View metadata, citation and similar papers at core.ac.uk

brought to you by CORE

provided by INRIA a CCSD electronic archive server

Opportunistic Scheduling in Clouds Partially Powered by Green Energy

Yunbo Li, Anne-Cécile Orgerie, Jean-Marc Menaud

To cite this version:

Yunbo Li, Anne-Cécile Orgerie, Jean-Marc Menaud. Opportunistic Scheduling in Clouds Partially Powered by Green Energy. IEEE International Conference on Green Computing and Communications (GreenCom), Dec 2015, Sydney, Australia. hal-01205911

HAL Id: hal-01205911

https://hal.archives-ouvertes.fr/hal-01205911

Submitted on 4 Dec 2015

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers.

L'archive ouverte pluridisciplinaire **HAL**, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d'enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.

Opportunistic Scheduling in Clouds Partially Powered by Green Energy

Yunbo Li
Ecole des Mines de Nantes, LINA
IRISA, Rennes, France
Email: yunbo.li@emn.fr

Anne-Cécile Orgerie
CNRS - IRISA, Rennes, France
Email: anne-cecile.orgerie@irisa.fr

Jean-Marc Menaud
Ecole des Mines de Nantes, LINA
Nantes, France
Email: menaud@emn.fr

***Abstract*—The fast growth of demand for computing and storage resources in data centers has considerably increased their energy consumption. Improving the utilization of data center resources and integrating renewable energy, such as solar and wind, has been proposed to reduce both the brown energy consumption and carbon footprint of the data centers. In this paper, we propose a novel framework oPportunistic schedulIng broKer infrAstructure (PIKA) to save energy in small mono-site data centers. In order to reduce the brown energy consumption, PIKA integrates resource overcommit techniques that help to minimize the number of powered-on Physical Machines (PMs). On the other hand, PIKA dynamically schedules the jobs and adjusts the number of powered-on PMs to match the variable renewable energy supply. Our simulations with a real-world job workload and solar power traces demonstrate that PIKA saves brown energy consumption by up to 44.9% compared to a typical scheduling algorithm.**

I. Introduction

Data centers have been the key system infrastructure for cloud computing. With the emergence of the Future Internet and the dawning of new IT models such as cloud computing, the usage of data centers, and consequently their power consumption, increases dramatically [1]. A data center (DC) is a facility used to house tens to thousands of computers and their associated components. These servers are used to host applications available in the Internet, from simple web servers to multi-tier applications, but also some batch jobs [2]. Besides the ecological impact, the energy consumption is a predominant criteria for DC providers since it determines the daily cost of their infrastructure. As a consequence, power management becomes one of the main challenges for DC infrastructures and more generally for large-scale distributed systems [1].

In parallel to the expansion of cloud computing, from this recent years, a new model emerges: decentralized cloud infrastructures [3]. To improve the performance of their cloud and to leverage their available infrastructure, telecom operators, like Orange, try to deploy micro data center (20 to 50 servers by micro-DC) at the network border, closer to customers. In this new model, by deploying data centers closer to the user, the response time and throughput would greatly improve.

From an energy point of view, these micro-data centers allow the study of new power supply solutions based on renewable energy, like wind or sun. Using these renewable energy sources can reduce the operating cost but, unfortunately, this kind of energy stays intermittent by nature. To address this problem, two solutions: investing in heavy expensive battery systems to smooth over the day the renewable energy production, or developing new applications management solutions adapted to the electricity production. In this paper, we propose to design a disruptive approach to Clouds resource management which takes advantage of renewable energy availability to perform opportunistic tasks.

The micro-DC receives a fixed amount of power from the regular electrical grid. This power allows it to run the usual tasks. In addition, the micro-DC is also connected to renewable energy sources (such as windmills or solar cells) and when these sources produce electricity, the micro-DC uses it to run more, less urgent, tasks. In order to achieve this energy-aware resource allocation, we distinguish two kinds of jobs to be scheduled on the data center: the web jobs which represent jobs requiring to run continuously (like web server), and the batch jobs which represent jobs that can be delayed and interrupted, but with a deadline constraint. The second type of jobs are the natural candidates of the opportunistic scheduling algorithm.

This paper presents PIKA, a framework aiming at reducing the brown energy consumption (ie. from non-renewable energy sources), and improving the usage of renewable energy for mono-site data center. It exploits jobs with slack periods, and executes or suspends them depending on the renewable energy availability. By consolidating the virtual machines (VM) on the physical servers, PIKA adjusts the number of powered-on servers in order for the overall energy consumption to match with the renewable energy supply. Using simulations driven by real-world workloads and solar power traces, we demonstrate that PIKA consumes 44.9% less brown energy and increases by 110.1% the renewable energy integration ratio in comparison with the baseline algorithm from literature.

The remainder of the paper is organized as follows. Section II presents related work. Section III formalizes the problem. We give a brief overview of PIKA framework in Section IV and we presents our formalization of PIKA in Section V. Section VI gives different resource overcommit policies and the experimental setup is explained in Section VII. Section VIII evaluates the various policies based on simulations. Section IX concludes this work.

II. Related works

We briefly describe related works in energy consumption management and renewable energy integration in data centers.

A. *Energy consumption management in data center(DC)*

Energy consumption management has become an important issue for data centers, but solutions mainly focus on the reduction of the brown energy consumption [1]. To save energy in a single data center, a common goal is to reduce the number of powered-on PMs (ON PMs). The performance of VM placement algorithm directly affects the number of ON PMs. The problem of VM placement is typically modeled as a n-dimensional bin-packing problem which is NP-Hard [4]. In this paper, we only consider the PMs CPU and RAM as resource constraints such that the VM placement problem becomes 2-dimension. The sum of VMs resources requirement on a PM can not exceed this PM's capacity. In [5], Wood *et al.* propose a metric *Volume* with $vol = \frac{1}{1-cpu} \times \frac{1}{1-mem} \times \frac{1}{1-net}$ where cpu, net and mem are the corresponding utilization levels of that resource for the VM on the PM. The higher utilization of a resource corresponds to a higher volume. However, this solution considers that all the PMs have identical hardware which is different from our context (heterogeneous environment). More important, if we swap two values among the three (e.g. cpu and mem), the volume is still holding the previous value, but may be ambiguous.

The above proposals of VM placement investigate the bin packing problem to use less PMs and to pack more VM. They consider the VM resources requirements as a bin size. But there is a a huge difference between the VM resources requirements and the real resources usage of a VM that may also lead a wastage of resources. In [6], Zhang *et al.* design an algorithm (called Scattered) to reduce the number of ON PMs by resource over-commitment. Similarly, in [7] and [8], the authors use the resource over-commitment technique to increase the resources usage. Hence, the over-commit resources technique is required for our framework to increase the PM usage and reduce the number of PMs if the VMs are not fully loaded. In addition, a single PM energy consumption is related to its different components [9]. For example, the CPU and RAM highly impact the PM performance and the energy consumption, more than the other components. Furthermore, the energy consumption of a PM mainly depends on the number of active cores and threads in comparison with the others hardware such as RAM, network card [10]. Thus, over-committing the CPU resource is considered prior than the other resources in our framework. However, the resource over-commitment may lead to overloading of the PM. Then, we need to migrate the VMs from an overloaded PM to another one. In [11], an optimization power-aware migration has been developed. It allows to migrate the VM to the most energy-efficient PM. However, it does not consider the number of migrations. Moreover, the migration also has an energy overhead that may lead to extra energy consumption and performance degradation.

Since in Cloud environments, each job has different arrival time and lifetime, the static VM placement cannot satisfy the dynamicity of the system. It necessitates adequate optimization operations to adjust the number of ON PMs in the current system. The dynamic VM consolidation techniques have now been widely employed in modern data centers, which enable the data center to decrease the number of ON PMs [1].

B. *Renewable energy in DC*

The energy supply can be roughly divided into fossil fuels energy (brown energy) and renewable energy such as the wind energy and solar energy. Integrating the renewable energy into data center can further reduce the brown energy consumption. Due to the intermittent and variable nature of renewable energy, we aim at scheduling opportunistically the jobs according to the renewable energy availability. In [12] and [13], the authors point out the slack time is a key feature that enables the jobs to be delayed until the renewable energy becomes available. However, the authors only take into account batch jobs in their work. In real-world workloads, there are massive number of jobs which do not have slack, like web services for instance. Furthermore, they propose a simple scheduling algorithm without VM consolidation and migration while both techniques can improve the energy consumption.

III. Problem formulation

This paper targets small/medium-size mono-site data center (typically between 20 and 150 servers). The data center consists of several *Physical Machines* (PMs). We assume the computing environment in the data center to be heterogeneous, meaning that the PMs can have different hardware. Each PM has limited resources (CPU, RAM, Network) and has its own disk storage. We assume that the data center has no centralized storage system (such as a NAS for instance) as described in [14]. The PMs with different capacities and performances may potentially lead to different energy consumption for a given VM.

We assume the data center has dual brown and renewable energy power supplies. Each PM has a switch connected with both energy supplies and opts for renewable energy only if there is enough of it. Otherwise, the PMs consume the brown energy from the regular grid. We assume there is no battery or the batteries are only used for emergency cases.

A. *Job*

Our system does not only accommodate periodic jobs which means that jobs may have different lifetimes and can arrive at anytime. Once a job is submitted by a user, it is encapsulated into an individual Virtual Machine (VM). In our system, a VM is considered as the basic unit of resource allocation. It demands two types of resources from PM: CPU and RAM. The lifetime of a VM depends on the job it accommodates. When a job is finished, the VM s destroyed and its reserved resources on the PM are released.

B. *Traces*

We studied anonymized traces provided by the University of Nantes, France. Theses traces concern a VM hosting provider with 55 servers. The traces stretch from the 25th of March 2014 to the 6th of July 2014. They consist in the logs for real CPU, RAM, network and disk utilization of each server every 90 seconds. They also contain the client's requests for VMs with CPU and RAM sizes, and the submission dates. These traces present a realistic scenario in our context.

The Figure 1 illustrates the average CPU and RAM utilization of all the PMs during a normal week in the data center.

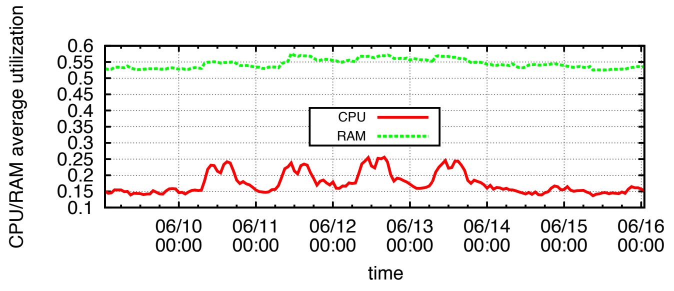

Fig. 1. CPU and RAM real utilization over one-week of real trace

Note that the average CPU utilization keeps low state and far below the average RAM utilization thus leading to a waste of resource. To reduce the number of ON PMs (powered-on Physical Machines), we now describe our proposed PIKA framework in the next section.

IV. PIKA OVERVIEW

Our proposed framework PIKA is designed as a centralized solution. It focuses on minimizing the brown energy consumption in a single small/medium-size data center. As the system is dynamic, PIKA performs the optimization operations periodically. The *optimization cycle* is defined as a slot, such that the *time* in our system is divided into a series of continuous *slots*. As shown in Figure 2, at the beginning of each slot, the broker executes the three main steps. First, the broker checks each PM's state and suspends some jobs from the overloaded PMs. Second, the renewable energy predictor predicts the amount of renewable energy for the current slot and informs the broker about it. Then, the broker determines the number of ON PMs that can be supported by the renewable energy supply. Finally, according to the available resources from these ON PMs, the broker schedules the jobs that can be executed during the current slot. Each component of PIKA is described in the remainder of this section.

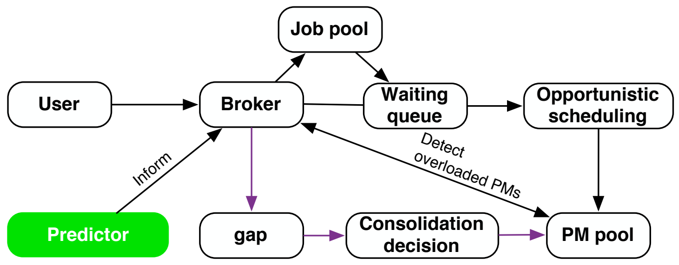

Fig. 2. PIKA framework

A. Renewable energy predictor

We make several simplifying assumptions. The renewable energy prediction performs at the beginning of each slot and the predicted renewable energy amount is used for only one slot. The short-term prediction's advantage is that it significantly reduces prediction errors caused by varying weather. Given an accurate prediction on renewable energy, the broker dynamically switches on and off PMs to adjust the energy consumption in order to maximize the renewable energy integration ratio.

B. Gap

PIKA is designed to be aware of the variable and intermittent nature of renewable energy supply, which makes the *gap* module in PIKA play an important role for the other operations. The renewable energy predictor provides the energy availability that can be consumed in the current slot. The key feature of *gap* enables the broker to dynamically adjust the number of ON PMs following by the renewable energy availability. We detail how the *gap* works in Section V-B.

C. Job pool

The key insight of PIKA is to align the energy consumption with the variable renewable energy supplies. Each job is submitted with the following parameters: $(t_b, T, t_e, t_d, v_i^{\text{cpu}}, v_i^{\text{ram}})$: beginning time, type, execution time, deadline, corresponding the i_{th} VM CPU and RAM requirement. They are pushed into the job pool by the broker. Once the broker gets the energy availability through the renewable energy predictor, it estimates the maximum number of ON PMs via the *gap* function in the current slot. To increase the chance to exploit more renewable energy, we employ the *slack time* for jobs into PIKA.

The *slack* is the most crucial factor in affecting the renewable energy integration ratio. It is given in Equation 1:

$$j_i^{\text{slack}} = t_d - t_b - t_e \tag{1}$$

Through j_i^{slack}, the broker enables a job to be delayed and this increases the chance of exploiting the renewable energy. Consequently, we classify the jobs into two types according to their characteristics: *web job* and *batch job*.

The *web-job* is non-interruptible with a little slack ($<$ 1slot). It has a higher priority than other jobs either on resource allocation or scheduling. For instance, when a web job is submitted, the broker pushes it into a specific job pool with higher priority than the waiting queue of the batch jobs. The web job is then placed on a PM which has sufficient resources to meet its VM resources requirements. Unlike the web job, the *Batch job* can be interrupted or delayed within a slack. Furthermore, because web jobs have higher priority, the batch jobs must wait for all the web jobs to be placed before being allocated only to a ON PM (PM already hosting some web jobs). The batch jobs are not allowed to switch on an OFF PM. Specially, when a batch job's slack is strict inferior to 1, it mutates as a web job that has the same priority as a regular web job. So, the *job pool* in PIKA is divided in two corresponding pools: the *web pool* and *batch pool*. Note that the VM placement process for web job and batch job is individual.

D. Flexible pooling of ON/OFF PMs

The PIKA offers a mechanism for dynamically adjusting the number of ON/OFF PMs that tries to follow the variable renewable energy supply if possible (according to the workload). Firstly, we divide the PMs into two categories: ON PM and OFF PM. Both *web job* and *batch job* are capable of being allocated on an already ON PM if it has sufficient resources to meet their VM resources requirements. An OFF PM can be switched on in the following two cases: 1) there is no more resources to meet the web job's VM resource requirements; 2) there is sufficient renewable energy for all the already ON

PMs at that time and an extra amount from renewable energy supply to switch on new OFF PMs (called *potential ON PM* later in the paper). A batch job is allowed to switch on an OFF PM if and only if it is mutated into a web jobs. The general VM placement process as shown Figure 3.

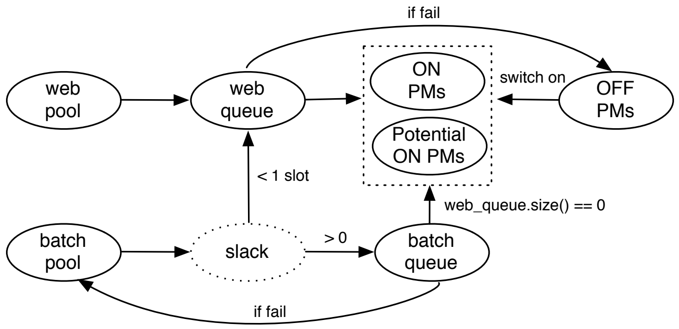

Fig. 3. VM placement for batch and web

In this section, we have given an overview of PIKA's architecture. The next section is dedicated to the formalization of the various algorithms used by PIKA for the resource management and job scheduling.

V. Resource management and job scheduling

The general process of PIKA consists of four major steps detailed in Algorithm 1.

Algorithm 1 General process of PIKA

```
1: Step 1: Detect the overloaded PMs;
2: Step 2: Launch calculating gap process and make the decision of
   whether consolidate depending on the count;
3: Step 3: Update each batch job's slack;
4: Step 4: Select the jobs that can be executed in the current slot and
   place them adequately on PMs (VM placement algorithm). The
   decision of VM consolidation relies on the result of 2nd step.
```

At the beginning of each slot, the steps aim at finding the CPU/RAM utilization of a PM which exceeds its capacity. The gap value is calculated at step 2, the broker makes the decision of either proactive consolidating or switching-on OFF PMs in near future. Then, the broker updates the slack of each batch job and places all the web jobs and the selected batch jobs in this slot. The consolidation decision at step 4 is based on the renewable energy availability.

A. Overloaded PM detection

Since we introduce the resource over-commit policy in PIKA, the varying CPU/RAM load of the jobs at each slot may lead the PMs to overload situations. At the beginning of each slot, if a PM utilization exceeds its capacity, the broker first verifies the batch jobs utilization on this PM and suspend them if necessary.

$$\begin{cases} \text{Suspend the corresponding batch jobs,} & \text{If } U_{batch} \geq U_{exceed}. \\ \text{Push the web job to the waiting queue,} & \text{otherwise.} \end{cases} \tag{2}$$

We first consider suspending the current running batch job on overloaded PM until the PM returns to normal state. If all the batch jobs on overloaded PM has been suspended and the PM still exceeds its capacity, the broker migrate the web jobs to other ON PMs. Assume, for example, there is a PM which exceeds it CPU capacity. Firstly, the broker sorts the VMs in ascending order of CPU utilization. Then, the broker pushes the web job with largest CPU utilization to the web job waiting queue until the PM state becomes normal. If the PM backs to normal state, the broker finishes the overload detection process. The queued web jobs are scheduled onto other ON PMs before any new allocation. The suspended batch jobs are pushed to the batch job pool, then they are either migrated to other ON PMs if possible or put in the waiting list of the next slot to run.

B. Gap

At the beginning of each slot, the broker receives the total electricity generated from renewable energy for the next slot. Ideally, if there is sufficient renewable energy to cover all the electricity need of all already executed jobs, the broker keeps the current PM state. Otherwise, the broker proposes a plan for VM consolidation in order to decrease the number of ON PMs. The metric $gap(t)$ is the key module of PIKA to aid the broker to dynamically adjust the number of ON PMs. The $gap(t)$ function is shown in Algorithm 2.

Algorithm 2 *gap* function

```
1: INPUT: pmList, renewable energy availability in the current
   slot: sumGreenPower OUTPUT: the number of PMs to be
   switched-on or -off
2: pmList.splitTwoSubList() → powerOnPmList and
   powerOffPmList
3: for each pm ∈ powerOnPmList do
4:     sumPmPower += CurrentPower(pm);
5: end for
6: gap = sumGreenPower - sumPmPower
7: count=0;
8: if gap >= 0 then
9:     powerOffServerList.sortByServerScore();
10:    for each server ∈ powerOffServerList do
11:        while gap > 0 do
12:            gap -= pm.getMaxPower();
13:            pm.setPotentialOn(true);
14:            count++;
15:        end while
16:    end for
17: else
18:    powerOffServerList.sortByNumVM();
19:    for each server ∈ powerOnServerList do
20:        while gap < 0 do
21:            gap += pm.getCurrentPower();
22:            pm.setPotentialOff(true);
23:            count−−;
24:        end while
25:    end for
26: end if
27: return count;
```

The $gap(t)$ (line 6) describes the difference between the renewable energy and current energy consumption in one slot. It is calculated as follows:

$$gap(t) = E_r(t) - \sum_i E_{s_i^{on}}(t) \tag{3}$$

Where $E_r(t)$ denotes the predicted amount of renewable energy at the t_{th} slot and $\sum_i E_{s_i^{on}}(t)$ denotes the current energy consumption of all the ON PMs (line 3-5). According to the $gap(t)$, the broker decides whether to consolidate at the t_{th} slot.

If $gap(t) \geq 0$ (line 8-17), it means there is extra renewable energy that may be used to switch on *n* OFF PMs and to run more jobs (opportunistically). Otherwise, it is necessary to switch off *m* PMs in order to reduce the current energy consumption. To determine the value of *n*, the broker first sorts the powered-off PMs list by the metric as follow:

$$e_{\text{efficient}} = \frac{1}{\frac{\max(E_s)}{\max(s_{mips})}} = \frac{\max(s_{mips})}{\max(E_s)} \tag{4}$$

Where s_{mips} is the CPU performance that is presented on megahertz (MHz), the $\max(E_s)$ is the maximum energy consumption of the PM *s* and the $\frac{\max(E_s)}{\max(s_{mips})}$ denotes energy consumption per unit of MHz. The lower the value of $\frac{\max(E_s)}{\max(s_{mips})}$, the more efficient the PM is. The higher is the value of $e_{\text{efficient}}$, the more efficient it is. Each time the broker sorts the OFF PMs list in decreasing order and selects the *n* first PMs to switch on where *n* is equal to *count* (line 27).

When $gap(t) < 0$ (line 18-26), there is no sufficient renewable energy to cover all the ON PMs energy consumption. It necessitates performing a consolidation to decrease the number of ON PMs. Due to VM migration also having an energy overhead, we need to minimize the number of migrations. Thus, the broker seeks a set of PMs which has fewer number of VMs. The size of this set is equal to the value of *count* (line 27).

C. *VM placement*

As described in Section II, by means of resource over-commit policy, the VM placement problem is transformed from 2-dimensional to 1-dimensional bin packing problem. In 1-dimensional bin packing problem, FFD (First Fit Decreasing) is a classic greedy algorithm which is proved to use: maximum 11/9 $\times n + 1$ bins where n presents the number of bins in the optimal solution [15]. The FFD and over-commit policy are combined in PIKA to optimize the resource allocation, the broker first places all the web jobs and then places all or a part of the batch jobs depending on the remaining ON resources. If there are some batch jobs that cannot successfully be scheduled in this step, the broker updates their slack time and reschedules them in a later slot.

But if there is no sufficient resources to place all the web jobs at the first step, the broker suspends a part of or all the batch jobs which are executed on current ON PMs. Then the broker places the web jobs on the current ON PMs. If it still cannot meet all the web jobs resources requirement, the broker actives one or more OFF PMs directly.

D. *Consolidation and migration*

Through the above analysis presented in Section V-C, we detail our heuristic for the problem of dynamic consolidation in context of renewable energy. We subdivide the consolidation problem into the two following issues: when and how to consolidate.

1) When to consolidate: There is not enough renewable energy to cover all the servers energy consumption for the current slot, i.e., the *count* is negative. In this way, the broker attempts to decrease the number of current ON PMs if possible.

2) How to consolidate: The broker gets the number of PMs that should be switched-off via *count*. The broker seeks the PM which has the least number of web jobs and tries to migrate all the web jobs from this PM to others. While a PM is selected to be switched-off, the broker tries to migrate all the web jobs from this PM to the others and interrupts all the batch jobs on this PM. The affected batch jobs are pushed into the batch pool and wait for the broker to complete the web job placement. If there is not enough free resources on other ON PMs to migrate all the VMs on this PM, the broker does not perform the migration and aborts the consolidation process. If each web job on this PM can find an another ON PM to host it, the broker migrates all of them and pushes all the batch jobs on this PM to the batch pool. Once all the VM migrations have been completed, the broker switches off this PM. The broker repeats this process until it reaches the expected number *count* if possible.

VI. Different over-commit resource policies

As stated in Section II, the resource over-commitment is an efficient method to increase the CPU/RAM utilization in order to minimize the number of ON PMs. Figure 1 provides an analysis on the real world workload traces. The VMs average CPU utilization is 15% and 50% of RAM utilization in comparison with their original VM resource requirement. It leads to a significant wastage of the CPU/RAM resources. The over-commitment can be used to further optimize the utilization of PMs. We define four resource over-commitment policies, which cover all the possible cases.

A. *Non over-commit policy*

Basic case, the resources actually allocated to the VMs as requested initially.

$$\begin{cases} \sum\limits_i v_i^{\text{cpu}} \leq S_n^{\text{cpu}}, \quad \forall v_i \in S_n. \\ \sum\limits_i v_i^{\text{ram}} \leq S_n^{\text{ram}}, \quad \forall v_i \in S_n. \end{cases} \tag{5}$$

Where $\sum\limits_i v_i^{\text{cpu}}$ denotes the CPU resource request of all the VM v_i^{cpu} on PM S_n^{cpu}. The first inequation denotes the PM *n* can simultaneously host multiple VM i if and only if the both CPU/RAM resource request of VM can not exceed the PM *n*'s capacity.

B. *Over-commit RAM policy*

In this case, the broker only guarantees the VM CPU resource requirement and over-allocate the RAM.

$$\sum v_i^{\text{cpu}} \leq S_n^{\text{cpu}}, \forall v_i \in S_n. \tag{6}$$

C. *Over-commit CPU policy*

In contrast with the Over-commit RAM policy. The over-commit CPU policy over-commits CPU resource instead of RAM resource. The broker considers the RAM capacity of server and there is no upper-bound of CPU resource.

$$\sum v_i^{\text{ram}} \leq S_n^{\text{ram}}, \forall v_i \in S_n. \tag{7}$$

D. Optimal Over-commit CPU/RAM policy

The optimal over-commitment policy that involved both CPU and RAM resources. This solution needs to analyze the history of each job and predict the job resource utilization for the near future. It is designed to keep the PM utilization always close to the upper bound. However, there is a risk that may lead the PM be overloaded if the prediction is not accurate. Consequently, a huge number of migrations may occur and lead to an additional energy consumption and performance degradation.

VII. Experimental setup

A. Trace-driven simulator

To evaluate the proposed algorithms in PIKA framework under different multiple-parameters configuration, we built a novel trace-driven simulator. The simulator is modeling a single energy-aware data center. It allows to simulate different resource allocation and scheduling policies.

B. Job workload

We use the trace as stated previously in Section III-B and use it for all the simulations in this paper. In this trace, each job consists of job id, time stamp, initial VM resource requirement, the Instantaneous CPU and RAM utilization. To eliminate the noise of CPU and RAM utilization for each job, the CPU and RAM utilization is averaged over the interval T (T = 1 slot)

TABLE I. Job characteristics (Hour)

Type	Number of jobs	Execution time	Slack time
Web	150	24	< 1
Batch	600	6	12

As shown in Table I, we extract a non-holiday week. It contains 750 jobs per day including 150 web jobs and 600 batch jobs. The slack time of a web job is defined as less than a slot. The slack time of batch job is defined as double time than its execution time. Half of jobs are submitted at anytime before noon (*0h - 12h*), and the other half are submitted from noon to night (*12h - 24h*).

C. Solar energy traces

To build the renewable energy workload in the simulator, we use the database provided by the University of Nantes, France. This database records the solar power data every hour (24 values per day). We choose the trace from the same week as the trace we used for the jobs.

VIII. Evaluation

A. The performance of resource over-commitment policy

We implement the aforementioned resource over-commitment policies on two PMs. PM-1 consists of 12 cores and 48GB RAM (2933 MHz each core). PM-2 consists of 16 cores and 192 GB RAM (2933 MHz each core). We submit the same number of jobs for the two PMs. We combine these resource over-commitment policies with FFD algorithm as mentioned in Section V-C.

Table II and Table III show the number of hosted VMs for different resource over-commitment (OC) compared to the

TABLE II. Number of VMs PM-1

Policy	Mean	St.dev.
NO.OC	6.356	2.154
RAM.OC	6.724	2.651
CPU.OC	27.035	2.744
OPT	60.994	9.775

TABLE III. Number of VMs PM-2

Policy	Mean	St.dev.
NO.OC	8.141	5.337
RAM.OC	11.451	2.656
CPU.OC	44.893	4.515
OPT	100.441	5.904

solution solution combining both over-commitment techniques (OPT). On PM1, the first two policies (NO OC and RAM OC) have nearly identical performance. CPU OC hosts more than 3 times more VMs than both NO OC and RAM OC policies . On PM-2, CPU OC also offers 400% better performance than both NO OC and RAM OC. This can be explained by an analysis of the trace: for the vast majority of VMs, the CPU resource requirement is higher than the RAM resource requirement (*i.e.* $\frac{\text{number of CPU cores}}{\text{RAM (GB)}}$, this ratio is > 1). Besides for PMs, the ratio are 1/4 and 1/12 respectively (PM1 and PM2). When the broker maps the VMs to the PMs, the total CPU resource requirements of VMs will reach the PM's CPU upper bound before the RAM reaches its upper bound. Therefore, the result of CPU OC it better than the NO OC and RAM OC policies. However, the OPT outperforms up by 50% in comparison with CPU OC. Recall that, OPT dynamically adjusts the over-commitment threshold under assumption with high accurate VM utilization predictor.

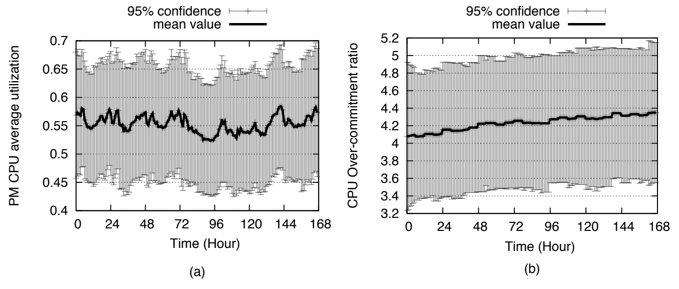

Fig. 4. (a) CPU real-time utilization; (b) total VM allocation ratio

Figure 4 shows the CPU over-commit ratio and the mean value of real PM CPU load of CPU OC policy. The mean value of real CPU load of PM is between 45% and 70%. It over-commits near 4.1 times more the CPU resource than the PM original CPU capacity over 168 hours.

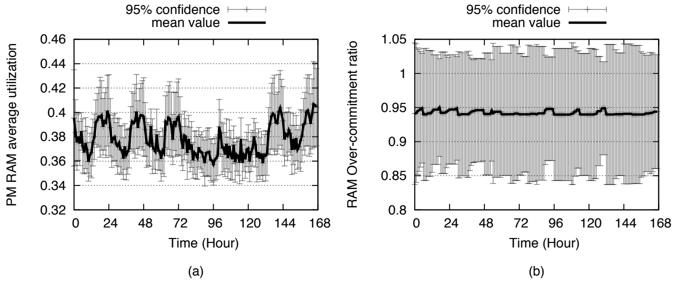

Fig. 5. (a) RAM real-time utilization; (b) total RAM allocation ratio

Figure 5 shows the RAM over-commit ratio and the mean value of real PM RAM load of CPU OC policy. The mean value of real CPU load of PM is between 34% and 44%. It does not over allocate the RAM resources and the mean value of real PM RAM load is near 95%. The above Figures 4 and

5 illustrate the fact that the CPU OC is offering a reasonable performance without a complete knowledge of future and rarely leads a PM to be overloaded. For this reason, we chose to implement CPU OC into PIKA.

B. Energy model

1) Physical machine model: The variation in energy consumption of a PM mainly depends on CPU utilization [1]. We experiment multiple tests on Taurus node at the Lyon site of Grid'5000, a large-scale and versatile test-bed for experiment-driven research. Each node has 12 cores and each core presents 8.3% overall CPU utilization. We use *stress benchmark* to vary CPU utilization in order to estimate the server's energy consumption according to its load. We activate one more core each time and keep the core at 100% utilization within 300 seconds. The test begins with 0 core (*idle state*).

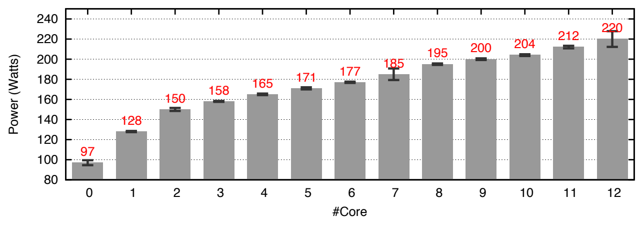

Fig. 6. Energy consumption of node Taurus at different number of cores in Watts on grid'5000 Lyon site. It owns 12 cores x 2933 MHz

Figure 6 presents the energy consumption with different numbers of active cores. Note that an idle PM (0% CPU utilization) consumes about half of full charged PM energy consumption (100% CPU charge). Similar observations can be found in literature [9]. We model the PM energy consumption as successive steps. The total CPU resource is divided into 12 intervals (i.e. number of cores). The number of cores transforms on CPU utilization (i.e. 0 core = 0%, 1 core = 8.33%, 2 cores = 16.66% and so on till 12 cores = 100% CPU utilization). We create a sequence of number corresponding to the cores. The real PM utilization is easily falling into an interval of two consecutive numbers *a, b* ($a < b$) from this sequence. The PM energy consumption is the difference between a's energy consumption and b's energy consumption. Then, we construct a linear model between the two numbers *a, b* to calculate the PM energy consumption, this model is similar to the one used in [11].

$$E_n = E^a + (E^b - E^a) \times \frac{u_n - a}{b - a} \text{ , where } a < u_n < b \quad (8)$$

where E^a denotes the lower bound a energy consumption and E^b denotes the upper bound b energy consumption. u_n present the PM CPU utilization and $\frac{u_n - a}{b-a}$ denotes the percentage of difference between PM utilization and the lower bound. Moreover, there is a switching cost, as the broker needs to dynamically switch on PMs or switch them to sleep mode. We define a fix energy overhead in the later simulation when broker switches on a PM, this value has been measured experimentally.

2) VM energy consumption model: The VM energy consumption model consists of two modules: VM creation energy overhead and VM migration overhead. The VM creation energy overhead is defined as a fixed value in the simulator. The energy consumption of VM migration depends on the following parameters: migration duration time and the energy consumption per unit of time. The study of [16] shows that the duration of live migration depends mostly on the memory and disk used by the migrated VM. The VM migration will sightly increase the CPU utilization on destination server. As above-mentioned, the energy consumption of PM increases almost linearly with CPU utilization, we formalize the linear model for energy consumption of VM live migration as following:

$$E_{Migration} = \frac{C_{RAM} + C_{DISK}}{B} \times E_{CPU} \quad (9)$$

where B denotes the bandwidth between the PMs. $C_{RAM} + C_{DISK}$ denotes the migration overhead that is the sum of the memory size and the disk size of a VM , $\frac{C_{RAM}+C_{DISK}}{B}$ denotes the duration of migration and E_{CPU} is the extra CPU energy consumption on destination PM per unit of time.

C. Energy consumption

The result of energy consumption for both baseline algorithm and PIKA are shown in Figure 7. The top curve presents the baseline result (with FFD algorithm to allocate the VMs) and the bottom corresponds to PIKA.

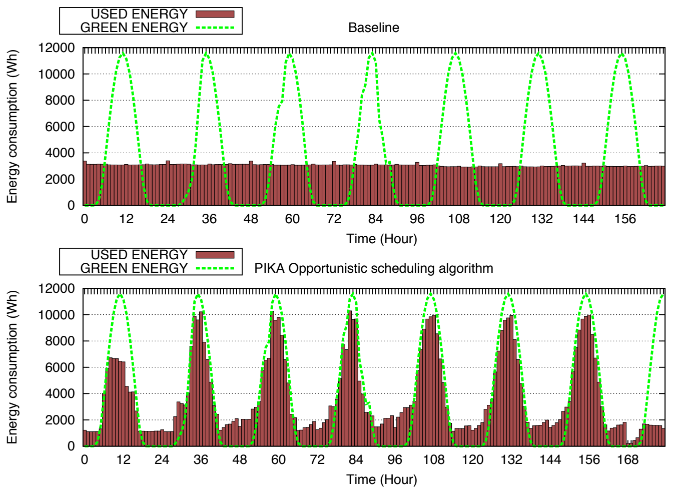

Fig. 7. Energy consumption : baseline vs PIKA

The energy consumption of baseline is flat. The workload scheduling is not affected by the variable renewable energy supply (the green curve). The energy consumption of PIKA is following the renewable energy variations. PIKA significantly increases the renewable energy integration into data center. While the renewable energy becomes unavailable, the broker switches off some ON PMs and only launches some essential jobs (web job, mutated batch job and a few of batch jobs on remaining resources). Due to this behavior, PIKA finishes all the job 11 hours later than the baseline. Indeed, in PIKA, the broker opportunistically schedules the batch jobs. So, some batch jobs are delayed when there is no sufficient renewable energy.

D. Simulation results

Table IV shows the result of brown-, renewable- and total-energy consumption for the baseline and PIKA.

TABLE IV. ENERGY SAVING RESULTS (KW)

Algorithm	Total E. C.	Brown E. C.	Renewable E. C.
Baseline	513.633	259.559	254.073
PIKA	676.895	142.957	533.938
	31% ↑	44.9%↓	110.1%↑

Compared to the baseline, PIKA reduces by 44.9% brown energy consumption and increases by 110.1% the renewable energy integration. The results show that PIKA significantly reduces the brown energy consumption in compare with the baseline, representing a typically energy-efficient algorithm (but not renewable-aware). The results also indicate that PIKA consumes 31% more energy in total. This is because PIKA performs dynamic VM consolidation to adjust the number of ON PMs and that leads to a large number of VM migrations compared with baseline (the migration in baseline is only in case of overloading PM). But all of this extra energy consumption comes from renewable energy supply. So, this extra energy is not used and thus wasted in the baseline case.

This work shows the opportunity created by medium-sized data centers partially powered by renewable energy in order to save energy for distributed Cloud infrastructures, such as the one promoted by the Future Internet.

IX. Conclusion and future work

Data centers partially powered by renewables become attractive for the new generation cloud architectures. It significantly reduces the traditional energy consumption and CO_2 footprint. This paper is dealing with resource allocation and opportunistic job scheduling in a small/medium mono-site data center. Our proposal is called PIKA and the preliminary results demonstrate that it can outperform the classical energy-efficient VM management algorithms.

Our next work involves studying the workload prediction, this method may still optimize the resource utilization to further reduce the brown energy consumption. Furthermore, we plan to integrate the management of the cooling system within PIKA in order to jointly and more efficiently tackle VM placement and job scheduling.

Acknowledgment

This work has received a French state support granted to the CominLabs excellence laboratory and managed by the National Research Agency in the "Investing for the Future" program under reference Nb. ANR-10-LABX-07-01.

References

[1] A.-C. Orgerie, M. D. d. Assunçao, and L. Lefèvre, "A Survey on Techniques for Improving the Energy Efficiency of Large-scale Distributed Systems," *ACM Computing Surveys*, vol. 46, no. 4, pp. 47:1–47:31, Mar. 2014.

[2] I. Menache, O. Shamir, and N. Jain, "On-demand, Spot, or Both: Dynamic Resource Allocation for Executing Batch Jobs in the Cloud," in *USENIX International Conference on Autonomic Computing (ICAC)*, Jun. 2014, pp. 177–187.

[3] M. Bertier *et al.*, "Beyond the Clouds: How Should Next Generation Utility Computing Infrastructures Be Designed?" in *Cloud Computing*, ser. Computer Communications and Networks. Springer, 2014, pp. 325–345.

[4] C. Chekuri and S. Khanna, "On Multi-Dimensional Packing Problems," in *Annual ACM-SIAM Symposium on Discrete Algorithms (SODA)*, 1999, pp. 185–194.

[5] T. Wood, P. J. Shenoy, A. Venkataramani, and M. S. Yousif, "Black-box and Gray-box Strategies for Virtual Machine Migration," in *USENIX Conference on Networked Systems Design & Implementation (NSDI)*, 2007.

[6] X. Zhang, Z.-Y. Shae, S. Zheng, and H. Jamjoom, "Virtual machine migration in an over-committed cloud," in *IEEE Network Operations and Management Symposium (NOMS)*, 2012, pp. 196–203.

[7] M. Dabbagh, B. Hamdaoui, M. Guizani, and A. Rayes, "Efficient datacenter resource utilization through cloud resource overcommitment," in *INFOCOM Workshops*, 2015, pp. 330–335.

[8] ——, "Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment," *IEEE Network*, vol. 29, no. 2, pp. 56–61, 2015.

[9] F. Quesnel, H. Kumar Mehta, and J.-M. Menaud, "Estimating the Power Consumption of an Idle Virtual Machine," in *IEEE International Conference on Green Computing and Communications (GreenCom)*, Beijing, China, Aug. 2013.

[10] X. Fan, W.-D. Weber, and L. A. Barroso, "Power provisioning for a warehouse-sized computer," in *ACM SIGARCH Computer Architecture News*, vol. 35, no. 2, 2007, pp. 13–23.

[11] A. Beloglazov and R. Buyya, "Energy efficient resource management in virtualized cloud data centers," in *IEEE/ACM International Conference on Cluster, Cloud and Grid Computing (CCGrid)*, 2010, pp. 826–831.

[12] A. Krioukov *et al.*, "Integrating Renewable Energy Using Data Analytics Systems: Challenges and Opportunities," *IEEE Data Engineering Bulletin*, vol. 34, no. 1, pp. 3–11, 2011.

[13] C. Chen, B. He, and X. Tang, "Green-aware workload scheduling in geographically distributed data centers," in *IEEE International Conference on Cloud Computing Technology and Science (CloudCom)*, 2012, pp. 82–89.

[14] N. Beldiceanu *et al.*, "The EPOC project: Energy Proportional and Opportunistic Computing system," in *International Conference on Smart Cities and Green ICT Systems (SMARTGREENS)*, Lisbonne, Portugal, May 2015.

[15] M. Yue, "A simple proof of the inequality $FFD(L) \leq \frac{11}{9}OPT(L)+1, \forall L$ for the FFD bin-packing algorithm," *Acta mathematicae applicatae sinica*, vol. 7, no. 4, pp. 321–331, 1991.

[16] Q. Huang, F. Gao, R. Wang, and Z. Qi, "Power consumption of virtual machine live migration in clouds," in *International Conference on Communications and Mobile Computing (CMC)*, 2011, pp. 122–125.