheduler is written in Python version 2.7 and takes less than 40 ms to execute. They run on a machine that is in the same local network with the node that does the actuation.

5.3. Sensor and Actuator

We instrument RUBiS and video applications to gather and send throughput statistic via UDP sockets to the optimizer. The incoming requests are also monitored to calculate the target throughput. The number of frames that need to be processed per second to meet the deadline is taken as a target throughput value for video VMs. The target values are sent to the optimizer via a TCP socket for making decision. And the power consumption is read from WattsUp meter.

The actuator uses three mechanisms to make the changes; i) the cpufrequtils package [23] to scale the CPU frequency of the cores. The cores can be scaled from 1.6 GHz to 2.4 GHz frequency. We changed the default *ondemand* CPU frequency scaling policy to a *usersspace* policy that gives us full control to change CPU frequency. ii) the taskset tool [24] to change the core allocation of VMs. The tool takes advantage of the Linux scheduler property that "bonds" a process to a given set of CPUs, ensuring that the process will not run on any other CPUs. iii)In order to attach/detach the FPGA, we use the KVM hotplug [25] pci with VT-d support. When the FPGA is free we establish a communication with the qemu monitor [26] to attach or detach the FPGA depending on the scheduler decision. Immediately, when a VM detects that it has an FPGA attached, it automatically starts using it. In addition, when the FPGA finish processing, it is released. All in all these techniques make overhead of actuation negligible.

5.4. Results

5.4.1. Models accuracy

To assess the prediction accuracy of performance and power models, we use two validation measures—*coefficient of determination* (R^2) and *mean absolute percentage error* ($MAPE$). These are calculated as $R^2 = 1 - \frac{\sum_{i=1}^{n}(y_i - f_i)^2}{\sum_{i=1}^{n}(y_i - \bar{y})^2}$ and $MAPE = \frac{1}{n}\sum_{i=1}^{n}|\frac{(y_i - f_i)}{y_i}|$, where n represents the total number of samples, y_i, f_i and $\bar{y}$ represent the measured value, predicted value and the sample mean of y respectively. Table 1 shows the values of these two measures for the power model and the three applications' performance models. The models predict system behavior with $MAPE$ below 10% and R^2 above 80%, which is considered sufficiently accurate [12].

Table 1: Prediction Accuracy of performance and power models.

	$MAPE$	R^2
Power	1.6%%	79.5%
Videos	8.5%	83%
RUBiS	5.4%	91%

5.4.2. Heterogeneity

We compared the impact of using heterogeneous resources (cores and FPGA) in terms of processing and energy efficiency against using only homogeneous server resources (cores). As shown in Table 2 the system is able to process more frames for video applications in the given time period and consumes less energy when using both regular cores and the FPGA. In fact, with a 7% increase in energy, 35% more number frames are processed and this results in an overall energy efficiency improvement of 27%. Here we have used power normalized throughput (frames/sec per watt) metric as a measure of energy efficiency. Figure 2 shows the details of this result such as the number of created VMs and the energy (power*time) used by each VM to complete its task. As shown in the figure less energy is needed to process each video in the heterogeneous setting. In addition, 28% more number of video VMs are processed during the same time span. These findings reflect the benefit of using FPGA in the cloud for improved performance and energy savings. For the experiments that follow, the use of cores and FPGA is assumed.

Table 2: Processing-and energy-efficiency comparison between with- and without-FPGA configurations.

	Frames Processed	Average Power (Watt)	Time (Sec)	Energy (Watts*Sec)	Frames-per-second-per-watt
withFPGA	1418456	108	6312	681696	2,08
withoutFPGA	995647	100	6312	631200	1.5

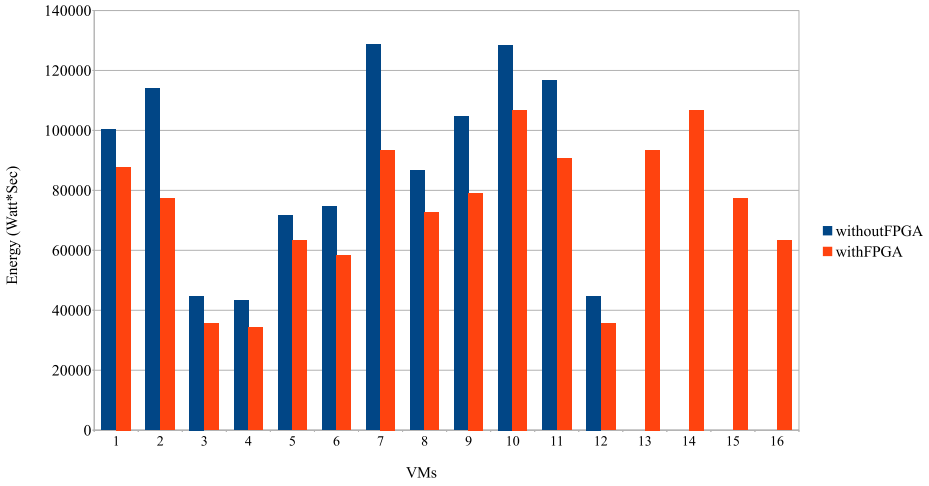
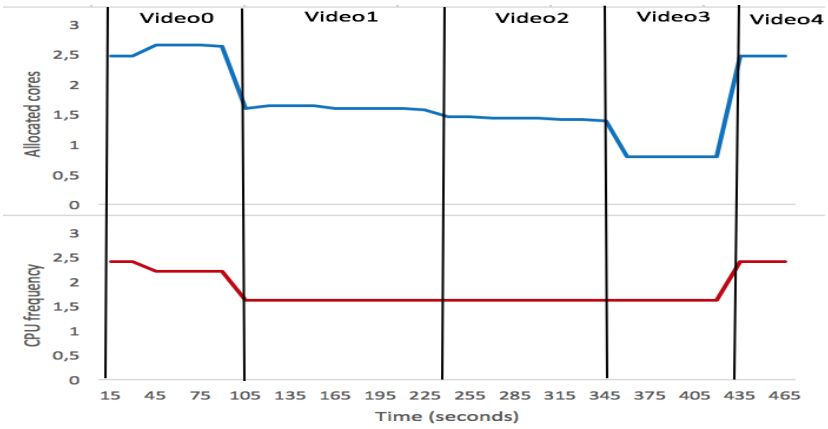


Figure 2: Number of video VMs processed and Energy per VM for FPGA and non-FPGA configurations

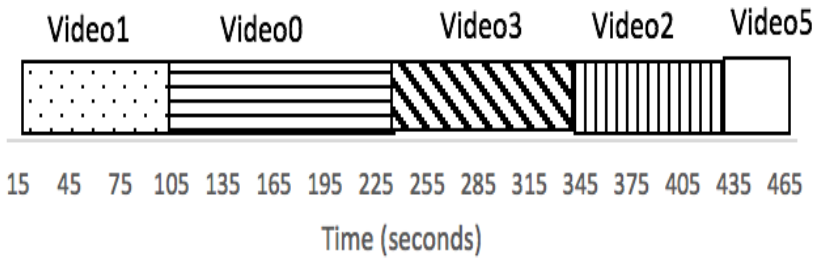
5.4.3. The optimizer and the scheduler

Both the optimzer and the scheduler work together to improve processing and energy efficiency of the system. The scheduler make efficient assignment of the FPGA to VMs that can make use of it. The optimizer complements the scheduler by assigning the right amount of resources to VMs that are not picked by the scheduler as well as VMs that can run only on server CPU.

To show how one supports the other, we perform an experiment by running three applications: RUBiS and one video application running on server CPU and a third video application using the FPGA. Figure 3 show the FPGA assignment and server resource allocation by the scheduler and the optimizer respectively. The scheduler selects a VM with high throughput(fps) and attaches it to the FPGA. As shown the Figure 3b the scheduler first selects video1 VM to use the FPGA while the optimzer determine the number of cores and CPU frequency for RUBiS and video0 VMs. Figure 3b shows the allocation for the video VM. Then a scheduling decision is made that attach the FPGA to a VM with higher target throughput(video1). Since video0 is accelerated well by the FPGA, the optimizer assign fewer number of cores to meet its performance target. For cases where a video throughput target is higher (the VM just started), shown in the right most side of the figure, the optimizer increases resource allocation to meet the VM's performance requirement. In this way, a



(a) Core and CPU frequency allocations by the optimizer for a video running on server CPUs.



(b) Assignement of video VMs by the scheduler.

Figure 3: Resource allocation for video applications by the scheduler and optimizer.

more power-efficient system is achieved through a more efficient assignment of the FPGA and the tuning of cores/frequency.

Furthermore, three assignation criteria for the FPGA have been tested:

- Random (RND) - The FPGA is assigned randomly to a VM.
- Earliest Deadline First (EDF) - The FPGA is assigned to the VM which runs the application with closest deadline.
- Highest Job First (HJF) - The FPGA is assigned to the VM with has more pending work to do, in relation with the deadline (i.e., it combines both the amount of work to do and the time to do it).

Table 3 shows that for all the FPGA assignation criteria, the HJF policy obtains the best time-to-complete the group of tasks and also the best performance in comparison with the amount of energy consumption. It is because the system is aware which VM should receive the FPGA to

Table 3: FPGA assignation criteria comparison

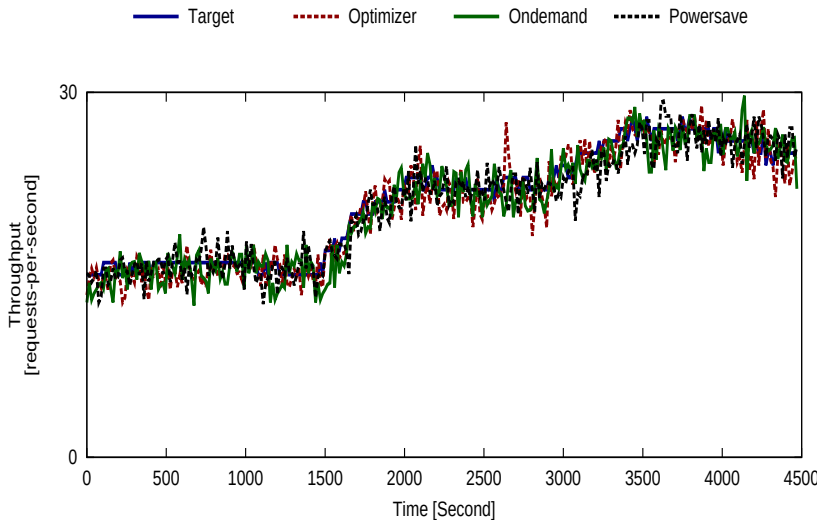
FPGA Assignation Criteria	Average Power (Watt)	Time (Sec)	Energy (Watts*Sec)	Frames-per-second-per-watt
RND	112,91	6079	686323	2.5
EDF	112,5	5913	665452	2.58
HJF	113	5543	626913	2.74

speed up the application depending on the load. And, due to the features of the FPGA, the impact of adding it is minimal.

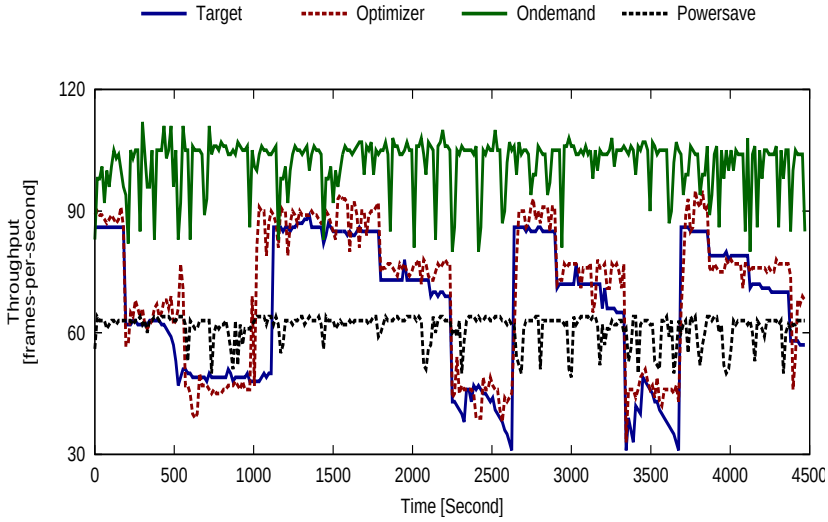
5.4.4. Comparison of the optimizer with linux governors

We compared the effectiveness of the optimizer, in terms of reduced power usage against the Linux *ondemand* and powersave CPU governors [27]. The ondemand governor manages the CPU frequency depending on system utilization: If current utilization is higher than an upper threshold (95%), the policy increases the frequency to the maximum. The powersave governor keeps the CPU at the lowest frequency irregardless of any changes in workload. A *work-conserving* resource allocation method [12] is used for both ondemand and powersave governors. In this allocation mode, the applications share all CPUs, i.e., they use any amount of CPU resource. For this experiment, the total number of cores assigned is less than the total available in the server to exclude the impact of resource contention among multiple running VMs.

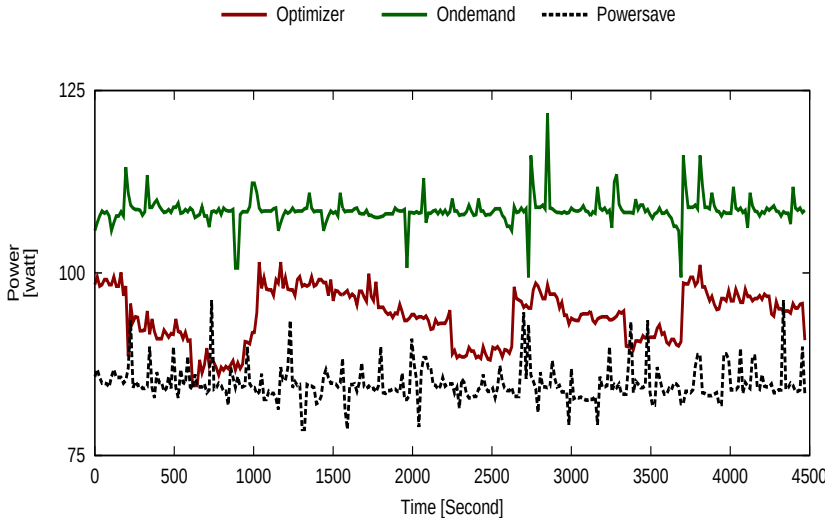
Figure 4 shows the results of these techniques in terms of applications' performance and energy usage of the server. As shown in Figure 4a all techniques meet the performance targets for RUBiS. Since the incoming load for RUBiS application is used as target, the measured throughput can never be higher than the target. The video applications (shown here for video1) intensively use any available CPU resource. As a result, the ondemand governor changes CPU frequency to the maximum due to high cpu usage. As shown for video1 in Figure 4b, this enables the application to achieve a throughput much higher than its performance targets. This also results in a higher server power consumption, shown in Figure 4c. The powersave governor consumes power the least since it sets the CPU statically to the minimum frequency. Given this fixed setup, the video application is unable to reach its performance targets at some times or exceed its targets other times. In addition, the governors are utilization-based, and are thus oblivious to the observed performance. The optimizer, on the other hand, uses performance and power models to select the right number of cores and CPU frequency, and minimizes power consumption while meeting performance targets. It achieves a 16% decrease in power consumption compared to the ondemand governor.



(a) RUBiS.



(b) Video1.



(c) Power.

Figure 4: Comparison of optimizer, powersave, and ondemand governor in terms of performance and power consumption under the two applications with different workloads.

5.4.5. Resource allocation by the optimizer

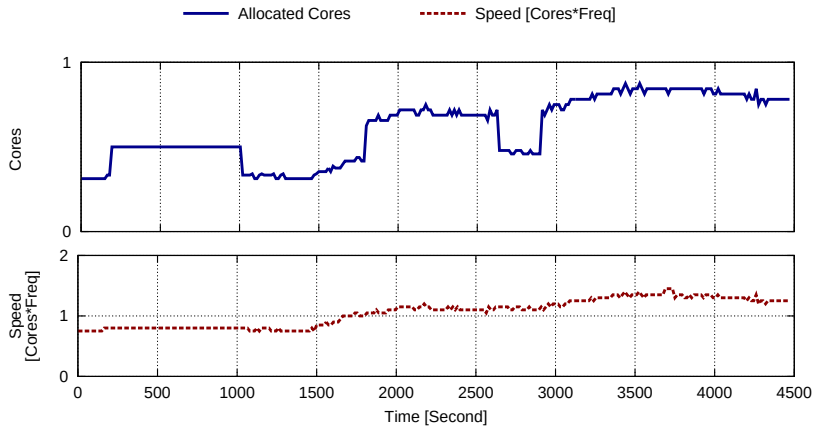
To help understand how the performance targets and server power consumption are achieved using the Optimizer, Figure 5 show the CPU and server frequency allocations to RUBiS and video1 applications. We use the same workloads and performance targets for the experiments shown in Figure 4. For a given CPU frequency, shown in 5c, the optimizer assigns the right number cores for RUBiS and video1 (shown in Figures 5a and 5b respectively) such that all of their targets is met and server power is minimized. It achieves this objective by using an algorithm, outlined in Algorithm 2, and is described as follows; if more cores are available than what is required for the lowest CPU frequency, the optimizer runs the server at that frequency to reduce power the most. By doing this, it improves energy efficiency at lower server utilization. However, due to factors like increased load, the number of cores required for a given frequency might exceed the cores available in the server. In that case the optimizer runs the server at a higher frequency in order to meet the performance targets. In summary, cloud providers can run a more energy efficient servers by using technique that dynamically adjust server CPU frequencies with methods that provide fine-grained resource allocations.

6. Related Work

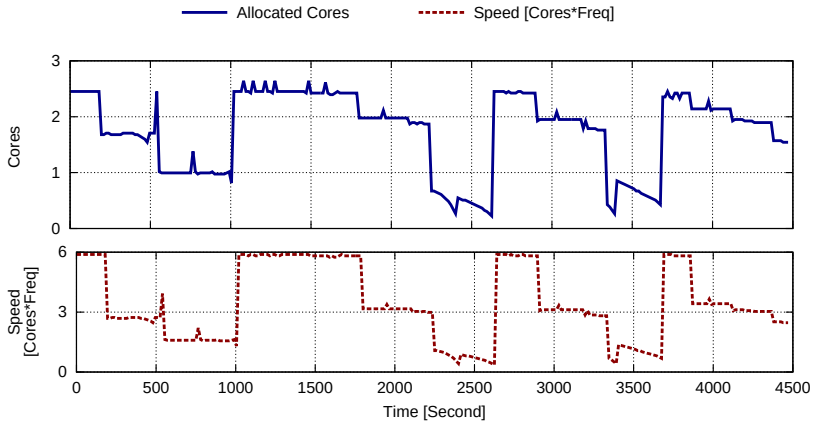
A brief overview of the work in fields related to this paper is given next.

6.1. FPGAs in Cloud as a code accelerators

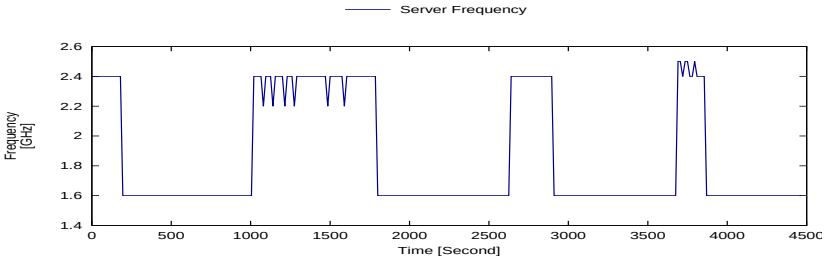
Catapult project [28] focus on designing a re-configurable fabric to accelerate portions of large-scale software services. Their results show great improvements in both the latency and the throughput of the service. Tsoi et al.[29] consider a heterogeneous computer cluster called Axel using devices such as FPGAs and GPUs. Axel is presented as a collaborative systems to improve the performance and the scalability of computer clusters. Byma et al. [30] present a hardware and software framework to enable the use of multiple virtual FPGAs as generic cloud resources on OpenStack. Chen et al. [5] propose a general framework for integrating FPGAs into the cloud based on a priority-based workload scheduling. Guohao et al. [31] design an online benefit-based scheduling algorithm for the use of FPGAs in cloud environment. Beach et al.[32] propose the use of a broker-based matchmaking system to select the most suitable accelerator device for a particular application kernel. The Application accelerators include GPUs, cell processors, and FPGAs among others. Chao et al.[33] propose a heterogeneous cloud framework using FPGA for big data genome sequencing for MapReduce[34]. All of these works



(a) RUBiS.



(b) Video1.



(c) server CPU frequency.

Figure 5: CPU allocation and CPU frequency scaling by the optimizer.

focus on the improvement of performance. However, none of the aforementioned works address the integration of FPGA with regarding to energy savings.

Ouyang et al. [35] show an accelerated neural networks using FPGAs in the datacenter and observe better performance and reduced power consumption. Coutinho et al. [36] present an heterogeneous architecture that integrates deferent types of devices such as FPGAs, programmable routers, and SSDs to reduce the power consumption in a cloud platform. In addition heterogeneous programming platforms are proposed. But the authors do not consider any strategy beyond of the use of more suitable elements for each application in order to save energy.

6.2. Dynamic Frequency

There are many works on power management mechanism that use DVFS technique [37, 38, 39, 40, 41]. Q. Wu et al. [37] propose a dynamic-compiler-driven runtime voltage and frequency optimizer to improve energy efficiency. Varma et al. [38] investigate the use of feedback loops for DVFS and consider system utilization for decision making. M. Kondo [39] propose algorithm for improving total instruction throughput, fairness, and energy efficiency of single chip multiprocessors These works do not consider the observed performance to inform the DVFS decision. In addition, the above methods cannot be directly applied in cloud environment.

6.3. Elasticity

Dawoud et al. [42] compare vertical elasticity with respect to horizontal elasticity and focus on vertical elasticity. They experimentally demonstrated that a fine-grained vertical elastic VM architecture consumes less resources and avoids scaling up overhead while guaranteeing SLAs. Concerning CPU elasticity, Kalyvianaki et al. [43] design a controller using Kalman filters to control allocation based on CPU utilization. The CPU utilization is tracked and the allocations are updated accordingly. Padala et al. [12] applied a proportional controller to dynamically adjust CPU shares to VM-based multi-tier web applications. These works do not consider power usage in decision making.

In contrast to the discussed existing works, we combine FPGA scheduling, vertical scaling of CPU and Frequency scaling for optimized power and performance in cloud environment.

7. Conclusions and Future Work

In this paper, we present an approach for power optimization that combines an optimizer for an optimal configuration of cores and CPU frequency and scheduling algorithm for the management of FPGA in heterogeneous cloud data centers. We develop power and performance models based on the number of CPUs, and CPU frequency. The models are used by the optimizer to select the best configuration. The scheduler selects the application the FPGA should run based on application deadline and power-efficiency. Our experimental evaluation using multiple applications shows the effectiveness of the proposed solution in power savings when both the optimizer and the scheduler are used.

For future work, we plan to share the FPGA among multiple VMs, by using techniques that virtualize the FPGA. Another direction is to allow memory elasticity in the system in order to achieve even more power efficient results.

Acknowledgement

This work was partially supported by the Swedish Research Council (VR) for the project Cloud Control, the Spanish Government under Grant TIN2012-38341-C04-04 and the Ecuadorian Government under the SENESCYT Scholarships Project. We also thank Eddie Waldrob for his feedback regarding optimizer formulation.

References

[1] Steve Crago, Kyle Dunn, Patrick Eads, Lorin Hochstein, Dong-In Kang, Mikyung Kang, Devendra Modium, Karandeep Singh, Jinwoo Suh, and John Paul Walters. Heterogeneous cloud computing. In *Cluster Computing (CLUSTER), 2011 IEEE International Conference on*, pages 378–385. IEEE, 2011.

[2] Suhaib A Fahmy and Kizheppatt Vipin. A case for fpga accelerators in the cloud.

[3] Srinidhi Kestur, John D Davis, and Oliver Williams. Blas comparison on fpga, cpu and gpu. In *VLSI (ISVLSI), 2010 IEEE computer society annual symposium on*, pages 288–293. IEEE, 2010.

[4] J. Becker, M. Huebner, and M. Ullmann. Power estimation and power measurement of xilinx virtex fpgas: trade-offs and limitations. In *Integrated Circuits and Systems Design, 2003. SBCCI 2003. Proceedings. 16th Symposium on*, pages 283–288, Sept 2003.

[5] Fei Chen, Yi Shan, Yu Zhang, Yu Wang, Hubertus Franke, Xiaotao Chang, and Kun Wang. Enabling fpgas in the cloud. In *Proceedings of the 11th ACM Conference on Computing Frontiers*, CF '14, pages 3:1–3:10, New York, NY, USA, 2014. ACM.

[6] Amazon: Amazon elastic compute cloud. http://aws.amazon.com/ec2/. Accessed: November 2015.

[7] Ewnetu Bayuh Lakew, Cristian Klein, Francisco Hernandez-Rodriguez, and Erik Elmroth. Performance-based service differentiation in clouds. In *Cluster, Cloud and Grid Computing (CCGrid), 2015 15th IEEE/ACM International Symposium on*, pages 505–514. IEEE, 2015.

[8] Rajkumar Buyya, Anton Beloglazov, and Jemal Abawajy. Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges. *arXiv preprint arXiv:1006.0308*, 2010.

[9] Luiz André Barroso and Urs Hölzle. The case for energy-proportional computing. *Computer*, 40(12):33–37, December 2007.

[10] A Starikovskiy V Palladi and Alexey Starikovskiy. The ondemand governor: past, present and future. In *Proceedings of Linux Symposium*, volume 2, pages 3–3, 2001.

[11] Yongqiang Gao, Haibing Guan, Zhengwei Qi, Bin Wang, and Liang Liu. Quality of service aware power management for virtualized data centers. *Journal of Systems Architecture*, 59(4):245–259, 2013.

[12] Pradeep Padala, Kai-Yuan Hou, Kang G Shin, Xiaoyun Zhu, Mustafa Uysal, Zhikui Wang, Sharad Singhal, and Arif Merchant. Automated control of multiple virtualized resources. In *Proceedings of the 4th ACM European conference on Computer systems*, pages 13–26. ACM, 2009.

[13] Yongqiang Gao, Haibing Guan, Zhengwei Qi, Bin Wang, and Liang Liu. Quality of service aware power management for virtualized data centers. *Journal of Systems Architecture - Embedded Systems Design*, 59(4-5):245–259, 2013.

[14] S.K. Tesfatsion, E. Wadbro, and J. Tordsson. A combined frequency scaling and application elasticity approach for energy-efficient cloud computing. *Sustainable Computing: Informatics*

and Systems, 4(4):205 – 214, 2014. Special Issue on Energy Aware Resource Management and Scheduling (EARMS).

[15] Xilinx ML605. Virtex-6 FPGA ML605 Evaluation Kit. `http://www.xilinx.com/publications/prodmktg/ml605productbrief.pdf`, 2015. [Online; accessed 21-December-2015].

[16] PowerMeterStore. Power Meter Store. Web page at url-http://www.powermeterstore.com/p1206/wattsuppro.php, Last access: 4th February, 2015. [On-line; accessed 21-December-2015].

[17] KVM. Kernel Based Virtual Machine (KVM), Last access: 29th July, 2014. Web page:http://www.linux-kvm.org/.

[18] Intel. Intel Virtualization Technology of Directed I/O, Architecture Specification, Rev. 2.2. `http://www.intel.com/content/dam/www/public/us/en/documents/product-specifications/vt-directed-io-spec.pdf`, 2013. [Online; accessed 21-December-2015].

[19] M. Jacobsen and R. Kastner. RIFFA 2.0: A reusable integration framework for FPGA accelerators. In *23rd International Conference on Field programmable Logic and Applications, FPL 2013*, pages 1–8, 2013.

[20] RUBiS: Rice University Bidding System, Last access: 29th July, 2014. cloud computing online:http://rubis.ow2.org.

[21] I Yasri, NH Hamid, and VV Yap. Performance analysis of FPGA based Sobel edge detection operator. In *Electronic Design, 2008. ICED 2008. International Conference on*, pages 1–4. IEEE, 2008.

[22] Kunwadee Sripanidkulchai, Sambit Sahu, Yaoping Ruan, Anees Shaikh, and Chitra Dorai. Are clouds ready for large distributed applications? *ACM SIGOPS Operating Systems Review*, 44(2):18–23, 2010.

[23] Linux Ubuntu. Ubuntu Manuals. `http://manpages.ubuntu.com/manpages/hardy/man1/cpufreq-set.1.html`, 2013. [Online; accessed 21-December-2015].

[24] Rober M. Love. Taskset. `http://linuxcommand.org/man_pages/taskset1.html`, 2003. [Online; accessed 05-January-2016].

[25] KVM. How to assign devices with VT-d in KVM. `http://www.linux-kvm.org/page/How_to_assign_devices_with_VT-d_in_KVM`, 2015. [Online; accessed 05-January-2016].

[26] Wikilibros. Qemu/monitor. Web page at `https://es.wikibooks.org/w/index.php?title=QEMU/Monitor\&oldid=225320`, 2014. On line; accessed 15-december-2015.

[27] Venkatesh Pallipadi and Alexey Starikovskiy. The ondemand governor: past, present and future. In *Proceedings of Linux Symposium*, volume 2, pages 223–238, 2006.

[28] Andrew Putnam, Adrian M Caulfield, Eric S Chung, Derek Chiou, Kypros Constantinides, John Demme, Hadi Esmaeilzadeh, Jeremy Fowers, Gopi Prashanth Gopal, Jordan Gray, et al. A reconfigurable fabric for accelerating large-scale datacenter services. In *Computer Architecture (ISCA), 2014 ACM/IEEE 41st International Symposium on*, pages 13–24. IEEE, 2014.

[29] Kuen Hung Tsoi and Wayne Luk. Axel: A heterogeneous cluster with fpgas and gpus. In *Proceedings of the 18th Annual ACM/SIGDA International Symposium on Field Programmable Gate Arrays*, FPGA '10, pages 115–124, New York, NY, USA, 2010. ACM.

[30] Byma, S. and Steffan, J.G. and Bannazadeh, H. and Leon-Garcia, A. and Chow, P. FPGAs in the Cloud: Booting Virtualized Hardware Accelerators with OpenStack. In *Field-Programmable Custom Computing Machines (FCCM), 2014 IEEE 22nd Annual International Symposium on*, pages 109–116, 2014.

[31] Guohao Dai, Yi Shan, Fei Chen, Yu Wang, Kun Wang, and Huazhong Yang. Online scheduling for fpga computation in the cloud. In *Field-Programmable Technology (FPT), 2014 International Conference on*, pages 330–333, Dec 2014.

[32] Thomas H Beach, Omer F Rana, and Nicholas J Avis. Integrating acceleration devices using cometcloud. In *Proceedings of the first ACM workshop on Optimization techniques for resources management in clouds*, pages 17–24. ACM, 2013.

[33] Chao Wang, Xi Li, Peng Chen, Aili Wang, Xuehai Zhou, and Hong Yu. Heterogeneous cloud framework for big data genome sequencing. *Computational Biology and Bioinformatics, IEEE/ACM Transactions on*, 12(1):166–178, Jan 2015.

[34] J. Dean and S. Ghemawat. MapReduce: Simplified Data Processing on Large Clusters. In *USENIX OSDI'04: 6th Symposium on Operating Systems Design and Implementation*, 2004.

[35] Jian Ouyang, Shiding Lin, Wei Qi, Yong Wang, Bo Yu, and Song Jiang. Sda: software-defined accelerator for large-scale dnn systems. In *Hot Chips: A Symposium on High Performance Chips*, 2014.

[36] JGF Coutinho, O Pell, E O'Neill, P Sanders, J McGlone, P Grigoras, W Luk, and C Ragusa. Harness project: Managing heterogeneous compute resources for a cloud platform. In *Proc. the 10th International Symposium on Applied Reconfigurable Computing (ARC)*, 2014.

[37] Qiang Wu, Margaret Martonosi, Douglas W Clark, Vijay Janapa Reddi, Dan Connors, Youfeng Wu, Jin Lee, and David Brooks. Dynamic-compiler-driven control for microprocessor energy and performance. *IEEE Micro*, (1):119–129, 2006.

[38] Ankush Varma, Brinda Ganesh, Mainak Sen, Suchismita Roy Choudhury, Lakshmi Srinivasan, and Bruce Jacob. A control-theoretic approach to dynamic voltage scheduling. In *Proceedings of the 2003 international conference on Compilers, architecture and synthesis for embedded systems*, pages 255–266. ACM, 2003.

[39] Masaaki Kondo, Hiroshi Sasaki, and Hiroshi Nakamura. Improving fairness, throughput and energy-efficiency on a chip multiprocessor through dvfs. *ACM SIGARCH Computer Architecture News*, 35(1):31–38, 2007.

[40] Tibor Horvath, Tarek Abdelzaher, Kevin Skadron, and Xue Liu. Dynamic voltage scaling in multi-tier web servers with end-to-end delay control. *Computers, IEEE Transactions on*, 56(4):444–458, 2007.

[41] Mootaz Elnozahy, Michael Kistler, and Ramakrishnan Rajamony. Energy conservation policies for web servers. In *Proceedings of the 4th conference on USENIX Symposium on Internet Technologies and Systems-Volume 4*, pages 8–8. USENIX Association, 2003.

[42] Wesam Dawoud, Ibrahim Takouna, and Christoph Meinel. Elastic virtual machine for fine-grained cloud resource provisioning. In *Global Trends in Computing and Communication Systems*, pages 11–25. Springer, 2012.

[43] Evangelia Kalyvianaki, Themistoklis Charalambous, and Steven Hand. Self-adaptive and self-configured cpu resource provisioning for virtualized servers using kalman filters. In *Proceedings of the 6th international conference on Autonomic computing*, pages 117–126. ACM, 2009.