Center for Open Access in Science ▪ https://www.centerprode.com/ojit.html
Open Journal for Information Technology, 2020, *3*(1), 1-10.
ISSN (Online) 2620-0627 ▪ https://doi.org/10.32591/coas.ojit.0301.01001m

MFF: Performance Interference-Aware VM Placement Algorithm for Reducing Energy Consumption in Data Centers

Derdus Mosoti & Vincent O. Omwenga

Strathmore University, Faculty of Information Technology, Nairobi, KENYA

Patrick Ogao

Technical University of Kenya, Nairobi, KENYA
Faculty of Engineering and Build Environments

Received 29 November 2019 ▪ Accepted 22 January 2020 ▪ Published Online 25 February 2020

Abstract

Virtualization is the main technology that powers cloud computing and has enabled the execution of multiple applications in same physical hardware using virtual machines (VM) for efficient utilization of resources and energy savings. Although virtualization successfully isolates co-resident VMs from a security perspective, it does not offer a guarantee from a performance interference perspective. This means that sharing of resources results in competition, which is the cause of performance interference. Performance interference is more pronounced in homogenous workloads, where applications workloads contend to the same shared resource. In this case, application workloads run for longer times due to reduced performance and thus consume more energy. To address this problem, a VM allocation policy should ensure that VM running homogeneous workloads is not co-located. In this paper, we propose a VM allocation algorithm called Minimum Interference First Fit (MFF), which co-locates dissimilar workloads. The algorithm clusters VMs using K-means based on resources usage. Before a VM is placed into a physical machine (PM), similarity index (SI) of all the active PMs is computed, the VM is then placed in a PM with least SI. MFF has been evaluated on a simulated data center using CloudSim Plus cloud simulator on application workloads logs obtained from a production data center. Results show that MFF outperforms well-known VM allocations algorithms such as first fit (FF), worst fit (WF) and best fit (BF) from an energy consumption perspective.

Keywords: cloud computing, virtual machine allocation, k-means, virtualization, data center energy consumption, performance interference.

1. Introduction

The growing appetite for workload processing power has resulted in cloud service providers (CSP) putting up many data centers. Unfortunately, data centers consume a lot of electrical energy resulting in high operating costs (Rallo, 2014). Besides, excessive energy consumption has a negative impact on the environment, which is the emission of carbon dioxide gas to the environment (Anton, 2013). Not all the energy that goes into a data center does useful work, some goes to waste. There are a number of known causes of energy wastage in data centers

Correspondence: Derdus Mosoto, Strathmore University, Faculty of Information Technology, Nairobi, KENYA. E-mail: dkenga@strathmore.edu.

such as low level of server utilization, wastage of server idle energy and consolidation of homogeneous workloads (Chaima, 2014; Derdus, Omwenga & Ogao, 2019). The problem of low server utilization and wastage of idle energy has been addressed in a number of research work such as in (Anton, 2013; Delimitrou, 2015).

Consolidation of workloads has an effect on performance and energy consumption issues. For instance, it has been shown that it is advantageous to co-located heterogeneous workload than homogenous workloads (Derdus, Omwenga & Ogao, 2019). Homogenous workloads have applications, which contend the same shared resource. Thus, if VMs run application workloads, which contend the same shared resource, the pressure put on that resource will make tasks to run longer because of reduced throughput. As a consequence, more energy is used (energy is a product of time and power). This is effect is known as performance interference. Although virtualization successfully isolates co-resident VMs from a security perspective, it does not offer a guarantee from a performance interference perspective (Tesfatsion, 2018). Performance interference can be very severe. For instance, network I/O bandwidth can vary by almost 50% due to inter-VM interference (Pu et al., 2010).

The methods used to address the problem of performance interference include hardware partition and VM placement and allocation policies (Tesfatsion, 2018; Amri, Hamdi & Brahmi, 2017). In former, the physical hardware resources are divided to enable hosted VMs to have exclusive access the resources. In the latter, VM placement policies are used, where the incoming VM's behavior is analyzed so as to co-locate VMs, which contend different resources. The commonly used bin packing based VM placement algorithms such as first fit (FF), worst fit (WF) and best fit (BF) are not suitable because they do not take into account VM characteristics before placement (Kumar, Sathasivam & Periyasamy, 2016; Dabbagh et al., 2015; Gohil et al., 2016). For instance, FF places an incoming VM in the first active PM or host, which has enough resources to accommodate the VM. If no suitable host is found, a new PM is activated. In BF, all PMs are checked for residual resources and the incoming VM is placed in a PM, which will suffer least resource wastage. In WF, the incoming VM is placed in a PM with most residual resources.

Various research works have been performed to address the problem of performance interference. Most of this research work is focused on either detecting, predicting or measuring performance interference in co-resident VMs with the goal of proposing a solution.

Xu, Liu and Jin (2016) discovered that apart from performance interference caused by executing application workloads, the existence of heterogeneous hardware similar VM instances can be a source of performance variations. Based on this claim, the authors have proposed a system called Heifer, which predicts the performance of hosted applications in a similar VM instance considering the performance interference in different hardware. Heifer provisions VM instances from the best performing hardware based on the predicted performance interference.

Chen et al. (2015) proposed a system called CloudScope, which is used in diagnosing performance interference among co-resident VMs. CloudScope measures performance interference using VM profiling information obtained from the hypervisor layer and then reassigns VMs to PMs in way that interference is minimized. CloudScope has been implemented on Xen and according to the authors, it achieves an average error of 9% in predicting interference and a 10% VM performance improvements as compared to Xen's default scheduler. The authors also argue that CloudScope is lightweight with less computation needs as compared to other approaches, which uses online training.

Amannejad, Krishnamurthy and Far (2015) proposed a machine learning based system for detecting if a web service running in hosted VMs is suffering from interference. The proposed system relies on VM customer accessible metric (such as transaction response time data collected by a web service) as a way of detecting performance degradation since hardware level metrics (such as CPU utilization) are only accessible by CSP. The actual performance values are

then compared to the expected performance. If performance interference is detected the system can mitigate it such as by using a different VM instance or using a different CSP.

In this paper, we extend our work (in Derdus, Omwenga & Ogao, (2019)), by proposing a VM allocation algorithm that maps a VM to a PM, which is hosting less similar VMs. In our approach, interference is estimated by computing a value called similarity index, which is a measure of how similar a host is to an incoming VM.

2. Materials and methods

2.1 *Cloud model and system process*

The target cloud deployment and service model for the proposed algorithm is a multi-tenant public Infrastructure as a Service (IaaS) cloud. Users request VM resources and the VM is placed in a PM in CSP's data centre. The user can then execute any type of applications in the VMs, and from a CSP's perspective, applications are a black box host in a VM. However, a CSP can access a VMs profile (such as VM resources consumption) from the hypervisor layer and in turn analyze the behaviour of user host application (Amannejad, Krishnamurthy, & Far, 2015). Further, the proposed system process is shown in Figure 1.

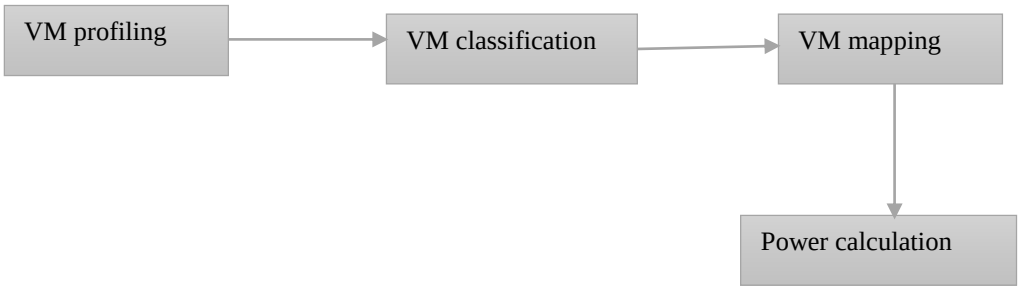

Figure 1: Proposed system process

VM profiling: in this part, the resource usage of the VM is monitored and recorded over a period. This is done by the CSP via the virtualization layer. The collected data forms the historical data, which fed into a VM classifier. This is possible and has been demonstrated in (Wajid, et al., 2016)

VM classification: this component receives historical VM resource usage from a profiling database and then classifies VM based on resource usage. In this paper, we have used k-means to classify VMs based on the following features:

- Average CPU usage – the average CPU used by a VM for the entire period of profiling.
- Average memory usage – the average RAM used by a VM for the entire period of profiling.

The choice of the k-means clustering algorithm is because of its success in clustering workloads in previous research such as the work by Alam, Shakil and Sethi (2016), Yousif and Al-Dulaimy (2017) and Di, Kondo and Cappello (2014). In the following sections, we will explain how k-means was applied in a dataset of choice.

VM mapping: the VM mapping component receives a VM and the class to which it belongs and then maps it to the appropriate PM. This is the component, which runs our algorithm. The algorithm logic has been explained in a later section.

Power calculation: this component computes the total amount of power used to execute VM workloads for a given VM scheduling algorithm. It is important to calculate this power because the aim of our algorithm is to reduce power consumption. Power, PT, is computed according to equation 1.

$$P_T = \sum_{i=1}^{n}((P_i^p - P_i^b) * \left(\frac{N_i}{100}\right) + P_i^b, \quad (1)$$

where n is the number of hosts in a data center, Pp is the peak power consumption of the ith host, Pb is the host's idle power and N is the percentage CPU utilization of the host. Energy, E, computed as shown in equation 2.

$$E = PT, \quad (2)$$

where P is equivalent to PT (measured in watts) and T is a time (in seconds) interval.

2.2 *Dataset used*

The dataset used in this paper is called c and is obtained from the Grid Workload Archive (GWA) (Delf University of Technology, 2018). The dataset is used in the following three ways: (1) for clustering purposes i.e. to demonstrate the use of k-means in clustering application workloads as well as the existence of groups of VMs, (2) to aid in determining characteristics of a data center to be used in evaluating our proposed algorithms, and (3) to aid in determining resource demands of VMs used to execute workloads in the process of evaluation. Materna trace is obtained from a VMware ESX environment consisting of 49 Hosts, 69 CPU cores and 6780 GB memory. This trace data is packaged in CSV files and it shows resources (such as memory, processor and storage) assigned to VMs and the resources that were actually used by the VMS. Materna trace consists of three different traces obtained from the infrastructure during three different times. The first trace, which has been used in this paper, consists of 520 VMs. Thus, there are 520 CSV files, each showing resources allocated and resources actually used by the VMs. Different VMs were allocated different resources. For instance, storage capacity ranged from 54 GB – 138 GB, memory was either 2 or 4 or 8 or 16 GB whereas a number of CPU cores were either 1 or 2 or 4 or 8. CPU utilization also showed the percentage usage in MHz. Resources allocated to VMs did not change the entire period of profiling. On the other hand, resources actually used by VMs varied the entire period. We have used the average VM resource usage to represent VM resource usage for the entire profiling period. Thus, we have created one CSV file with 520 records showing VM resource usage averages from the 520 CSV files. Classes of VMs are created via k-means using average memory usage and average CPU usage as a feature set.

2.3 *Algorithm design*

This algorithm determines the PM to host a VM, based on the VM's class. The algorithm minimizes the number of VMs of a similar class running in the same PM. The algorithm can be summarized using the following steps and the flow chart shown in Figure 2.

Step 1: From all host machines (PMs), identify all PMs with enough resources to accommodate the incoming VM. This is the *candidate host list*.

Step 2: Compute the similarity, *S*, of the incoming VM with each of the *candidate hosts*. *S* is computed according to equation 3.

Step 3: Sort the *candidate host list* in ascending order using *S*.

Step 4: Place incoming VM in the first host of the sorted *candidate host list*.

$$S = \frac{i}{j} \qquad (3)$$

where *i* is the number of VMs in a particular host, which belong to the same class as the incoming host and *j* is the total of VMs in that host.

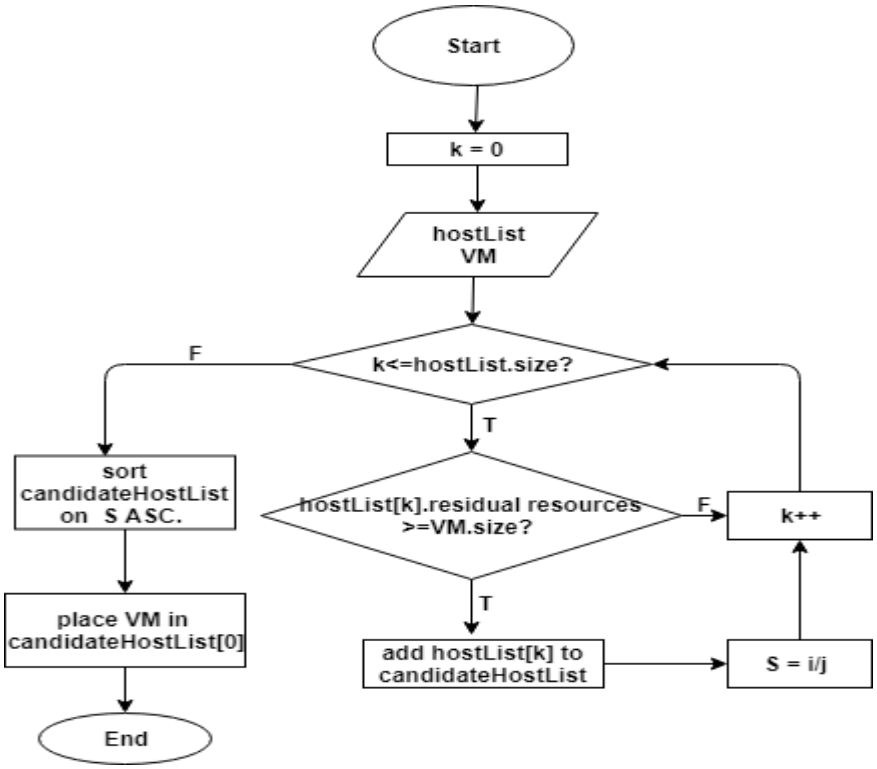

Figure 2: Flow chart for the proposed algorithm

2.4 *Algorithm evaluation*

Our algorithm is evaluated using GWA-T-13 Materna (explained earlier) on CloudSim Plus simulator, CloudSim Plus is a java-based cloud simulator forked from CloudSim (Rodrigo et al., 2011; Manoel et al., 2017). CloudSim Plus is easier to use because it follows software engineering standards with code duplication entirely removed. CloudSim Plus perfectly emulates a cloud datacenter – it has the following components; a Cloudlet, VM, Broker, Host and Datacenter. A cloudlet is similar to user applications, which are executed inside VMs. VMs are held in hosts, which are typically servers in a datacenter. A datacenter is comprised of hardware with physical computing resources and all the software that is used to manage the hardware. CloudSim Plus framework allows the creation of the aforementioned components in Java code. It also provides interfaces and abstract classes, which can be implemented and extended respectively, to enable the creation of own algorithm to determine how VMs are mapped to PMs. Some of the commonly used VM allocation algorithms such as FF, WF and BF have been implemented in the default installation of CloudSim Plus simulator. In this work, we have simulated a datacenter with 49 hosts and 520 VMs. The datacenter has 69 CPUs (454 cores) and 6780 GB of memory. The memory and CPU allocation to each VM is in line with GWA-T-13 Materna dataset. The cloudlets are also simulated to consume the same amount of resources depicted in the workload. The idle

power for each host is set at 60% of the host's peak power. For algorithm implementation, we have created a class, which inherits from *VmAllocationPolicyAbstract* and implemented the algorithm in a method called *findHostForVm*. The workload depicted in the dataset is then executed in the datacenter using FF, WF, BF and MFF VM allocation algorithms in turn. At the end of each execution, energy consumption by the data center is recorded.

3. Results and discussion

The results presented and discussed in this section relates to the outcome of clustering of the used dataset for the purpose of determining VM allocation and the performance of the proposed algorithm as compared to other algorithms. Figure 3 shows the clustering results of GWA-T-13 Materna. Indeed, it shows that there existed groups of VMs based on VM resources consumption (CPU and memory). K-means revealed the existence of 4 groups, which can be described as extra small VMs, small VMs, medium VMs and large VMs. The population of each group and a description of the VM groups have been summarized in Table 1.

Table 1. Population and description of VM groups resulting from clustering GWA-T-13 Materna dataset

VM Cluster group	VM population (number and %)	VM group description
Large VMs	1 (≈ 0.2%)	We have considered this group to contain an outlier because there is only one member. The VM in this groups shows a high memory consumption, with moderate CPU consumption.
Medium VMs	29 (≈ 5.6%)	The VMs in this group shows a moderate memory consumption with varying CPU usage.
Small VMs	96 (≈18.4%)	The VMs in this group shows a moderate memory consumption with varying CPU usage. However, the memory demand for this group is lower compared to medium VMs.
Extra small VM	394 (≈75.8%)	The VMs in this group shows low consumption in both memory and CPU.

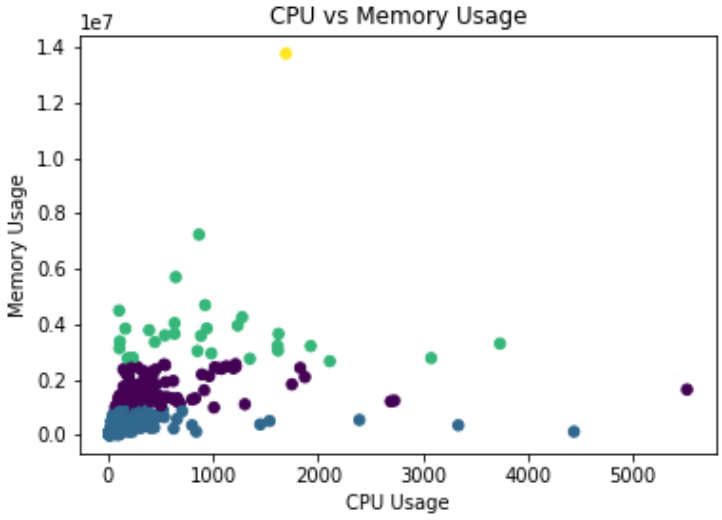

Figure 3: Scatter plot showing results of clustering of GWA-T-13 dataset

Further, Figure 4 shows the amount of energy consumed by the datacenter (consisting of 46 hosts) while executing the workload using a different VM allocation algorithm. We compared our proposed algorithm, MFF with FF, BF and WF. From the results, it is noticeable that MFF consumes the least amount of energy (18766 joules). On the other hand, BF consumes the highest amount of energy to execute the same workload to completion (22674 joules). The reason for MFF's better performance from an energy perspective is because it co-locates VMs, which content different computing resources (heterogeneous VM). This type of allocation reduces inter-VM interference caused by VM when they compete with hypervisor capacity. By ensuring that dissimilar VMs are co-located, all computing resources are used in a balanced manner, which also means that physical resources' idle power is put into useful processing.

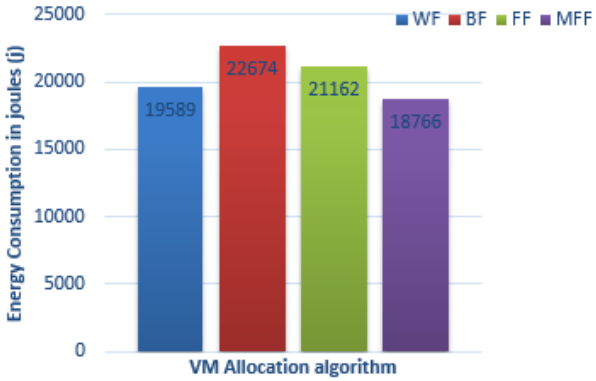

Figure 4: Energy consumed by datacenter hosts to execute same amount of workload across different VM allocation algorithms

4. Conclusion

In this paper, we have proposed a VM allocation algorithm, MFF, for mapping VMs to PMs, based on historical resources consumption of VMs. MFF is motivated by the fact that inter-VM interference of co-resident VM is reduced when the VMs content different computing resources. We have used a k-means clustering algorithm to identify groups of VMs in a dataset used. Simulated results show that MFF beats FF, BF and WF VM algorithms and we consider this an achievement. As future work, we plan to apply MFF to a wide range of real cloud workloads. We also plan to combine MFF and BF to further enhance resource utilization efficiency.

Acknowledgements

This research did not receive any specific grant from funding agencies in the public commercial, or not-for-profit sectors.

The authors declare no competing interests.

References

Alam, M., Shakil, K., & Sethi, S. (2016). Analysis and clustering of workload in google cluster trace based on resource usage. *2016 IEEE Intl Conference on Computational Science and Engineering (CSE) and IEEE Intl Conference on Embedded and Ubiquitous Computing (EUC) and 15th Intl Symposium on Distributed Computing and Applications for Business Engineering (DCABES)*. Paris, France.

Amannejad, Y., Krishnamurthy, D., & Far, B. (2015). Detecting performance interference in cloud-based web services. *2015 IFIP/IEEE International Symposium on Integrated Network Management (IM)*. Ottawa, ON, Canada: IEEE.

Amri, S., Hamdi, H., & Brahmi, Z. (2017). Inter-VM interference in cloud environments: A survey. *2017 IEEE/ACS 14th International Conference on Computer Systems and Applications*. Hammamet, Tunisia: IEEE.

Anton, B. (2013). *Energy-efficient management of virtual machines data centers for cloud computing*. The University of Melbourne, Department of Computing and Information Systems.

Chaima, G. (2014). *Energy efficient resource allocation in cloud computing environment*. Paris, France: Institut National des T´el´ecommunications.

Chen, X., Rupprecht, L., Osman, R., Pietzuch, P., Franciosi, F., & Knottenbelt, W. (2015). CloudScope: Diagnosing and managing performance interference in multi-tenant clouds. *2015 IEEE 23rd International Symposium on Modeling, Analysis, and Simulation of Computer and Telecommunication Systems*.

Dabbagh, M., Hamdaoui, B., Guizani, M., & Rayes, A. (2015). Toward energy-efficient cloud computing: Prediction, consolidation, and overcommitment. *IEEE Network*, *29*(2).

Delf University of Technology (2018). *GWA-T13-materna-trace* (Delf University of Technology). Retrieved 23 November 2018, from http://gwa.ewi.tudelft.nl/datasets/gwa-t-13-materna.

Delimitrou, C. (2015). *Improving resource efficiency in cloud computing*. Stanford University.

Derdus, K. M., Omwenga, V. O., & Ogao, P. J. (2019). The effect of cloud workload consolidation on cloud energy consumption and performance in multi-tenant cloud infrastructure. *International Journal of Computer Applications (0975-8887), 181*(37), 47-53.

Di, S., Kondo, D., & Cappello, F. (2014). Characterizing and modeling cloud applications/jobs on a Google data center. *The Journal of Supercomputing*, *69*(1), 139-160.

Gohil, B., Shah, S., Golechha, Y., & Patel, D. (2016). A comparative analysis of virtual machine placement techniques in the cloud environment. *International Journal of Computer Applications, 156*(14), 12-18.

Kumar, A., Sathasivam, C., & Periyasamy, P. (2016). Virtual machine placement in cloud computing. *Indian Journal of Science and Technology, 9*(29).

Manoel, F., Oliveira, R., Monteiro, C., Inácio, P., & Freire, M. (2017). CloudSim Plus: A cloud computing simulation framework pursuing software engineering principles for improved modularity, extensibility and correctness. *2017 IFIP/IEEE Symposium on Integrated Network and Service Management (IM)*. Lisbon, Portugal.

Pu, X., Liu, L., Mei, Y., Sivathanu, S., Koh, Y., & Pu, C. (2010). Understanding performance interference of I/O workload in virtualized cloud environments. *2010 IEEE 3rd International Conference on Cloud Computing*. Miami, FL, USA: IEEE.

Rallo, A. (2014). *Industry outlook: Data center energy efficiency*. Retrieved 4 August 2015, from Data Center Journal: http://www.datacenterjournal.com/industry-outlook-data-center-energy-efficiency/.

Rodrigo, C., Rajiv, R., Anton, B., Cesar, D. R., & Rajkumar, B. (2011). CloudSim: A toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms. *Journal of Software: Practice and Experience*, *4*(1), 23-50.

Tesfatsion, S. K. (2018). *Energy-efficient cloud computing: Autonomic resource provisioning for datacenters*. Umea: Umea University.

Wajid, U., Cappiello, C., Plebani, P., Pernici, B., Mehandjiev, N., Vitali, M., . . . Sampaio, P. (2016). On achieving energy efficiency and reducing CO2 footprint in cloud computing. *IEEE Transactions on Cloud Computing*, *4*(2).

Xu, F., Liu, F., & Jin, H. (2016). Heterogeneity and interference-aware virtual machine provisioning for predictable performance in the cloud. *IEEE Transactions on Computers*, *65*(8), 2470-2483.

Yousif, S., & Al-Dulaimy, A. (2017). Clustering cloud workload traces to improve the performance of cloud data centers. *Proceedings of the World Congress on Engineering*. London, UK.


C O A S