Load and Thermal-Aware VM Scheduling on the Cloud

Yousri Mhedheb[1], Foued Jrad[1], Jie Tao[1], Jiaqi Zhao[2], Joanna Kołodziej[3], and Achim Streit[1]

[1] Steinbuch Center for Computing, Karlsruhe Institute of Technology, Germany
{yousri.mhedheb,foued.jrad,jie.tao,achim.streit}@kit.edu
[2] School of Basic Science, Changchun University of Technology, China
scorpiozhao@yahoo.com.cn
[3] Institute of Computer Science, Cracow University of Technology, Poland
jokoldziej@pk.edu.pl

Abstract. Virtualization is one of the key technologies that enable Cloud Computing, a novel computing paradigm aiming at provisioning on-demand computing capacities as services. With the special features of self-service and pay-as-you-use, Cloud Computing is attracting not only personal users but also small and middle enterprises. By running applications on the Cloud, users need not maintain their own servers thus to save administration cost.

Cloud Computing uses a business model meaning that the operation overhead must be a major concern of the Cloud providers. Today, the payment of a data centre on energy may be larger than the overall investment on the computing, storage and network facilities. Therefore, saving energy consumption is a hot topic not only in Cloud Computing but also for other domains.

This work proposes and implements a virtual machine (VM) scheduling mechanism that targets on both load-balancing and temperature-balancing with a final goal of reducing the energy consumption in a Cloud centre. Using the strategy of VM migration it is ensured that none of the physical hosts suffers from either high temperature or over-utilization. The proposed scheduling mechanism has been evaluated on CloudSim, a well-known simulator for Cloud Computing. Initial experimental results show a significant benefit in terms of energy consumption.

Keywords: Cloud Computing, Green Computing, Virtualization, VM Scheduling, Thermal-aware Scheduler, Load Balancing.

1 Introduction

Cloud Computing [16,26] is a novel computing paradigm. It provisions computing capacities, including hardware, software, applications, networks as well as storage, as services with a business model of pay-as-you-use. Its special features lie in that users can access the computing resources via Internet with a thin-client, such as a Web browser, without the interaction of administrators. Additionally, Cloud Computing shows the advantages in elasticity, system management, cost-efficiency, customized environment and on-demand resource provision [23,17]. Therefore, an increasing number of Cloud infrastructures [5,29,18] have been established after the first computing Cloud, the Amazon Elastic Compute Cloud [1].

J. Kołodziej et al. (Eds.): ICA3PP 2013, Part I, LNCS 8285, pp. 101–114, 2013.

Several underlying technologies enable Cloud Computing, where the virtualization technology is especially important because virtual machines are the base for delivering any Cloud services, which are categorized as Infrastructure as a Service (IaaS), Platform as a Service (PaaS) and Software as a Service (IaaS) [16]. Actually, the virtualization approach has been used for 60 years with an initial application of running different binary codes on the expensive hardware in the late 50's. Today, Cloud Computing makes virtualization a hot topic again because it relies on this technology to provide on-demand, elastic computing resources. The virtualization itself has also been developed from simple approach to mature techniques with a standard virtualization layer called hypervisor or virtual machine monitor. This layer is responsible for resource allocation and the virtualization of the processor, the memory and devices.

An important issue in Cloud Computing is the scheduling of virtual machines on physical hosts. A Cloud centre is equipped with several thousands of physical hosts, each of them can host an incoming virtual machine request. 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 to host the VM.

Scheduling is not a new topic. This issue exists in various scenarios, like task scheduling in parallel and distributed systems [10,24,25,19] and job scheduling in computing Grids [20,11,12]. Researchers have also proposed a number of algorithms, including those targeting on energy consumption. In the field of Cloud Computing, saving energy is especially important because it adopts a business model and the Cloud providers are surely expecting a low operation overhead. Therefore, this work developed a specific algorithm for scheduling the virtual machines on the Cloud. 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, hence to avoid hot spots with respect to load and temperature. We implemented this algorithm on CloudSim [4], a well-known simulation platform for research work in Cloud Computing. The initial experimental results show the feasibility of the developed algorithm, especially in saving energy consumption.

The remainder of the paper is organized as following. 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. The evaluation results are then depicted in Section 5. The paper concludes in Section 6 with a brief summary and future directions.

2 Related Work

Task scheduling has been a hot topic in various research domains. As a result, a lot of research works have been performed for investigating the scheduling strategies on different systems with features like load-balancing and energy-awareness.

The work described in [27] and [24] exploited Dynamic Voltage Frequency Scaling (DVFS) to implement a power-aware task clustering algorithm for parallel HPC tasks. Authors of [14] also relied on DVFS to schedule independent tasks on a single processor. The work presented in [15] used DVFS to build a hybrid global/local search

optimization framework for reducing the energy requirement of multiprocessor system with dynamic workloads.

Authors of [13] applied a thermal-aware strategy based on the RC-Thermal model [21] to reduce the peak temperature of HPC servers under stochastic workloads. The approach in [30] combines both techniques used in [27] and [13] for solving a temperature-aware scheduling problem. For this, the authors implemented an approximation algorithm based on the Lumped RC-Thermal model and DVFS to study the effect of using the thermal constraints on maximizing the performance of tasks running on some CPU architectures.

Concretely on virtualized machines, task scheduling at the high level is actually VM scheduling that handles the allocation of virtual machines to the physical hosts. Over the last years, the topic of VM scheduling has been addressed. The work presented in [8] implemented a guest-aware priority-based scheduling scheme 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 thus to reduce the response time to the I/O events of latency-sensitive workloads. Authors of [28] proposed a novel VM scheduling algorithm for virtualized heterogonous multicore architectures. The algorithm exploits core performance heterogeneity to optimize the overall system energy efficiency.

The work in [7] proposed 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 [9] 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 [2] 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 [4] to evaluate different power-aware scheduling policies for VMs running on a large scale Cloud centre. Furthermore, they showed how the VM migration and VM pining techniques can optimize the load-balancing and the total energy consumption of the datacenter. Our work is similar to this work, however, we combine both power and thermal-aware scheduling policies to reduce the energy consumption. More importantly, we extend 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 The Thermal-Aware VM Scheduling Scheme

Modern processors have a tolerance limit to the on-chip temperature. A higher temperature over this limit (i.e., case temperature) not only increases the energy consumption but also may result in defect in hardware. We designed a novel thermal-aware scheduling scheme in order to avoid the occurrence of this scenario.

The proposed scheduler is called ThaS (Thermal-aware Scheduler). As shown in Figure 1, 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.

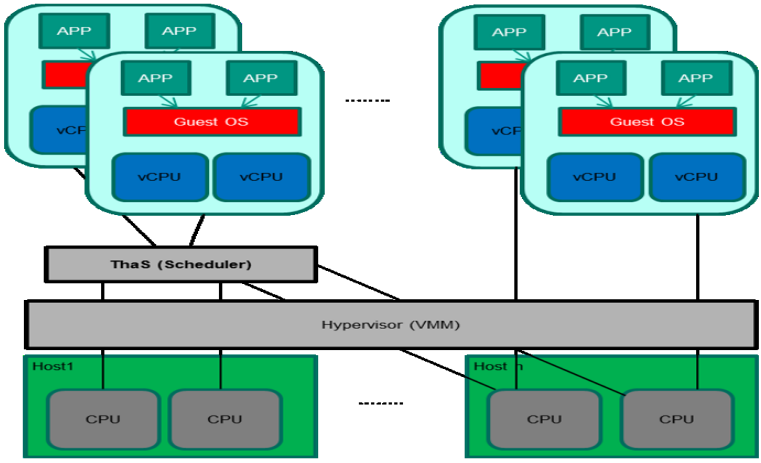

Fig. 1. Software architecture of the Thermal-aware Scheduler

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 runtime based on the current load and temperature state. The main tasks of our scheduler ThaS are the following:

- **Thermal-Aware Energy-Management:** The first purpose of ThaS is to schedule VMs with respect to the temperature of the processors. Such scheduling strategy needs a temperature model that describes the changes of this parameter as applications (here virtual machines) are running. We applied the lumped RC thermal model for this initial work due to its simplicity. This model is however limited to a single-core processor. Therefore, ThaS supports now only single-core machines. For the next version of ThaS we will adopt the Hotspot [6] tool 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.
- **Power-Aware Energy Management:** For power management we apply DVFS. Since the power consumption depends on the CPU usage, this metric has to be

measured before each VM scheduling step in the whole simulation process in order to calculate the current consumed energy by the CPU.

– **Migration of Virtual Machines:** 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, we implemented two strategies in ThaS: 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 must be running 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 datacenter 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, the proposed approach schedules the virtual machines not only for minimizing the energy consumption but also for load-balancing.

4 ThaS Implementation

In order to verify the concept and to validate the functionality of the proposed scheduling strategies, we implemented a Java-based simulation environment for ThaS. The prototypical implementation is based on CloudSim [4], a well-known simulator for research work on the Cloud.

An important task of ThaS is to figure out the critical hosts. The following pseudocode shows how the scheduler performs this task. As can be seen in the code, 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 MigratingFromHosts. This list is adopted in the second step as the input list.

```
Input: HostList,VmList
Output: MigratingFromHosts
For each host in hostList Do
    If isHostOverThresholdTemperature(host) Then
        overThresholdTempHosts <- add host
    Else
        If isHostOverThresholdUtilization(host) Then
```

```
            overUtilizedHosts <- add host
        Endif
    Endif
Endfor
MigratingFromHosts <- overThresholdTempHosts+overUtilizedHosts
Return MigratingFromHosts
```

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 for VMs running on them. These virtual machines are the concrete migration candidates. The candidate VMs are then sorted by their CPU usage. The VMs with minimal CPU usage have higher priority of migration in order not to bring high workload on the target host, thus to avoid 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 VMstoMigrateList that contains all VMs, which are the actual migration candidates.

```
Input: MigratingFromHosts
Output: VMstoMigrateList
For each host in MigratingFromHosts Do
    While true Do
    vm <- getVmtoMigrate(host)
    If vm = Null Then
    break
    Endif
    VMstoMigrateList <- add vm
    host <- deallocate vm
    If !(isHostOverThresholdTemperature(host) &&
        isHostOverThresholdUtilisation(host)) Then
    break
    Endif
    Endwhile
Endfor
Return VMstoMigrateList
```

The last step of the VM migration process is to find an appropriate target host for hosting the migration candidates in the list created in the last step. Here, the workload requirements (e.g., needed resources) have to be taken into account. Our scheduler 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.

```
Input: MigratingFromHosts, vmstoMigrateList
Output: MigrationMap
MigrationMap <- null
vmstoMigrateList.SortDecreasingCPUUtilisation
For each vm in vmstoMigrateList Do
    allocatedHost<- null
    minPower<- Max
    For each host not in MigratingFromHost DO
        If host has enough resources for vm Then
            power <- estimatePower(host,vm)
        Endif
        If host switchedOff && host overUtilizedAfterAlloc||
            overThresholdTemparureAfterAlloc(power)
            Then continue
        Endif
        If power < minPower Then
```

```
            allocatedHost <- host
            minPower <- power
        Endif
        If allocatedHost != NULL Then
            allocate vm to allocatedHost
        Endif
    Endfor
    MigrationMap <- add (vm, allocatedHost)
Endfor
Return migrationMap
```

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 the sleep mode for the purpose of saving energy. The idle hosts are not candidates of destination host for VM migration.

5 Experimental Results

5.1 Simulation Setup

As mentioned above, the prototype of the scheduler is implemented on top of CloudSim, which models large datacenters provisioning computing infrastructures as services. CloudSim implements a view of infinite computing resources. This feature is important for us to evaluate the proposed thermal-aware scheduling algorithms on a large virtualized datacenter infrastructure. In contrast, validation on a real Cloud infrastructure will be extremely difficult for performing different experiments in order to examine the full functionality of the implemented scheduler and the impact of the scheduling strategies.

The simulation duration was set to one day with a scheduling interval of five minutes in the simulation. In addition, a workload of 50 cloudlets (applications) was modeled, each with a CPU core and a computational requirement of 2500 MIPS.

In order to calculate the CPU temperature at a specific timestamp, our thermal model requires the current performance and consumed power of the processors. The later is determined using a power model. The power consumption of computing resources in a datacenter is mainly determined by the total consumed CPU, memory, disk, and cooling power. It has been shown that the power consumption of a server can be described exactly by a linear relationship between the power consumption and CPU utilization (u), even if Dynamic Voltage and Frequency Scaling (DVFS) is employed. An idle server usually uses 70% of the maximum power consumption [3].

Table 1 lists all setup parameters applied in the simulation based tests. The first block in the table shows the thermal constants for the lumped RC thermal model. The values in the table are typical values of a single core CPU obtained from [6].

We configured CloudSim for a datacenter with 50 diverse hosts, each of them composing half of the HP ProLiant G4 servers. The other half is modeled as the HP ProLiant G5 servers. The characteristics of the servers are given in the second block of Table 1. 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

Table 1. The experimental setups

	Thermal Parameter	Value	Unit
Thermal Constants	Initiale CPU Temperatur (T_{init})	318	Kelvin
	Ambiente Temperatur (T_{amb})	308	Kelvin
	Case Temperatur (T_{case})	353	Kelvin
	Thermal Capacity (C_{th})	340	Joule/Kelvin
	Thermal Resistance (R_{th})	0.34	Kelvin/Watt
Simulated physical machines	Server Host Type	HP_Proliant_G4	HP_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
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

from SpecPower08 [22]. 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 last block of Table 1 gives the properties of the four modeled VM types with the assumption that all VMs are running on single core machines. As shown in the table, we use different VMs with various values in MIPS and RAM to model real scenarios. The bandwith and VM size for all simulated virtual machines are set as 100 Mbit/s and 2.5 GB individually.

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 [2] 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 we demonstrate how we achieved an optimal threshold of 343 Kelvin with a trade-off between power consumption and SLA violation.

5.2 Simulation Results

The first experiment was performed for studying the impact of the Temperature_threshold. Figure 2 demonstrates the experimental results. The upper picture in

the figure depicts the impact on the energy consumption and the middle one shows the resulted number of migrations by different thresholds, while the lower figure depicts the impact of the threshold on the Service Level Agreement (SLA) violations. The thresholds range from 333 to 360, as presented in the x-axis. The SLA violation metric represents the percentage of unallocated CPU performance relative to the total requested performance in the workloads.

Observing all three diagrams, it can be seen that the lines can be divided into four areas with different impact values, as marked in the upper diagram. In the first area, i.e., the Temperature_threshold between 333K (initial temperature) and 335K, the energy consumption remains constant and its value is 52 KWh per day. Our ThaS scheduler only decides to migrate the VMs when the temperature of the source host has reached the Temperature_threshold. If a VM should be migrated there must be a destination host with a CPU temperature below the threshold. Because, logically, all hosts have a temperature above the Temperature_threshold, our scheduler does not find any target host on which the VMs can be migrated. Hence, there is no VM migration in this area, as shown in the middle diagram. The scheduler has also no influence at all (see the lower diagram), which leads to increased and constant energy consumption and no SLA violations.

In the second region ($335K < Temperature\_threshold < 340K$) the effectiveness of the scheduler can be seen clearly. In contrast to the first area, the scheduler finds now hosts with temperatures below the threshold Temperature_threshold. These hosts are selected by the scheduler as the target hosts for VM migration. The number of target hosts increases as the threshold Temperature_threshold being enlarged, because with a higher threshold there must be more hosts whose temperature is below the threshold. As a result, also more VM migrations are performed as depicted in the middle diagram. As any VM migration causes a modeled CPU performance degradation of 10%, the percentage of SLA violations increases with a large number of migrations. Overall, the second area shows that i) The scheduler starts VM migration only from a certain Temperature_threshold (here 335 K); ii) The higher the threshold value for the temperature is, the more target hosts are available as candidates for VM migrations.

In the third area ($340Kelvin < Temperature\_threshold < 353Kelvin$), the variation of the Temperature_threshold has no influence on the power consumption any more since the number of the destination hosts ($hosttemperature < Temperature\_threshold$) remains constant in this region. Correspondingly the number of migrations is nearly not changed. This results in a nearly constant power consumption and SLA violation percentage.

In the fourth area ($Temperature\_threshold > 353Kelvin$) the energy consumption is reduced slightly. The reason for this is that the scheduler sets all hosts with CPU temperature over the case temperature (T_case=353 Kelvin) in the sleep mode. It moves away all running VMs on these hosts and then puts the hosts in the idle status. This mechanism results in a small reduction of energy consumption. However, a higher SLA violation is resulted, as can be seen in the lower picture, due to the fact that not all user requirements can be fulfilled with a few numbers of active hosts.

To summarize the results in Figure 2: the number of VM migrations significantly depends on the value of the Temperature_threshold; the energy consumption remains

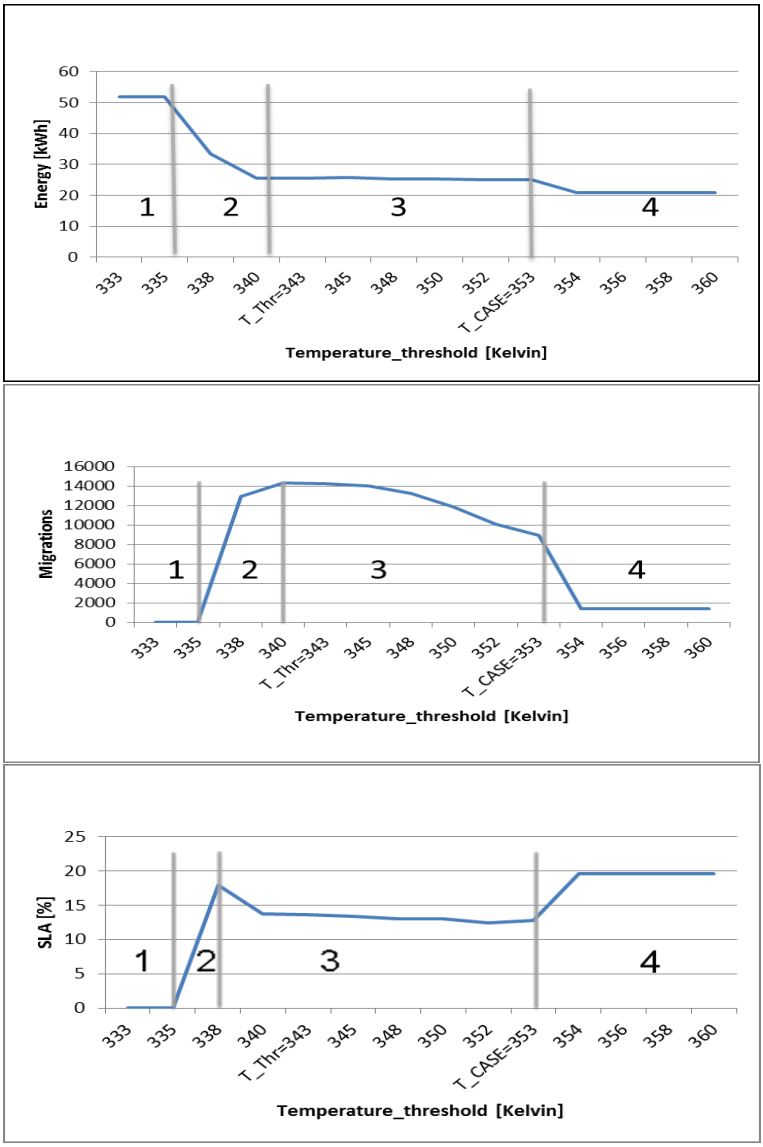

Fig. 2. The impact of Temperature_threshold on the energy consumption (upper diagram), the number of migrations (middle diagram) and the SLA violation (lower diagram)

high when no migration is possible and only after a certain threshold with the temperature VM migration is performed, which leads to a reduction of energy consumption; when the case temperature (T_case) is reached most hosts are put into sleep mode, which affects the CPU Utilization_threshold that in turn leads to a low number of VM migrations.

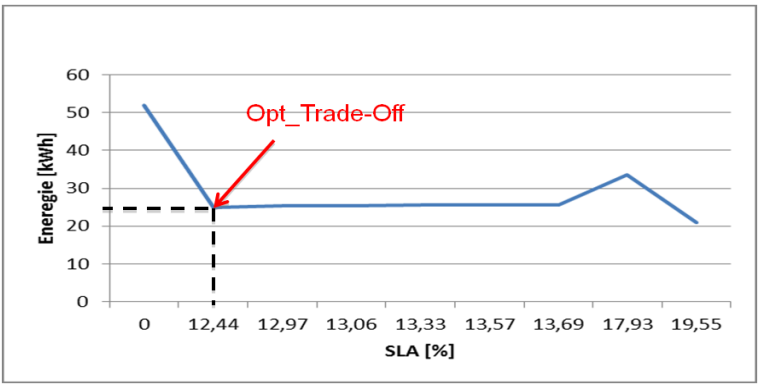

Fig. 3. Optimal trade-off for the Temperature_threshold

Our goal in thermal-aware scheduling is to minimize both the energy consumption and the SLA violation as possible. Therefore, we try in the selection of threshold values (Temperature_threshold and Utilization_threshold) to take a trade-off, 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 more clear view about this optimal threshold of temperature, we created another diagram containing the energy consumption and the SLA violation. Figure 3 depicts this graph. 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 experiments.

To further examine the efficiency of the implemented scheduler, we compared the results of our ThaS scheme with three other scheduling schemes. The first one is Non_power_aware, which was implemented without consideration of the CPU usage. It reflects the energy consumption in a datacenter with full CPU power.

The second scheme is DVFS. It schedules tasks on 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. No migration is performed by the DVFS scheduling. 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 [2]. This scheduling algorithm focuses on minimizing the CPU usage by setting up physical hosts in the idle mode. It migrates the running VMs of a host with CPU usage over a threshold to other hosts. We choose a Utilization_threshold of 0.9 for this scheme.

In contrast, our scheduling algorithm ThaS performs VM migration based not only on the CPU usage but also on the CPU temperature. Here, we adopted the lumped RC thermal model for computing the temperature. The two thresholds in the VM migration are set with the Utilization_threshold of 0.9 and the Temperature_threshold of 343 Kelvin.

Figure 4 depicts the result of the experiment, where the energy consumption was measured during a single simulation run with all four algorithms. 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 per day, while the energy consumption with other schemes are Non_power_aware of 150.68 KWh, DVFS of 52.98 KWh and Power_aware_ThrMu of 28.9 KWh. We conclude: i) The support of VM migration mechanisms is required for efficiently using Cloud resources. ii) Combining the power-aware with the thermal-aware scheduling strategies provides the best results for energy consumption.

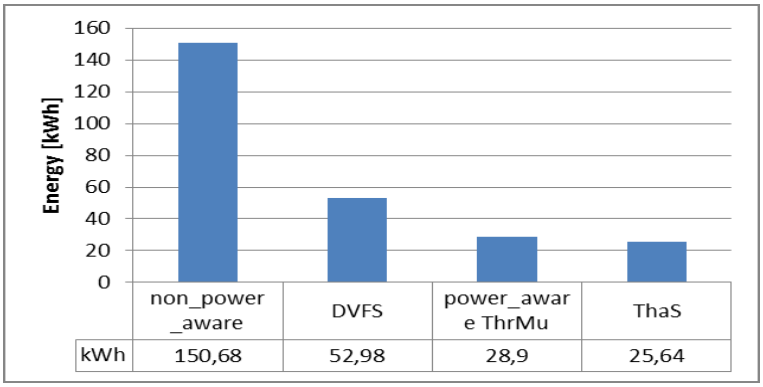

Fig. 4. Comparison of ThaS with other scheduling schemes

6 Conclusions

Scheduling is a hot topic in different scientific domains, including distributed systems, peer-to-peer environments, High Performance Computing as well as Cloud Computing. In Cloud Computing the scheduling problem concerns majorly the scheduling of virtual machines on a physical host. In this paper we propose and implement a load-aware and thermal-aware scheduling mechanism that is capable of preventing the occurrence of over-loaded or over-heated physical machines hence balancing the entire system. The validation results show the benefit of the developed mechanism in terms of energy consumption.

This is our initial work in the research field of energy issues in Cloud centers. The proposed mechanism achieves a reduction of power consumption but we still need a better algorithm for finding the best location to host a virtual machine towards the lowest energy consumption in the complete system. We are currently studying vision cognitive algorithms, which are usually used for solving global optimization problems, and will apply them to schedule the virtual machines on the Cloud. Furthermore, we will improve the accuracy of our thermal model to support multi-core CPU architectures used by current Cloud infrastructures.

References

1. Amazon Elastic Compute Cloud, `http://aws.amazon.com/ec2/`
2. Beloglazov, A., Buyya, R.: 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 (2012)
3. Beloglazov, A., Buyya, R.: 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 (2010)
4. Calheiros, R.N., Ranjan, R., Beloglazov, A., De Rose, C.A.F., Buyya, R.: 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 (2011)
5. Google App Engine, `http://code.google.com/appengine/`
6. Hotspot, `http://lava.cs.virginia.edu/HotSpot/`
7. Hu, J., Gu, J., Sun, G., Zhao, T.: 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, pp. 89–96 (2010)
8. Kim, D.-S., Kim, H., Jeon, M., Seo, E., Lee, J.: Guest-Aware Priority-Based Virtual Machine Scheduling for Highly Consolidated Server. In: Luque, E., Margalef, T., Benítez, D. (eds.) Euro-Par 2008. LNCS, vol. 5168, pp. 285–294. Springer, Heidelberg (2008)
9. Knauth, T., Fetzer, C.: Energy-aware scheduling for infrastructure clouds. In: Proceedings of the IEEE International Conference on Cloud Computing Technology and Science, pp. 58–65 (2012)
10. Kolodziej, J., Khan, S., Wang, L., Byrski, A., Nasro, M., Madani, S.: Hierarchical Genetic-based Grid Scheduling with Energy Optimization. In: Cluster Coimputing (2013), doi:10.1007/s10586-012-0226-7
11. Kolodziej, J., Khan, S., Wang, L., Kisiel-Dorohinicki, M., Madani, S.: Security, Energy, and Performance-aware Resource Allocation Mechanisms for Computational Grids. In: Future Generation Computer Systems (2012), doi:10.1016/j.future.2012.09.009
12. Kolodziej, J., Khan, S., Wang, L., Zomaya, A.: Energy Efficient Genetic-Based Schedulers in Computational Grids. In: Concurrency and Computation: Practice & Experience (2013), doi:10.1002/cpe.2839
13. Lin, S., Qiu, M.: Thermal-Aware Scheduling for Peak Temperature Reduction with Stochastic Workloads. In: Proceedins of IEEE/ACM RTAS WIP, pp. 53–56 (April 2010)
14. Manzak, A., Chakrabarti, C.: Variable voltage task scheduling algorithms for minimizing energy/power. IEEE Transactions on Very Large Scale Integration System 11(2), 270–276 (2003)

15. Martin, S., Flautner, K., Mudge, T., Blaauw, D.: Combined dynamic voltage scaling and adaptive body biasing for lower power microprocessors under dynamic workloads. In: Proceedings of the 2002 IEEE/ACM International Conference on Computer-aided Design, pp. 721–725 (2002)
16. Mell, P., Grance, T.: The NIST Definition of Cloud Computing, http://csrc.nist.gov/publications/drafts/800-145/Draft-SP-800-145_cloud-definition.pdf
17. Menzel, M., Ranjan, R.: CloudGenius: Decision Support for Web Service Cloud Migration. In: Proceedings of the International ACM Conference on World Wide Web (WWW 2012), Lyon, France (April 2012)
18. The Rackspace Open Cloud, http://www.rackspace.com/cloud/
19. Ranjan, R., Buyya, R., Harwood, A.: A Case for Cooperative and Incentive Based Coupling of Distributed Clusters. In: Proceedings of the 7th IEEE International Conference on Cluster Computing (Cluster 2005), Boston, Massachusetts, USA, pp. 1–11 (September 2005)
20. Ranjan, R., Harwood, A., Buyya, R.: 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), Barcelona, Spain, pp. 1–8 (September 2006)
21. Skadron, K., Abdelzaher, T., Stan, M.R.: 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 2002, p. 17. IEEE Computer Society, Washington, DC (2002)
22. SpecPower08, http://www.spec.org
23. Wang, L., Khan, S.: Review of performance metrics for green data centers: a taxonomy study. The Journal of Supercomputing 63(3), 639–656 (2013)
24. Wang, L., Khan, S., Chen, D., Kolodziej, J., Ranjan, R., Xu, C., Zomaya, A.: Energy-aware parallel task scheduling in a cluster. Future Generation Computer Systems 29(7), 1661–1670 (2013)
25. Wang, L., Khan, S., Dayal, J.: Thermal aware workload placement with task-temperature profiles in a data center. The Journal of Supercomputing 61(3), 780–803 (2012)
26. Wang, L., Laszewski, G., Younge, A., He, X., Kunze, M., Tao, J., Fu, C.: Cloud Computing: a Perspective Study. New Generation Computing 28(2), 137–146 (2010)
27. Wang, L., Tao, J., von Laszewski, G., Chen, D.: Power Aware Scheduling for Parallel Tasks via Task Clustering. In: Proceedings of the IEEE 16th International Conference on Parallel and Distributed Systems, ICPADS (2010)
28. Wang, Y., Wang, X., Chen, Y.: 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, pp. 288–294 (2012)
29. Windows Azure Platform, http://www.microsoft.com/windowsazure
30. Zhang, S., Chatha, K.S.: Approximation Algorithm for the Temperature-aware Scheduling Problem. In: Proceedins of IEEE/ACM International Conference on Computer-Aided Design, pp. 281–288 (November 2007)