Efficient Virtual Machine Sizing For Hosting Containers as a Service

Sareh Fotuhi Piraghaj, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya

Cloud Computing and **D**istributed **S**ystems (CLOUDS) Laboratory
Department of Computing and Information Systems
The University of Melbourne, Australia
Email: {sarehf@student, amir.vahid@, rnc@, rbuyya@}unimelb.edu.au

***Abstract*—There has been a growing effort in decreasing energy consumption of large-scale cloud data centers via maximization of host-level utilization and load balancing techniques. However, with the recent introduction of Container as a Service (CaaS) by cloud providers, maximizing the utilization at virtual machine (VM) level becomes essential. To this end, this paper focuses on finding efficient virtual machine sizes for hosting containers in such a way that the workload is executed with minimum wastage of resources on VM level. Suitable VM sizes for containers are calculated, and application tasks are grouped and clustered based on their usage patterns obtained from historical data. Furthermore, tasks are mapped to containers and containers are hosted on their associated VM types. We analyzed clouds' trace logs from Google cluster and consider the cloud workload variances, which is crucial for testing and validating our proposed solutions. Experimental results showed up to 7.55% improvement in the average energy consumption compared to baseline scenarios where the virtual machine sizes are fixed. In addition, comparing to the baseline scenarios, the total number of VMs instantiated for hosting the containers is also improved by 68% on average.**

I. Introduction

Cloud computing is a term for utility-oriented computing on a pay as you go basis. As stated by Armbrust et al. [1], cloud computing has the potential to transform a large part of the IT industry while making software even more attractive as a service. However, the major concern of cloud data centers is the drastic growth in energy consumption. Data center energy consumption results in increases in Total Cost of Ownership (TCO) while decreasing the Return of Investment (ROI) of the cloud infrastructure. Data center energy consumption has also a great impact on carbon dioxide (CO_2) emissions, which are estimated to be 2% of global emissions [2]. In this respect, there has been a growing effort in decreasing energy consumption, what resulted in considerable improvements in many aspects of cloud data center operations, including cooling system and server efficiency. Nevertheless, emerging new technologies make room for further improvements in server efficiency. A promising source for further improvements in energy efficiency of cloud servers are the solutions that are tailored for specific cloud service models.

In addition to traditional cloud services, namely Infrastructure as a Service (IaaS), Platform as a Service (PaaS), and Software as a Service (SaaS), recently a new type of service—called Containers as a Service (CaaS)—has been introduced. An example of container management system is docker which is a tool that allows developers to define containers for applications [1]. CaaS lies between IaaS and PaaS: while IaaS provides virtualized compute resources and PaaS provides application specific runtime services, CaaS is the missing layer that glues these two layers together. As illustrated in Figure 1, CaaS services are usually provided on top of IaaS' virtual machines. CaaS providers, such as Google and AWS, argue that at the same time as containers might offer appropriate environment for semi-trusted workloads, virtual machines provide another layer of security for untrusted workloads.

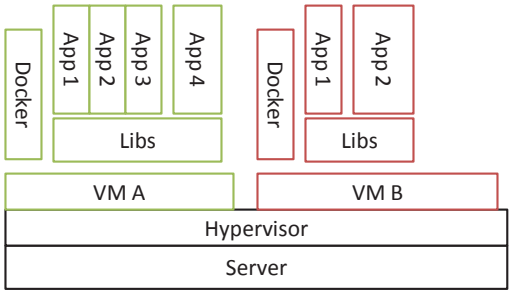

Fig. 1. A Simple CaaS Deployment Model on IaaS.

To reduce energy consumption of CaaS, one may choose virtual machine consolidation, Dynamic Voltage and Frequency Scaling (DVFS), or both of them combined. However, these efforts would be in vain if VM sizes are not customized to better support deployed containers. For example, as shown in Figure 1, size of *VM B* in comparison to*VM A* is not container-optimized. As a result, there is resource wastage that results in inefficiency in terms of energy consumption regardless of how effective and energy efficient is the VM consolidation technique in place.

We tackle the issue of energy efficiency in the context of CaaS, this paper focuses on finding efficient virtual machine sizes for hosting containers in a such way that the workload is executed with minimum wastage of energy. To achieve this, we present a technique derived from the analysis of real world

[1] Docker: https://www.docker.com/

clouds' trace logs that takes into consideration cloud workload variances, which is crucial in testing and validating the proposed solutions. Cloud computing traces released publicly are limited to those made available by Google. The first Google log provides the normalized resource usage of a set of tasks over a 7 hour period. The second version of the Google traces, which was released in 2012, contains more details in a longer time frame. The paper main contribution is an approach for efficient allocation of resources (in the form of virtual machine with defined CPU and memory entitlement) that matches closely the actual resource usage of deployed containers.

In this paper, the focus is on measuring impact of VM sizing strategies on data center power consumption for a specific period of time. To this end, we assume that we have a perfect knowledge of workload (to eliminate the prediction inaccuracy effect) and that no virtual machine consolidation approach is in place. This work is carried out in three steps. In the first step, we explored task usage patterns. We found that there are similarities in the utilization patterns reported in the traces, which is confirmed by previous studies [3], [4]. These similarities helped us to group the tasks based on average resource usage via clustering techniques. In the second step, the clustering output is used for identification of customized virtual machine types for the studied traces. Finally, in the third step, each cluster of tasks is mapped to a corresponding virtual machine type. We compare our VM sizing technique with fixed VM size baseline scenarios. The experimental results show that our proposed approach results in less number of servers, which in turn results in less energy consumption in the data center.

II. Related Work

There is a vast body of literature that considers power management in virtualized and non-virtualized data centers via hardware and software-based solutions [5]–[7]. Most of the works in this area focus on host level optimization techniques neglecting energy-efficient virtual machine size selection and utilization. These approaches are suitable for IaaS cloud services, where the provider does not have any knowledge about the applications running in every virtual machine. However, for SaaS and CaaS service models, information about the workload on the virtual machines and improvements on the size selection and utilization efficiency on VM level could be the first step towards more energy efficient data centers, as results presented in this paper demonstrate.

Regarding comparison between hosting SaaS on bare-metal servers or virtual machines, Daniel et al. [8] explored the differences between fixed virtual machine sizes and time shares. Although they concluded that the time share model requires less number of servers, they have not considered dynamic VM size selection in their experiments. Similarly, in the SEATS (smart energy-aware task scheduling) framework, Hosseinimotlagh et al. [9] introduced an optimal utilization level of a host to execute tasks which decrease the energy consumption of the data center. In addition, they also presented a virtual machine scheduling algorithm for maintaining the host optimal utilization level while meeting the given QoS. Apart from the level of optimization (host-level, data center-level, or virtualization level), most of the research in the area lack the analysis of real cloud backend traces and the variance in the cloud workload in the proposed solutions.

To address this issue, Mishra et al. [3] and Chen et al. [4] explored the first version of the Google cluster traces and introduced two approaches for workload modeling and characterization. Mishra et al. [3] used the clustering algorithm k-means for forming the groups of tasks with more similarities in resource consumptions and durations. Likewise, Chen et al. [4] used k-means as the clustering algorithm. In their experiments, the authors classified jobs[2] instead of tasks. In our proposed model, we use X-means for grouping the tasks with similarities in their usage patterns. The clustering output is used for determining the virtual machine configurations and estimating the average usage of a typical task in a specific cluster. We consider the second version of the trace released in late 2011.

Through configuring bare-metal servers and applying Google cluster trace, Zhang et al. [11] investigated the problem of dynamic capacity provisioning in a real production data center. The authors introduced a heterogeneity-aware resource provisioning environment named Harmony and dynamically adjusted number of active machines by considering heterogeneity in both workload and available machine hardware. k-means is used for forming workload classes with similar characteristics and resource requirements. Applying Google cluster trace, it is stated that Harmony can improve the energy consumption of the data center up to 28%. Although the energy consumption is improved, the virtualization technology as the building block of a cloud data center is not considered.

Different from previous works, we leverage virtualization and containerization technology together and map the groups of tasks to containers and containers to VMs. The configuration of the container-optimized VMs is chosen based on the workload characteristics, which results in less resource wastage. In our methodology, the problem of high energy consumption that results from low resource utilization is also addressed, which is not explored in most of the previous studies.

III. System Model

The targeted system is a CaaS service placed between a Platform as a Service (PaaS) and an IaaS data center operating as a private cloud. Users can submit their applications in one or more programming models supported by the provider to the platform. For example, the platform can support MapReduce or Bag of Tasks (BoT) applications. The users interact with the system by submitting applications supported by the platform. Each application consists of a set of jobs to be executed on the infrastructure. In our studied scenario, the job itself can be compromised of one or more tasks that are executed in containers managed by the CaaS provider.

User Request Model: In the proposed model, the infrastructure is abstracted from the users. Thus, the only action required from users of the service is the submission of their application along with estimated resources required for their execution. Parameters of a task submitted by a user are: Scheduling class;

[2]A job is compromised of one or more tasks [10].

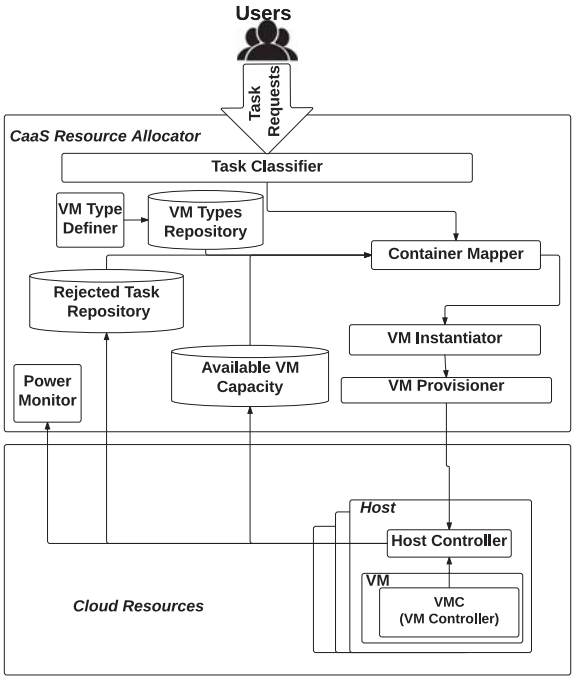

Fig. 2. Proposed system model. Cloud users submit requests for execution of applications in form of the container requests to the data center. Each task is executed in a container that meets application requirements. The resource allocator is responsible for mapping these containers to the relevant and available virtual machine configurations.

Task priority; and Required resources (i.e., number of cores and amounts of RAM and storage).

Notice that currently available commercial CaaS solutions require users to determine the required resources.

Cloud Model: The cloud model explored in this paper contains three layers:

- **Infrastructure layer:** This layer contains physical server that are partitioned to virtual machine through the next layer;
- **Virtualization layer:** The virtualization technology is taken into consideration as it improves the utilization of resources by sharing them between virtual machines;
- **Container layer:** Containers run on virtual machines and all share the same Linux kernel as the OS. The combination of these technologies (i.e., container and virtualization) has been recently introduced in Google Container Engine[3], which uses the open source technology Kubernetes[4].

System Objective: The objective of the proposed architecture is to find efficient virtual machine sizes for hosting the containers in a way that the workload is executed with minimum wastage of energy. The energy wastage is caused by underutilization of resources (i.e., the virtual machines). Therefore, one of the challenges is finding an optimal configuration of VM for a container, in such a way that its residing tasks have enough resources to be executed and the virtual machine's resources are not wasted during the operation.

The proposed model can be utilized by both Software as a Services and private clouds so the advantages of virtual machines and containers can be leveraged at the same time that energy costs are reduced through efficiently utilization of data centers. These systems are targeted because they have limited number of applications and enough information about them so that the typical cloud usage can be profiled.

Usually, profiles will be available for returning users of the infrastructure. In such a case, previous information about tasks needs enable the optimization of the task needs. For new customers, where there is no historical information enable profiling, the resources requested at request submission time can be used as an estimation of resources needs. As discussed previously, resource requirements are a common input parameter in commercial providers of similar services.

IV. Architecture

Figure 2 shows our proposed architecture that is responsible for selecting the right size of virtual machines that host containers running users' tasks. The components of the proposed architecture are further discussed in the next subsections. In the pre-execution phase, based on historical information on resource utilization of containers, our proposed system determines the virtual machine sizes. After that, in the execution phases, each container (based on its features) is mapped to a virtual machine with a specific type.

A. Components Involved in the Pre-execution Phase

In the pre-execution phase, some components of the architecture need to be tuned or defined before the system runtime.

- **Task Classifier:** The classifier is the entry point of the streaming of tasks submitted by cloud users. It is responsible for categorizing the submitted tasks to predefined classes based on the desired features. The features are selected according to the objective of the grouping process and the workload characteristics. The goal of the component is to identify tasks with similar usage patterns in terms of CPU, memory, and disk utilization. These patterns are obtained with application of clustering algorithms. The classifier must be trained with historical data before the system start up. The objective of clustering tasks is understanding, via profiling, the resource needs of tasks that are later used in the process of defining VM types and estimating the number of tasks that can be packed on each VM.
- **VM Type Definer:** This component is responsible for defining VM sizes based on the information provided by the Task Classifier. This is because determination of the optimal VM size requires analysis of the historical data about usage patterns of the task containers, and identification of groups of tasks with similar usage patterns reduces the complexity of estimating the average usage for a typical group. The output of this component is then saved in the VM Types Repository.

[3]Google Container Engine: https://cloud.google.com/container-engine/docs/
[4]Kubernetes: http://kubernetes.io/

- **VM Types Repository:** In this repository, the available virtual machine sizes including the CPU, memory, and disk specification are saved.

B. Components Involved in Execution Phase:

In this section we discuss the components which are involved during the execution phase of the system. By execution phase, we mean the activities carried out when the system is started up and accepting job submissions.

- **Container Mapper:** Clustering results from the Task Classifier are sent to the *Container Mapper*. Firstly, this component maps each task to a suitable container. Then, based on the available resources in the running virtual machines and the available VM types in the *VM Types Repository*, this component estimates the number and type of new virtual machines to be instantiated to support the newly arrived tasks. Apart from new VM instantiation when available VMs cannot support the arriving load, this component also reschedules rejected tasks that are stored in the rejected task repository to the available virtual machines of the type required by the VM (if any). This component prioritizes the assignment of newly arrived tasks to available resources before instantiating a new virtual machine.
- **Virtual Machine Instantiator:** This component is responsible for the instantiation of a group of VMs with the specifications received from the Container Mapper. This component decreases the start-up time of the virtual machines by instantiating a group of VMs at a time instead of one VM per time.
- **Virtual Machine Provisioner:** This component is responsible for determining the placement of each virtual machine on available hosts and turning on new hosts if required to support the new VMs.
- **Rejected Task Repository:** The tasks that are rejected by the VM Controller are submitted to this repository, where they stay until the next upcoming processing window to be rescheduled by the Container Mapper.
- **Available VM Capacity Repository:** IDs of virtual machines that have available resources are registered in this repository. It is used for assigning tasks killed by the Virtual Machine Controller along with newly arrived ones to available resource capacity.
- **Power Monitor:** This component is responsible for estimating the power consumption of the data center based on the resource utilization of available hosts.
- **Host Controller:** The Host Controller runs on each host of the data center. It periodically checks virtual machine resource usage (which is received from the Virtual Machine Controllers) and identifies underutilized machines, which are registered in the available resource repository. This component also submits killed tasks from the VMs running on its host to the *Rejected Task Repository* so that these tasks can be rescheduled in the next processing window. It also sends the host usage data to the Power Monitor.
- **Virtual Machine Controller (VMC):** The VMC runs on each VM of the data center. It monitors the usage of the VM and, if the resources usage exceeds the virtual machine capacity, it kills a number of containers with low priority tasks so that high priority ones can obtain the VM resources they require. In order to avoid task starvation, this component also considers the number of times a task is killed. Then the Controller sends killed tasks to the Host Controller to be submitted to the global rejected task repository. As mentioned before, the killed tasks are then rescheduled on an available virtual machine in the next processing window.

As we have described in this section, Task Clustering and VM Type Definer are two core components of the architecture. Therefore, in the next two sections we elaborate more on approaches that these components use to achieve effective and container-optimized VM size selection.

V. Task clustering

For validation of the proposed architecture, we use Google cluster traces. For the purpose of the evaluation of the approach, we selected to use the second day of the traces as it had the highest number of task submissions. As the first step, we have selected a subset of relevant features of tasks.

A. Clustering Feature Set

As our feature set, we used the following characteristics of each task:

Task Length: The time during which the task was running on a machine;

Submission Rate: The number of times that a task is being submitted to the data center;

Scheduling Class: How sensitive to latency is the task/job. In the studied traces the scheduling class is presented by an integer number between 0 and 3. The task with a 0 scheduling class is a non-production task. The higher the scheduling class is, the most latency sensitive the task is;

Priority: The priority of a task shows how important a task is. The high priority tasks get the preference for resources over the low priority ones [10]. The priority is an integer number in the range from 0 to 10;

Resource Usage: The average resource utilization U_T (calculated based on Equation 1) of a task T in terms of CPU, memory, and disk during the observed period, defined as:

$$U_T = \frac{\sum_{m=1}^{nr} u_{(T,m)}}{nr} \tag{1}$$

Where nr is the number of times that the task usage (u_T) is being reported in the observed period.

The selected features of the data set were used for estimating the number of task's clusters and determining the suitable virtual machine configuration for each group.

B. Clustering Algorithm

Clustering is the process of grouping the objects with the objective of finding the subsets with the most similarities in terms of the selected features. Therefore, the objective of the grouping and the number of groups both would affect the results of clustering. In our specific experiment, we focus on finding groups of tasks with similarities in their usage pattern so that we can allocate the available resources efficiently. A primary task in clustering is identifying the number of clusters for which we used X-means algorithm.

X-means [12] is the extended version of k-means [13] used for estimating the number of groups present in the incoming tasks. K-means is a computationally efficient partitioning algorithm for grouping N-dimensional dataset into k clusters via decreasing within-class variance. However, supplying the number of groups (k) as an input of the algorithm is the major challenge in K-means. However, X-means itself estimates the number of clusters. The other difference of two is in the searching process, k-means search is prone to local minima, while X-means efficiently searches the space of cluster locations and number of clusters in order to optimise Bayesian information Criterion (BIC). The BIC is a criterion for selection of the best fitting model amongst a set of available models for the data [14]. Optimising the BIC criterion (which is achieved in X-means) results in a better fitting model, which in the context of this work means finding the optimum number of clusters for the input data set.

In our implementation, we use R [15] as an environment for our statistical analysis of the data set and the statistical procedures including X-means clustering algorithm and data pre-processing. In this respect, RWeka [16] is used as our R interface to connect to the applied machine learning software Weka [17]. We used X-means from the clusterer package of Weka for estimating the number of clusters.

VI. Determination of VM Types

Once clusters that represent groups of tasks with similar characteristics in terms of the selected features are defined, the next step is to assign a VM type that can efficiently execute tasks that belong to the cluster. By efficiently, we mean successfully executing the tasks with minimum resource wastage. Parameters of interest of a VM are number of cores, amount of memory, and amount of storage. Apart from these, we added one more parameter named task capacity, which is the maximum number of tasks that can run in one VM without resulting in task rejection.

VM Sizing Strategy: For each group (cluster) of tasks, in order to determine the virtual machine's parameters, the average amount of resources required per hour for serving the user requests during the 24 hours observation period is estimated. Thus, the amount of CPU required per hour (CPU_h) is defined as follows:

$$CPU_h = \overline{nT} * \overline{CPU}_u \tag{2}$$

where $\overline{CPU}_u$ is the average amount of CPU usage on an hourly basis and $\overline{nT_h}$ is the average number of tasks per hour that are present and running in the system.

When the average amount of required CPU per hour is estimated, a limit should be set for the maximum amount of the CPU that a virtual machine can obtain. This amount should be set according to the underlining infrastructure. For example if the servers have 32 cores of CPU and the provider wants to host at least two virtual machines per server, then each VM can obtain at most 16 cores. If we assume that the largest virtual machine in the system would have less than 16 cores, then if the CPU_h is above 16, therefore the load should be served by more than one virtual machine. It has to be divided by the first integer n found between 2 to 9 in a way that the residual of CPU_h/n is zero (For example if the CPU_h is equal to 21, n would be 3, which results in 3 VMs with 7 cores each.) . In other words, n would be the number of virtual machines with $VM_{CPU} = CPU_h/n$ which is enough for serving the requests for every hour. Further, if the CPU_h is bellow our virtual machine maximum CPU limit, then one virtual machine would be enough to serve the requests and n is equal to 1.

Then, for defining the VM memory size, Equation 3 is used.

$$VM_{memory} = \frac{\overline{nT} * \overline{memory_u}}{n} \tag{3}$$

In our studied trace, since the tasks required small amount of storage, the allocated amount of storage for all of the virtual machines are assumed to be 10 GB, which is enough for the OS installed on the VM and the tasks disk usage. Furthermore, the VM's task capacity VM_{tc} is calculated based on Equation 4.

$$VM_{tc} = min(VM_1/\overline{t_{U_1}}, ..., VM_i/\overline{t_{U_i}}), i = \{CPU, memory\} \tag{4}$$

where $\overline{t_{U_i}}$ is the average resource usage during the observed period.

Thus, for each cluster, the VM type is defined in terms of CPU_h and VM_{memory}. Once VM Types are determined, they are stored in the VM types repository, so future selection of better matches for required containers hosting the tasks are chosen based on such defined types. The effectiveness of this strategy for selection of VM Types based on clustering of tasks is evaluated in the next sections.

VII. Evaluation

In this section, we discuss the experiments that we conducted to evaluate our proposed approach in terms of its efficiency in task execution and power consumption.

The data set used in this paper is derived from the second version of the Google cloud trace log [10] collected during a period of 29 days. The log consists of data tables describing the machines, jobs, and tasks. In the trace log, each job consists of a number of tasks with specific constraints. Considering these constraints, the scheduler determines the placement of the tasks on the appropriate machines. The event type value in the job and tasks are reported in the event table. For the purpose of this evaluation, we utilize all the events from the trace log and we assume that all the events are happening as reported in the trace log.

TABLE I. VIRTUAL MACHINE CONFIGURATIONS FOR 18 CLUSTERS.

VM Type	Number of Tasks	vCPU	Memory (GB)	VM Type	Number of Tasks	vCPU	Memory (GB)
TYPE 1	142	14	17.93	TYPE 10	418	8	14.8
TYPE 2	79	1	2.82	TYPE 11	471	9	28.98
TYPE 3	261	3	11.70	TYPE 12	87	2	4.11
TYPE 4	292	2	2.39	TYPE 13	439	1	3.68
TYPE 5	1836	1	5.35	TYPE 14	185	3	22.48
TYPE 6	107	3	8.1	TYPE 15	78	1	2.72
TYPE 7	70	2	3.21	TYPE 16	220	3	10.12
TYPE 8	585	1	0.93	TYPE 17	254	1	14.14
TYPE 9	336	1	2.06	TYPE 18	1200	1	1.52

TABLE II. VIRTUAL MACHINE SPECIFICATIONS OF RFS AND THE SELECTED AMAZON EC2 INSTANCES.

RFS				Amazon EC2			
VM Type	Number of Tasks	vCPU	Memory (GB)	VM Type	C_{max}	vCPU	Memory (GB)
TYPE 1	452	1	5.42	m3.large	150	1	7.5
TYPE 2	696	9	16.1				
TYPE 3	273	3	5.32	m3.medium	150	1	3.75
TYPE 4	1639	11	24.29	t2.medium	266	2	4
TYPE 5	357	1	3.29	t2.small	133	1	2

In the available traces, resource utilization measurements and requests are normalized, and the normalization is performed separately for each column in relation to the highest amount of the particular resource found on any of the machines. In this context, to get a real sense of the data, we assume the largest amount of resources including CPU, memory, and disk to be 100% of a core of the largest machine CPU (3.2 GHz), 4GB and 1 GB respectively. Therefore, for every column we multiply each recorded data by the related amount (e.g for recorded memory utilization we have $Real_{util} = Recorded_{Util} * 4$).

The Google cluster hosts are heterogeneous in terms of the CPU, memory and disk capacity. However, the hosts with the same platform ID have the same architecture. In order to eliminate the placement constraints and decrease placement complexity, only the tasks scheduled on one of the three available platforms are considered. The configurations of the simulated data center servers are inspired by Google data center during the studied trace period. As suggested by Garraghan et.al [18], the servers in our selected platform are PRIMERGY RX200 S7. For the PRIMERGY platform, the tasks scheduled during the second day of the traces are all scheduled on one server type with 32 cores of CPU (3.2 GHz), 32 GB of memory and 1 TB of disk. The power profile of this server type are extracted from the SPECpower_ssj2008 results [19] reported for PRIMERGY. This power profile is then used for determining the constants of the applied power linear model presented in Equation 5 (in this equation n is the CPU utilization percentage of each server). We focus on energy consumption of CPU because this is the component that presents the largest variance in energy consumption in regards to its utilization rate [20].

$$P_n(t_i) = (P_{max} - P_{idle}) * n/100 + P_{idle} \quad (5)$$

The proposed system is simulated and the tasks are assigned to the containers. Then, these containers are hosted in the corresponding virtual machine types during each processing window (one minute for the purposes of these experiments). The simulation runtime is set to 24 hours. The number of rejected tasks is reported for each experiment separately. Since the virtual machines placement also affects the simulation result, First Fit placement policy is used for all of the experiments. This placement algorithm reports the first running host that can provide the resources for the VM and if no running host is found for placing the VM then a new host will be turned on.

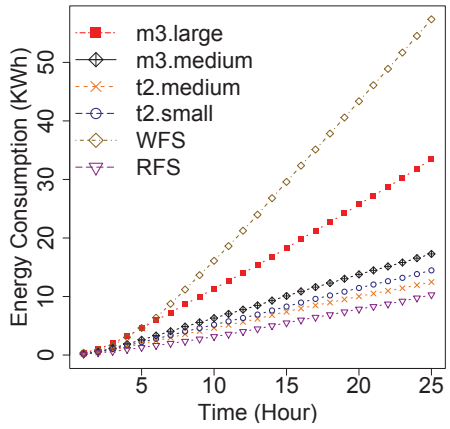

Fig. 3. Energy consumption of the data center for the usage-based fix VM size approach versus RFS and WFS

Output metrics of interest are number of instantiated VMs, task rejection rate, and the energy consumption (Subsection VII-A3). In order to ensure the quality of service, task rejection rate is defined as the number of rejected tasks in each 1 minute processing window.

A. Results

Initially, we explore the factors that affect the resulting virtual machine sizes (types). The number of VM types is directly affected by the number of task clusters (k), which

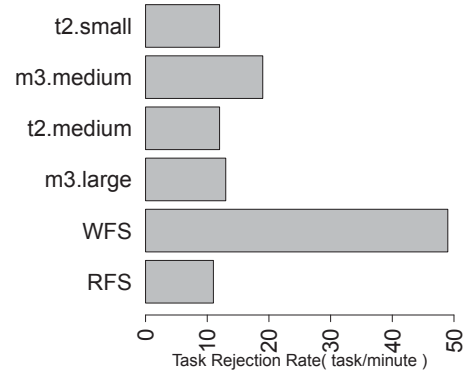

Fig. 4. Task rejection rate for WFS, RFS and the fixed VM sizes considering the usage-based approach

is estimated via applying X-means on the selected feature set. The details are further discussed in Section VII-A1.

The efficiency of the proposed VM size selection technique is then explored considering the baseline scenarios in which the virtual machine types are fixed. For the baseline scenarios, the containers are assigned to VMs considering two different criteria discussed in Subsection VII-A2.

1) Feature set selection: The following approaches are considered in the investigation the effect of feature selection on virtual machine sizes:

Whole Feature Set (WFS) approach: In this approach, the whole feature set listed in Section V-A is considered as the input of the X-means algorithm, which resulted in 18 clusters of tasks. The virtual machine sizes are defined following the procedures in Section VI. The obtained VM sizes are listed in Table I. As we mentioned in Section VI, 10 GB of storage is assigned to all VM types in order to have enough space for the OS installed on each VM.

Reduced Feature Set (RFS) approach: By assigning the same amount of storage, WFS approach leads to the observation that the most effective parameters in VM size selection for this specific workload are the average CPU and memory utilization. Therefore, for the RFS approach, the clustering feature set is reduced to these two main features. The application of the new feature set as input of X-means resulted in 5 clusters, which consequently results in 5 different VM sizes (shown in Table II).

Between these two approaches, the one that could save more energy via reducing the number of servers has the more efficient virtual machine sizes. As Figure 3 shows, RFS results in 81% less energy consumption in comparison to WFS. This is because RFS causes less resource fragmentation and that leads to less number of servers. Therefore, it can be concluded that tailoring the size of virtual machines to container requirement is crucial, however it has to be done in such a way that avoids resource fragmentation. Apart from the energy perspective, comparing to WFS approach, RFS results in 86% and 77% less number of instantiated VMs and rejection rate respectively.

2) Baseline scenarios: In order to investigate the effect of defining the virtual machine sizes according to the workload (container optimized), in these experiments we consider scenarios where VM sizes are fixed. In this respect, we consider the existing Amazon EC2 instances listed in Table II. In these scenarios, the scheduler assigns the containers to only one specific VM size (e.g t2.small instance) considering the following approaches:

Request-based resource allocation approach: In the request-based resource allocation approach, tasks are given the exact amount of resources requested and specified by the user while submitted. Therefore, in this approach containers are packed in the virtual machines according to their containing task specifications, as submitted by users.

Usage-based resource allocation approach: In this approach, the maximum number of containers (c_{max}) that can be hosted in one VM instance is specified in the pre-execution phase leveraging the average resource utilization of the tasks. c_{max} is reported for each of the Amazon instances in Table II. This approach is included in the baseline scenarios, since the concept of utilizing the usage of the tasks is also taken into account in our VM size selection technique.

$$c_{max} = min(VM_r/\overline{Usage_r}), r = CPU, Memory, Disk \tag{6}$$

3) Discussion: The results obtained with the utilization of the aforementioned approaches are compared considering three different matrices. The first matrix, which shows the main effect of our proposed VM size selection selection technique, is the energy consumption matrix obtained through Equation 5. Figure 3 shows the energy consumption for the baseline scenarios considering the usage-based approach and the proposed VM size selection technique. The data center energy consumption for the baseline scenarios considering the requested-based approach is also plotted in Figure 5. Because users overestimate resource requirements, the energy consumption is improved in the usage-based approach by almost 50% comparing to the requested-based approach in the baseline scenarios, for all of the fixed VM sizes. However, RFS still outperforms all of the experiments in terms of the data center energy consumption. For the usage-based baseline scenarios, RFS shows 33%, 44%, 24%, and 70% less energy consumption on average comparing to t2.small, t2.medium, m3.medium, and m3.large VM sizes respectively.

For the second comparison matrix, we consider the number of instantiated virtual machines (n_{VM}). The increase in n_{VM}, results in more virtualization overhead and also longer delays in scheduling because of the VM instantiation delay. Results show that RFS can execute the workload with the smallest n_{VM}. It outperforms the usage-based baseline scenarios by 82%, 75%, 37%, and 86% less number of instantiated VMs for the t2.small, t2.medium, m3.medium, and m3.large VM sizes respectively.

In order to ensure the quality of service, the task rejection rate is considered as the third comparison matrix. Task rejection rates for the usage-based baseline scenarios, WFS, and RFS approaches are shown in Figure 4. As in previous cases, RFS VM size selection approach outperforms the usage-based baseline scenarios by 8%, 42%, 8%, and 15% less task rejection rate for the t2.small, t2.medium, m3.medium, and m3.large VM sizes respectively. It is worth mentioning that,

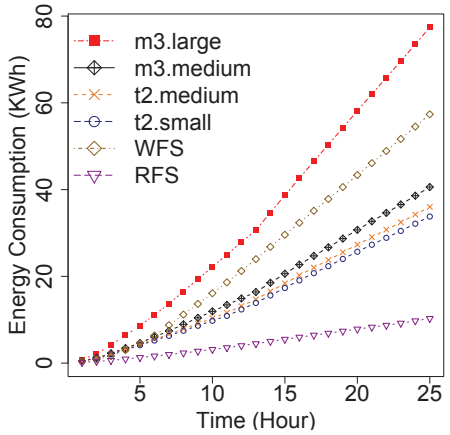

Fig. 5. Energy consumption of the data center for the request-based fix VM size approach versus RFS and WFS.

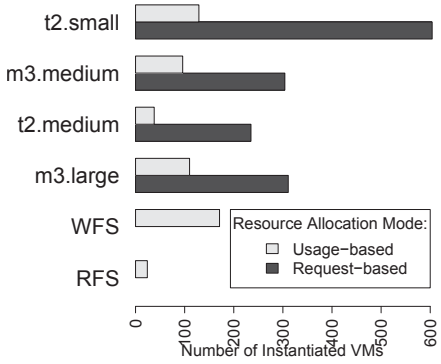

Fig. 6. Number of instantiated virtual machines for the applied approaches.

because of the over allocation of resources for the requested-based baseline scenarios, the task rejection rate is equal to zero for all of the VM types.

In summary, the experiments show that our VM size selection technique combined with workload information, saves considerable amount of energy with minimum task rejections.

VIII. Conclusions and Future Directions

In this work, we investigated the problem of inefficient utilization of data center infrastructure resulted from the users overestimation of required resources on virtual machine level. To address the issue, we considered the recently introduced CaaS cloud service model and presented a technique for finding efficient virtual machine sizes for hosting containers considering their actual resource usage instead of users estimated amounts.

To investigate the efficiency of our VM sizing technique, we considered baseline scenarios in which the virtual machine sizes are fixed. Because of user overestimation of resources, the usage-based approach (where VM sizes are chosen based on actual requirements of applications rather than amount requested by users) outperforms the requested-based approach by almost 50% in terms of the data center average energy consumption. Despite this improvement, our approach outperforms the usage-based baseline scenarios by almost 7.55% in terms of the data center energy consumption. Apart from the energy perspective, our approach results in less number of VM instantiation comparing to all other mentioned experiments.

As future work, we will consider workload prediction to estimate the real usage of containers, and will apply correlation analysis to place VMs that are not correlated in one host to minimize rejection rate.

References

[1] M. Armbrust *et al.*, "Above the clouds: A Berkeley view of cloud computing," University of California at Berkeley, Berkeley, Technical Report UCB/EECS-2009-28, Feb. 2009.

[2] R. Buyya, A. Beloglazov, and J. Abawajy, "Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges," in *PDPTA 2010: Proc. of the 2010 International Conference on Parallel and Distributed Processing Techniques and Applications*. CSREA Press, pp. 6–17.

[3] Mishra *et al.*, "Towards characterizing cloud backend workloads: insights from Google compute clusters," *ACM SIGMETRICS Performance Evaluation Review*, vol. 37, no. 4, p. 3441, 2010.

[4] Y. Chen, A. S. Ganapathi, R. Griffith, and R. H. Katz, "Analysis and lessons from a publicly available google cluster trace," Tech. Rep., 2010.

[5] A. Kansal *et al.*, "Virtual machine power metering and provisioning," in *Proc. of the 1st ACM symposium on Cloud computing*, 2010.

[6] R. Nathuji and K. Schwan, "VirtualPower: coordinated power management in virtualized enterprise systems," in *ACM SIGOPS Operating Systems Review*, vol. 41, 2007, pp. 265 –278.

[7] K. H. Kim, A. Beloglazov, and R. Buyya, "Power-aware provisioning of cloud resources for real-time services," in *Proc. of the 7th International Workshop on Middleware for Grids, Clouds and e-Science*, 2009, p. 1.

[8] D. Gmach, J. Rolia, and L. Cherkasova, "Selling t-shirts and time shares in the cloud," in *Proc. of the 2012 12th IEEE/ACM International Symposium on Cluster, Cloud and Grid Computing (ccgrid 2012)*. IEEE Computer Society, 2012, pp. 539–546.

[9] S. Hosseinimotlagh, F. Khunjush, and R. Samadzadeh, "Seats: smart energy-aware task scheduling in real-time cloud computing," *The Journal of Supercomputing*, vol. 71, no. 1, pp. 45–66, 2015.

[10] C. Reiss, J. Wilkes, and J. L. Hellerstein, "Google cluster-usage traces: format+ schema," Tech. Rep., 2011.

[11] Q. Zhang, M. F. Zhani, R. Boutaba, and J. L. Hellerstein, "HARMONY: Dynamic heterogeneity-aware resource provisioning in the cloud," *The 33rd International Conference on Distributed Computing Systems (ICDCS)*, July 2013.

[12] D. Pelleg and A. W. Moore, "X-means: Extending k-means with efficient estimation of the number of clusters," in *Seventeenth International Conference on Machine Learning*. Morgan Kaufmann, 2000.

[13] J. Hartigan and M. Wong, "Algorithm as 136: A k-means clustering algorithm," *Applied Statistics*, vol. 28, no. 1, pp. 100–108, 1979.

[14] G. Schwarz *et al.*, "Estimating the dimension of a model," *The annals of statistics*, vol. 6, no. 2, pp. 461–464, 1978.

[15] R. C. Team, *R: A Language and Environment for Statistical Computing*, R Foundation for Statistical Computing, Vienna, Austria, 2013.

[16] K. Hornik *et al.*, "Open-source machine learning: R meets Weka," *Computational Statistics*, vol. 24, no. 2, pp. 225–232, 2009.

[17] M. Hall, E. Frank, G. Holmes, B. Pfahringer, P. Reutemann, and I. H. Witten, "The weka data mining software: an update," *ACM SIGKDD explorations newsletter*, vol. 11, no. 1, pp. 10–18, 2009.

[18] P. Townend, J. Xu, and I. S. Moreno, "An analysis of failure-related energy waste in a large-scale cloud environment," *IEEE Transactions on Emerging Topics in Computing*, p. 1, 2014.

[19] S. P. E. Corporation. (2012) Specpower_ssj2008 results. [Online]. Available: http://www.spec.org/power_ssj2008/results/

[20] M. Blackburn, Ed., *Five ways to reduce data center server power consumption*. The Green Grid, 2008.