Energy-efficient Task Scheduling in Data Centers

Yousri Mhedheb and Achim Streit
Steinbuch Centre for Computing, Karlsruhe Institute of Technology, Karlsruhe, Germany

Keywords: Cloud Computing, Distributed Systems, Energy Efficiency, Power Management, DVFS.

Abstract: A data center is often also a Cloud center, which delivers its computational and storage capacity as services. To enable on-demand resource provision with elasticity and high reliability, the host machines in data centers are usually virtualized, which brings a challenging research topic, i.e., how to schedule the virtual machines (VM) on the hosts for energy efficiency. The goal of this Work is to ameliorate, through scheduling, the energy-efficiency of data center. To support this work a novel VM scheduling mechanism design and implementation will be proposed. This mechanism addresses on both load-balancing and temperature-awareness with a final goal of reducing the energy consumption of a data centre. Our scheduling scheme selects a physical machine to host a virtual machine based on the user requirements, the load on the hosts and the temperature of the hosts, while maintaining the quality of the service. The proposed scheduling mechanism on CloudSim will be finally validated, a well-known simulator that models data centers provisioning Infrastructure as a Service. For a comparative study, we also implemented other scheduling algorithms i.e., non power control, DVFS and power aware ThrMu. The experimental results show that the proposed scheduling scheme, combining the power-aware with the thermal-aware scheduling strategies, significantly reduces the energy consumption of a given Data Center because of its thermal-aware strategy and the support of VM migration mechanisms.

1 INTRODUCTION

Cloud Computing (Mell and Grance, 2011; Wang et al., 2010) has emerged as a novel computing paradigm and this paradigm is so attracting that doing things "in the cloud" has become a common catch phrase these days. For that reason, increasing number of Cloud infrastructures (GAE; Windows; rackspace,) have been established after the first computing Cloud – the Amazon Elastic Compute Cloud (EC2) (AmazonEC2). These Cloud providers are also data center operators, where computing and storage facilities are now provisioned as Cloud services. With a business model of "pay as you use", Cloud Computing offers services not only in hardware and storage, but also software, applications as well as network. The power of Cloud Computing is that users can access the computing resources via Internet using a thin-client such as a Web browser. Additionally, Cloud Computing shows the advantages in elasticity, system management, cost-efficiency, customized environment and on-demand resource provision (Wang and Khan, 2013; Menzel and Ranjan, 2012). Currently, data centers deliver mainly three kinds of Cloud services, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (IaaS) (Mell and Grance, 2011; Kahn et al., 2013) . IaaS targets on an on-demand provision of the computational resources, normally in the form of virtual machines (VM). Customers can install software packages on the machines to establish their own computing environments. SaaS provides the customers a feature of using the provider's applications in the form of Web services that run centrallyon the computing infrastructure of the service provider. PaaS targets on a complete platform including the hardware and the application development environment, which allows the users to develop their applications using the provider's computing platform. Existing Clouds are mostly IaaS-featured. Examples include the Amazon EC2, SmartCloud, Flexiscale as well as the academic implementations like Eucalyptus (Nurmi et al., 2008), OpenNebula (Sotomayor et al., 2008), Nimbus (Keahey and Freeman, 2008) and OpenStack (Openstack, 2013). Independent of which services a data center offers, data centers commonly require a virtualized hardware platform for either IaaS or running the provisioned software (SaaS) or user applications in the case of PaaS. Therefore, virtual machines are the base to deliver any cloud services. Since a Cloud center is equipped with thousands of physical hosts, where each of them is potentially the target of a re-

Mhedheb, Y. and Streit, A.
Energy-efficient Task Scheduling in Data Centers.
In *Proceedings of the 6th International Conference on Cloud Computing and Services Science (CLOSER 2016) - Volume 1*, pages 273-282
ISBN: 978-989-758-182-3

quested incoming virtual machine, there arises an important issue of scheduling virtual machines on the physical hosts. A traditional approach of scheduling virtual machines is a kind of FIFO approaches, where all hosts are contained in a list and the first physical machine that matches the requirement of the VM is selected as the target host.

Issue with scheduling exists in various scenarios, like task scheduling in parallel and distributed systems (Mhedheb et al., 2013; Ranjan et al., 2005) and job scheduling in computing Grids (Ranjan et al., 2006; Kolodziej et al., 2012; Kolodziej et al., 2013). Various algorithms have been proposed by researchers including those focusing on Energy Consumption. In the field of Cloud Computing, Cloud providers are surely expecting a low operation overhead. Considering the importance of energy saving in the field of Cloud Computing a specific algorithm for scheduling the virtual machines on the Cloud was here developed. This algorithm first performs an initial scheduling and then inspects the change of workload and temperature on the host. In case of over-loading or over-heating, the VM is migrated to another host thereby to avoid hot spots with respect to load and temperature. We implemented this algorithm on CloudSim (Calheiros et al., 2011), a well-known simulation platform for research work in Cloud Computing. In addition to the proposed scheduling scheme we also implemented several other scheduling policies for a comparative study. The experimental results show that the proposed approach performs better than others with the lowest energy consumption while the service quality is granted in most cases.

The rest of this paper is organized as follows: Section 2 introduces the related work in energy-aware scheduling algorithms. Section 3 describes the concept of the proposed approach followed by the implementation details in Section 4. Section 5 depicts the evaluation details and Section 6 concludes the paper with a brief summary and future directions.

2 RELATED WORK

Task scheduling has been investigated for many years and is currently still a hot topic in various research domains. For example, Cluster Computing relies on good scheduling technologies for achieving the performance of parallel applications; Grid Computing requires efficient job scheduler to best utilize the Grid resources and thereby reduce the job waiting time. Over the last years, many research works have been performed for investigating the scheduling strategies on different systems with features like load-balancing and energy-awareness.

As computing systems consume more and more energy, increasing research works on task scheduling show the feature of energy-awareness (Hsu and Feng, 2005; Wang et al., 2011). Most of these research works adopt the technique of Dynamic Voltage and Frequency Scaling (DVFS). DVFS has been proven to be a feasible solution for reducing the processor energy consumption (Hsu and Feng, 2005). By lowering the processor clock frequency and supply voltage during some time slots (for example, idle or communication phases), energy consumption can be significantly reduced with a slight performance loss. Therefore, the DVFS technique has been applied in various domains, such as the high performance computing areas and large data centers, to reduce the energy consumption while gaining high reliability and availability. As the virtualization technology is getting hot, the problem of virtual machine scheduling has also been studied. The authors of (Kim et al., 2008) implement a guest-aware priority-based scheduling scheme, which is specifically designed to support latency-sensitive workloads. The proposed scheduling scheme prioritizes the virtual machines to be allocated by using the information about priorities and status of guest-level tasks in each VM. It preferentially selects the VMs that run latency-sensitive applications to be scheduled and in this way to reduce the response time to the I/O events of latency-sensitive workloads.

The authors of (Wang et al., 2012) propose a novel VM scheduling algorithm for virtualized heterogonous multicore architectures. The algorithm exploits the core performance heterogeneity to optimize the overall system energy efficiency. It uses a metric termed energy-efficiency factor to characterize the power and performance behavior of the applications hosted by VMs on different cores. The VM's energy-efficiency factors are calculated and based on the values virtual machines are mapped to heterogeneous cores with a final goal of maximizing the energy efficiency of the entire system. The authors of (Takouna et al., 2011) also address on the VM scheduling of heterogeneous multicore machines. A scheduling policy is designed to schedule each virtual machine to an appropriate processing core based on the performance sensibility to the CPU clock frequency and the performance dependency on the host.

In addition to the research work on VM schedulers of general purposes, the specific problem of VM scheduling on the Cloud has also been addressed over the last years.

The work presented in (Fang et al., 2010) develops a two-layer scheduling model, where the first layer creates the description of resource requirement of a virtual machine, while the second layer takes care of the VM scheduling on the host. By allocating the VMs to the hosts that closely meet the resource requirement this approach tries to better use the Cloud hardware resources. The approach was evaluated on CloudSim. The authors of (Li et al., 2011) propose a dynamic round-robin scheduling scheme for deploying and migrating virtual machines to or across the servers on the Cloud. The scheme uses two scheduling rules, whereby the first rule avoids allocating additional virtual machines to a retiring physical machine that will be shut down while the second rule speeds up the consolidation process. The main goal of this approach is to reduce the number of used physical machines for saving energy. The work in (Hu et al., 2010) propose a strategy for VM scheduling on the Cloud with load balancing. The scheduling decision is based on the historical information and current state of the system. Authors of (Knauth and Fetzer, 2012) proposed a scheduler, which schedules the virtual machines based on the knowledge about the duration of timed instances to optimize the virtual to physical machine assignment. The main goal of this scheduler is to reduce the cumulative machine uptime and thereby save the energy consumption. The work in (Beloglazov and Buyya, 2012) is also one of the few approaches that deal with energy-aware scheduling of Cloud-based resources. The authors implemented a simulation environment based on CloudSim (Calheiros et al., 2011) to evaluate different power-aware scheduling policies for VM running on a large scale Cloud centre. Furthermore, they show how the VM migration and VM pining techniques can optimize the load-balancing and the total energy consumption of a data center.

Our work is similar to the last related work. However, both power and thermal-aware scheduling policies will be combined to reduce the energy consumption. More importantly, the extension of this work to support using temperature constraints as new scheduling parameters. The evaluation on CloudSim has shown the improvement of this approach over the existing one in terms of saving energy consumption. The experimental results will be given after the description of the proposed scheduling algorithm and its implementation.

3 THERMAL AWARE VM SCHEDULING

Modern processors have a tolerance limit to the on-chip temperature named case temperature. A higher temperature over this limit may reaches the Junction temperature and not only increases the energy consumption but also may result in some defect in the hardware i.e., silicon chip is going to melt the internals. The design goal of our scheduling scheme is to be aware of the changing temperature of the processing cores to avoid crossing the limit. Our thermal-aware scheduling concept is based on several existing strategies, which are applied for the requirement of different scheduling scenarios. These strategies are individually integrated in our framework at the run-time based on the current load and temperature state. The main tasks of our scheduler, called Thermal aware Scheduler (ThaS), are the following:

3.1 Power-aware Energy Management

Applying the DVFS technique for power management. Since the power consumption depends on the CPU usage, this metric has to be measured before each VM scheduling step in the whole execution process in order to calculate the current consumed energy by the CPU.

3.2 Thermal-aware Energy Management

The main focus of our ThaS scheme is to schedule virtual machines with respect to the temperature of the processors. Such a scheduling strategy needs a temperature model to describe the property of this parameter while applications (here virtual machines) are running. The lumped RC thermal model (Skadron et al., 2002) will be used due to its simplicity. According to (Skadron et al., 2002), for an integrated circuit at the die level, heat conduction is the dominant mechanism that determines the temperature. There exists a well-known duality between heat transfer and electrical phenomena. Any heat flow can be described as a *current* and the passing of this heat flow through a thermal *resistance* leads to a temperature difference equivalent to a *voltage*. The RC model can be described with $T_{amb} = RC\frac{dT}{dt} + T - RP$, where C stands for the thermal capacitance of the processor, R for the thermal resistance, P for the processor dynamic energy consumption, and T_{amb} for the ambient temperature. This model is however limited to a single-core processor. Therefore, ThaS supports now only single-core machines.

3.3 Migration of Virtual Machines

Our ThaS scheme inspects the change of load and temperature on a host machine. In case that a host is about to be over-heated or over-loaded it performs VM migration to avoid critic scenarios. Migration or live migration refers to the process of moving a running virtual machine or application from one physical machine to another. In addition to the machine image, the data storage and network connectivity of the virtual machines have also to be transferred from the source host to the destination. The proposed scheduling scheme implements this kind of VM migration and runs the entire migration process transparently to the user. The migration contains several steps, including the Push phase, the Stop phase, the Copy phase and the Pull phase. The VM migration can take a long time when a VM has a large amount of memory. In order to save the unnecessary energy consumption during the VM migration, two strategies in ThaS are implemented: i) the Pining strategy that allows the allocation of multiple VMs on the same host to free other physical hosts; ii) the energy-save-modus strategy that sets the unused hosts (or CPUs) in the idle mode.

ThaS decides during the runtime which VM shall be allocated on which host. It works in the following way: As a starting point a VM request coming from the user is scheduled on a physical host based on the traditional round-robin scheme. In order to make a decision, ThaS calls the thermal model and the power model to acquire all scheduling parameters including the current CPU temperature, the CPU usage for each host, the data center configuration and the application (VM) requirements. In case that a physical host approaches to the critical temperature (Temperature_threshold) or the critical CPU utilization value (Utilization_threshold), ThaS looks for another host with better temperature or load criteria and migrates the running VM from the source host to the destination host. Therefore, our ThaS schedules the virtual machines not only for minimizing the energy consumption but also for load-balancing. ThaS is designed to be integrated in a Cloud middleware to replace its original one, usually a round-robin scheme that allocates the VMs to each host equally in a cyclic order. ThaS implements an interface between the VMM (hypervisor) and the virtual machines in a Cloud centre. It replaces the conventional scheduling algorithm of a hypervisor to map a virtual machine request to a physical machine with consideration of the load and temperature on the hosts. The deployment, such as start, stop, migration, etc., of the virtual machines on the physical host remains the task of the hypervisor. In this way, our scheduler acts as an allocation decision component for the hypervisor.

4 IMPLEMENTATION

A Java-based simulation environment, to verify the concept and to validate the functionality of the proposed scheduling strategies for ThaS was implemented. The prototypical implementation is based on a well-known simulator, CloudSim (Calheiros et al., 2011), for research work on the Cloud. CloudSim models large scale data centers with internal broker, sensors, physical host, as well as virtual machines. CloudSim also provides the functionality to configure data center resources. This allows us to model cloud providers with different system scale and hardware resources.

By an incoming VM request our scheduler ThaS starts first with an initial task of allocating the VM to one of the hosts that meet the requirement described in the request. In the following, it performs the runtime tuning with respect to load and temperature.

The first task of ThaS for runtime tuning is to detect the critical hosts with either higher temperature or workloads. The following pseudo-code illustrates how the scheduler performs this task. As can be seen in the algorithm, the scheduler goes through all the available hosts to find the hosts, whose temperature and CPU usage exceed the specified thresholds. The detected hosts are then marked as candidates for VM migration and added to the list 'Migrating From Hosts'. This list is applied in the second step as the input list.

Algorithm 1: Algorithm for detecting critical hosts.

```
Input: HotList, VmList
Output: MigratingFromHosts
 1: for each host in hostlist do
 2:     if isHostOverThresholdTemperature (host) then
 3:         $overThresholdTempHosts \leftarrow addhost$
 4:     else
 5:         if isHostOverThresholdUtilization ( host ) then
 6:             $overUtilizedHosts \leftarrow addhost$
 7:         end if
 8:     end if
 9:     $overHosts \leftarrow overThresholdTempHosts + overUtilizedHosts$
10:     $MigratingFromHosts \leftarrow overHosts$
11: end for
```

In the second step, the list of critical hosts, which has been created by the scheduler in the first step, is processed again for finding the VMs running on them. These virtual machines are the concrete candidates for migration. The candidate VMs are then sorted by their CPU usage. The VMs with minimal CPU usage are marked with a higher priority of migration for the reason of not to bring high workload on the target host, thus to avoid possible further migrations. For the same reason the scheduler must also ensure that the temperature on the target host does not exceed the threshold. At the end of processing, this scheduling step creates a list 'VMs to Migrate List' that contains all VMs, which are the actual migration objects.

Algorithm 2: Algorithm for detecting of migratable VMs.

```
Input: MigratingFromHosts
Output: VMsToMigrateList
 1: for each host in MigratingFromHosts do
 2:     while true do
 3:         $vm \leftarrow getVmToMigrate(host)$
 4:         if then$vm = Null$
 5:             break
 6:         end if
 7:         $VMstoMigrateList \leftarrow addvm$
 8:         $host \leftarrow deallocatevm$
 9:         if then$!(isHostOverThresTemp(host)$ & $isHostOverThresholdUtilisation(host))$
10:             break
11:         end if
12:     end while
13: end for
```

The third and last step of the VM migration procedure is to find an appropriate target host for hosting the migration objects in the list created in the last step. Here, the workload requirements (e.g., needed

Algorithm 3: Algorithm for detecting of target Hosts.

```
Input: MigratingFromHosts, VMsToMigrateList
Output: MigrationMap
 1: $MigrationMap \leftarrow null$
 2: $vmstoMigrateList.SortDecreasingCPUUtilisation$
 3: for each VM in VMsToMigrateList do
 4:     $allocatedHost \leftarrow null$
 5:     $minPower \leftarrow Max$
 6:     for each host not in MigratingFromHost do
 7:         if host has enough resources for vm then
 8:             $power \leftarrow estimatePower(host, vm)$
 9:         end if
10:         if then$power < minPower$
11:             $allocatedHost \leftarrow host$
12:             $minPower \leftarrow power$
13:         end if
14:         if then$allocatedHost != NULL$
15:             allocate vm to allocatedHost
16:         end if
17:     end for
18:     $MigrationMap \leftarrow add(vm, allocatedHost)$
19: end for
```

resources) have to be taken into account. Our ThaS first observes the temperature on the destination host. If this temperature is below the threshold value (Temperature_threshold) and the requirement of the VM is fulfilled, the observed host is selected as the target host. In case that several target hosts are found, the one with the minimum energy consumption is chosen for hosting the VM to be migrated.

To further improve the scheduling efficiency in terms of energy consumption, a reallocation of VMs is designed in the proposed scheduler. This scheme concerns the hosts that are underutilized. In case that the CPU usage of a physical host is below the minimal value, the VMs on it are migrated to other hosts. The underutilized hosts are then set in a sleep mode for the purpose of saving energy. The idle hosts are not candidates of destination host for VM migration.

5 EVALUATION RESULTS

5.1 Experimental Setup

ThaS is evaluated using a simulation environment CloudSim, which models large data centers provisioning computing infrastructures as services. Simulation is a widely applied approach for validating novel concepts and implementations. For this work a simulation platform is especially important because of aiming to evaluate the thermal-aware scheduling algorithm on a large virtualized data center infrastructure. For this purpose the status information of physical hosts on a Cloud centre is needed. In this case, validation on a real Cloud infrastructure would be extremely difficult or not possible due to the missing information, which inhibits us to perform different experiments in order to examine the full functionality of the implemented scheduler and the impact of the scheduling strategies. In contrast, CloudSim implements a view of infinite computing resources and it is easy to extend this simulator for both monitoring and additional functionalities.

We modeled a scenario of scheduling the virtual machine requests of three days with a randomly generated VM request every five minutes. A VM request contains four requirement attributes, i.e., the number of CPU cores, the CPU frequency, the RAM size and the bandwidth. The properties of the modelled VM types are described in Table 1. As shown in the table, different VMs with various values in MIPS and RAM size to model real scenarios are used. With assuming that all VMs run on single core machines. The bandwidth and VM size for all simulated virtual machines are set as 100 Mbit/s and 2.5 GB individually.

Table 1: Virtual machine configuration.

VM Type	VM_MIPS	VM_RAM [MB]
1	500	613
2	1000	1740
3	2000	1740
4	2500	870
VM_Cores	VM_BW [Mbit/s]	VM_Size [GB]
1	100	2.5

Table 2: Thermal constants.

Thermal parameter	Value	Unit
Initial CPU temperatur (T_{init})	318	Kelvin
Ambiente temperatur (T_{amb})	308	Kelvin
Case temperatur (T_{case})	353	Kelvin
Thermal capacity (C_{th})	340	J/K
Thermal resistance (R_{th})	0.34	K/W

Table 3: Simulated physical machines.

Server Host Type	Proliant_G4	Proliant_G5
Host_MIPS	1860	2660
Host_Cores	1	1
Host_RAM [MB]	2048	4096
Host_BW [Gbit/s]	1	1
Host_Storage [TB]	1	1

As mentioned, a lumped RC thermal model is applied to describe the relationship between the temperature of the processor and its dynamic energy consumption. Table 2 depicts the used thermal constants for the experiments. The values in the table are typical values of a single core CPU obtained from (Hotspot,). CloudSim was here configured for modeling two data centers with one equipped with 50 and the other 250 different hosts. A half of the hosts are modelled as the HP ProLiant G4 servers. The other half is modeled as the HP ProLiant G5 servers. Table 3 shows the setup parameters of the physical machines simulated for the testing. The frequency of each core on the HP ProLiant G4 Server is 1860 MIPS and for the HP ProLiant G5 Server the value is 2660 MIPS. Each server is modeled with a connection of 1 GB/s bandwidth. The corresponding power model used by each server is gathered from SpecPower08 (spec08). The simulation of less powerful CPUs is advantageous for a better evaluation of the effect of the VM migration because few workload is required to result in the overload of a server. The RAM size for both machines are configured to 2GB and 4GB individually.

The final goal of our ThaS approach is to reduce the energy consumption of data centers. To evaluate ThaS with respect to this performance metric a model to compute the energy was needed consumption of the host machines (i.e., servers) in a data centre. Recent studies (Beloglazov and Buyya, 2010) show that the energy consumption by servers can be accurately described by a linear relationship between the energy consumption and the CPU utilization, even when the DVFS scheme is applied. The reason lies in the limited number of states that can be set to the frequency and voltage of the CPU and the fact that voltage and performance scaling are not applied to other system components, such as the memory and the network interface. Moreover, these studies show that on average an idle server consumes approximately 70% of the energy consumed when it is fully utilized. Therefore, the energy consumption of a server E can be modelled as a function of the CPU utilization u using the form $E(u) = (1-u)*0.7*E_{max} + u*E_{max}$, where E_{max} is the power consumption of a server in full use. For example, a server of a data centre is 90% used and 10% idle. In this case, the energy consumption of the server is $0.9*E_{max}$ in busy status and $0.1*0.7*E_{max}$ in idle status. The total energy consumption of the server is the sum of the two parts.

The implemented scheduler relies on two thresholds for migration decisions, one is the Temperature_threshold and the other is the Utilization_threshold. In order to have a simulation-based evaluation applicable, it is important to perform experiments with workload traces of a real system. The simulation experiments (Beloglazov and Buyya, 2010) have demonstrated that energy consumption from a CPU utilization rate of 90% rises very quickly. Therefore we have chosen a value of 0.9 as the Utilization_threshold. If the CPU utilization reaches this threshold, the VMs running on it may be migrated to another host with lower CPU usage. The selection of the Temperature_threshold is not as easy as the Utilization_threshold. In the following subsection is demonstrated how to achieve an optimal threshold of 343 Kelvin with a trade-off between power consumption and SLA violation.

To validate ThaS several other scheduling schemes for a comparative study were also implemented. The first one is called Non_power_control, which schedules the virtual machines without considering the CPU usage. We use this scheme to study the energy consumption of a data center with full CPU utilization. The second scheme is DVFS. It schedules tasks on the basis of the CPU voltage and frequency. It relies on the information from the CPU performance and power model to set the priorities for the VM placement. The energy consumption is calculated as a function of the CPU usage and is regulated automatically and dynamically based on DVFS. The last scheme is Power_aware_ThrMu (Beloglazov and Buyya, 2012). This scheduling algorithm focuses on minimizing the CPU usage by setting up hosts in the idle mode. It migrates the running VMs of a host with CPU usage over a threshold to other hosts. An utilization threshold of 0.9 for this scheme is used, the same as our scheduler.

5.2 Experimental Results

As described above, we use a temperature threshold to limit the temperature of the processors. This means that in case of higher temperature than the threshold some workload on the host has to be moved to other hosts in order both to avoid hardware defect and for energy reasons. Therefore, our first three experiments were done for studying the impact of the Temperature_threshold using a number of values ranging from 333 Kelvin to 360 Kelvin.

Figure 1 shows the results of the first experiment, where we measured the energy consumption of the two simulated data centers in a single day using our VM scheduler. The x-axis of the figure demonstrates the thresholds, while the y-axis presents the energy consumption in Kwh. For each temperature threshold there are two values with one for the first data center (DC-1) with 50 hosts and the second for the other one (DC-2) with 250 hosts. The data are presented in two forms, one in bars and the other in curves. Observing the curves in the figure, it can be seen that the energy consumption with different temperature thresholds can be divided into four areas. In the first area, i.e., the temperature threshold between 333 and 335, the energy consumption maintains constant with a value of 52 Kwh for DC-1 and 102 for DC-2. In the second region, i.e., $335 < temperaturethreshold < 340$, the energy consumption goes down as the threshold getting larger. This behavior stops at threshold 340, where a new phase starts with again a constant energy consumption. After this phase, a slight reduction in energy consumption can be seen but generally the temperature threshold does not change the energy behavior in phase 4.

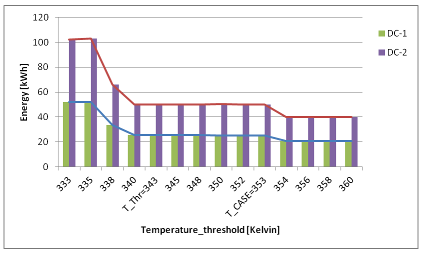

Figure 1: Energy consumption.

The next experiment explains why such scenarios occur with the thresholds. The results are depicted in Figure 2, which shows the number of VM migrations with different threshold values. Observing the figure it can be seen that there is no any migration with the low thresholds 333 and 335. The reason is: Our ThaS migrates the VMs when the temperature of the source host has reached the Temperature_threshold. However, there must be a destination host with a CPU temperature below the threshold. Logically, all hosts may have a temperature above the Temperature_threshold. In this case, our scheduler cannot find any target host on which the VMs can be migrated. This is exactly the case with threshold 333 and 335. Increasing the temperature threshold, i.e., starting from 338, VM migrations can be seen because the scheduler now finds target hosts with temperatures below the threshold Temperature_threshold. It can be also observed that the number of migrations growing up with the threshold. This is because with a higher threshold there must be more hosts whose temperature is below the threshold. Therefore, more migrations can be performed. However, starting from threshold 343 the number of migrations is first not changed and then decreased significantly. The former may be caused by non more target hosts for migrations and the latter is related to the ThaS strategies. Our ThaS sets all hosts with CPU temperature over the case temperature (T_case=353 Kelvin) in the sleep mode for cooling down. Therefore, the available hosts for hosting a migrated VM get fewer.

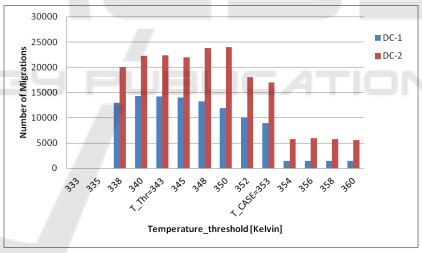

Figure 2: Number of Migrations.

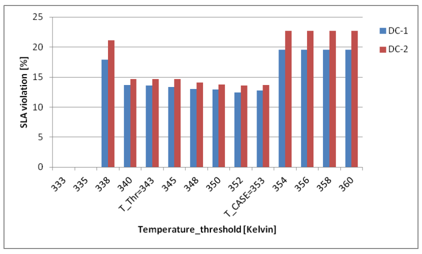

Figure 3: Service level agreement (SLA) violation.

The last experiment with temperature threshold was done for studying the violation to the Service Level Agreement (SLA). that defines the quality of the services. We define the SLA violation with the percentage of unallocated CPU performance relative to the total requested performance in the workloads.

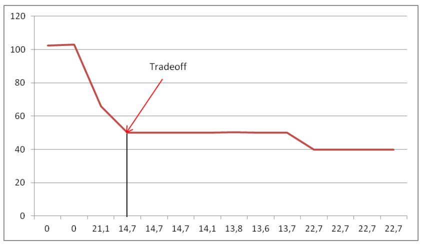

Figure 4: Optimal trade-off for the temperature threshold (x-axis: SLA violation, y-axis: energy consumption).

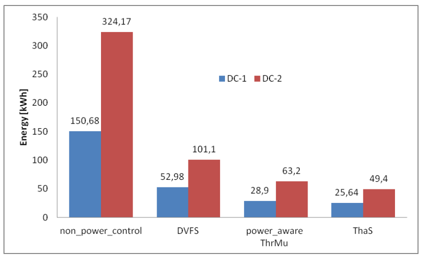

Figure 5: Comparaison of ThaS with other scheduling schemes.

The experimental results are shown in Figure 3. Similar to the results with energy consumption, there is no SLA violation for the first phase (threshold 333 and 335) again due to zero migrations. For threshold 338, where VM migrations are shown in Figure 2, we see an SLA violation. The reason is that a VM migration causes the CPU performance degradation and hence the percentage of SLA violations increases with the number of migrations. For thresholds between 340 and 353 the SLA violation remains constant, which is resulted by nearly unchanged number of migrations. In the last phase from threshold 354, however, the SLA violation arises. This is again because that many hosts are set to idle and therefore not all user requirements can be fulfilled with a small set of active hosts. To summarize the results in the three figures: The number of VM migrations significantly depends on the value of the Temperature_threshold; the energy consumption remains high when no migration is possible; VM migration is performed only after a certain threshold; when the case temperature (T_case) is reached most hosts are put into the sleep mode, which affects the CPU Utilization_threshold that in turn leads to a low number of VM migrations.

Our goal in thermal-aware scheduling is not only to minimize the energy consumption but also not result high SLA violation. Therefore, we have to make a trade-off in selecting the threshold value, where the energy consumption and the SLA violation shall both remain at a minimum. From the previous simulation results we have observed that the third area of the different waveforms is the optimum range and the threshold value for the temperature shall be chosen from this field.

In order to give a clearer view about this optimal threshold of temperature, we made a diagram with the energy consumption and the SLA violation. Figure 4 depicts this graph, where the data are from the tests with the second data center. The x-axis shows the SLA violation while the y-axis presents the energy consumption. Observing the graph in the figure, it can be seen that there is a point where both the energy and the SLA violation are low. This point (threshold 343 Kelvin) is exactly the optimal threshold we are looking for. Hence, we selected 343 Kelvin as the optimum value of the Temperature_threshold for our ThaS scheme. The last experiment studies the feasibility of our scheduler by comparing the energy consumption with ThaS and with other three scheduling schemes, i.e., non_power_control, DVFS and power_aware_ThrMu. Figure 5 depicts the result of the experiment, where the energy consumption was measured during a single simulation run with all four algorithms. We measured the energy consumption of the two data centers in a single day.

Comparing ThaS with the other scheduling algorithms, it can be observed that ThaS achieves the lowest energy consumption with a value of 25.64 Kwh with DC-1 and 49.4 Kwh with DC-2. The energy consumption with other schemes are Non_power_control of 150.68 and 324.17, DVFS of 52.98 and 101.1, and Power_aware_ThrMu of 28.9 and 63.2. For the first data center with 50 hosts we achieved a reduction of 83% to Non_power_control, 52% to DVFS and 11.3% to power_aware_ThrMu. For the other data center with 250 hosts the reduction in energy consumption by ThaS is 85% to non_power_control, 52% to DVFS and 21.8% to power_aware_ThrMu. It can be seen that ThaS results a higher reduction in energy consumption for larger data centers than the smaller ones, especially in comparison with the power-aware scheduling scheme ThrMu.

Overall, we can conclude that the support of VM migration mechanisms is required for efficiently using Cloud resources and combining the power-aware with the thermal-aware scheduling strategies provides the best results for reducing the energy consumption of data centers.

6 CONCLUSION

Today, energy-efficient computing is becoming more and more important due to the huge amount of energy consumption of the data centers. As a result, researchers are investigating on various methods to reduce the energy requirement of a computer system. This work targets on a scheduling algorithm aiming at allocating virtual machines to physical hosts of data centers in such a way that the target host will not be overloaded or over-heated. This means that we schedule virtual machines with respect to the temperature and CPU utilization of processors. We validated the novel scheduler on a simulation environment and compared our achievement with several other scheduling schemes. The experimental results show a clear benefit with our scheduler.

In the next step of this research work we will address on the optimization of the proposed scheduler with respect to its performance. We plan to use those intelligent global optimization algorithms to speed up the procedure of filtering unqualified hosts.For the next version of ThaS the Hotspot (Hotspot,) tool will adopted, that models multicore architectures with more accuracy but not more complexity. Hotspot uses an analogy between electrical circuit phenomena and a heat transfer phenomena. The heat flow between the internal CPU chip blocks is modeled by connecting thermal resistors and thermal storage in blocks. The power consumed by each chip (which typically corresponds to a function unit) is modeled by a power source. In addition, it is also our future work to improve the models for calculating processor temperature in order to support the study on many-core servers.

REFERENCES

AmazonEC2. Amazon Elastic Compute Cloud. [Online]. http://aws.amazon.com/ec2/.

Beloglazov, A. and Buyya, R. (2010). Adaptive threshold-based approach for energy-efficient consolidation of virtual machines in cloud data centers. In *Proceedings of the 8th International Workshop on Middleware for Grids, Clouds and e-Science*.

Beloglazov, A. and Buyya, R. (2012). Optimal Online Deterministic Algorithms and Adaptive Heuristic for Energy and Performance Efficient Dynamic Consolidation of Virtual Machines in Cloud Datacenters. *Concurrency and Computation: Practice and Experience*, 24(3):1397–1420.

Calheiros, R. N., Ranjan, R., Beloglazov, A., e Rose, C. A. F. D., and Buyya, R. (2011). CloudSim: A Toolkit for Modeling and Simulation of Cloud Comp uting Environments and Evaluation of Resource Provisioning Algorithms. *Software: Practice and Experience*, 41(1):23–50.

Fang, Y., Wang, F., and Ge, J. (2010). A task scheduling algorithm based on load balancing in cloud computing. In *Proceedings of the 2010 international conference on Web information systems and mining*, pages 271–277.

GAE. Google App Engine. [Online]. http://code.google.com/appengine/.

Hotspot. Hotspot. [Online]. http://lava.cs.virginia.edu/HotSpot/.

Hsu, C. and Feng, W. (2005). A Feasibility Analysis of Power Awareness in Commodity-Based High-Performance Clusters. In *Proceedings of Cluster Computing*, pages 1–10.

Hu, J., Gu, J., Sun, G., and Zhao, T. (2010). A Scheduling Strategy on Load Balancing of Virtual Machine Resources in Cloud Computing Environment. In *Proceedings of the International Symposium on Parallel Architectures, Algorithms and Programming*, pages 89–96.

Kahn, S., Bilal, K., Zhang, L., Li, H., Hayat, K., Madani, S., Min-Allah, N., Wang, L., Chen, D., Iqbal, M., Xu, C., and Zomaya, A. (2013). Quantitative Comparisons of the State of the Art Data Center Architectures. *Concurrency and Computation: Practice & Experience*. DOI=10.1002/cpe.2963.

Keahey, K. and Freeman, T. (2008). Science Clouds: Early Experiences in Cloud Computing for Scientific Applications. In *Proceedings of the First Workshop on Cloud Computing and its Applications*.

Kim, D., Kim, H., Jeon, M., Seo, E., and Lee, J. (2008). Guest-Aware Priority-based Virtual Machine Scheduling for Highly Consolidated Server. In *Proceedings of the 14th International Conference on Parallel and Distributed Computing (Euro-Par 2008)*, pages 285–294.

Knauth, T. and Fetzer, C. (2012). Energy-aware scheduling for infrastructure clouds. In *Proceedings of the IEEE International Conference on Cloud Computing Technology and Science*, pages 58–65.

Kolodziej, J., Khan, S., Wang, L., Kisiel-Dorohinicki, M., and Madani, S. (2012). Security, Energy, and Performance-aware Resource Allocation Mechanisms for Computational Grids. *Future Generation Computer Systems*. DOI: 10.1016/j.future.2012.09.009.

Kolodziej, J., Khan, S., Wang, L., and Zomaya, A. (2013). Energy Efficient Genetic-Based Schedulers in Computational Grids. *Concurrency and Computation: Practice & Experience*. DOI=10.1002/cpe.2839.

Li, K., Xu, G., Zhao, G., Dong, Y., and Wang, D. (2011). Cloud Task Scheduling Based on Load Balancing Ant Colony Optimization. In *Proceedings of the Six Annual Chinagrid Conference*, pages 3–9.

Mell, P. and Grance, T. The NIST Definition of Cloud Computing. [Online]. http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf.

Menzel, M. and Ranjan, R. (2012). CloudGenius: Decision Support for Web Service Cloud Migration. In *Proceedings of the International ACM Conference on World Wide Web (WWW 2012)*, Lyon, France.

Mhedheb, Y., Jrad, F., Tao, J., Kolodziej, J., and Streit, A. (2013). Load and Thermal-Aware VM Scheduling on the Cloud. In *Algorithms and Architectures for Parallel Processing*, pages 101–114.

Nurmi, D., Wolski, R., Grzegorczyk, C., Obertelli, G., Soman, S., Youseff, L., and Zagorodnov, D. (2008).

The Eucalyptus Open-source Cloud-computing System. In *Proceedings of Cloud Computing and Its Applications*. Available: http://eucalyptus.cs.ucsb.edu/wiki/Presentations.

Openstack (2013). OpenStack Cloud Software. [Online] http://openstack.org/.

rackspace. The Rackspace Open Cloud. [Online]. http://www.rackspace.com/cloud/.

Ranjan, R., Buyya, R., and Harwood, A. (2005). A Case for Cooperative and Incentive Based Coupling of Distributed Clusters. In *Proceedings of the 7th IEEE International Conference on Cluster Computing (Cluster 2005)*, pages 1–11, Boston, Massachusetts, USA.

Ranjan, R., Harwood, A., and Buyya, R. (2006). A SLA-Based Coordinated Super scheduling Scheme and Performance for Computational Grids. In *Proceedings of the 8th IEEE International Conference on Cluster Computing (Cluster 2006)*, pages 1–8, Barcelona, Spain.

Skadron, K., Abdelzaher, T., and Stan, M. R. (2002). Control-theoretic techniques and thermal-rc modeling for accurate and localized dynamic thermal management. In *Proceedings of the 8th International Symposium on High-Performance Computer Architecture*, HPCA '02, pages 17–, Washington, DC, USA. IEEE Computer Society.

Sotomayor, B., Montero, R., Llorente, I., and Foster, I. (2008). Capacity Leasing in Cloud Systems using the OpenNebula Engine. In *The First Workshop on Cloud Computing and its Applications*.

spec08. SpecPower08. [Online] http://www.spec.org.

Takouna, I., Dawoud, W., and Meinel, C. (2011). Efficient Virtual Machine Scheduling-policy for Virtualized heterogeneous Multicore Systems. In *Proceedings of the International Conference on Parallel and Distributed Processing Techniques and Applications (PDPTA2011)*.

Wang, L. and Khan, S. (2013). Review of performance metrics for green data centers: a taxonomy study. *The Journal of Supercomputing*, 63(3):639–656.

Wang, L., Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., and Fu, C. (2010). Cloud Computing: a Perspective Study. *New Generation Computing*, 28(2):137–146.

Wang, L., von Laszewski, G., Huang, F., Dayal, J., Frulani, T., and Fox, G. (2011). Task scheduling with ann-based temperature prediction in a data center: a simulation-based study. *Engineering with Computers*, 27(4):381–391.

Wang, Y., Wang, X., and Chen, Y. (2012). Energy-efficient virtual machine scheduling in performance-asymmetric multi-core architectures. In *Proceedings of the 8th international conference on Network and service management and 2012 workshop on systems virtualiztion management*, pages 288–294.

Windows. Windows Azure Platform. [Online] http://www.microsoft.com/windowsazure.