Journal of Harbin Engineering University
ISSN: 1006-7043

Vol 45 No. 7
July 2024

Energy Efficient Aware Computational Load Balancing Framework Algorithm for Fog-Cloud Data Centers

[1]Khulekani Sibiya, [2]Bakhe Nleya, Mlungisi Molefe[3]

[1,2,3] Department of Electronic and Computer Engineering, Durban University of Technology, Durban, South Africa

Abstract: Data centers form the foundation of Cloud computing systems. The overall operational costs are directly influenced by the resource management algorithms that assign virtual machines (VMs) to physical servers and the ability to relocate them in emergencies, such as power losses due to excessive heating. This paper presents a hierarchical SLA-based distributed-based resource provisioning and optimization framework algorithm. This framework considers constraints like energy consumption, cooling-related energy consumption, and scalability issues. We also include a load-balancing algorithm to minimize the operational costs of the distributed data center cloud system. Simulation results demonstrate that this approach, together with load balancing, significantly reduces the operational costs of the entire cloud system. This reduction is significantly pronounced when considering dynamic energy tariffs from integrating renewable power sources to supply server power.

Keywords: *resource optimization, management, peak power dynamic pricing*.

1 Introduction

The continuing global surge in various cloud services has led to a sudden increase in the demand for Datacenters. A data center, as defined, is a physical site where businesses and other entities store important data and applications. It is designed around a system of computing and storage resources that allow for the distribution of shared data and applications., [1].

Notable advantages of Data Centers include but are not limited, to their ability to provide services to end-users based on affordable rates in various plans as per contractual agreements. They also offer a robust hardware ecosystem as well as software. In operational terms, data centers offer a reliable and enhanced system performance by way of carefully distributing the traffic loads uniformly across the cluster nodes.

End users are excused from maintenance responsibilities. Data centers also afford instant scalability based on changing capacity demands by users. To enhance the fail-safe abilities of data canters, backup systems are incorporated, [2], [3].

A notable drawback of Data centers is the high power consumption which up both CAPEX and OPEX costs. For example, it is prohibitively costly to erect robust cooling systems for a large-scale data center. The same cooling system ought to be scalable to accommodate future expansions of the data centers in terms of new services that may require new hardware to be incorporated. Thus scalability of energy supply capacity is quite a challenge. Thus, the objective is to maximize power utilization and at the same time optimize the performance per power budget. Overall the operational costs of Data centers directly link the resource management algorithms implemented to assign virtual machines (VMs) to actual hardware servers and degrees of flexibility to relocate them elsewhere in case of emergencies usually associated with power losses of excessive heating of system elements. Load balancing applies to both physical servers and VMs. Associated algorithms can either be initiation process-based (sender or receiver-initiated) or current state-based (static or dynamic). The first type is further divided into three types, i.e. sender initiation based, [4].

Static load balancing has since been proved to be quite inefficient and hence we will only review dynamic load balancing, Various categories of dynamic and adaptive load balancing techniques have been explored. These include traditional, nature-inspired, agent-based, real-time as well as adaptive/hybrid load balancing. We define each of these as follows, [5].

Traditional load balancing: This encompasses various techniques. Examples include the breadth-first search (BFS) algorithm which operates mainly at the Fog layers. Another example of techniques in this category is the dynamic resource allocation (DRA) which provides equilibrium in load balancing among servers running at varying speeds. In short, the algorithm will prefer a faster running server when allocating tasks. In that way, load

balancing is achieved. A real-time efficient scheduling algorithm (RTES) was also proposed to balance loads by way of executing all real jobs before their time outs. In that way, the turnaround times experienced by end-users would be very minimal. The throttled algorithm is another example of traditional load balancing that assigns tasks to VMs based on the latter's capabilities and size. All active VMs are indexed as either busy or idle. When new jobs arrive an assigned manager will search the index table to determine which VMs are available and allocate the jobs to them, [4].

Nature-inspired load balancing: An example is the ant colony algorithm, which generally brings about a reduction of the makespan and thus consequently balances the load. The honey bee behavior-based load balancing (HBB-LB, external optimization (EO), and ant colony optimization (ACO) are also example techniques in this category, [5], [7], [8].

Agent-based approaches for load balancing: This operates on a client-server style basis in which a load balancer is fed with load levels of each active server via agents. The agents themselves are incorporated into each server and relay the information in real-time to the load balance. The load balancer utilizes this information when assigning workloads to servers. Example algorithms in this category include agent-based automated service composition (AASC).

Real-time load balancing: This is a category of algorithms that gear themselves towards minimizing latency and execution times. Examples include the real-time efficient (RTES) algorithm which tends to force the Fog layer ends to give priority to real-time tasks so that they can be served and completed before timeouts. Other example algorithms in this category include round-robin (RR) and first come first serve (FCFS).

Adaptive/ Hybrid balancing: These will generally incorporate one or more of the categories already discussed in a bid to achieve good load balancing, [5]

In this paper, we propose and analyze a framework for load balancing in a Fog-Cloud Datacenter. In so doing we assume that the Datacenter architecture is 3-layered,i.e it comprises the end-users, Fog, and Cloud layers. As alluded to before the Fog layer is dominated by resource-constrained terminals and as such provides a processing platform for real-time jobs only. Otherwise, the rest of the tasks are relayed to the cloud layer. Because the arrival volumes, as well as patterns, are unpredictable, it becomes necessary to address load imbalances as soon as the need arises. An example is that at each layer, should some VMs be overwhelmed, then some of the load is diverted instead to the less loaded VMs. In formulating our problem we take into cognizance that:

- The end-user layer comprises millions of IoT-enabled and other terminals that frequently generate huge volumes of data that require processing either in real or non-real-time frames.
- Nodes at both layers (Fog and Cloud) of the data center will utilize lesser power during periods of idleness in comparison to periods of activeness.

A desire thus arises for a need to design an energy-efficient framework that will drastically reduce power consumption within the data center, i.e. at all nodes in both the cloud and Fog layers.

That frequently the volumes of requests from end users may be overwhelming such that the aggregated Data Center resources also become overwhelmed hence overall QoS degrades.

The intermittency nature of request arrival together with the heterogeneity nature of the workloads thus provides a case for designing an energy-aware load balancing mechanism that will serve both layers.

Figure 1 is representative of such a framework. As can be seen from the same figure, three layers are distinguishable. These are the end-user, Fog, and cloud.

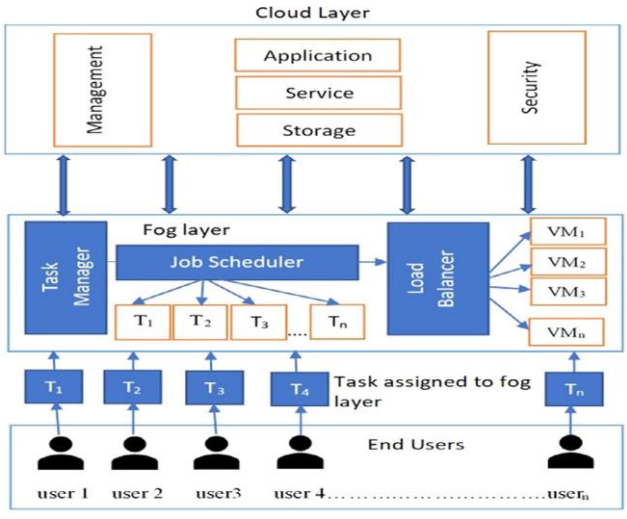

Figure 1:Load balancing framework for Fog computing.

The layers are summarily defined as follows:

End-User layer: This is the main origin of all requests. Note that an end-user request can also be in the form of requests coming from a neighboring data center. The end-user layer connects to the cloud layer via the Fog. Depending on the resource requirements as well as the time-sensitive nature of a given request from this layer, it can be processed at the Fog layer or relayed to the Cloud.

Fog Layer: The layer is mostly defined by Fog nodes designed to receive, and authenticate, requests originating from the user-end layer. Depending on the resource and QoS requirements, this layer may immediately process and post back the results or may refer the request(s) to the upper (cloud) layer. It is at this same layer that scientific workflow tasks are mapped directly to its nodes. The incorporated task manager module coordinates the handling of tasks based on a priority basis. Likewise, the scheduling will be implemented on a priority basis by the Scheduler module. The scheduler will help to set the priority of execution of the tasks. To prevent some VMs from being overwhelmed whilst others are idling, a load balancer carefully manages the process by routinely evenly spreading the newly arriving jobs (tasks). This also consequently reduces both power consumption as well as CAPEX and OPEX.

Cloud layer: The layer is equipped with high computing and storage capabilities. All data requiring long term storage may utilise this layer. In addition services such as PaaS. SaaS and IaaS together with security are provisioned at this same layer.

2. Data Center Model Overview

We assume the Data center is distributed and hence has many interconnected processing nodes (PNs), such that the entire Fog-Cloud Data center system is defined by:

$$PN = \langle PN_1, PN_2, \ldots . PN_n \rangle \quad (1)$$

Likewise, each PN comprises several parallel serving VMs ;

$$VM = \langle VM_1, VM_2, \ldots VM_n \rangle \quad (2)$$

The user layer is an originator of so many requests (Rs), i.e defined by;

$$R = \langle R_1, R_2, \ldots . R_y \rangle \quad (3)$$

The aggregate number of requests within a given measured interval would be;

$$R_{aggregate} = \sum_{i=0}^{y} R_y \quad (4)$$

To sustain the workload, the associated costs are determined by summing the cost of each VM , hence we have;

$$C_{total} = C_{\cos t_{VM}} + C_{\cos t_{DT}} + C_{\cos t_{storage}} \quad (5)$$

where in the last equation;

$C_{\cos t_{VM}}$ -is the cost of running VMs. . Note that the physical servers are also incorporated as the earlier is implemented in a physical server.

$C_{\cos t_{DM}}$ -costs associated with data transfer. This can be locally or between the Fog and cloud layers.

$C_{\cos s_{storage}}$- data storage-related costs.

The aggregate workload (WL) is determined from;

$$WL_{total} = \sum_{i=0}^{n} r * \left\langle \sum T_{VM} \right\rangle^{*} f \quad (6)$$

where, T_{VM} is the duration of task completion per VM at frequency (f).

The aggregated power consumption is approximated from;

$$E_{k_{flow}} = E_{b_{flow\_}} N_{bit_k} \quad (7)$$

In which case the variables contained in the last equation are defined as follows;

$E_{b_{flow\_}}$ -is the network's energy consumption per bit.

N_{bit_k} -denotes the total number of bits exchanged among the k services.

A summary of the load balancing approach (algorithm) at both the Fog and Could layer VMs is summarized in Figure 2. The algorithm overall ensures that all the resources at both layers are efficiently, rationally, and maximally utilized. In that way, maximum energy savings is achieved. As can be observed from the same figure, it is noted that as far as the workload assignment is concerned, nodes are initially analyzed for their readiness in terms of resource availability and power consumption. Given that not all nodes are active at the same time, then the algorithm will

avoid any inactive ones if one or more of the currently active ones can. In that way, both power and running costs are optimized.

.

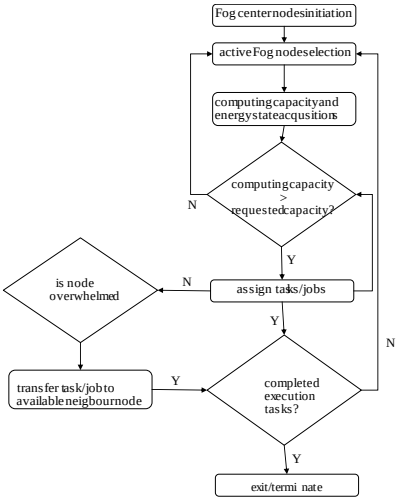

Figure 2. Load balancing summarised algorithm

3 Capacity and Performance Metrics

VM Capacity

For each active VM, its capacity versus its rated workload are also compared. We assume that there are three threshold levels of caparisons, high (WL_H), moderate (WL_M), and low(WL_L). If we assume the number of workloads served to a single VM to be $P_{WL_{number}}$ and $P_{WL_{speed}}$ is the number of executable instructions (in millions). per second, then the VM capacity is ultimately determined from;

$$C_{VM} = P_{WL_{number}} \times P_{WL_{speed}} \tag{8}$$

By further assuming that independence of the workloads; i.e they are independent of each other are independent;

$$\langle WL_1, WL_2, ...WL_n \rangle \tag{9}$$

The VM load is expressed as;

$$VM_{loading} = \frac{\sum_{j=1}^{n} WLL_j}{n} \tag{10}$$

where L_j is the workload duration (length).

The VM loadings can thus be compared in phases as follows;

Case I :

$$VM_{loading} < C_{VM} \times WLL \tag{11}$$

The above infers that the VM is underloaded and thus can accept more workloads.

Case II :

$$VM_{loading} > C_{VM} \div TL_L \quad \&\& \quad VM_{loading} \le C_{VM} \times WL_L \tag{12}$$

In the case the VM is running in optimum state.

Case III :

$$VM_{loading} > C_{VM} \div WL_L \quad \&\& \quad VM_{loading} \le C_{VM} \times WLH \tag{13}$$

This is a fairly highly balanced state.

Case IV :

$$VM_{loading} > C_{VM} \times WL_H \tag{14}$$

This is a state signaling that the VM is overloaded. We can also compute the expected workload completion time.

$$E[T] = \frac{WLL}{C_{VM}} \tag{15}$$

The above metric is also referred to as the execution time. Consequently, the estimated completion time is approximated according to the following equation;

$$ECT = E[T] + VM_{loading} \tag{16}$$

Overall, we conclude that for a VM to be selected, the following equation must be satisfied;

$$VM_{loading} = VM_{loading} + WLL \tag{17}$$

Performance Metrics

This set of metrics discussed in this section directly dictates the overall performance of the Fog-Cloud data center.

Makespan: This is the total duration (in seconds) required to execute to completion of all workloads in the VM's existing queue;

$$Makespan = \max\langle CT_i \rangle \quad i \in VMs \tag{18}$$

Average Makespan: Note that the makespan frequently fluctuates such that taking its overall average value would be more practical. This is determined from;

$$\overline{Makespan}[Av] = \frac{\sum_{k=1}^{m} Makespan}{m} \quad (19)$$

where m denotes the number of averaged values.

Response times(ART): This is the aggregate time to make the necessary searches as well as execute the workload.

$$RT_i = FT_i - SB_i \quad (20)$$

In the above equation FT_i is the completion (finish) time of a given workload and SB_i is its actual submission time to the VM .

The value tends to fluctuate hence Its overall average can be determined;

$$E[RT] \div \frac{\sum_{i=1}^{m} E[RT_i]}{m} \quad (21)$$

Execution Times: This is the time-lapse between workload completion time to its executing start time (EST).

$$ET_i = FT_i - EST_i \quad (22)$$

4 Analysis and Results

The proposed framework's algorithm provided in Figure 3 is implemented in Cloud Sim, [8], [9], [10], [11].

initialise
set n( )
set m( )
setting VM set;
{
$VM = [VM_1, VM_2, ...VM_n]$
initialising workload : $WL = \langle wl, wl_2, ...wl_m \rangle$
while
{
VM = active but not maximally loaded
do
assign work load from list
do
VM = active but not loaded
shift wl to VM_j
next state= desired goal_state?? = YES
return
end if
end while
end while
}

Figure 3: Load balancing algorithm code

In our evaluation, we make use of two types of VMs each with RAM size of 4 G, HDD size of 500 G and supported by a bandwidth of up to 1 GBps. For the various workloads, we assume that the lengths are variable. The file sizes are however fixed at 500 MB. Energy consumption figures for the two types of servers are provided in Table 1.

Table 1

server	*idle*	25%	45%	75%	95%
I	88	93.4	98	105.7	115.6
II	95.7	100.3	109.3	119.4	131.5

We also provide additional simulation parameters in Table 2 . Note that in this case, we set our Fog-Cloud Data Center to be distributed in nature, i.e it is located in 5 different places.

Table 2

center	*user* reques rate (per/hr)	*peak times*	*average peak* users	*off – peak users*
A	300	5-8 am	1500	150
B	1500	7-11 am	1500	150
C	2500	6-11 am	1500	150
D	370	5-10 am	1500	150
E	555	7-9 am	1500	150

We further evaluate the overall performance of the Fog-Cloud data center, in this case, we take into consideration that a significant proportion of the user-generated workloads are ultimately processed at the Cloud layer. We thus make a brief performance evaluation comparison of our framework's algorithm with other equivalent algorithms, such as the Max-Min, SJF and Round Robin[, 12].

Table 3: Aggregated execution times

Trial	$makespan[Av]$	$E[RT]$	AET
150 Jobsin 10 VMs	371	282	**31 352**
200 Jobs in 13 VMs	435	305	**51 325**
250 Jobs in 16 VMs	441	311	**71 346**
300 Jobs in 18 VMs	456	319	**100 324**
350 Jobs in 21 VMs	**463**	**331**	**121 897**

Table 3 provides the average makespan, response time and average execution times of the proposed framework's algorithm. As expected, as offered workloads increase so would be these parameters. Figure 4 demonstrates that the algorithm comparably reduces the makespan.

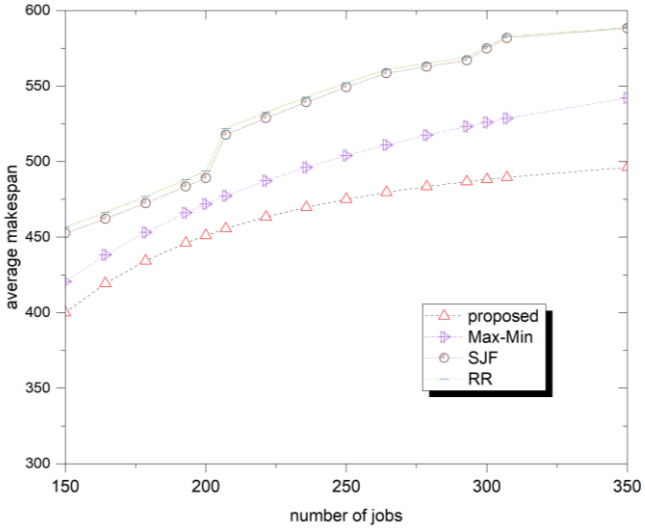

Figure 4: Average makespan.

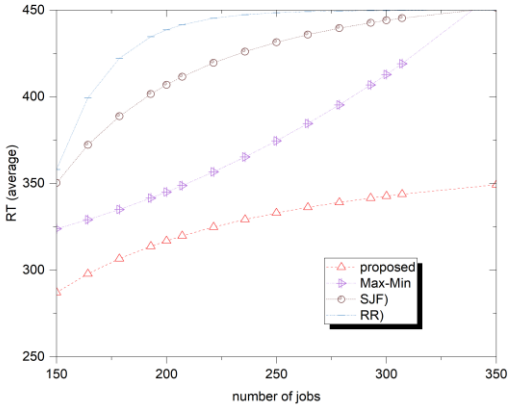

Figure 5: Average response times

Similarly, it is observed from Figure 5 that the algorithm significantly reduces the response times in comparison with the other algorithms.

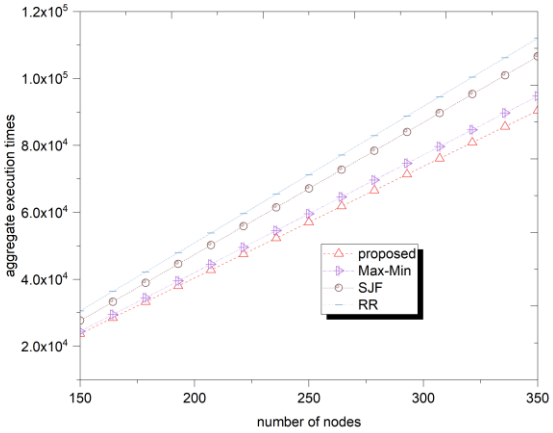

Figure 6: Aggregated execution times

Finally, we also compared the total execution times As can be observed from Figure 6, as the number of nodes increases, so will the overall aggregated execution times.

CONCLUSION

A load-balancing framework together with a load-balancing algorithm proposed and analyzed in respect of the key QoS metrics that would affect both t users' satisfaction as well as overall energy efficiency A comparative performance evaluation of the algorithm was carried out at Fog as well as at Fog-Cloud data center levels. Overall it is concluded that load balancing at VMs level coupled with sound scheduling are key to achieving energy efficiency. The choice for a Fog-Cloud paradigm is that the Fog servers are normally placed in proximity to end-users and as such QoS related issues such as latencies for critical mission services and other applications can easily be overcome. We also note that the key to achieving energy efficiency in its operations would be sound load balancing among the active servers as well as appropriate scheduling. The latter is necessary to ensure that we maximize the completion of jobs in the shortest time possible using the minimum possible resources. In that way, much less power will be consumed.

REFERENCES

[1] M. Y. uddin and S. Ahmad, "A Review on Edge to Cloud: Paradigm Shift from Large Data Centers to Small Centers of Data Everywhere," 2020 International Conference on Inventive Computation Technologies

(ICICT), 2020, pp. 318-322, doi: 10.1109/ICICT48043.2020.9112457.

[2] C. Lee and A. Fumagalli, "Internet of Things Security - Multilayered Method For End to End Data Communications Over Cellular Networks," 2019 IEEE 5th World Forum on Internet of Things (WF-IoT), Limerick, Ireland, 2019, pp. 24-28, doi: 10.1109/WF-IoT.2019.8767227.

[3] M. Y. uddin and S. Ahmad, "A Review on Edge to Cloud: Paradigm Shift from Large Data Centers to Small Centers of Data Everywhere," 2020 International Conference on Inventive Computation Technologies (ICICT), 2020, pp. 318-322, doi: 10.1109/ICICT48043.2020.9112457.

[4] Y. Zhao, W. Zhang, M. Yang and H. Shi, "Network Resource Scheduling For Cloud/Edge Data Centers," 2020 IEEE 39th International Performance Computing and Communications Conference (IPCCC), 2020, pp. 1-4, doi: 10.1109/IPCCC50635.2020.9391529.

[5] P. Geetha and C. R. R. Robin, "A comparative-study of load-cloud balancing algorithms in cloud environments," 2017 International Conference on Energy, Communication, Data Analytics and Soft Computing (ICECDS), 2017, pp. 806-810, doi: 10.1109/ICECDS.2017.8389549.

[6] A. Agarwal, G. Manisha, R. N. Milind and S. S. Shylaja, "Performance analysis of cloud based load balancing techniques," 2014 International Conference on Parallel, Distributed and Grid Computing, 2014, pp. 49-52, doi: 10.1109/PDGC.2014.7030714.

[7] R. R. Kumar, S. K. Jha, D. Garg and S. Vaishnav, "Evaluation of Load Balancing Algorithm Using Cloudsim," 2018 3rd International Conference on Inventive Computation Technologies (ICICT), 2018, pp. 78-81, doi: 10.1109/ICICT43934.2018.9034367.

[8] H. Xu, G. Wang, L. Luo and M. Lei, "The Design of Reliability Simulation of Cloud System in the Cloudsim," 2018 15th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP), 2018, pp. 215-219, doi: 10.1109/ICCWAMTIP.2018.8632572.

[9] P. Humane and J. N. Varshapriya, "Simulation of cloud infrastructure using CloudSim simulator: A practical approach for researchers," 2015 International Conference on Smart Technologies and Management for Computing, Communication, Controls, Energy and Materials (ICSTM), 2015, pp. 207-211, doi: 10.1109/ICSTM.2015.7225415.

[10] S. Santra and K. Mali, "A new approach to survey on load balancing in VM in cloud computing: Using CloudSim," 2015 International Conference on Computer, Communication and Control (IC4), 2015, pp. 1-5, doi: 10.1109/IC4.2015.7375671.

[11] M. Haghi Kashani and E. Mahdipour, "Load Balancing Algorithms in Fog Computing: A Systematic Review," in IEEE Transactions on Services Computing, doi: 10.1109/TSC.2022.3174475.

[12] J. Yan, J. Wu, Y. Wu, L. Chen and S. Liu, "Task Offloading Algorithms for Novel Load Balancing in Homogeneous Fog Network," 2021 IEEE 24th International Conference on Computer Supported Cooperative Work in Design (CSCWD), 2021, pp. 79-84, doi: 10.1109/CSCWD49262.2021.9437748.