Sustainable Computing: Informatics and Systems xxx (2014) xxx–xxx

Contents lists available at ScienceDirect

Sustainable Computing: Informatics and Systems

journal homepage: www.elsevier.com/locate/suscom

RESCUE: An energy-aware scheduler for cloud environments

Quan Zhang*, Grace Metri, Sudharsan Raghavan, Weisong Shi

Department of Computer Science, Wayne State University, Detroit, MI, United States

ARTICLE INFO

Article history:
Received 28 February 2014
Received in revised form 7 July 2014
Accepted 11 August 2014

Keywords:
Power management of data centers
Energy efficient scheduling
Virtualization
Cloud computing

ABSTRACT

Cloud computing has become an attractive platform, offering on-demand computing resources and storage capacity for both personal and commercial use. However, the data centers hosting these clouds use a staggering amount of energy, making energy consumption a major expense. For both homogeneous and heterogeneous clouds, energy consumption varies significantly, owing to the heterogeneity of hardware and software. Therefore, to reduce a data center's energy consumption, our approach leverages knowledge of the power consumption behavior of the underlying hardware and the characteristics of workloads in order to increase their overall energy efficiency. Our ultimate goal is to provide a scheduling mechanism that allows cloud providers to reduce the energy consumption of their data centers without needing to replace the underlying hardware, and to do so seamlessly, without impacting clients' performance requirements.

In this paper, we introduce RESCUE, an energy-aware scheduler for heterogeneous cloud environments. RESCUE ranks nodes within the cloud based on their application-specific energy efficiency (ASEE), which implies the correlation between the various hardware and software and the energy efficiency. According to ASEE, RESCUE can assign the whole workload to the most energy-efficient machine while keeping the same performance. We evaluated RESCUE on a private cloud using three benchmarks: BS Seeker, Matrix Stressmark, and TPC-W. We implemented RESCUE with Eucalyptus and compared the energy consumed using RESCUE as the scheduling policy to the other Eucalyptus built-in policies: Round Robin and Greedy. The results show that with a nonaggressive control policy where the machine remains in active mode (ACPI C0 state), RESCUE reduces the overall energy consumption by up to 16.7% on average for BS Seeker and Matrix Stressmark, while for TPC-W, RESCUE saves up to 6.6% of the total energy. Furthermore, with an aggressive control policy that set the idle machines in "sleep" mode (ACPI S3 state), RESCUE-with-sleep can further reduce the total energy consumption by 51.5% for BS Seeker, 48.8% for Matrix Stressmark, and 21.3% for TPC-W compared to RESCUE.

1. Introduction

Cloud computing is a new computing reality used by a diverse range of clients. These clients can be classified into a wide range of categories, each of which uses it for several purposes. For instance, some clients use the cloud for personal reasons, such as backing up their own smartphones. Other clients may use it to support their business operations, whereas researchers often use it for conducting scientific computing. Through virtualization, data center hosts can provide tailored resources for each client based on their needs. However, the data centers hosting these clouds consume a staggering amount of energy every year, making energy consumption one of their major expenses. A single data center can have up to $9.3 million a year in energy expenses [1]. Also, according to Koomey [2], worldwide electricity used by data centers doubled from 2000 to 2005, representing an aggregate annual growth rate of 16.7% per year for the world. Therefore, reducing the energy consumption in data centers can lead to significant gains for cloud providers, where even a small improvement in energy efficiency can have a great impact on reduction of operating costs and carbon emissions, due to economies of scale.

With the explosion of data centers and their extensive power consumption, energy efficiency has become a very important research topic. Energy consumption of homogeneous hardware may be quite different even with an identical workload. The heterogeneities of hardware and software make energy consumption more complex in a heterogeneous cloud. Many works already focus on energy-aware computing for both homogeneous and heterogeneous clouds. [3–5] leverage dynamic voltage and frequency scaling (DVFS) to reduce the CPU's energy consumption because the CPU dominates the entire system power. Similar research has

* Corresponding author. Tel.: +1 3132127260.
E-mail addresses: quan.zhang@wayne.edu (Q. Zhang), gmetri@wayne.edu (G. Metri), sudharsan@wayne.edu (S. Raghavan), weisong@wayne.edu (W. Shi).

http://dx.doi.org/10.1016/j.suscom.2014.08.008
2210-5379/

also focused on the memory subsystem [6–9]. Some other works [10–12] proposed resource allocation based on workload requirements. Those works usually come with performance degradation and violation of the service-level agreement (SLA) due to the frequency or voltage modeling and resource reallocation determining process.

As a result, we propose RESCUE, an energy-aware scheduler that efficiently utilizes the data center's resources while reducing energy consumption and maintaining the same performance and SLA. We focus in this paper on heterogeneous platforms for the following reasons. First, hardware components are known to fail. As a result, data center technicians tend to replace the failed parts with new ones that are not necessarily of the same configuration as the old ones. In addition, another key feature of data centers is their scalability. As demand increases, new servers get added to the old ones. Both of these factors give the data centers their heterogeneous nature. We also focus on scheduling based on application type because a computing platform is a combination of hardware, software, and other technologies. Every running workload requires different types of resources, and each application can impact a server's energy consumption differently. As a result, we make the application type an essential component of our scheduling mechanism. By applying RESCUE, cloud providers can reduce the operating cost of electricity by improving the data center's energy efficiency, and therefore can become more attractive and competitive with a lower service price. The end users benefit directly from the lower price because they spent less while enjoying the same quality of service.

In this paper, we implemented RESCUE, an energy-aware scheduler for virtualized cloud environments. The basic idea is to choose the most energy-efficient machine for a specific application type and workload. To accomplish this purpose, we first rank the nodes within the cloud according to their application-specific energy efficiency (ASEE), as defined in [13]. In the second step, the cloud controller can schedule and start virtual machines on nodes on the basis of their energy efficiency ranking. Therefore, RESCUE can assign the whole workload to a minimal set of machines while keeping the same performance. This provides great flexibility for a data center operator to set those idle machines to sleep mode, which is the most energy-saving strategy shown in previous works [11,14]. We evaluated RESCUE with a heterogeneous cloud using three heterogeneous benchmarks: BS Seeker, Matrix Stressmark, and TPC-W. For comparison, we demonstrate the energy consumed using the RESCUE scheduling policy with the built-in scheduling policies (Round Robin and Greedy) of Eucalyptus.

The results show that with a nonaggressive control policy that keeps all nodes in active mode even when there is no workload running on some machines, RESCUE reduces the overall energy consumption by up to 16.7% on average for BS Seeker and Matrix Stressmark, whereas for TPC-W, RESCUE saves up to 6.6% of the total energy. Furthermore, with an aggressive control policy that set the idle machines in "sleep" mode (ACPI-defined S3 state), RESCUE-with-sleep can further reduce the total energy consumption by 51.5% for BS Seeker, 48.8% for Matrix Stressmark, and 21.3% for TPC-W compared to RESCUE.

The remainder of the paper is organized as follows: we discuss some background in Section 2. RESCUE, our energy-aware scheduler for Eucalyptus design, is explained in Section 3. Section 4 and Section 5 describe the experimental setup and results. Finally, we summaries the related work and conclude in Section 6 and Section 7, respectively.

2. Background

The development of cloud computing has raised a lot of virtualization techniques. Eucalyptus [15] is an open-source Infrastructure as a Service (IaaS) that allows its clients to create a private cloud using their available hardware resources. Within the Eucalyptus framework, there are two different types of nodes: a head node and worker nodes. Clients' requests of virtual machines (VMs) are submitted to the head node, and the VMs use the resources on the worker nodes. Eucalyptus has three main components: Node Controller (NC), which starts, inspects, shuts down, and cleans up the VMs; Cluster Controller (CC), which monitors the whole cluster and determines which NC can be assigned to start an instance; and Cloud Controller (CLC), which processes incoming user-initiated or administrative requests. For the scheduling policies, Eucalyptus has two built-in choices: Round Robin (RR) and Greedy (GDY). The policies are described in the following manners: (1) using RR policy, Eucalyptus schedules the instances uniformly among the available nodes according to the order in which the nodes are registered; (2) with GDY policy, the CC will keep adding to each node more instances until no more resources are available. In this paper, we implemented RESCUE in the CC module of the Eucalyptus platform and compared RESCUE with two built-in schedulers. The performance metric used in RESCUE is ASEE. The ASEE is defined as following:

$$\text{ASEE} = \frac{Load\_of\_the\_Application}{Energy\_Consumed\_by\_the\_Application} \quad (1)$$

where *Load_of_the_Application* can differ based on the type of application. For instance, it can be defined as the number of operations for an arithmetic application, the size of processed data for a data-processing application, or throughput for a web service application. The detailed discussion of ASEE is in our previous work [13]. In this paper, we focus on using ASEE as our ranking metric because we assume that each computer component consumes different energy values; because data centers can run heterogeneous types of applications, each of which requires the use of different components, then nodes will consume different amounts of energy based on the application type. For instance, an application (A) may run more efficiently on server (X) compared to server (Y), but an application (B) may run more efficiently on server (Y) compared to server (X). From those observations, we propose RESCUE to schedule the VMs in Eucalyptus.

3. RESCUE

RESCUE is an energy-aware scheduler for Eucalyptus. It consists of two modules, as shown in Fig. 1. The first module is for *preprocessing*, and the second one is for *node ranking and scheduling*. In the first module, we profile the ASEE with different applications and hardware configurations. In the second module, we rank all nodes based on their ASEE, and assign the application's request to the node with highest ASEE of that application. We profile the energy efficiency of servers according to their application type. On the basis of our profiling results, we simply assign a new workload to a node according to its ASEE; the running workload will rarely be migrated unless hardware failure occurs, and it will be rescheduled among the current active nodes. At the same time, we also keep the machine running at the highest speed leading to a shorter execution time, and then stay in idle mode, which operates at the lowest speed to save energy. If there is no instance running on a certain machine, RESCUE will set this machine to sleep mode as an aggressive power-saving method. By applying such a strategy, RESCUE can reduce the overall energy consumption without any performance loss and keep itself flexible and simple.

3.1. Preprocessing

In the preprocessing step, we profile the ASEE with different applications and hardware. We broadly categorized the

Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008

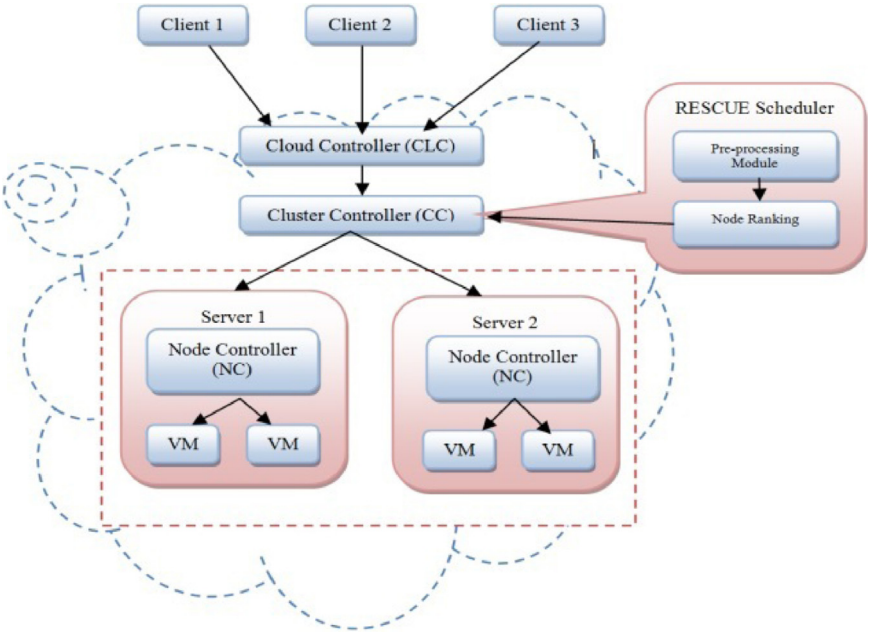


Fig. 1. An overview of the RESCUE architecture.

benchmarks into three categories: CPU-intensive, memory-intensive, and I/O-intensive workloads. Before adding a node to the cloud, we profile it using a "profiling imag". In the profiling image, three benchmarks have been installed along with a Python script that automated the testing process. After a VM is started on the node using the profiling image, the script starts the process by serially running the benchmarks while collecting the power consumption using an external power meter for the entire node. Next, we repeat this with different types of VMs that have distinct hardware resource configurations. Upon completion of the benchmarking, the CC processes the data collected using the following steps. (1) It calculates the average power consumed during the application runtime. We use an external power meter to get the power of the entire node at fixed intervals. (2) The CC calculates the energy consumption by multiplying the average power by the duration required to process the application's request. (3) Finally, the ASEE is derived and recorded as defined in Eq. (1). When these calculations are done, the new node is added to the cloud, and the ASEE results are used by the scheduler for ranking this node. RESCUE does not profile every possible scaling factor since it simply use an approximation thus reducing the number of pre-processing VMs and the data kept for each node. For example, if we profiled for Node(i) with 1 VM, 2 concurrent VMs, and 4 concurrent VMs. Then, after placing 2 VMs on Node(i), we get the ASEE for 4 VMs if a 3rd VM is requested.

Each node added to the cloud will maintain three structures: *POSSIBLE_INSTANCES, AVAILABLE_INSTANCES, and RUNNING_INSTANCES*. *POSSIBLE_INSTANCES* contains the collection of these tuples *(VM_SIZE, SCALING_FACTOR, APP_TYPE, ASEE)*. The *VM_SIZE* indicates the hardware resources including the number of CPU cores, memory size, and hard disk size, which are allocated for the application. All different types of VM are summarized in Section 4. Due to hardware heterogeneity, the *POSSIBLE_INSTANCES* varies for different nodes and depends on the *VM_SIZE*. Upon the profiling of a node, *POSSIBLE_INSTANCES* gets filled with the appropriate information. For each type of *VM_SIZE*, the *SCALING_FACTOR* indicates the maximal number of such type VMs this node can host. The *VM_SIZE* and *APP_TYPE* decides its corresponding *ASEE* value. *AVAILABLE_INSTANCES* tracks the remaining resources and also save a list of *(VM_SIZE, APP_TYPE*. *RUNNING_INSTANCES* contains the list of running instances where for every running instance we keep track of *(VM_SIZE, APP_TYPE)*. All those three structures are maintained by each NC. When a new request is submit to the CC, the CC will collect those three tables to run the Map_Request algorithm as shown in Algorithm 1.

3.2. Node ranking and scheduling

Upon completion of the preprocessing step, the CC gets the ASEE data for the node. Upon a user's request, it compares the ASEE of the available nodes and schedules the VM on the node with the max ASEE. Normally, when a client requests a new VM using Eucalyptus, the user enters only the VM size requested. Using RESCUE, the user must enter the VM size and one of the following application categories: CPU intensive, memory intensive, or I/O intensive. The scheduler matches the request to the corresponding node with the highest ASEE and will start the new instance on the selected node.

3.2.1. Mapping of user requests

In RESCUE, all workload is assigned to the node with the highest ASEE value through *Map_Request(APP_TYPE, VM_SIZE)* as shown in Algorithm 1. The users need to specify the their requests along with *APP_TYPE* and *VM_SIZE*. For each request, all nodes are evaluated to find the node with highest ASEE. If there is no running instance on a node, the ASEE of this node, which is stored in the *POSSIBLE_INSTANCE* structure, is decided by the *APP_TYPE* and VM_SIZE (as shown in lines 6–12 of Algorithm 1). When a node host multiple types of applications, the ASEE for a new VM request will be the lowest ASEE among all those applications (as shown in lines 14–24 of Algorithm 1). Since the ASEE is defined for each application, and for the node running mixing types of applications we conservatively use the minimum ASEE. If no more resource on the machine that has the highest ASEE for a new request is not available, the new request will be assigned to the machine that has the second highest ASEE. Once a VM is assigned to a Node(i), if Node(i) is in sleep mode, then RESCUE brings the node to active mode first as shown in Algorithm 1, line 27 and 28. Next, RESCUE starts the instance on the corresponding node after fixing the list of *AVAILABLE_INSTANCES* and *RUNNING_INSTANCE* of Node(i) as shown in Algorithm 2. Finally, when an instance is terminated

Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008

on Node(i), RESCUE terminates that instance and fixes the *AVAILABLE_INSTANCES* and *RUNNING_INSTANCE* of Node(i) as shown in Algorithm 3.

In RESCUE, we make a machine in a ACPI-defined S3 Sleep/Standby mode to save more power when the machine has no running instances. Some other works [11,16,17] prefer to shutdown the machine to achieve a more aggressive power saving, but the cost (startup latency) for completely shutdown a machine is much larger than that of making it in sleep/standby mode. The ACPI-defined S3 sleeping state is a low wake latency sleeping state where CPU, cache, and chip set context are lost except memory in this state. Those techniques are used in PowerNap [18]and Blink [19] to save system power while maintains a low wakeup latency. Thus, in RESCUE, we also use the similar mechanism to save power as shown in Algorithm 3, lines 6–9, where the node is set to ACPI S3 mode if there is no running instances. We do not set the node to sleep mode immediately after a fixed period as shown in line 7 of Algorithm 3

Algorithm 1. Map_Request

```
1.  Map_Request(APP_TYPE, VM_SIZE)
2.  BEGIN
3.  best_node = NULL, Max_ASEE = 0
4.  for all node in node_list do
5.    if node.available_instances has (APP_TYPE,VM_SIZE) then
6.      if node.running_instances == ∅ then
7.        current_ASEE = get_ASEE(APP_TYPE, VM_SIZE)
8.        if current_ASEE > Max_ASEEthen
9.          Max_ASEE = current_ASEE
10.         best_node = node
11.       end if
12.     end if
13.   else
14.     Min_ASEE = get_ASEE(APP_TYPE, VM_SIZE)
15.     for all APP_TYPE in running_instances do
16.       current_ASEE = get_ASEE(APP_TYPE, VM_SIZE)
17.       if current_ASEE < Min_ASEE then
18.         Min_ASEE = current_ASEE
19.       end if
20.     end for
21.     if Min_ASEE > Max_ASEE then
22.       Max_ASEE = current_ASEE
23.       best_node = node
24.     end if
25.   end if
26. end for
27. if best_node is in SLEEP then
28.   wakeup()
29. end if
30. Start_Instances(best_node, APP_TYPE, VM_SIZE)
31. END
```

Algorithm 2. Start_Instance

```
1:  Start_Instances(Node, APP_TYPE, VM_SIZE)
2:  BEGIN
3:  available_instances = available_instances - (APP_TYPE, VM_SIZE)
4:  running_instances = running_instances + (APP_TYPE, VM_SIZE)
5:  Eucalyptus.start_instance(Node, VM_SIZE)
6:  END
```

Algorithm 3. Terminate_Instance

```
1:  Terminate_Instances(Node, APP_TYPE, VM_SIZE)
2:  BEGIN
3:  Eucalyptus.stop_instance(node, VM_INDEX)
4:  available_instances = available_instances + (APP_TYPE, VM_SIZE)
5:  running_instances = running_instances - (APP_TYPE, VM_SIZE)
6:  if Node.running_instances == ∅ then
7:    wait(timeout)
8:    sleep()
9:  end if
10: END
```

3.2.2. How can RESCUE scale as the number of nodes increases?

Eucalyptus can have multiple NCs within the same cluster. Therefore, as the number of nodes scales up, we can configure the cluster to have multiple NCs. Each NC can have a subset of nodes. Then, upon the request of a new VM, all the NCs can determine, in parallel, the local optimal node with the Max_ASEE. Then, when all the NCs return their Max_ASEE for the requested VM, we can assign the VM to the NC with the highest Max_ASEE, and that NC starts the instance on the local optimal node.

Table 1
Nodes hardware configuration.

Server	Processor	Cache	RAM	HD	Network
Node-1 and Node-2	Xeon E5260 2.4 GHz 16 cores	L1 256 KB L2 1 MB L3 13 MB	16 GB DDR3 1333 MHz	1TB	100 Mbps Full Duplex
Node-3	Xeon E5260 2.4 GHz 16 cores	L1 256 KB L2 1 MB L3 13 MB	24 GB DDR3 1066 MHz	1 TB	100 MBps Full Duplex
Node-4	Xeon E5404 2.0 GHz 8 cores	L1 256 KB L2 12 MB	16 GB DDR2 667 MHz	320 GB	100 Mbps Full Duplex
Node-5	Xeon 2.8 GHz 1 core	L1 16 KB L2 1 MB	4 GB DDR 400 MHz	500 GB	100 MBps Half Duplex
Node-6	Xeon 2.8 GHz 1 core	L1 16 KB L2 1 MB	4 GB DDR 400 MHz	40 GB	100 MBps Half Duplex

Table 2
Types of available instance.

Virtual machine size	# Cores	RAM	HD
Small (S)	1	512 MB	5 GB
Medium (M)	1	1 GB	10 GB
Large (L)	1	2 GB	10 GB
Extra large (XL)	2	2 GB	10 GB
Extra extra large (XXL)	4	4 GB	20 GB

4. Experimental setup

We implemented RESCUE on a heterogeneous cloud created using Eucalyptus. This cloud includes six nodes as described in Table 1. Node-1 is our head node, and Node-2 to Node-6 are the data nodes. Our cloud supports multiple types of VMs, as described in Table 2. Because of the heterogeneous nature of our nodes, the support of VMs differs from one node to another. The nodes and VMs ran CentOS 5.5 with Xen virtualization module. To implement RESCUE in Eucalyptus, we modified the source code of the CC and added the following three functions: $Map_Request$, $Start_Instance$, and $Terminate_Instance$.

We used three different benchmarks. The first benchmark is BS Seeker, a CPU-intensive benchmark. BS Seeker is a bioinformatics application designed to map bisulfite-treated reads in genome-wide measurements of DNA methylation at a single nucleotide resolution. We deployed the Python implementation of BS Seeker [20] in the VMs' images. We supplied BS Seeker with a 100-kB input file and used Bowtie [21] to align the reads.

The second benchmark is Matrix Stressmark. It is a part of DARPA Data Intensive Systems (DIS) Stressmark suite. It is characterized by its irregular accesses to memory and calculating an equation over a sparse $n \times n$ matrix. Thus, Matrix Stressmark is a memory-intensive benchmark, and we use it to simulate the behavior of common scientific applications [22]. For all running instances including profiling image, we set the vector to be of size 6000 and the number of nonzero items to be 1,000,000.

The last one is TPC-W, which is an online bookstore serving as a transactional e-commerce benchmark. We deployed a Java implemented version based on TPC-W specification 1.0.1 [23].

Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008

Table 3
The order of creating instances for each sequence.

Order	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
Seq. 1	M	M	XL	S	L	L	S	XL	XXL	M	S	M	L	M	L
Seq. 2	S	XL	S	XXL	L	L	M	XL	M	L	S	XL	M	L	L
Seq. 3	S	S	L	XL	S	M	XXL	XL	L	S	M	L	L	M	S

TPC-W uses Emulated Browsers (EBs) to emulate traffic generated by customers. Each EB session consists of a series of sequential interactions such as searching for a product, browsing the list of products, adding items to the shopping cart, and so on. For TPC-W, we used Apache Tomcat version 5.5.20 as the application server and MySQL 5.0.77 as the database server. A total of 10,000 items were added to the database, and we ran 250 concurrent clients to generate the workload. Using those benchmarks, RESCUE ranked the nodes according to their ASEE.

Compared to the RR and GDY scheduling policies, RESCUE can schedule every instance on the basis of its energy profile without any performance degradation. Furthermore, the scheduling also provides an opportunity to save more energy by setting idle machines in sleep mode. Finally, we evaluated our scheduling mechanism by collecting the total energy consumed while running each scheduling mechanism – RESCUE, RR, and GDY – for the same set of requests.

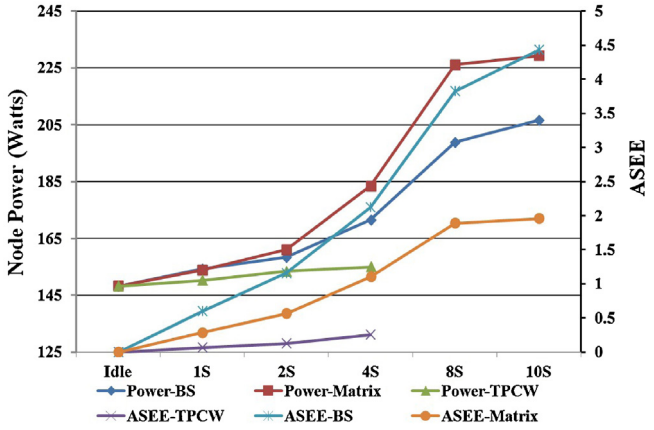


Fig. 2. The power and ASEE of running 1, 2, 4, 8, and 10 Small VMs on Node-2.

5. Evaluation

The entire evaluation includes two phases: preprocessing and scheduling. In the preprocessing phase, we ran the profiling image on Node-2 to Node-6. For each VM type, we created different numbers of instances on a single node. During the execution of the benchmark, the CC collects the power and other needed parameters and then uses those data to derive the ASEE for each node.

In the scheduling phase, we used three randomly selected user request sequences to generate the workload. Table 3 shows the order of user requests and the VM size for each request. For each sequence, we compared the energy consumption with three scheduling policies: RR, GDY, and RESCUE (RSC).

First we let all our nodes go to idle in order to start with the same baseline for all the servers. Then, we started the instances as shown in Table 3. Finally, we collected the energy consumed for each scheduling policy. Our power consumption measurements were collected using an electronic watt meter called Wattsup?/PRO/ES/.Net [24]. The voltage is 120 VAC, 60 HZ and the max wattage is 1800 watts. The measurement accuracy is ±1.5%, and the selected interval of time between records is 1 second.

5.1. Preprocessing results

We capture the preprocessing results of each benchmark separately. Ideally, within a data center there are heterogeneous types of applications running at the same time. However, for comparison clarity, we focus on running each benchmark separately and displaying the results accordingly.

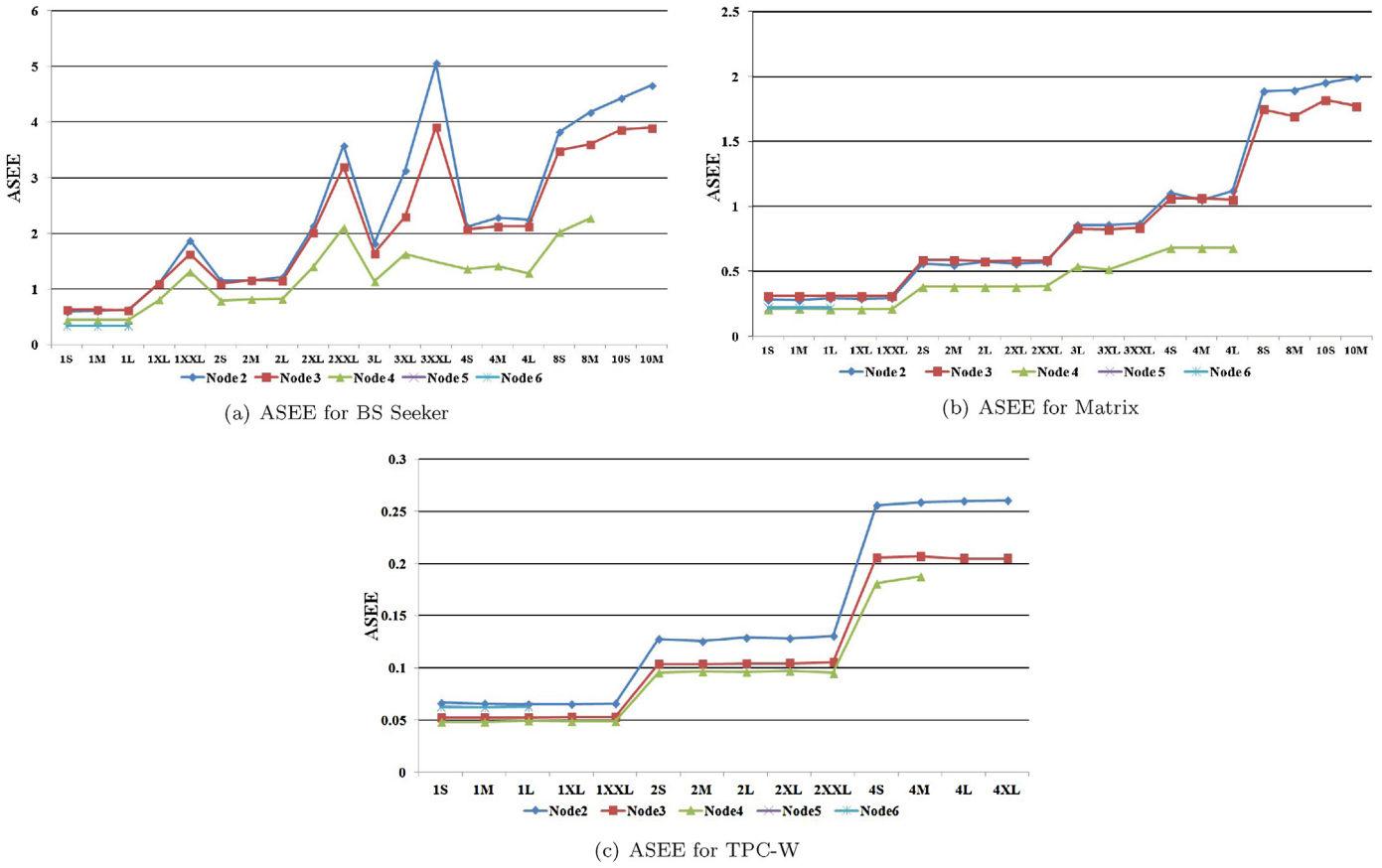
Fig. 2 shows the power and ASEE when running 1, 2, 4, 8, and 10 Small VMs on Node-2. The power and ASEE have a similar trend because the total workload is proportional to the number of VMs. Since all VMs have same workload, the total workload increases when the number of running VMs increases. For the power, more VMs running on the same machine leads to higher power, but it increases relatively slower. The execution time of all cases is similar. Thus, based on the definition of ASEE, the power dominates the increasing trend of ASEE. For a particular application, such as BS Seeker, the power is not in a linear relationship with the number of VMs, and the power increases more quickly between 2 S and 8 S. Furthermore, the BS Seeker consumes more power than the other two benchmarks because it is CPU intensive, and the CPU dominates the power consumption of a node. Matrix is memory intensive, so it consumes less power than BS Seeker when running the same number of VMs. TPC-W is an I/O (disk and network) intensive workload that is sensitive to user behavior, and it consumes much less power than CPU- or memory-intensive jobs.

Fig. 3(a), (b), and (c) shows the ASEE for running BS Seeker, Matrix Stressmark, and TPC-W, respectively, with different VM sizes on all five nodes. We first analyze the ASEE for each benchmark and then compare it across all three benchmarks. Owing to the hardware limitations, Node-5 and Node-6 cannot support VM sizes other than 1S, 1M, and 1L. Similarly, Node-4 cannot support all combinations of VM sizes.

5.1.1. BS seeker analysis

First, Fig. 3(a) represents the ASEE data for BS Seeker. Node-2 and Node-3 are closer to each other. Node-2, Node-3, and Node-4 have the same pattern when the VM size changes. Switching between (S), (M), and (L) had a negligible effect on the ASEE because the nature of BS Seeker is CPU intensive. Moreover, all (S), (M), and (L) are single-core VM. Therefore, keeping the number of CPUs consistent did not affect the ASEE much. If we fix the same VM size, the ASEE differences are caused by the different processors' performance, as shown in Table 1. When the VM size switches between (L), (XL), and (XXL), the ASEE increases sharply for Node-2, Node-3, and Node-4. This increment is brought by the number of cores doubling. More CPU resources can reduce the execution time and increase the ASEE. Another noteworthy observation is the ASEE gap between Node-2 and Node-3, where switching the VM size from (XL) to (XXL) becomes larger and larger. Although Node-2 and Node-3 have the same processor type, when the number of instances goes up, the competition on the other shared resources, such as the L2 and L3 cache, is intense. Thus, for one and two running instances, the gap is smaller, and for three or more running instances, the gap is larger. Furthermore, the range of ASEE is also wide; for all data points, the highest ASEE is about 5× the lowest one.

Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008



(a) ASEE for BS Seeker

(b) ASEE for Matrix

(c) ASEE for TPC-W

Fig. 3. ASEE for different VM sizes and nodes.

5.1.2. Matrix Stressmark analysis

Second, we present the result of the Matrix Stressmark benchmark in Fig. 3(b), which displays the data we collected regarding the energy consumed for the same workload. We noticed that Node-2 and Node-3 consume a similar amount of energy to perform the entire calculation, leading to a similar ASEE. For 8 and 10 running instances, the ASEE gap becomes larger because Node-2 has faster memory frequency than Node-3. Comparing Node-4 and Node-2 (or Node-3), Node-4 consumes more than 25% extra energy than Node-2 (or Node-3) in order to perform the same job. This is attributed to the fact that Node-2 and Node-3 have an L3 cache and faster memory frequency. Because the Matrix Stressmark is memory intensive, with higher-level caches and faster memory frequency, Node-2 and Node-3 had fewer cache misses and could finish the job faster and thus save on energy. However, the difference between Node-2 and Node-4 is not so wide as in BS Seeker. This narrow change range is also caused by the nature of the Matrix Stressmark, in that it generates irregular memory access, and to a certain extent this can weaken the effect of larger cache and faster memory frequency. Another important observation is that Node-5 and Node-6 have a higher ASEE than Node-4, whereas Node-4 has a more powerful hardware configuration. The reason here is that the entire system power is lower for Node-5 and Node-6 than for Node-4. Less-powerful hardware requires less power. The average power for Node-5 and Node-6 running 1S (or 1M, or 1L) is about 195 watts (144 watts when it is idle). For Node-4, the average power is around 217 watts (196 watts when it is idle) when running the same size of VM. The idle power of Node-4 is even higher than the average power of Node-5 and Node-6. Thus, although the execution time of Node-5 (or Node-6) is longer than Node-4, the total energy may be lower because of the lower average power. However, this is perfect if there are no tight performance requirements. In reality, this is impossible because any work must be done within a time limit.

5.2. TPC-W analysis

Third, the results for TPC-W with 250 EBs is shown in Fig. 3(c). Based on the results, it is clear that the ASEE of TPC-W has a similar trend with the ASEE of Matrix Stressmark. No matter what VM size it is, the ASEE changes a little. Similarly, with the same running instances, the ASEE of each node is also close. We also witness that Node-5 and Node-6 have a higher ASEE than Node-3 and Node-4. As we mentioned before, lower system power may obtain a lower total energy consumption even though the execution time is longer. TPC-W fits this example perfectly. TPC-W is neither CPU nor memory intensive, but it needs a lot of disk and network I/O operation. Disk and network devices contribute only a fraction to the total system power, and in Fig. 2 we already show that the average power of running TPC-W is several watts higher than the idle power (7 watts higher running 4 Small VMs on Node-2). The execution times for all test cases are 1197 ± 12 seconds. Thus, the node with lower idle power becomes more energy efficient. In particular, the idle power for Node-2 to Node-6 in an increasing fashion are Node-5 and Node-6 (144 watts), Node-2 (148 watts), Node-3 (182 watts), and Node-4 (196 watts). The ASEE ranking of each node in a decreasing fashion is Node-2, Node-5 and Node-6, Node-3, and Node-4, which is almost the same with the order based on idle power. This implies that reducing the idle power is sometimes more useful.

Last, if we compare the ASEE of three benchmarks, the rank order of each benchmark is different. Node-2 is always the most energy-efficient node. The other nodes may be efficient in one benchmark, but not as efficient in another benchmark. This verifies our claim that ASEE varies by hardware configuration. Another

Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008

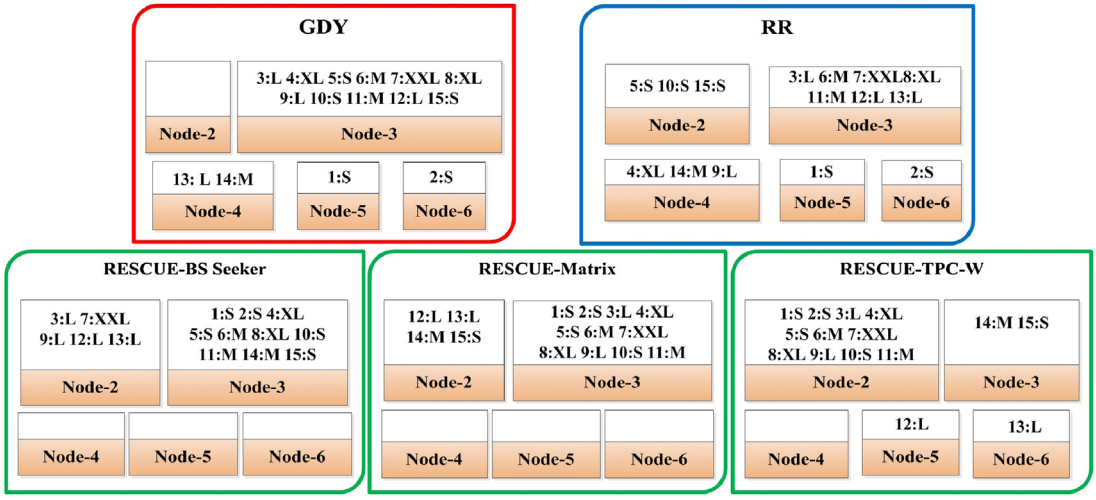


Fig. 4. A sample VM allocation with Seq. 3.

observation is that the ASEE-based node rank can change when the workload changes. In Fig. 3(b), when the running instances are less than or equal to 2, Node-3 has a higher rank than Node-2. On the contrary, Node-2 becomes more efficient than Node-3 when the workload is heavy. Finally, all these benchmarks reinforce the notion that energy efficiency differs according to the server and application types. Regardless of why these differences occur, they should be considered when scheduling instances within a data center.

5.3. Scheduling results

In order to compare RESCUE to the current Eucalyptus scheduling policies, we used the three test sequences shown in Table 3 to run 15 concurrent instances. The nodes were connected to the cloud (Node-1 is the head node running CC and CLC) in the following order: Node-5, Node-6, Node-3, Node-4, and Node-2. To illustrate the differences in scheduling allocation of VMs, we show a sample of VM allocation using Seq. 3 in Fig. 4. Each small rectangle represents a node, and an empty rectangle means that no VM was assigned to this node. The *digital* : *letter* pair such as $1 : S$ within a rectangle indicates the first VM of size Small was assigned to this node.

All three test sequences were scheduled, and the energy consumption was collected. Fig. 5 represents the total energy consumed using the three scheduling policies for BS Seeker, Matrix Stressmark, and TPC-W, respectively. The nodes hosting no VM are in idle mode, but not standby mode. For all three benchmarks, RESCUE required less energy compared to Greedy and Round Robin. The energy savings for BS Seeker and Matrix are up to 16.7% and 14.4%, respectively. For TPC-W, energy reduction is between 2.6% and 6.6%.

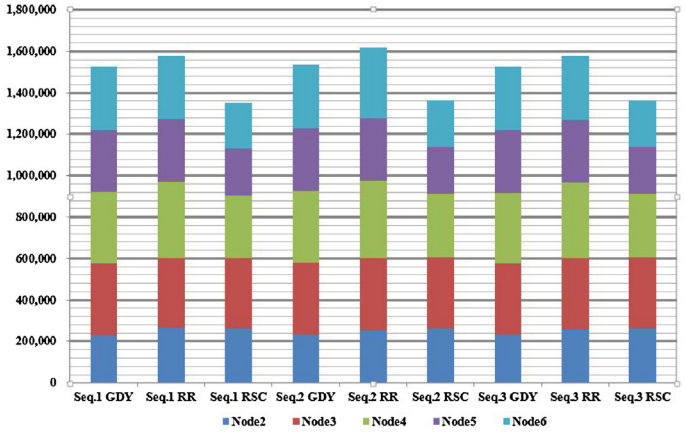
Another interesting observation is that the RR scheduler always consumes the most energy in all three benchmarks, except running TPC-W with Seq.3. Compared to the GDY scheduler, RR distributed the VMs evenly across all nodes. Using the example in Fig. 4, all five nodes host at least one VM using RR, but for GDY, Node-2 hosts no VM. As we discussed earlier, using fewer resources and making as many nodes idle as possible could save more energy than making all nodes run at a lower power state. Thus, the RR scheduler consumes more energy than the GDY scheduler. With RESCUE, it amplifies the effect of making more nodes in idle mode because RESCUE clusters the VMs using ASEE to fewer, but more energy-efficient, resources. As a result, RESCUE reduced the total energy consumption for all test cases.

In the case of BS Seeker and Matrix Stressmark, the greater energy saving was witnessed, where the greatest savings reached up to 16.7%. The high energy saving of those two benchmarks is clear that fewer nodes (Node-2 and Node-3 only) are used than that of GDY and RR schedulers. Furthermore, BS Seeker and Matrix Stressmark are CPU and memory intensive and the dynamic power range is wide, so making more nodes idle can save more energy. However, even though RESCUE saves energy for TPC-W, the differences are tighter than the other benchmarks. The small energy-saving gap occurs for two reasons. First, the dynamic power range of running TPC-W is narrow. Thus, an idle node cannot save more energy than a node running multiple instances. In Fig. 5(c), for Seq.1 and Seq.2, the total energy consumed by the RR scheduler is higher than that of the GDY scheduler. However, for Seq.3, the GDY scheduler consumed more energy than the RR scheduler, even though Node-2 is idle in GDY but hosting three instances in RR, as shown in Fig. 4. Second, all nodes have a similar ASEE as TPC-W. (We addressed this in the previous Section 5.1.) Thus, RESCUE saves energy because it allocates VMs with more energy efficiency than GDY and RR. The scheduling results in Fig. 4 show that in the RR scheduler, Node-3 and Node-4 host more instances, whereas in RESCUE, Node-2 hosts more instances and Node-4 becomes completely idle because Node-2 has a higher ASEE than Node-3 and Node-4. Although the allocation results are quite different, the energy saving is hardly great.

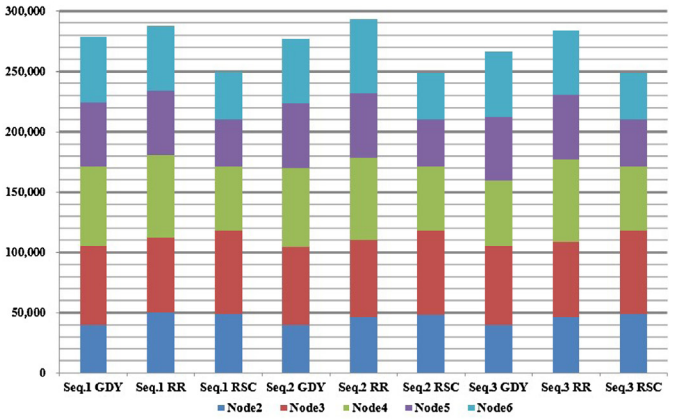
5.4. Aggressive energy saving

As shown in Fig. 4, using RESCUE can use fewer nodes to host all VMs to save energy. However, even an idle machine still consumes a large fraction of observed peak power. In our case, Node-2 to Node-6 consume 53.89%, 58.91%, 61.46%, 63.32%, and 62.43% of the observed peak power, correspondingly. Because using RESCUE leads to more idle nodes, in an aggressive way, we set all idle nodes in an ACPI-defined S3 state to further reduce the total energy consumption. The nodes in the ACPI S3 state consume less than 15 *watts . . . s* energy per second, and setting the nodes hosting no VM achieves huge energy savings. Fig. 6 shows the average normalized energy savings of using RESCUE with sleep state. For the example shown in Fig. 4, RESCUE sets Node-4, Node-5, and Node-6 into a sleep state when running BS Seeker and Matrix, and for TPC-W, only Node-4 is set to sleep mode. As a result, further energy saving is achieved. Compared to RESCUE, RESCUE-with-sleep can reduce energy consumption by 51.5% for BS Seeker, 48.8% for Matrix, and 21.3% for TPC-W, on average, for all three test sequences.

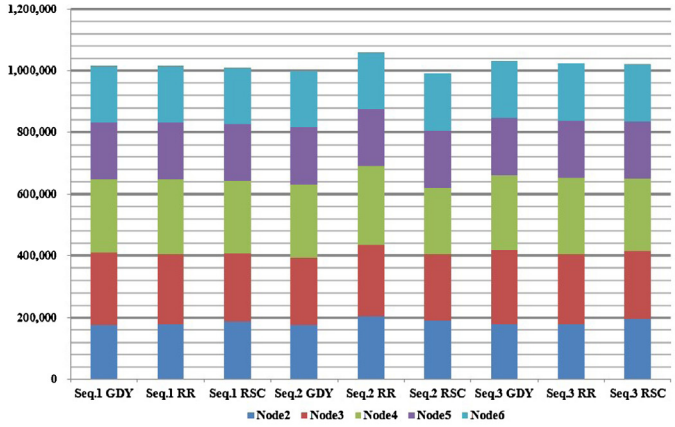
Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008



(a) Total Energy in Joules for BS Seeker.



(b) Total Energy in Joules for Matrix.



(c) Total Energy in Joules for TPC-W.

Fig. 5. Energy consumption for three benchmarks with GDY, RR, and RSC schedulers.

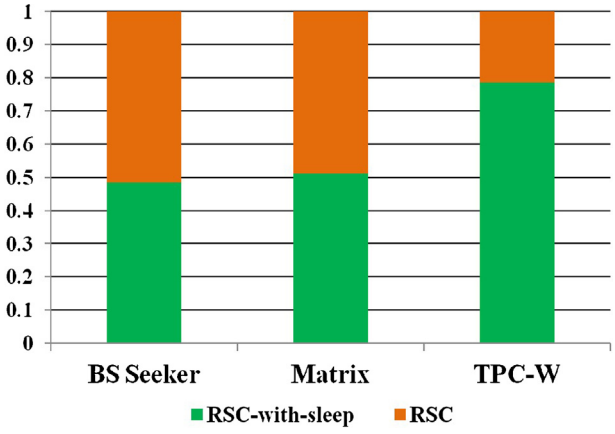


Fig. 6. Normalized average energy savings with sleep state enabled.

5.5. Performance impact

In the design of RESCUE, we schedule the workload on the basis of its ASEE, by which we can reduce the total energy consumption. The performance impact caused by applying RESCUE is negligible. RESCUE does not need real-time performance-counter-involved adjustment during execution. The execution time of CPU-intensive and memory-intensive benchmarks differs a little (within 1%) compared with a system without RESCUE. For the I/O-intensive benchmark, the execution time varies up to 3% because the requests were generated following a nonuniform distribution. The potential performance vibration is that RESCUE assigns the application to a slower machine rather than a faster one due to the slower machine has higher ASEE, but the SAL is guaranteed even running on a slower machine.

5.6. Limitations

It is worth nothing that the proposed scheme performs well under certain scenarios, but it has several limitations that require further investigation. First, RESCUE does not leverage virtual machine migration and proper consolidation with the creation and/or termination of instances in order to increase energy efficiency. In RESCUE, we rarely migrate a VM unless the hardware fails. However, the software or hardware could fail at any time and location. Although frequent VM migration is time consuming, it is still necessary to achieve higher energy efficiency. In RESCUE, we already store all the running instances and its host, so we can run an instance reallocation process to reach a global optimal ASEE with minimal migration operations.

Second, DVFS is another widely used method to conserve energy. In our implementation, RESCUE makes a node operating at the highest frequency when the first VM is assigned to this node. However, always having the CPU running at full speed is not good if only a few VMs are running on a node. Thus, tuning the performance of CPU and memory to make them proportional to the workload requirement is still a hot topic in data centers. Both VM migration and DVFS come with performance overheads naturally. It is easy to combine them with RESCUE, but it breaks the simplicity.

Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008

Finally, the ASEE is defined as the ratio of completed workloads over energy consumed. It is easy to define the workload for a single machine. However, getting the ASEE of a rack or a cluster can be difficult because the workload for a rack or a cluster is usually a mixture of multiple application types, making the question of how to uniformly represent the workload a key problem. This may be a problem when a higher-level scheduler that wants to leverage ASEE is deployed.

Overall, through analyzing the ASEE result of three benchmarks, we show that hardware has a "preference" of application's energy efficiency. Thus, RESCUE takes the facility of such a preference and schedules a user request on the basis of the application's ASEE. The scheduling results also reflect that RESCUE can assign the workload to the most energy-efficient node. As a result, significant energy savings are achieved for three benchmarks, and the aggressive energy-saving method further reduces the total energy consumption. Moreover, even though our implementation is based on the Eucalyptus environment, the same idea can be implemented on other cloud environments such as OpenStack [25].

6. Related work

Despite the advance in system power management, current servers are not energy proportional. To improve proportionality, researchers have proposed an active low-power mode for both CPU and memory. Many works [3–5,11] have used DVFS to control the CPU power because DVFS is well-suited for most workloads. However, CPU DVFS usually depends on modeling and prediction to determine the new frequency and voltage combination for the next epoch. Although CPU power optimization has long been a focus, memory power management has become a new interest. To harness the same benefits of DVFS, [6–9] showed that active low-power modes are also available at the memory, memory bus, and memory controller. Recently, a CPU and memory-cooperated DVFS mechanism was proposed in [26,27], where the disadvantage of performance degradation still exists in DVFS-based power management. In RESCUE, the CPU is set to the highest frequency to reduce the potential performance loss. In addition, reducing the frequency or voltage of the underlying physical CPU can have a negative effect on those VMs that need high-performance virtual CPUs.

Another approach of server power management is workload migration. In the work of [12,28], fewer resources were consolidated to fewer physical resources, by which the total energy consumption can be reduced. However, implementing power reallocation and workload migration brings performance and energy overheads. To overcome such defects, we propose that RESCUE schedule applications on the basis of their energy efficiency. RESCUE needs only one profiling step, and during the running time, no modeling or prediction process is needed, so that there is no performance degradation.

For virtualized platforms, Nathuji and Schwan [29] proposed the VirtualPower approach, which supports online power management, including *hardware scaling* and *soft scaling*. For hardware scaling, the hardware setting is determined by VM-level resource sharing. In particular of DVFS management, the voltage provided to the chip is sufficient for core operation at the highest frequency. The soft scaling uses resource scheduling to emulate the hardware performance change. The soft scaling modified the hypervisor's scheduling attributes by scaling the time slice or issued instructions. Both hardware scaling and soft scaling depend on a real-time estimation model that leverages hardware performance counters, and consequently the estimation overhead leads to performance degradation. Unlike VirtualPower, RESCUE does not need real-time estimation while a preprofiling is still needed. VirtualPower adjusts the hardware performance to reduce the energy consumption, but the application may not run efficiently. Higher CPU frequency does not always consume more energy, because the execution time is less than that of running on a lower CPU frequency, whereas the power of a higher-frequency CPU is higher. An application's energy consumption depends on both hardware and software. Thus, RESCUE allocates applications on the basis of their ASEEs, which takes advantage of the power characteristics of hardware and software.

Verma et al. [30] have studied the problem of optimal VM allocations with the constraints of power consumption and SLA. The authors proposed a power-aware application placement framework, called pMapper, which can improve the energy efficiency of a virtualized data center. The pMapper consists of three managers: a performance manager, a power manager, and a migration manager. Those three managers coordinate the VM's action and resize VMs according to current resource requirements and SLA. To reduce energy consumption, all VMs are consolidated to fewer physical resources, and the idle machines are turned off. In RESCUE, we do not migrate VMs unless a hardware failure occurs. However, VM migration does not guarantee that VMs are assigned to the most energy-efficient hardware. RESCUE assigns VMs on the basis of ASEE, which already maximizes the energy efficiency. Furthermore, pMapper turns off idle machines, which leads to long latency (up to minutes) to switch on machines when the workload increases. Contrarily, RESCUE uses the ACPI S3 state, which has a shorter latency (less than 10 s) to transit to a standby state.

Feller et al. [31] have proposed Snooze for private cloud management. Snooze implements dynamic VM consolidation as a new feature of cloud management platform. The VM consolidation policies leverage the VMs and hosts' resources utilization collected from the performance counters. However, the consolidation policies are implemented at the group manager level which includes only a subset of physical machines. In contrast to Snooze, RESCUE can optimize the VM allocation globally since CC has a global view of each NCs, and therefore VM can be allocated to the most energy efficient node.

There are also research on energy-aware scheduling for specific application types. In [32] and [33], the authors have proposed dynamic VM controller which optimizes power consumption and performance benefits for Web application. The work of [34] has proposed dynamic cluster reconfiguration algorithm to adjusts the active servers for HPC workload. The authors also proposed a server scheduling strategy according to the workload distribution. Those works need the knowledge of application characteristics, which is similar to RESCUE. However, RESCUE can handle multiple types of applications. For different application, the ASEE represents the energy efficiency knowledge of both hardware and software, while those prior work only uses software workload knowledge.

7. Conclusion

Energy efficiency is highly desirable for data centers because energy costs constitute a large portion of the cost of operating data centers. In this paper, we presented RESCUE, an energy-aware scheduler for cloud environments that increases energy efficiency using ASEE. We ran benchmarks on a private cloud with six nodes, showing that our methodology of ranking nodes by their ASEE and then scheduling VMs based on that ranking can lead to more energy-efficient computing. More aggressively, setting node hosting from no VM to an ACPI-defined S3 state, RESCUE can reduce total energy consumption by more than 50%.

Acknowledgment

This work is supported in part by NSF grants CNS-1205338 and CCF-0643521. This material is based upon work supporting while serving at the National Science Foundation.

Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008

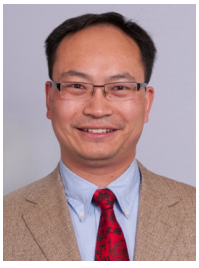
References

[1] A. Greenberg, J. Hamilton, A. David, P. Patel, The cost of a cloud: research problems in data center networks, ACM SIGCOMM Comput. Commun. 39 (2009) 68–73.
[2] J.G. Koomey, Worldwide electricity used in data centers, Environ. Res. Lett. 3 (3) (2008) 034008.
[3] S. Herbert, D. Marculescu, Analysis of dynamic voltage/frequency scaling in chip-multiprocessors, in: Proc. of ISLPED'07, 2007.
[4] C. Lefurgy, et al., Energy management for commercial servers, IEEE Comput. 36 (2003).
[5] D. Snowdon, S. Ruocco, G. Heiser, Power management and dynamic voltage scaling: myths and facts, in: Proc. of Power Aware Real-time Computing, 2005.
[6] Q. Deng, D. Meisner, A. Bhattacharjee, T.F. Wenish, R. Bianchini, Multiscale: memory system DVFS with multiple memory controllers, in: Proc. of ISLPED'12, 2012.
[7] Q. Deng, D. Meisner, L. Ramos, T.F. Wenish, R. Bianchini, Memscale: active low-power modes for main memory, in: Proc. of ASPLOS'11, 2011.
[8] B. Diniz, D. Guedes, W.M. Bianchini R. Jr., Limiting the power consumption of main memory, in: Proc. of ISCA'07, 2007.
[9] A. Merkel, F. Bellosa, Memory-aware scheduling for energy efficiency on multicore processors, in: Workshop on Power Aware Computing and Systems (Hot-Power8), 2008.
[10] J.S. Chase, et al., Managing energy and server resources in hosting centers, in: Proceedings of ACM Symposium on Operating System Principles, 2001, pp. 103–116.
[11] A. Gandhi, et al., Optimal power allocation in server farms, in: Proceedings of International Join Conference on Measurement and Modeling of Computer Systems, 2009, pp. 157–168.
[12] Y. Song, et al., Multi-tiered on-demand resource scheduling for VM-based data center, in: Proceedings of International Symposium on Cluster Computing and the Grid (CCGrid'09), 2009, pp. 148–155.
[13] G. Metri, S. Srinivasaraghavan, W. Shi, M. Brockmeyer, Experimental analysis of application specific energy efficiency of data centers with heterogenous servers, in: Proc. Of IEEE Cloud'12, 2012.
[14] E.L. Sueur, G. Heiser, Dynamic voltage and frequency scaling: the laws of diminishing returns, in: Proceedings of the 2010 International Conference on Power Aware Computing and Systems, 2010.
[15] D. Nurmi, R. Wolski, C. Grzegorczyk, G. Obertelli, S. Soman, L. Youseff, D. Zagorodnov, The eucalyptus open-source cloud computing system, in: Proceedings of IEEE Symposium on Cluster Computing and Grid, 2009.
[16] M.B. Srivastava, A.P. Chandrakasan, R.W. Brodersen, Predictive system shutdown and other architectural techniques for energy efficient programmable computation, IEEE Trans. VLSI Syst. 4 (1996) 42–55.
[17] C.H. Hwang, A.C. Wu, A predictive system shutdown method for energy saving of event-driven computation, ACM Trans. Des. Autom. Electron. Syst. 5 (2000).
[18] D. Meisner, B.T. Gold, T.F. Wenisch, Powernap: eliminating server idle power, in: ACM Sigplan Notices, vol. 44, 2000.
[19] N. Sharma, et al., Blink: managing server clusters on intermittent power, in: ACM SIGARCH Computer Architecture News, vol. 39, 2011.
[20] BS Seeker. http://pellegrini.mcdb.ucla.edu/BS_Seeker/BS_Seeker.html
[21] B. Langmead, et al., Ultrafast and memory-efficient alignment of short DNA sequences to the human genome, J. Genome Biol. 25 (2009) 423–433.
[22] B. Gaeke, P. Husbands, X. Li, L. Oliker, K. Yelick, R. Biswas, Memory-intensive benchmarks: Iram vs. cache-based machines, in: Proceedings on International Parallel and Distributed Processing Symposium (IPDPS), 2002.
[23] Java TPC-W implementation. http://pharm.ece.wisc.edu/tpcw.shtml
[24] Watts up? meters. http://www.wattsupmeters.com
[25] Openstack. http://www.openstack.org/
[26] X. Li, R. Gupta, S. Adve, Y. Zhou, Cross-component energy management: join adaption of processor and memory, in: ACM Trans. Archit. Code Optim., 2007.
[27] Q. Deng, et al., Coscale: coordinating CPU and memory system DVFS in server systems, in: 45th Annual IEEE/ACM International Symposium on Microarchitecture (MICRO), 2012.
[28] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proceedings of the Workshop on Power Aware Computing Systems (HotPower'08), 2008.
[29] R. Nathuji, K. Schwan, Virtualpower: coordinated power management in virtualized enterprise systems, ACM SIGOPS Oper. Syst. Rev. 41 (6) (2007) 265–278.
[30] A. Verma, P. Ahuja, A. Neogi, pMapper: power and migration cost aware application placement in virtualized systems, in: Middleware 2008, Springer, 2008, pp. 243–264.
[31] E. Feller, L. Rilling, C. Morin, Snooze: a scalable and autonomic virtual machine management framework for private clouds, in: Proceedings of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (CCGrid 2012), IEEE Computer Society, 2012, pp. 482–489.
[32] G. Jung, M.A. Hiltunen, K.R. Joshi, R.D. Schlichting, C. Pu, Mistral: dynamically managing power, performance, and adaptation cost in cloud infrastructures, in: IEEE 30th International Conference on Distributed Computing Systems (ICDCS), IEEE, 2010, pp. 62–73.
[33] X. Wang, Y. Wang, Coordinating power control and performance management for virtualized server clusters, IEEE Trans. Parallel Distrib. Syst. 22 (2) (2011) 245–259.
[34] J. Yang, K. Zeng, H. Hu, H. Xi, Dynamic cluster reconfiguration for energy conservation in computation intensive service, IEEE Trans. Comput. 61 (10) (2012) 1401–1416.

Quan Zhang received his BSc degree in information security from Tongji University, Shanghai, China. He is currently a PhD candidate in computer science at Wayne State University, Detroit, Michigan. His research interests include distributed systems, cloud computing, and energy efficient computing system.

Grace Metri is a software tools development engineer for energy and power analysis at Intel Corporation. Her research interests include power-aware computing, energy-efficiency of mobile devices, and energy-efficiency within the cloud. Metri received her PhD and MS in computer science from Wayne State University, Detroit, Michigan.

Sudharsan Raghavan did his bachelors in Electronics & Communications Engineering – Anna university-India. Then he completed his Master's in computer science in Wayne state university with a thesis on energy efficient cloud computing. Currently he is working in Robert bosch as software engineer.



Weisong Shi is a professor of computer science at Wayne State University, where he leads the Mobile and Internet Systems Laboratory. He received his BE from Xidian University in 1995, and PhD from the Chinese Academy of Sciences in 2000, both in Computer Engineering. His research interests include computer systems, sustainable computing, mobile computing, and smart health. Dr. Shi has published over 140 peer-reviewed journal and conference papers and has an H-index of 30. He is the chair of the IEEE CS Technical Committee on the Internet, and serves on the editorial board of IEEE Internet Computing, Elsevier Sustainable Computing, Journal of Computer Science and Technology (JCST) and International Journal of Sensor Networks. He was a recipient of National Outstanding PhD dissertation award of China (2002) and the NSF CAREER award (2007), Wayne State University Career Development Chair award (2009), and the Best Paper award of ICWE'04, IEEE IPDPS'05, HPCChina'12 and IEEE IISWC'12. He is a senior member of the IEEE and ACM, a member of the USENIX.

Please cite this article in press as: Q. Zhang, et al., RESCUE: An energy-aware scheduler for cloud environments, Sustain. Comput.: Inform. Syst. (2014), http://dx.doi.org/10.1016/j.suscom.2014.08.008