Vehicular Communications 9 (2017) 199–210

Contents lists available at ScienceDirect

Vehicular Communications

www.elsevier.com/locate/vehcom

An evergreen cloud: Optimizing energy efficiency in heterogeneous cloud computing architectures [☆]

Mohamed Abu Sharkh [*], Abdallah Shami

Department of Electrical and Computer Engineering, Western University, London, Canada

ARTICLE INFO

Article history:
Received 7 June 2016
Received in revised form 23 December 2016
Accepted 23 February 2017
Available online 4 March 2017

Keywords:
Cloud computing
Energy efficiency
Scalability
Virtualization
Network and systems monitoring and measurements

ABSTRACT

In a heterogeneous Cloud network scenario where a Cloud computing data center serves mobile Cloud computing requests, Cloud providers are expected to implement more innovative and effective solutions for a list of long standing challenges. Energy efficiency in the Cloud data center is one of the more pressing issues near the top of that list. Cloud providers are in constant pursuit of a system that satisfies client demands for resources, maximizes availability and other service level agreement metrics while minimizing energy consumption and, in turn, minimizing Cloud providers' cost.

In this work, we introduce a novel mathematical optimization model to solve the problem of energy efficiency in a cloud data center. Next, we offer a solution based on VM migration that tackles this problem and minimizes energy efficiency in comparison to other common solutions. This solution includes a novel proposed technique to be integrated in any consolidation-based energy efficiency solution. This technique depends on dynamic idleness prediction (DIP) using machine learning classifiers. Moreover, we offer a robust energy efficiency scheduling solution that does not depend on live migration. This technique, termed Smart VM Over Provision (SVOP), offers a major advantage to cloud providers in the cases when live migration of VMs is not preferred due to its effects on performance. We evaluate the aforementioned solutions in terms of a number of critical metrics, namely, energy used per server, energy used per served request, acceptance rate, and the number of migrations performed.

1. Introduction

Cloud providers are in continuing pursuit of solutions to adapt to the surge in cloud technology demand levels. The percentage of Internet of Things devices is gradually increasing on the cloud client side. This means cloud providers need to cater for requests from client devices from heterogeneous distributed and more diverse than ever. In a heterogeneous cloud network scenario where cloud computing data center serves mobile cloud computing requests, major challenges with veracity are suffered [1]. The sheer volume of the connected devices along with the connection speed and user expectations for response from the server put a stress both on the network and on back end cloud data center to perform up to the required level [2].

Simultaneously, cloud providers are under a lot of pressure to manage costs in a way that the performance demands does not cause unrealistic operational cost which could comprise the business objectives [3]. The aim here for a cloud provider is to serve the high volume of requests which are received continuously from a diverse set of constantly changing (and moving) devices [4]. As illustrated in Fig. 1, this is done by successfully receiving the request data from client device and then scheduling these requests to the corresponding virtual machine in the cloud data center based on the functionality or application required. From there, the computational part is done and then the results are sent back to the client. A major challenge in this scenario is how to serve these requests with the required performance while minimizing the energy the cloud data center uses.

To tackle this challenge, cloud providers are expected to implement more innovative and effective solutions for a list of long standing challenges faced by the industry. Energy efficiency in the cloud data center (DC) is one of the more pressing issues near the top of that list [1–3,5–7]. As DCs expand, so does their energy consumption. Electricity used by servers doubled between 2000 and 2005, from 12 to 23 billion kilowatt hours [8]. This is not only due to the increasing amount of servers per DC, but also the individual server consumption of energy has increased too. The increase in energy consumption is a major concern to the data center own-

☆ This work is supported by the Natural Sciences and Engineering Research Council of Canada (NSERC-STPGP 447230).

* Corresponding author.
E-mail addresses: mabusha@uwo.ca (M. Abu Sharkh), ashami2@uwo.ca (A. Shami).

http://dx.doi.org/10.1016/j.vehcom.2017.02.004
2214-2096/© 2017 Elsevier Inc. All rights reserved.

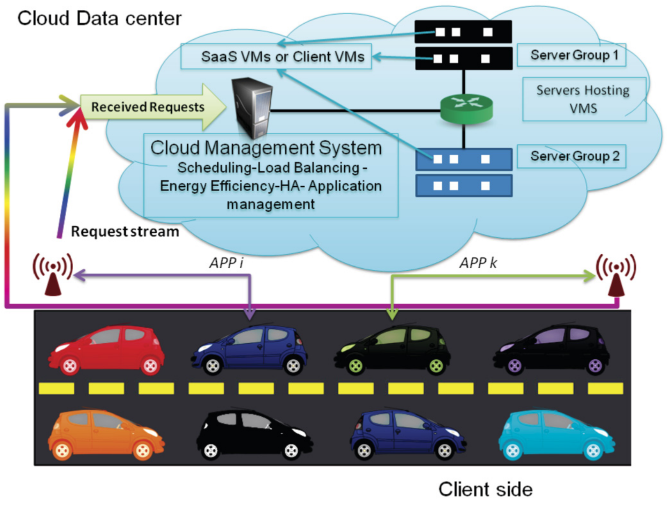

Fig. 1. Heterogeneous IoT clients sending diverse computational requests to the public cloud.

ers because of its effect on the operational cost. It is also a major concern for governments because of the increase in data centers' carbon footprint [9].

Cloud technology adoption rates are another factor to look out for. 78% of U.S. small businesses will have fully adopted cloud computing by 2020, more than doubling the current 37% [10]. The percentage grows to 90% when looking at large businesses (larger than 1000 employees) [11]. The U.S. Small and Medium Business (SMB) cloud computing and services market will grow from 43 billion dollars in 2015 to 55 billion dollars in 2016 [10]. This trend is consistent in Europe as well. The percentages of small, medium and large businesses adopting cloud technologies in UK are 46, 63 and 82% respectively. In Germany, the percentages are 50, 65 and 86%.

Client demand growth shows promising potential as well. In a survey conducted by the rightscale.com team, 68% of enterprises indicated they run less than a fifth of their application portfolios in the cloud. 55% of enterprises report that a significant portion of their existing application portfolios are not in cloud, but are built with cloud-friendly architectures [12]. This is used to serve major functions like data protection (backup), business continuity (replication, disaster recovery), archiving and file services and office enablement (sharing, synchronization, collaboration).

Power consumption in cloud data centers is a pressing issue for cloud providers. Power costs represent between 25% and 40% of the operational expenses of a data center [13]. The Natural Resources Defense Council (NDRC) published a data center efficiency assessment in Aug. 2014, as an attempt to depict the scale of data centers around the world [9]. The study mentions that "U.S. data centers are on track to consume roughly 140 billion kilowatt-hours of electricity annually by 2020, equivalent to the output of 50 large power plants (each with 500 megawatts capacity)" [9]. Another fact here is their assessment of energy efficiency. "An analysis by the NRDC in partnership with Anthesis finds that up to 30% of servers are obsolete or not needed and no longer needed, other machines are grossly underutilized" [9]. Persistent issues obscuring efficiency include:

- Peak provisioning.
- Limited deployment of virtualization technology.
- Failure to power down unused servers,
- Challenges with efficiency incentive programs.
- Competing priorities: (keeping costs low and maintaining high levels of security, reliability, and uptime for their clients.)

These are 5 out of the 8 main factors cited that affect power efficiency and stand in the way of a staggering 40% potential improvement in power consumption (with the other 3 being: challenges with efficiency incentive programs, short sighted procurement practices, and split incentives between the parties paying the power bills and those managing the IT equipment). These 5 factors are all largely affected by data center load planning and management. An efficient scheduling energy – aware algorithm is in need. This algorithm should exploit the benefits that come from virtualization technologies, optimal demand driven provisioning, and efficient load modeling.

In this work, we explore the possible venues to reach this robust energy efficiency solution for cloud environments.

Upon reviewing the available solutions as can be seen in detail in the following section, a need arises for a more complete solution for energy efficiency in cloud data centers. Each one of the solutions surveyed – summarized in Tables 1 and 2 – despite covering a significant flavor of the problem, does not offer a comprehensive vision. A mathematical model would be crucial to provide the theoretical base for the solution. To achieve an outstanding solution for such a layered problem, we offer the following contributions:

1 – We first formulate the problem as mathematical optimization problem with the objective of minimizing the energy consumption.

2 – A new technique called Dynamic Idleness Prediction (DIP) is introduced where the future demands for VMs are considered when placing/scheduling the VM on a host. This technique is based on using an artificial intelligence classifier (in our case REPtree) to predict the nature of the load every VM will receive in a prespecified future period. A novel scheduling/placement technique was constructed by combining DIP and the resource based scheduling technique we introduced in an earlier publication [14]. In this technique, DIP dictates VM initial placement and VM migration in order to reach more efficient consolidation-based energy efficiency as opposed to traditional methods like first fit, round robin or greedy methods.

3 – We introduce a novel technique for energy efficient VM management that does not depend on Migration. The technique is called Smart VM Overprovision (SVOP). This technique depends on the DIP technique described earlier to dictate the overprovision of VMs by choosing the mostly idle VMS to be switched off and in turn minimize lost requests.

4 – We use a data set published by of Google (data center traces [15]) to evaluate the two proposed methods. The performance of these methods is compared to common scheduling algorithms in terms of critical energy efficiency metrics including: energy used per server, energy used per served request, service rate, and number of migrations performed.

The remaining sections are divided as follows. The related work is visited in Section 2. Section 3 presents the system model. The mathematical formulation for the energy efficiency problem in virtualized cloud data centers is presented in Section 4. In Section 5, a consolidation-based energy efficiency solution is proposed and details are put forward. Section 6 presents our novel non-consolidation-based energy efficiency solution: Smart VM Over Provision (SVOP). The experimental setup is described in Section 7 followed by analysis of the experimental results. Section 8 concludes the paper.

2. Related work

2.1. Server consolidation

Classic solutions to the energy efficiency problem in the cloud – if we disregard cooling processes – stem primarily from two ideas: consolidating loads on fewer servers (hosts) or using variations of

Table 1
A comparison of energy efficiency virtualization based efforts in cloud environments – Part 1.

Technique	Offer Optimization model	Scheduling considers network & computational resources	VM modeling	Application layer	Network model
[16]	No	No	CPU and storage requirements	No	No
[17]	No	No	MIPs,ram,BW, VMs can be resized	No	BW
[18]	Not full	No	VM host applications of different types and up to one replica per VM	Yes	No
[19]	Yes	No	CPU & memory requirements (replicas of a VM share CPU), 1 client per VM	Yes	No
[20]	Yes	Yes	Fixed scheduling, no migration	No	Full
[21]	No	No	N/A	N/A	N/A
[22]	No	No	CPU, storage, memory, BW,communication demands	Tasks request Comp+ network resource	BW +fixed source and destination network
This solution	Yes	Yes	Detailed resources: User & workload profile	Yes	BW

Table 2
A comparison of energy efficiency virtualization based efforts in cloud environments – Part 2.

Computational resources	Live migration (major method used)	Guaranteed reservation	Centralized/ decentralized	Scheduling algorithm
CPU (numerical) & storage Counting by usage not absolute	No migration, Best allocation at a desired utilization level vs. degradation considered	Yes	C	Bin packing modified + maximize euclidean distance
CPU (multi-core) ram	Live migration according to current utilization of resources	SLA violation	D	Multi-dimensional bin packing
CPU	Yes (for replicas of Apps and for VMS)	Adaptable (affects performance)	C	Multi-layer Bin packing, Looking for to satisfy power-performance while minimizing the number of hosts
CPU & memory	No migration, Dynamic programming	Yes	C	Dynamic algorithm to place VMs and local search to check servers to be consolidated
No	No	Yes	C	Focused on traffic engineering (network optimization in the cloud)
CPU	No	N/A	C	No scheduling algorithm, CPU idle intervals dynamically predicted
CPU, Mem, storage	Enabled enabling DVFS using multiple power models	Yes	C	Green-scheduling and round robin scheduling
CPU, Memory, storage, user defined resources	Migrated VM chosen via VM idleness estimator	Yes	C or D	First fit scheduling + Dynamics prediction

dynamic voltage and frequency scaling (DVFS). The latter includes algorithms that exploit dynamic power management in servers. Server computational power/speed can be toned down and thus energy consumption decreases. Server consolidation can be seen in early papers like [16]. The algorithm proposed in [16] executes the consolidation of different applications on cloud computing data center servers. The idea is to consolidate VMs on the least amount of servers and then switch the unused servers off or to an idle state. That problem is modeled as a bin-packing problem with the assumption that the servers are the bins and they are full when their resources reach a predefined optimal utilization level. This utilization level is calculated and set beforehand. The optimal utilization is a level where the a balance is reached between the resource utilization and performance degradation caused by pressuring the resources. The issues faced by over utilization are cache contentions, conflicts of CPU functional units, desk scheduling and desk write buffer issues and that is only from the computational side of things.

The final objective is to minimize energy consumption per transaction. Resources used are processor and disk space. The heuristic algorithm is then used to allocate workloads to servers or bins. This heuristic tries to maximize the Euclidean distance between the current allocations of the servers and the optimal utilization point of each server. There were no comparisons to the optimal solution. Also, power consumption by network components is not considered. Another issue here is that it is debatable whether finding an optimal point for each server is only based on utilization without considering other factors like the type of the application.

Another approach can be seen in [17], where methods for live migration of VMs according to the current utilization of resources are introduced. Each node has a CPU, which can be multi-core, with performance defined in Millions of Instructions Per Second (MIPS). Besides that, a node is characterized by the amount of RAM and network bandwidth. The aim is to prevent service level agreement (SLA) violations which occur when a VM cannot get the requested amount of resource, which may happen due to VM consolidation. A decentralized resource allocation system is offered containing a dispatcher, global and local managers.

Only utilization of CPU is considered. “The main idea of the policies is to set upper and lower utilization thresholds and keep total utilization of CPU created by VMs sharing the same node between these thresholds” [17] using live migration. A challenge here is to determine values of the utilization limits (thresholds).

In [18], the authors offer a migration algorithm that depends on copying the VMs and dividing the original CPU allocation among the copied VMs while keeping the memory allocated at the same level. This would cause an additional resource consumption. Allocation is done using dynamic programming and a local search is run after to find opportunities for consolidation. This algorithm is run periodically to improve performance. The period the algorithm is run can have a great impact on the success of the method. The balance between the running cost and the running gain along with the workload change period should be investigated.

2.2. Adaptive allocation

Consolidation is far from a standalone solution though. For changing workload patterns, the cost of moving VMs in and out of a server in terms of performance and power could be worse than keeping them where they are. A careful consideration of the amortization period of the migration costs is required for a successful decision. The authors of [19] consider this when presenting their solution. They propose a central controller (termed Mistral) that balances steady state performance and power with the dynamic adaptation costs under changing workloads.

The authors assume workloads from multi-tiered applications are being scheduled on cloud VMs. Each application type is associated with a set of transaction types through which users access its services. Each transaction type corresponds to a unique call graph of some of the application types. The mean request rates for transaction types are combined in a vector to specify the workload of applications. Mistral controllers are called periodically to impact the VM locations and their CPU allocations.

“Costs of these adaptation actions are measured experimentally offline for different workloads and VM placements, and are stored in tables used at runtime” [19]. This includes adaptation duration, change in response time for the application being adapted as well as co-located applications, and change in power consumption during the adaptation. A model is introduced to predict workloads of each component and define the stability interval following the current adaptation. No details are offered as for the structure or operation of the workload predictor. Experimental results were shown for data centers of up to just 8 servers. Also, calculating the costs offline sacrifices precision and often would not take interactions into account as it assumes that every time the decision will cost the same in terms of power and performance for example.

2.3. Methods focusing on CPU utilization

We can see a detailed discussion of the network resource energy consumption in the cloud data centers in sources like [20]. VMs are assigned to servers with the objective of reducing the amount of traffic and generating favorable conditions for traffic engineering. Moreover, the number of active switches and balance traffic flows is decreased depending on the relation between power consumption and routing, to achieve energy conservation.

There are a few existing schemes that transition a CPU into various low-power and sleep states to reduce its idle power. One of the more recent efforts using this approach is in [21]. The paper offers a method to predict which CPUs will be idle based on analyzing the reading of each CPU hardware parameters. This helps the decision making process in order to achieve intelligent sleep states. This means that a more accurate prediction of the period a CPU might be idle reflects on the choice of which sleep state the CPU is moved into. The CPU performance metrics monitored are IPC, cache miss rates, structure occupancies, branch predictor statistics, and others. These readings are used as an input to an expert system based on classifiers like boosted regression trees. The output is the length of the CPU idle interval. The cost of monitoring here could be a decisive factor. Decision based CPU readings are on different level of speed to processes like running a VM on a server or switching off the server.

As one of the most detailed cloud simulators available [22–26, 30,31], Greencloud arises as a powerful tool to evaluate energy efficiency in cloud environments. GreenCloud was developed as a simulator with a focus on energy efficiency and fine grained networking capabilities. The prime purpose cited for building GreenCloud is mitigating overprovision issues [22]. Overprovision happens in a data center due to the loads constantly changing on the computational and communication resources. The average load can be as low as 30% of the data center server and network capacity [22]. This, in turn, causes the data center to systematically use more power than the optimal value. In GreenCloud, solutions implementing server consolidation and dynamic server power management are simulated with an option to expand to a hybrid solution containing both.

GreenCloud offers simulation capabilities including multiple topology choices (2 layers and 3 layers) and it offers communication through packets using the underlying NS-2 simulator features. GreenCloud also offers the choice of scheduling tasks (user requests) on hosts directly or on virtual machines which reside on hosts. Tasks are modeled as unit requests that contain resource specification in the form of computational resource requirements (MIPs, memory and storage) in addition to data exchange requirements (task size variable representing the process files to be sent to the host the task scheduled on before execution, data sent to other servers during execution and output data sent after execution).

A comparison of the most prominent methods and techniques this problem has been approached with is introduced in Tables 1 and 2. We depict whether each work offers and optimization model, resource types considered, VM modeling, application layer modeling, network model, computational resource modeling, whether live migration is used, whether users are offered guaranteed reservation, whether the method is centralized or decentralized, and the scheduling algorithm in brief.

3. System implementation scenario

A typical cloud data center contains hosts that are available for clients with multiple lease terms on offer. Every client profile is tailored based on their budget, applications and general portfolio. For every rented VM, a client specifies the time scale (or at least the start time), the resource requirements in terms of computational and network resources, and operating systems in some of the cases. While the VM is alive in the data center, a continuous stream of requests arrive at the data center. Requests (tasks or Cloudlets) Tasks are modeled as unit requests that contain resource specification in the form of computational resource requirements (MIPs, memory and storage) in addition (sometimes) to data exchange requirements. The power consumption in the data center amounts to the total power consumption of all the resources residing in it. As the computational resources – specifically the hosts – consume most of the data center power, we will focus our optimization efforts on optimizing the power consumed by hosts not the network resources like switches, routers, etc. Power calculation models used in the literature are mostly utilization based [16,19,22]. We will introduce the formula in detail in the following section. Therefore, workload distribution on available hosts and VMs has a prime impact on the energy efficiency. This includes, VM initial placement, migration decision management in terms of migration frequency and the choice of the migrated VM and also the scheduling of Cloudlets (requests) on the corresponding VMs. Regardless weather the Cloudlets belong to one client or multiple clients, efficient scheduling that focuses on energy efficiency is a demand. In the following section, we offer a substantiation of this problem in the form of mathematical formulation. This is a preceding step to discussing a more scalable solution in Section 5.

4. System model

To solve the problem of scheduling tasks in a cloud computing environment while minimizing energy consumption, we introduce an analytical model where we formulate the problem as a mixed integer linear problem. The optimization problem is modeled focusing on two objectives, namely, minimizing the required power to serve a specific load of tasks and minimizing the load that needs to be migrated/recovered in case of a data center component failure. This model's purpose is to demonstrate the major constraints imposed on the problem and the potential search space size.

4.1. Notations

Environment parameters are described below. A set of resource providers (servers or VMs) are represented by S. CLT is a set of tasks (Cloudlets) sent by cloud clients. These tasks demand specific amounts of resources to run as per the scenarios discussed in earlier sections. CAP_{sm} represents the amount of resources (e.g. memory) available on a server (resource provider) where $s \in S$ and $m \in \{memory(me), CPUunit(c), storage(st)\}$ such that $CAP_{sMem} = 30$ indicates that available memory on server s or memory capacity is 30 GB. DEM is used to represent the demand matrix or the amount of resources needed for every requested task (Cloudlet). Memory and storage requirements are measured by GB while CPU requirements are measured by the task size shown in Million instructions (MI) or million instruction per second (MIPs) and duration. They could alternatively be measured by the fraction of processor power required. Moreover, the same model could be applied when using other common metrics for processing demand like the amount of employee data processed per hour (employee/hour) or Java server side operations per second (JOPs). $DEM_{CletCPU} = 700$ indicates that the task (Cloudlet) $Clet \in CLT$ demands computational power to run 700 Million instructions, $v \in V$ requires 7 GB of memory assuming that m denotes memory resource on a server.

The matrix D contains the deadline of every Cloudlet (denoting request lifetime). $D_Clet = t$ means that Cloudlet $Clet \in CLT$ has to be served before time unit t. All the specification of a Cloudlet could in the same way be applied to VMs. This depends whether the problem is just scheduling Cloudlets on VMs (servers) or it includes scheduling both VMs on servers and Cloudlets on VMs. The parameters $Pidle_s$ and $Pmax_s$ indicate the amount of power consumed by server s at the idle state and at maximum utilization respectively.

4.2. Optimization problem formulation

The formulation is based on two decision variables. Y_{sClet} is a binary decision variable such that $Y_{sClet} = 1$ if Cloudlet $Clet \in CLT$ is scheduled on server (resource provider) s and 0 otherwise. X_{tClet} is a binary decision variable such that $X_{tClet} = 1$ if Cloudlet $Clet \in CLT$ is served at time unit t and 0 otherwise.

The problem is formulated as a mixed integer linear programming (MILP) problem with two possible objectives. The first one is to minimize power consumption needed to perform a specific load (minimize the kilo watt hours required to run a specific set of tasks (requests)). The second objective is more high availability-oriented. The goal is to minimize the potential migration load in case of a failure to any server (or by extension to any component in the data center hierarchy). Scheduling the tasks in a way that minimizes the migration load affects the performance positively as it decreases the performance hiccup induced by failures and aids in reaching a more seamless failure event handling. This decreases the amount of resources dedicated to maintain high availability as the work load is less.

4.2.1. Power consumption

Formulating power consumption in severs is a standing challenge in the literature. Some of the commonly used efforts can be seen in [16,22] where linear models are proposed. These models are based on the assumption that servers consume minimal power when idle and that level of consumption increases linearly while the computational capacity increases. Thus, power consumption at a specific processor utilization percentage u is calculated as:

$$P_u = P_{idle} + (P_{max} - P_{idle}) \times u \tag{1}$$

Hence, power consumption can be minimized by directly minimizing the average utilization.

$$MIN \quad Z1 \tag{2}$$

$$Z1 = \sum_{t \in T} \sum_{s \in S} (Pidle_s + (Pmax_s - Pidle_s) \times (\sum_{Clet \in CLT} X_{tClet} Y_{sClet} DEM_{CletCpu}) \div CAP_{sCPU}) \tag{3}$$

4.2.2. Migration load

It is desirable to minimize the load to be migrated in case of component failures. That translates to the data load on the server at the moment of failure. To minimize that we use a min max approach where we try to minimize the maximum computational load on any of the servers in the data centers. This is calculated based on the computational capacity of Cloudlets (tasks) scheduled on servers. However, it can be easily adjusted to accommodate other components (VMs, racks, etc.) or to take into consideration other resources when calculating the load (like memory, storage, bandwidth).

$$MINMAX \quad Z2 \tag{4}$$

$$Z2 = \sum_{Clet \in CLT} X_{tClet} Y_{sClet} DEM_{CletCpu} \quad \forall s \in S \; \forall t \in T \tag{5}$$

This model assumes an equal Mean Time To Fail (MTTF) for all servers in the data center. A weighted function based on the MTTF would be added in case of using varying values for servers.

The objective function is subjected to the following constraints:

$$\textstyle\sum_{t\in T} X_{tClet} >= DEM_{CletCpu}/Cap_{CletCpu}, \quad \forall Clet \in CLT \tag{6}$$

$$\sum_{s\in S} Y_{sClet} = 1, \quad \forall Clet \in CLT \tag{7}$$

$$\textstyle\sum_{Clet\in CLT} X_{tClet} Y_{sClet} DEM_{Cletm} <= CAP_{sm}, \quad \forall t \in T, s \in S, m \in \{me, c, st\} \tag{8}$$

$$X_{tClet}.t <= D_{Clet} \quad \forall Clet \in CLT, \forall t \in T \tag{9}$$

$$X_{tClet}, Y_{sClet}, \in \{0, 1\} \tag{10}$$

In Constraint (7), we ensure that a Cloudlet (request) will be assigned exactly to one server (service provider). In Constraint (6), we ensure that a Cloudlet is scheduled for enough time units to satisfy its computational demand given the server computational capacity. In Constraint (8), we guarantee that Cloudlets will be allocated on servers with enough capacity of the computational resources required by the Cloudlets. In Constraint (9), we ensure that a Cloudlet is served before its deadline. Constraint (10) guarantees the binary constraints of the problems.

5. Consolidation-based energy efficiency

Moving to more practical solutions, we start by proposing a consolidation-based solution. Then, we move to discussing the impacts of consolidation-based solutions and live migration leading to the proposal of a novel non-consolidation-based solution.

5.1. VM dynamic idleness prediction (DIP)

The challenge of choosing which VM to move or migrate is central to any consolidation technique and therefore crucial to energy efficiency policies. A core activity of consolidation-based energy efficiency is migrating VMs in order to empty a machine so it can be switched off or moved to an idle state. This decision affects the performance highly, first by specifying the amount of data to be moved and the nature/amount of resources required at the destination host. In addition, regardless of how much algorithms are able to minimize the live migration time, there will always be a certain amount of delay or performance degradation. This means that depending on the load expected of the VM and SLA agreement, an SLA violation is highly possible. This way, the priority should be given to VMs with more strict SLAs, and critically, with less activity when the migrated VMs are chosen. We propose to tackle this challenge by introducing a scoring system for the VMs to decide which ones to shut down and move based on the use of an expert system. The hypothesis here is that with the successful prediction of which VMs will be idle (or least active) and for how long, we will have a clear advantage in terms of migration decision management and the consolidation process in general. This step, which we termed dynamic idleness prediction (DIP) will make an instant impact in terms of the total power consumed by the data center with all the other factors unchanged.

5.2. Classification parameters

The classification parameters are the parameters used by the system to predict the state of the VM for a preset future period. They include variables that would affect the system's expectation of the VM future behavior. Some of these variables would affect the VM activity directly (for example: redundancy models specifies the frequency of backup/redundancy activities). Some affect the VM indirectly and contribute to behavior patterns than are not specified explicitly (for example: User location or the type of application served by the VM). The more parameter values that can be collected for the VM, the more reflective the profile built by the classifier will be. In turn, the results will be more reflective of the VM activity level. It is critical in this type of experiments to gain an insight into the demanded resources by the request (CPU and memory, for example) as well the time bounds if any. However a more coherent profile for the VM can be constructed by collecting parameters like: User ID, User location, User VMs, Type of contract (rental term), VM reserved resources, VM start time/reserved time, Redundancy model, Redundancy activity frequency, Component type, Request types and frequency, Communication/data exchange request Dependencies, and response time required.

5.3. How DIP works

First, the classifier is fed a list of records containing the parameter readings or values for the VMs and their resulting states for a certain time period. This would be the training data set Then the classifier uses this training data set to build behavioral models for the VMs in question. Alternatively, cross validation method can be used. These behavioral models would depend on the classification method used (for example: decision trees, Naive Bayes or Support vector machines). From now on, the Classifier would be able to predict (classify) the number of requests sent to the VMs in a fixed future period. Next, this information can be used by the scheduling component (centralized to for the whole data center or decentralized) to rank the VMs and either:

A – Choose the VM with predicted least received requests in time period t2–t1.

B – Set a cutoff threshold (CO) such that VMs with incoming future request in a pre-specified period less than CO are considered idle.

Once the idle VMs list is generated, the list is used in the consolidation step as explained in the flowchart in Fig. 3.

5.4. The proposed consolidation-based algorithm

The proposed method is illustrated in the flowchart in Fig. 2. First, the algorithm starts by initializing the major parameters including parameters in Table 3. VMs are placed based on one of the following 3 methods: first fit scheduling, round robin or the resource based scheduling technique with the variation proposed in [14]. Then, as the requests for resources keep arriving, these request are scheduled based on the availability of host and VM resources. Periodically, and based on a preset consolidation trigger parameter, the VM shuffle process starts. This trigger defines the period or the frequency of revisiting the VM placement and running the shuffle process and the consolidation function. The shuffle process starts by constructing a list of VMs to be switched off. These VMs are chosen based on VM choice method parameter (or in other words, idle VMlist construction technique). We chose 4 options to test in our experiment. These options are choosing the VMs randomly, choosing the VMs based on a greedy method that chooses VMs hosted on the least used server(s), exploiting the proposed technique DIP as explained in the previous subsection and finally, we set the fourth option to be keeping the VMs running without any switching off. This would aid us in calculating how much of an improvement these techniques are offering as opposed to not using any technique. Next, after constructing the Idle VMlist, these VMs are switched off and the data center loads are consolidated into fewer servers. This shuffle process is repeated on a periodical basis depending on the trigger parameter mentioned earlier. Another step that is done periodically is swapping the VMs that are switched off in order not to starve a certain VM and to ensure fairness. In the following section, we propose

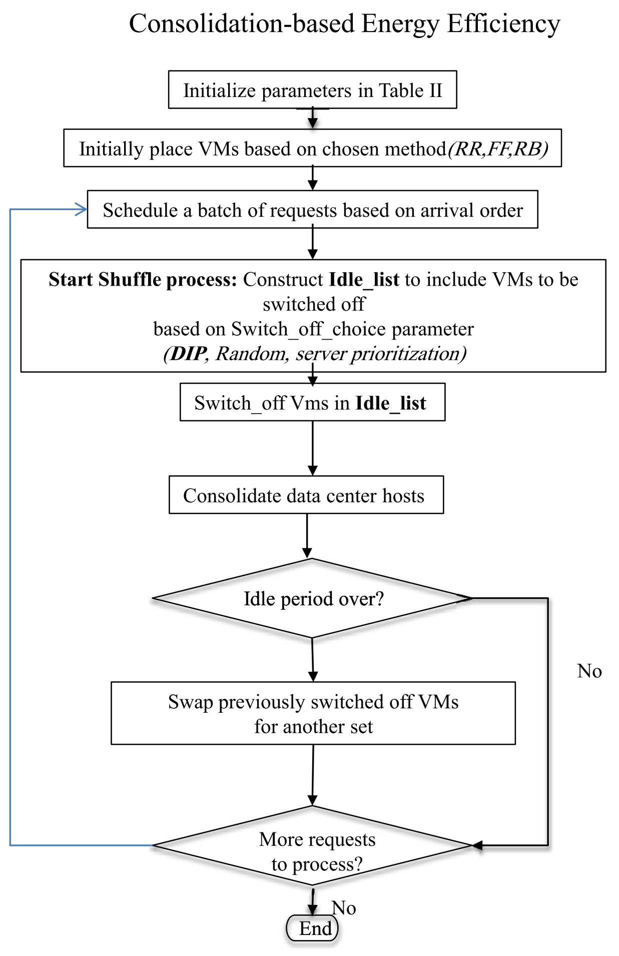
Consolidation-based Energy Efficiency
Initialize parameters in Table II
Initially place VMs based on chosen method(RR,FF,RB)
Schedule a batch of requests based on arrival order
Start Shuffle process: Construct Idle_list to include VMs to be switched off based on Switch_off_choice parameter (DIP, Random, server prioritization)
Switch_off Vms in Idle_list
Consolidate data center hosts
Idle period over?
No
Swap previously switched off VMs for another set
More requests to process?
No
End

Fig. 2. Consolidation-based energy efficiency flowchart.

Table 3
Energy efficiency problem – major parameters.

Parameter	Description
Placement method	How are VMs initially scheduled on hosts round robin First fit or Resource based scheduling proposed [14]
Switched off (consolidated) VM choice method (Idle VMlist construction technique)	No switch off, DIP, random, greedy-servers
Cutoff limit (CO)	The amount of requests in the future below which the VM is considered idle
Consolidation trigger	The frequency of revisiting the VM placement and running consolidation function
Migration allowed	If migration is used for this experiment or fixed VM placement is imposed

a solution based on a technique using DIP that is not dependent of live migration. Then, we present the experimentation results for both migration-based and non-migration based techniques. Fig. 3 shows the pseudo code for the Idle VMlist construction function. The algorithm processes a set of virtual machines VM, a specific future period (FP) and a cutoff limit of requests (CO). The algorithm then decides which VMs will be allowed to run and which will be idle.

```
1: Algorithm: Construct Idle List
2: Input: Virtual machine set VM,
3:         FP, CO
4: Output: VM state values assigned correctly
5: for $VM_i \in VM$ do
6:     $VM_i.state = running$
7: end for
8: for $VM_i \in VM$ do
9:     if ( $getClassifierPredictedValue(v, FP) <= CO$ ) then
10:        $VM_i.state = idle$
11:    end if
12: end for
```

Fig. 3. Idle VMlist construction function using Dynamic Idleness Prediction technique (DIP).

6. Non-Consolidation-based energy efficiency

6.1. Live migration: why not?

Taking migration as automatic solution is far from agreed upon. Migration typically imposes performance degradation to the extent of having an off time which is not welcome by the clients. This time could span through an unexpected range based on the efficiency of the process and the network bottlenecks in the data center at the time and the amount data included. VMware, for example, offers live migration as major feature introduced in vMotion where a VM can be moved from a host to another without having to shut it down. Larger providers like Amazon, for example, do not depend on this. Reasons, cited in [27], include the fact that AWS (and Rackspace as well) keep the VM data in the local disk. This makes it harder to send all the data across the network. “Evacuating a given host, particularly one at capacity can take hours” [27]. This casts doubts over the practicality of using live migration in principle. More so, it casts doubts on using migration with the freedom and frequency suggested in some of the energy efficiency solutions, where VMs are to be consolidated periodically in fewer servers. Finding a solution that does not depend on live migration or at least minimizing the number of migrations performed is a pressing requirement.

6.2. Smart VM overprovision (SVOP)

We introduce a novel technique for energy efficient VM management that does not depend on migration. The technique is called Smart VM Overprovision (SVOP). This technique depends on the DIP technique described earlier to dictate the overprovision of VMs by choosing the mostly idle VMS to be switched off and in turn minimize lost requests. SVOP is illustrated in the flowchart Fig. 4. This method works in two phases. First, the initial VM profile building phase. Then, the regular VM operation phase. The first phase starts by the initializing the parameters then scheduling the VMs on the corresponding host based on RB scheduling explained the previous section. Next, a test batch of requests is scheduled to build each VM’s profile. These requests can be real requests demanded by the VM or client. They can alternatively be a training set of request constructed based on the client and VM profile in order to be used for the following steps. After that, the classifier builds a profile for the VM and a predicted value of the future demand is calculated. Then the overbook-shuffle process is started. An idle VMlist is constructed using the DIP technique discussed previously. The list of idle VMs are separated from the active VMs. They are scheduled on a separate set of hosts where the concept of overbooking is applied. An overbooking factor is used to define the overbooked load on each of these overbooked hosts. For example, if the host’s capacity is 4 VMs and the overbooking factor is 150%, then the Overbook-idle-list function will book up to 6 VMs on this host (assuming all VMs have requested equal amounts of resources in that case). From here, these VMs can alternatively share the

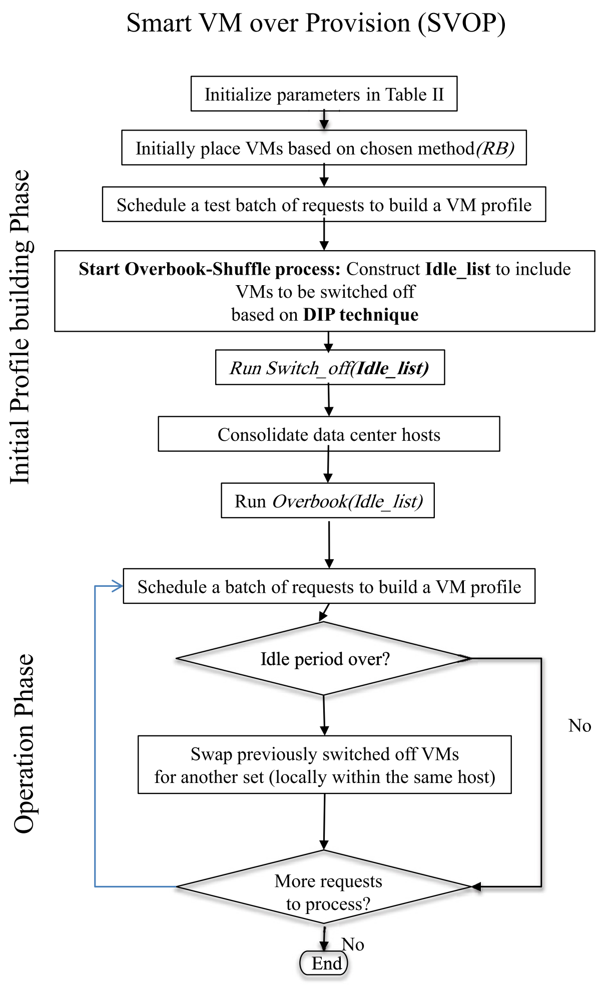
Smart VM over Provision (SVOP)
Initial Profile building Phase
Initialize parameters in Table II
Initially place VMs based on chosen method(RB)
Schedule a test batch of requests to build a VM profile
Start Overbook-Shuffle process: Construct Idle_list to include VMs to be switched off based on DIP technique
Run Switch_off(Idle_list)
Consolidate data center hosts
Run Overbook(Idle_list)
Operation Phase
Schedule a batch of requests to build a VM profile
Idle period over?
No
Swap previously switched off VMs for another set (locally within the same host)
More requests to process?
No
End

Fig. 4. Consolidation-based energy efficiency flowchart.

```
1: Algorithm: $Overbook\ Idle\ List$
2: Input: $Virtual\ machine\ set\ VM,$
3:        $where\ VM_i.state\ is\ either\ 'idle'\ or\ 'running'$
4:        $with\ states\ assigned\ based\ on$
5:        $Construct_i dle_l ist function()$
6:        $(OBF)\ as\ overbooking\ factor$
7: Output: $VMs\ with\ idle\ state\ are$
8:        $scheduled\ using\ SVOP\ method$
9:        $hostS = leastUsedServer()$
10: for $VM_i \in VM$ do
11:   if $(VM_i.state = idle)$ then
12:       $ScheduleVM(i, S, OBF)$
13:     if $(isHostfull(S, OBF))$ then
14:         $S = leastUsedServer()$
15:     end if
16:   end if
17: end for
```

Fig. 5. Idle VMlist overbooking function.

overbooked resources (which is the concept we used in our implementation). Another technique that can alternatively be used here is dividing resources between the VMs in a way that each one would have reduced capacity. In the second phase of the SVOP technique, the operation phase, the incoming requests are served based on the setup in the first phase. Another step that is done periodically is swapping the VMs that are switched off in order not to starve a certain VM and to ensure fairness. Fig. 5 contains the pseudo code for Idle VMlist overbooking function included in the SVOP method.

7. Performance evaluation

7.1. Data set

To perform the experiment, we used a data set taken from Google's cluster workload traces. These are traces of workloads running on Google compute cells. The dataset provides traces from a Borg cell that were taken over a 7 hour period. The workload consists of a set of tasks, where each task runs on a single machine. Tasks consume memory and one or more cores (in fractional units). Each task belongs to a single parent; a parent may have multiple tasks (e.g., mappers and reducers). In our work, the parent is represented by the VM the task belongs to. "The data have been anonymized in several ways: there are no task or job names" [15], just numeric identifiers; timestamps are relative to the start of data collection; the consumption of CPU and memory is obscured using a linear transformation. The data are structured as blank-separated columns. Each row reports on the execution of a single task during a five minute period.

- Time (int) – time in seconds since the start of data collection
- parentID (int) – Unique identifier of the job to which this task belongs (may be called ParentID)
- TaskID (int) – Unique identifier of the executing task
- Type (0, 1, 2, 3) – class of job (a categorization of work)
- Normalized Task Cores (float) – normalized value of the average number of cores used by the task
- Normalized Task Memory (float) – normalized value of the average memory consumed by the task Using classifiers

7.2. Classifier and classification tool

Machine learning (ML) classifiers automatically analyze a large data set composed of several attributes and decide what information is most relevant. This builds the classifier's ability to predict the values of a specific preselected attribute. This value (which could be qualitative or quantitative) is the classification. Classifiers are used in many application fields. A commonly used tool that has a variety of the most common classifiers readily implemented is Weka [28].

Weka is a software workbench that includes several ready to use ML techniques [28]. Once the data is formatted in the format readable by Weka (.arff format) which defines what is the relation name, the attributes and their possible values and the data rows themselves, the tool can pre-process and classify. The relation defined for this work to predict the number of future requests is VM-predictor. We have tested multiple classifiers to find the classifier most suitable to our DIP technique and the energy efficiency problem. Table 4 contains the classifier names as in Weka and the classification precision measured using root absolute error. It is seen from the table that classifiers differ in their achieved precision. The highest performing classifiers for this specific case is REPtree with a root absolute error of 7.8% and then meta bagging and KStar classifiers. Fig. 6 shows a sample of the visual results gained for individual prediction using the REPtree classifier. Most of the values lie in or around the line which has a slope of 1. This indicates the equality of the predicted and the actual values of the number of future requests.

7.3. Simulation parameters

A discrete event simulator was built in C++ to evaluate and compare the aforementioned techniques. As for the VM resource specification, we based it on some of the offered VMs by Amazon AWS [29]. The simulated time reached 6500 time units. The power calculation model used is a linear power consumption model as

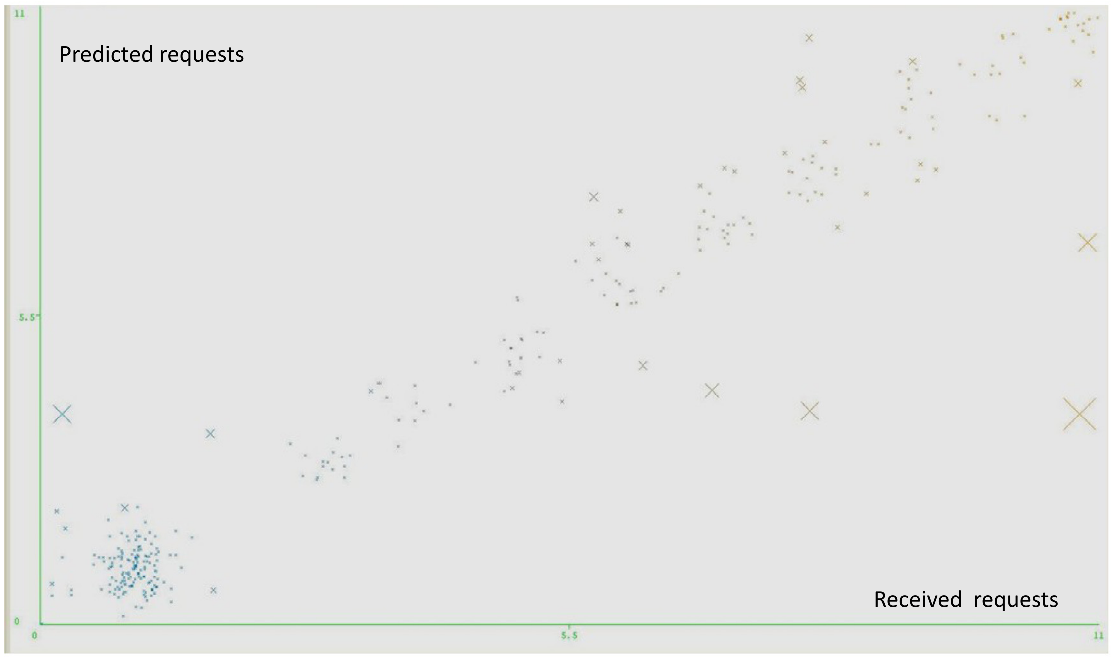

Fig. 6. VM prediction results using REPtree classifier.

Table 4
Classifier prediction precision comparison.

Classifier (as named in Weka)	Relative absolute error
Decision Table	18.5557%
MSRules	18.9886%
Conjunctive Rule	55.0279%
Gaussian Processes	52.0167%
Multi-Layer perception	34.8747%
IBK	17.3399%
KStar	11.9271%
LWL	44.7885%
meta bagging	10.0447%
Random sub-space	16.1515%
Regression by discretization	13.0296%
MSP tree	17.2206%
REPtree	7.7719%

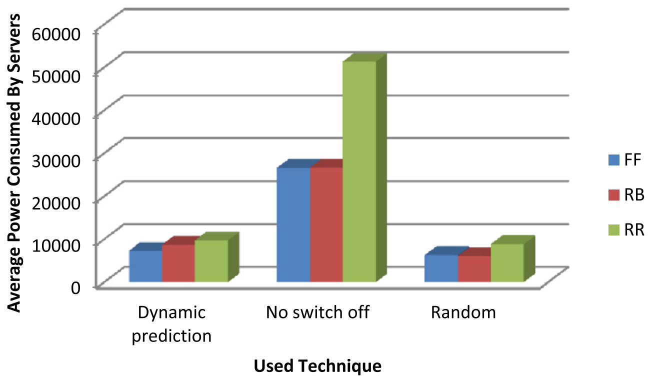

Fig. 7. Comparing request acceptance rate for different placement methods.

in [22]. This could easily be swapped with any other model as per the cloud provider's preference. Each request was considered a fixed-duration request for the corresponding memory and CPU values that are taken from the Google data trace. This helps in eliminating any distortion caused by the request duration distribution and increasing the dominance of the evaluated parameters over the results. More details on the specifications of each of the evaluated techniques are presented in the following section.

7.4. Comparing placement methods

We start analyzing the results by a comparison of the placement methods used in the VM initial placement step. A look at Fig. 7 shows the major advantage each of FF and RB methods has over RR when either of these methods are combined with any of the idle list construction (switch off factor) methods. This is due to the fact round robin mainly focuses on distributing the load on as many hosts as possible. This means starting many unnecessary hosts and while this method has advantages in terms of high availability and minimizing network bottlenecks, it is not really suitable for energy efficiency purposes. Moreover, the other two techniques perform comparatively mainly because of their tendency to fill the hosts before looking at using new ones. This happens in a greedy way (FF) or in resource oriented way (RB).

7.5. Evaluated methods (energy efficiency solutions)

Next, we evaluate the aforementioned solutions in terms of a number of critical metrics, namely, energy used per server, energy used per served request, request acceptance rate, and number of migrations performed. It can be seen that the multiple factors considered in this problem and the possible methods employed can yield a high number of solution permutations. Due to space constraints, we will show the results for the 9 methods with high performing or significant results for any of these metrics. Table 5 explains each method in terms of nature and the techniques used in it. The table specifies if the method is consolidation-based or not (depends on migration or not), which placement method is used for the initial placement, how the idle VM list (mentioned in flowcharts 1 and 2) is constructed, if the VM switch off act is permanent (until the end of the experiment) or temporary and interchangeable between VMs as explained in the previous section and finally, it discusses the frequency the VM consolidation technique is called whenever it is used. The last factor was added to show the effect of increasing the frequency of calling the VM consolidation method on the evaluated metrics. From this point on, we will refer to each of the evaluated methods with the abbreviated name used in Table 5.

Table 5
Evaluated methods (energy efficiency solutions).

Method	Consolidation-Based?	Placement method	IdleList construction technique	Switch off duration	VM consolidation frequency
FF-DIP-PermSW	Yes	FF	DIP	Perm	Normal
RB-DIP-TmpSW	Yes	RB	DIP	Temp	Normal
RB-DIP-TmpSW-Hfreq	Yes	RB	DIP	Temp	High frequency
RB-Greedy-PermSW	Yes	RB	Server greedy	Perm	Normal
RB-Greedy-TmpSW	Yes	RB	Server greedy	Temp	Normal
RB-Rnd	Yes	RB	Random	Temp	Normal
RR-NoSW	No	RR	No switch off	-	Normal
FF-NoSW	No	FF	No switch off	-	Normal
SVOP	No	RB	DIP	Temp	Normal

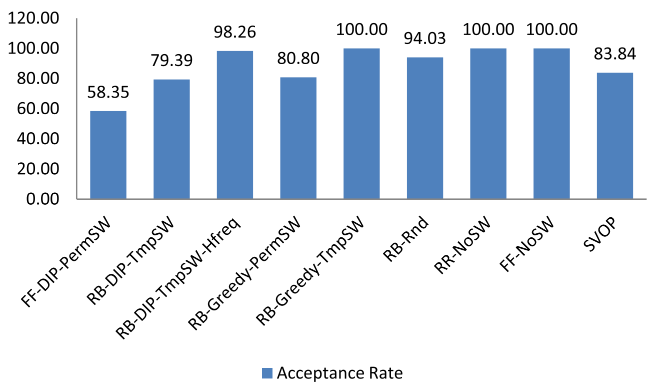

Fig. 8. Request acceptance rate for different energy efficiency methods.

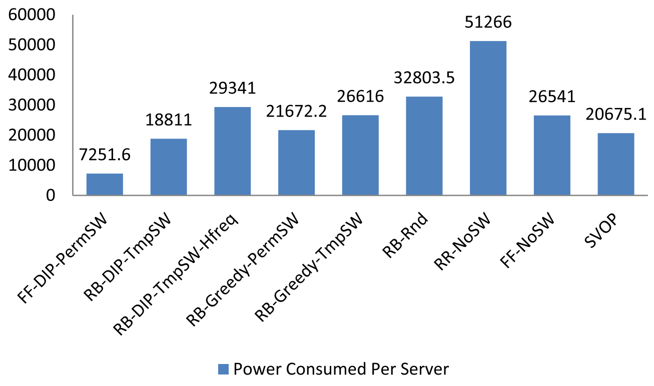

Fig. 10. Power consumed per for different energy efficiency methods.

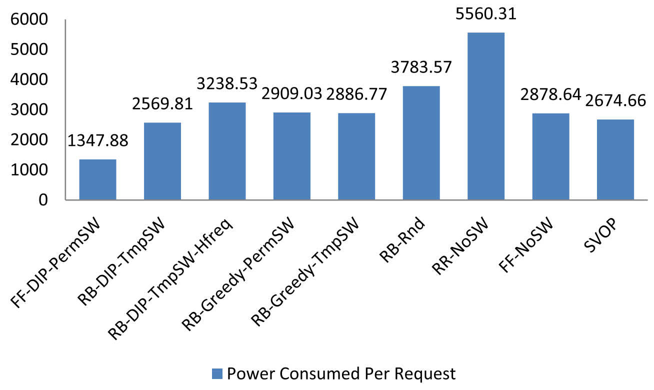

Fig. 9. Power consumed per request for different energy efficiency methods.

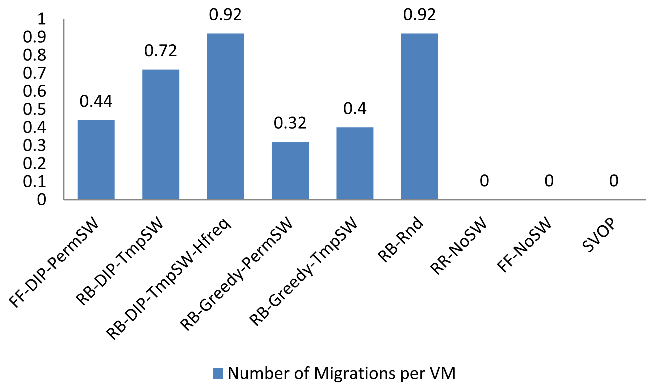

Fig. 11. Number of migrations per VM for different energy efficiency methods.

7.6. Consolidation frequency

Consolidation frequency defines how often the consolidation function is called to look for space on the hosts to be saved. The effect of increasing the consolidation frequency can be seen by comparing the metric readings for RB-DIP-TmpSW and RB-DIP-TmpSW-Hfreq in Figs. 8 to 12. It can be inferred that increasing the consolidation increases the acceptance rate significantly. However, this increase is paid in the form of system load. RB-DIP-TmpSW-Hfreq scores highest in terms of the number of migration per VM (in Fig. 11) and in terms of the total number of servers used for a fixed load (in Fig. 12). A balanced level of frequency needs to be reached for each specific case to reach a trade off between the load caused by the high number of migrations might cause the gain in accepted requests caused by updating the system to reflect the momentarily loads states of the VMs.

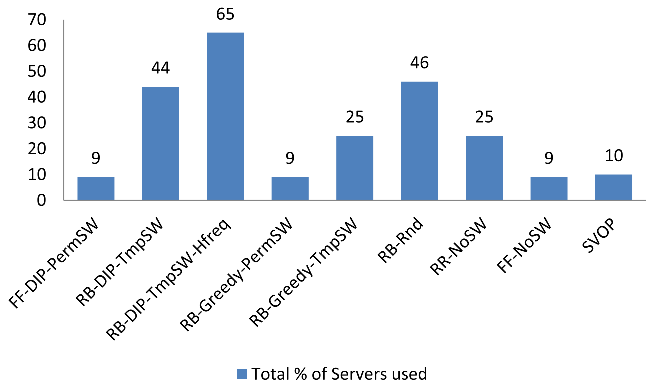

Fig. 12. Percentage of used servers for different energy efficiency methods.

7.7. Permanent vs. temporary switch off

As seen in Figs. 9 and 10, using permanent switch off for inactive VMs will yield significant power savings. This applies regardless of the Idle list construction technique. In Fig. 10 for example, FF-DIP-PermSW consumes 7251.6 compared to the 18811 consumed by RB-DIP-TmpSW while RB-Greedy-PermSW consumes 21672 compared to the 26616 consumed by RB-Greedy-TmpSW. However, the acceptance rate losses caused by the permanent switch off of idle VMs are very high. As in Fig. 8, both FF-DIP-PermSW and RB-Greedy-PremSW gain largely discounted acceptance rates (58.35% and 80.80%) compared to their counter part methods (79.39 % and 100%). This confirms the notion that using permanent switch off even for the most idle VMs is not effective in terms of scheduling fairness and general acceptance rate. Therefore, using permanent switch of should be saved only for cases where the data center cannot serve the load for all the VMs requested at a certain moment.

7.8. DIP technique's impact on the consolidation based techniques

Looking at DIP's impact when it is introduced as the technique of choice to construct the Idle List during any consolidation based technique, it is found that this impact is significant. In Fig. 10, we notice that FF-DIP-PermSW and RB-DIP-TmpSW have a clear advantage in terms of power consumed per server specially compared to the other consolidation-based techniques. This is supported by an advantage in terms of power consumed per request where these two methods ranked 1 and 2 again. RB-DIP-TmpSW specifically performs favorably in terms of energy efficiency and acceptance rate (81% of request). However, when looking at the number of migrations per VM, we notice that this technique requires a relatively high number. In the cases where migration is not a preferred option, there is a critical need for another method which performs comparatively to RB-DIP-TmpSW and that does not depend on migrations.

7.9. Smart VM over provision (SVOP) as a method that is not dependent on migration

Two methods which serve as a benchmark for our solution are the No switch off methods (RR-NoSW and FF-NoSW). In these two methods, the initial placement of the VMs is the only step performed. All VMs are given high priority for the resource allocation. No Vm is switched off or migrated. Naturally, this means that most or even all requests are accepted. However, the energy efficiency is far from optimal. Also, the initial placement method is the dominant factor that affects the method performance. A look at Figs. 9 and 10 shows that RR-NoSW method has the highest value for power consumed per request and power consumed per server metrics and by a distance. Fig. 12 (as discussed earlier) shows that the same method used a higher percentage of the data center servers even than some of the methods that use migration. This leaves as with FF-NoSW. When comparing our proposed method SVOP with the best performing non-consolidation-based method (which is FF-NoSW), encouraging results are seen. Although SVOP does not quite reach 100% acceptance rate, SVOP consumes lower power per server than FF-NoSW and comes third for that metric only after FF-DIP-Perm and RB-DIP-TmpSW. Both of those methods are consolidation-based and both achieved lower acceptance rates than SVOP. As for the power consumed per request metric, SVOP comes in third and consumes lower power than all methods with 80% acceptance rate or higher. SVOP consumed power per request is close to the value achieved by the best consolidation-based technique RB-DIP-TmpSW. SVOP also offers the advantage of not needing any migrations in the operation phase and using considerably lower number of servers on average than RB-DIP-TmpSW.

Therefore, from combining the previous results, the consolidation-based method RB-DIP-TmpSW (which depends on the proposed DIP technique) is the best performing method in terms of energy efficiency with a viable acceptance rate. However, the proposed non-consolidation-based method SVOP comes very close in terms of energy efficiency while offering the added advantage of less/no migration load.

8. Conclusion

In a heterogeneous cloud network scenario where a cloud computing data center serves mobile cloud computing requests from diverse IoT devices, major challenges with performance are faced.

A prominent objective in this scenario for cloud providers is how to serve these requests with the required performance while minimizing the energy the cloud data center uses. To satisfy clients' demands, cloud providers are in constant pursuit of a system that satisfies client demands for resources, maximizes availability and other service level agreement metrics while minimizing energy consumption and, in turn, minimizing cloud providers' cost.

We introduced a novel mathematical optimization model to solve the problem of energy efficiency in a cloud data center. Next, we offered a solution based on VM migration that tackles this problem and minimizes energy efficiency in comparison to other common solutions. This solution includes a novel proposed technique to be integrated in any consolidation-based energy efficiency solution. This technique depends on dynamic idleness prediction (DIP) using machine learning classifiers. Potential classifiers were evaluated and a recommendation with regards to the most suitable classifiers was made. Moreover, a robust energy efficiency scheduling solution that does not depend on VM consolidation or live migration was introduced. This solution, termed Smart VM Over Provision (SVOP), offers a major advantage to cloud providers in the cases where live migration of VMs is not preferred. 9 candidate solutions with multiple energy efficiency techniques were evaluated for a number of critical metrics, namely, energy used per server, energy used per served request, acceptance rate, and number of migrations performed.

The experimental results gained from testing these methods on data taken from the Google trace data set showed that the consolidation-based method RB-DIP-TmpSW (which depends on the proposed DIP technique)was the best performing method in terms of energy efficiency with a viable acceptance rate. However, the proposed non-consolidation-based method SVOP came very close in terms of energy efficiency while offering the added advantage of less/no migration load.

In future work, we will strive to build on SVOP's success and improve the energy efficiency gain while increasing the achieved acceptance rate.

References

[1] A. Nadjar, S. Abrishami, H. Deldari, Load dispersion-aware VM placement in favor of energy-performance tradeoff, J. Supercomput. (ISSN 0920-8542) (October 2016) 1–20, http://dx.doi.org/10.1007/s11227-016-1842-4.

[2] A. Qouneh, M. Liu, T. Li, Optimization of resource allocation and energy efficiency in heterogeneous cloud data centers, in: 44th International Conference on Parallel Processing, ICPP 2015, Beijing, 2015, pp. 1–10.

[3] M. Guzek, D. Kliazovich, P. Bouvry, HEROS: energy-efficient load balancing for heterogeneous data centers, in: IEEE 8th International Conference on Cloud Computing, CLOUD 2015, ISSN 2159-6190, 2015, pp. 742–749.

[4] M. Abu Sharkh, M. Jammal, A. Ouda, A. Shami, Resource allocation in a network-based cloud computing environment: design challenges, IEEE Commun. Mag. 51 (11) (November 2013) 46–52.

[5] H. Goudarzi, M. Ghasemazar, M. Pedram, SLA-based optimization of power and migration cost in cloud computing, in: 12th IEEE/ACM International Symposium on Cluster Cloud and Grid Computing, CCGrid 2012, 2012, pp. 172–179.

[6] L. Wang, G.v. Laszewski, J. Dayal, F. Wang, Towards energy aware scheduling for precedence constrained parallel tasks in a cluster with DVFS, in: 10th IEEE/ACM International Conference on Cluster Cloud and Grid Computing, CCGrid 2010, 2010, pp. 368–377.

[7] D. Theng, K.N. Hande, VM management for cross-cloud computing environment, in: International Conference on Communication Systems and Network Technologies, CSNT 2012, 2012, pp. 731–735.

[8] L. Minas, B. Ellison, The Problem of Power Consumption in Servers, March 2009, prepared in Intel Lab, Dr. Dobbs Journal. Online. Available: http://www.drdobbs.com/the-problem-of-power-consumption-in-serv/215800830 (accessed March 2017).

[9] National Resources Defense Council, America's data centers are wasting huge amounts of energy, online. Available: http://www.nrdc.org/energy/files/data-center-efficiency-assessment-IB.pdf (accessed October 2015).

[10] L. Columbus, Roundup of small & medium business cloud computing forecasts and market estimates. Online. Available: http://www.forbes.com/sites/louiscolumbus/2015/05/04/roundup-of-small-medium-business-cloud-computing-forecasts-and-market-estimates-2015/#5bc3cc621646.

[11] P. Lamson, A look at U.S. & European cloud adoption trends & forming lasting partnerships, online. Available: https://gtdc.org/evs/Carbonite_USvsEU.pdf (accessed October 2015).

[12] K. Weins, Cloud computing trends: 2015 state of the cloud survey, online. Available: http://www.rightscale.com/blog/cloud-industry-insights/cloud-computing-trends-2015-state-cloud-survey (accessed October 2015).

[13] C. Tulkoff, Using immersion cooling technology for enhanced efficiency & reliability for data center servers, online. Available: http://www.slideshare.net/CherylTulkoff/df-r-webinar-immersion-cooling-technology, Oct. 2015 (accessed October 2015).

[14] M. Abu Sharkh, A. Ouda, A. Shami, A resource scheduling model for cloud computing data centers, in: 9th International Wireless Communications and Mobile Computing Conference, IWCMC, 1–5 July 2013, 2013, pp. 213–218.

[15] Google, Google cluster data – trace version 1, Available: https://github.com/google/cluster-data/blob/master/TraceVersion1.md.

[16] S. Srikantaiah, A. Kansal, F. Zhao, Energy aware consolidation for cloud computing, in: Proc. of the 2008 Conference on Power Aware Computing and Systems, pp. 10–16.

[17] A. Beloglazov, R. Buyya, Energy efficient resource management in virtualized cloud data centers, in: 10th IEEE/ACM International, Conference on Cluster, Cloud and Grid Computing, 2010.

[18] H. Goudarzi, M. Pedram, Energy-efficient virtual machine replication and placement in a cloud computing system, in: Proc. IEEE Cloud, June 2012, pp. 750–757.

[19] 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: Proc. IEEE 30th Int'l Conf. Distributed Computing Systems, ICDCS, 2010.

[20] L. Wang, et al., GreenDCN: a general framework for achieving energy efficiency in data center networks, IEEE J. Sel. Areas Commun. 32 (1) (January 2014) 4–15.

[21] L. Duan, Z. Dongyuan, J. Hohnerlein, Optimizing cloud data center energy efficiency via dynamic prediction of CPU idle intervals, in: IEEE 8th International Conference on Cloud Computing, CLOUD, 2015, pp. 985–988.

[22] D. Kliazovich, P. Bouvry, S.U. Khan, A packet-level simulator of energy-aware cloud computing data centers, J. Supercomput. 62 (3) (2012) 1263–1283.

[23] D. Bruneo, A. Lhoas, F. Longo, A. Puliafito, Modeling and evaluation of energy policies in green clouds, IEEE Trans. Parallel Distrib. Syst. 26 (11) (2015) 3052–3065.

[24] R.N. Calheiros, R. Ranjan, A. Beloglazov, C.A.F. De Rose, R. Buyya, CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms, Softw. Pract. Exp. 41 (1) (2011) 23–50.

[25] S.K. Garg, R. Buyya, NetworkCloudSim: modeling parallel applications in cloud simulations, in: 4th IEEE International Conference on Utility and Cloud Computing, 2011, pp. 105–113.

[26] S.H. Lim, B. Sharma, G. Nam, E.K. Kim, C.R. Das, MDCSim: a multi-tier data center simulation, platform, in: IEEE International Conference on Cluster Computing and Workshops, 2009, pp. 1–9.

[27] B. Darrow, is-live-migration-coming-to-amazon-web-services-smart-money-says-yes/, available: https://gigaom.com/2014/10/12/is-live-migration-coming-to-amazon-web-services-smart-money-says-yes/.

[28] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, I.H. Witten, The WEKA data mining software: an update, SIGKDD Explor. 11 (1) (2009).

[29] Amazon, Amazon elastic compute cloud (Amazon EC2), online. Available: http://aws.amazon.com/ec2/.

[30] M. Abu Sharkh, A. Kanso, A. Shami, P. Öhlén, Building a cloud on earth: a study of cloud computing data center simulators, Elsevier Comput. Netw. J. 108 (October 2016) 78–96.

[31] M. Jammal, A. Kanso, A. Shami, High availability-aware optimization digest for applications deployment in cloud, in: Proc. ICC, IEEE, June 2015, pp. 6822–6828. Available: http://vixra.org/pdf/1410.0193v1.pdf.