# Energy and QoS Optimization in Fog/Edge Computing Via Brownout

MSc Research Project
Programme Name

Nandhini Venkatesan
Student ID: 20236158

School of Computing
National College of Ireland

Supervisor: Sean Heeney

**National College of Ireland**
**Project Submission Sheet**
**School of Computing**

| | |
|---|---|
| **Student Name:** | Nandhini Venkatesan |
| **Student ID:** | 20236158 |
| **Programme:** | Programme Name |
| **Year:** | 2022 |
| **Module:** | MSc Research Project |
| **Supervisor:** | Sean |
| **Submission Due Date:** | 15/08/2022 |
| **Project Title:** | Energy and QoS Optimization in Fog/Edge Computing Via Brownout |
| **Word Count:** | 5900 approx |
| **Page Count:** | 18 |

| | |
|---|---|
| **Signature:** | Nandhini Venkatesan |
| **Date:** | 18th September 2022 |

# Energy and QoS Optimization in Fog/Edge Computing Via Brownout

Nandhini Venkatesan
20236158

**Abstract**

Cloud computing is used to store and handle the enormous and diverse amounts of data produced by the Internet of Things (IoT). However, there are other difficulties with the cloud, such as an increase in latency and reaction time for real-time applications. Additionally, this can lead to a rise in energy use and resource waste. In order to address this issue, a new paradigm for cloud computing called fog/edge computing has been created. Despite offering a solution to the problems described, this paradigm for real-time applications still requires the development of resource-aware and energy-efficient techniques due to the limited computational power available in fog nodes and the battery life of edge devices. This study proposed a controller for container-based micro-service application in cloud-fog environment based on brownout method, and evaluated energy consumption, SLA, Execution .The experiments shows that the SLA violation and energy consumption has positive impact using the brownout approach but not the Execution time.

## 1 Introduction

Cloud computing has been widely used in many sectors, but the rise of IoT has created a number of issues for it. Cloud computing is faced with difficulties due to high-latency and network congestion for applications that are delay-sensitive and location-aware due to the massive volume of data being generated by IoT devices. These issues ultimately result in high energy use and resource waste. Cisco introduced fog computing in 2012, Bonomi, F. et al. (2012) and Luo, J. et al. (2019) characterized it as a solution to these problems. Fog/edge computing, a development of cloud computing, was created to solve the limitations of cloud computing. Fog computing will move storage and processing from the network core to the network edge in order to support real-time applications and reduce the network burden on data centers (Vaquero and Rodero-Merino, 2014).

### 1.1 Motivation

To address issues with high latency, reliability and security, high performance, mobility, and interoperability, fog/edge computing architecture adds an additional resource-rich layer between end devices and the cloud. This layer consists of a significant number of nodes, some of which may be routers and switches. Computerized fog is made up of resources that have a dynamic character and are unknown, which causes the primary problem with not using them effectively, which results in high energy usage. Even so, some works have been conducted, this topic still need further investigation to be done.

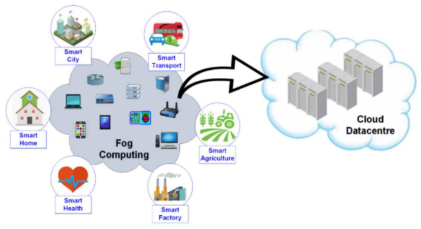


Figure 1: Interactions between IoT systems, Fog and Cloud computing  Mahmud and Buyya (2019).

Additionally, the high energy consumption of fog-assisted cloud systems has significantly raised operational costs, which has had a negative impact on the environment. The cloud data centers have raised power bills and carbon emissions, Mishra, B. S. P. et al. (eds.) (2018). Additionally, as more and more apps move to the cloud, the demand for energy keeps increasing, making it hard to satisfy. Therefore, if a practical power management strategy is created, where a lot of analytical activities may be carried out on fog/edge nodes that are closer to the data source instead of in the cloud, could provide a solution to the escalating energy needs in cloud data centers, Varghese, B. et al. (2016) Although some study has been done in the area of energy-efficient fog/edge computing, more work has to be done in this area. This will reduce the amount of energy used in cloud data centers since fog and cloud should cooperate to gain the most performance from the system.

Fog/edge computing should take QoS factors like latency into account while managing energy. Because delay-sensitive applications may experience excessive latency when resources for fog/edge computing are used to optimize energy efficiency. Fog/edge computing resources are known to have limited compute capability. Therefore, when building the strategy, there should be a balance between energy management and QoS in terms of delay.

Brownout, a novel methodology that allows optional components or micro-services to be temporarily activated or deactivated. It has been used in cloud computing systems for a variety of optimizing objectives, including load balancing, energy efficiency, latency awareness, and cost awareness. Additionally, as containers provide the ability to instantly start and stop the services, this strategy may be combined with them. Therefore, brownout with cloud-based containers might enhance scheduling performance (Xu and Buyya, 2020). However, in fog/edge computing, this technique has to be researched for a number of system optimizations.

## 1.2 Research Question

**Can energy consumption, SLA and Execution time in fog/edge computing be improved by efficient resource utilization using brownout paradigm and simulated with iFogSim.**

The main advantage of fog computing is to provide the device with the best possible service while being quick and effective. However, there is extreme energy consumption in fog when enormous number of Internet-of-Things are used, and this is one of the major issues in Fog environment which in turn the service's performance will be affected. So, in this research tried to solve this issue using brownout approach.

## 1.3 Research Objectives and Contributions

In this study, a cloud-fog/edge computing environment with a container-based micro-service application has been taken into consideration. The controller has been developed using a brownout approach in which the selection policy has been passed in order to deactivate the containers. The newly developed controller has been placed in fog layer, and the simulated this environment using iFogSim. The simulation results are used to measure energy consumption, latency, network usage and CPU utilization metrics.

The rest of the paper is organized as follows: Section 2: Literature review begins with the importance of reducing the energy consumption in fog/edge computing of cloud-fog/edge framework followed by related works has been carried out in follwoin1) Different Energy optimization approaches;2) Works Related to Containers in cloud/fog/edge; 3) Brownout related works in cloud/fog/edge. Section 3: Research methodology, Section 4: Design specification, Section 5: Implementation, Section 6: Evaluation, Section 7: Conclusion and Future Work

# 2 Related Work

The drawbacks of traditional cloud computing, such as latency, QoS, and energy consumption, are addressed by fog/edge computing, which can manage the diverse and massive amounts of data produced by the expanding IoT edge devices. Similar to cloud data centers, the energy usage of fog/edge data centers rises, having a negative impact on the environment worldwide. Numerous studies have been done to address the energy issue in cloud data centers, but only few have addressed fog/edge data centers.So, in this section , discussed about the existing works related to the research will see the works which has been carried out in

## 2.1 Different Energy optimization approaches in cloud/fog/edge computing:

Baccarelli et al. (2017),depending on the minimal hop count, a dynamic cluster policy technique for task scheduling in the fog layer has been presented. The function is run either in the fog gateway or device, depending on the outcome, to decrease energy consumption and reaction time. The "Dynamic Cluster Method" described has been compared to various existing algorithms, including resource-aware and latency-aware algorithms, and the results reveal that the proposed algorithm outperformed them both.

Kim et al. (2019), to utilize both both idle servers and free cores of the CPU in the server, novel DTM (Dynamic Thermal Management) technique for heterogeneous data centers was proposed.By recognizing the memory intensity and virtual machine utilization, this technique migrates the VMs among CPU cores to keep the server temperature stable when the temperature of the CPU core in the servers exceeds a pre-defined thermal threshold, which ultimately saves energy and enhances performance. When temperature fluctuations have an influence, however, SLA is not taken into account.

In Ilager et al. (2019), the overall energy consumption has been reduced byan ETAS(Energy and Thermal-Aware Scheduling)Algorithm where VMs were consolidated dynamically, and prevented hotspots proactively.In Yang et al. (2016), to improve response time and limit energy usage, a game-theoretic framework was provided along with related algorithms. The model outperformed NPA, DVFS, and DCP schemes in comparison, it has been noted. Xiao et al. (2015), by turning off the actual machine, evolutionary game theory has been utilized among virtual machines to reduce power usage. This strategy outperformed the First Fit, the Best Fit Increasing, the Best Fit Decreasing, the Greedy, and the Load Balance strategies in tests. When PMs are repeatedly turned on and off, there will be an increase in power consumption and a decline in performance.

In Pierson et al. (2020), for spatiotemporal thermal awareness scheduling,MILP (Mixed Integer Linear Programming) formulations has been suggested by taking heat generation and dissipation dynamics into account.The efficiency of cloud data centers is increased by maximizing energy and processing speed. The impact of the quantity of VM migrations on system performance and SLA, however, is not investigated in this study.

In Gai et al. (2021),for time-constrained applications operating in fog computing systems, a unique technique called EFRO (Energy-aware Fog Resource Optimization) model has been presented. This model created a heuristic approach to reduce the cost of energy and the amount of time used. The hill-climbing process used by this model to create near-optimal resource allocation solutions led to a decrease in energy usage and scheduling time without breaking the time restriction. Additionally, this model improves the MESF and RR schemes' energy efficiency.

In order to reduce energy consumption with little delay and energy when computation activities are completed, an energy-efficient computation offloading technique was created in fog computing, Chen, Zheng, Wang and Lu (2019). The authors also suggested an accelerated gradient method to deal with the issue of energy reduction and to arrive at the best offloading choice. The suggested algorithm has been shown to be speedier than the conventional algorithm.

To lower the overall energy consumption of the cloud-fog network, the SMETO (stable matching algorithm for energy-minimized task offloading) model with two algorithms, EEDA and EMTA, was presented, Zu et al. (2019). By allocating the subtasks to helpers based on the result matching made by the EEDA algorithm, the EMTA algorithm lowers the network's energy usage.

Wu and Lee (2018), proposed ILP(Integer Linear Programming model),which is used to get the factors that can minimize the energy consumption in a distributed fog system.And, developed an algorithm energy minimization scheduling (EMS) based on the observations from ILP model.This algorithm has combination of different policies which are used to achieve optimal scheduling on heterogeneous fog computing architectures to reduce the energy consumption for IoT workflows.However, it has some shortcomings such as :(i)this algorithm does not consider the tasks which needs to be completed in particular time; (ii)different types of computing nodes are not considered; (iii)it is not validated with larger size of workload to find the efficiency of the algorithm.

Mishra et al. (2018), proposed a metaheuristic-based service allocation framework to solve the NP-hard problem in fog computing system.The framework used three techniques of metaheuristic such as PSO(particle swarm optimization), binary PSO, and bat algorithm in order to deal with the resources which are heterogeneous in fog computing environment.And, this metaheuristic techniques are used to achieve desired QoS, and also to achieve energy efficiency for the service allocation.

| Reference Paper | Method/Approach/Algorithm | Objective | Limitations |
|---|---|---|---|
| Baccarelli et al. (2017) | Dynamic Cluster Algorithm | Dynamic cluster policy for scheduling the task in fog layer to reduce the energy consumption and response time based on the minimum hop count | N/A |
| Kim et al. (2019) | DTM | Saves energy and improves performance in cloud by VM migration | SLA is not considered |
| Ilager et al. (2019) | ETAS algorithm | Consolidate VMs to reduce energy consumption in cloud | N/A |
| Yang et al. (2016) | Game-theoretic framework | Reduces energy consumption and response time | N/A |
| Gai et al. (2021) | EFRO | For time-constrained applications in fog, resources are optimized and measured in terms of energy consumption ,time consumption | N/A |
| Zu et al. (2019) | SMETO | Energy consumption reduced in cloud-fog network | N/A |
| . Wu and Lee (2018) | ILP;EMS | For distributed fog system , energy consumption reduced | Time-sensitive tasks are not considered and heavy workloads not considered |
| . Mishra et al. (2018) | Metaheuristic-based service allocation framework for fog | Energy consumption reduction, meets QoS | N/A |
| **THIS ONE** | Brownout-based Framework | To Improve the utilization and measure in terms of energy consumption, latency | |

Table 1: Summary of Literature Review for Different Energy optimization approaches

## 2.2 Containers in Cloud/Fog Computing Survey

Zhang et al. (2017), presented a linear programming methodology for container deployment to computational nodes. In this model, several optimization factors, including network cost and energy usage, have been taken into account. When comparing the experimental model results to the Docker Swarm binpack technique, there was a cost decrease of about 45%

Wan et al. (2018), used ILP formulation to provide a framework to reduce the cost of deployment in a Docker container environment for micro-service-based applications. This method has been used to handle the scheduling problem in a distributed way while also taking into account the cost of the network. However, this method has resulted in increased computing complexity, and other goals like saving energy or speeding up reaction times were not taken into account.

Mendes et al. (2019),presented an improved Docker Swarm scheduling technique to improve the energy efficiency of Docker Swarm in a fog/edge computing scenario. Since the available resources in this environment are few, an oversubscription method was implemented to reduce energy consumption and improve resource utilization. The findings of the suggested solution demonstrated that resources had been used effectively, which resulted in a decrease in energy usage. When compared to Docker Swarm techniques, the scheduler took longer to generate the solution since the decision-making process was more complicated.

Chen, Zhou and Shi (2019), suggested a method for container scheduling to increase the energy efficiency of edge computing systems based on the Min-Min heuristic. The authors have changed the original Min-Min method such that less energy is used while assigning containers on computer nodes. When the experimental results of this method are compared to the First-Fit container placement strategy, there has been an energy consumption decrease of roughly 56%.

Kaur et al. (2019), with the aim of reducing carbon footprint rate by improving performance and energy-efficiency, KEIDS (Kubernetes-based energy and interference driven scheduler), a Kubernetes-based controller for managing containers on edge clouds has been introduced. The authors developed a MOOP to efficiently map and schedule containers to available nodes, and its performance was compared with that of other schedulers (FCFS) using real-time Google traces. The outcomes demonstrated that the suggested model performed better in terms of lowering the carbon footprint rate, and interference from containers that co-existed on the same or separate nodes was reduced.

Medel et al. (2018),to study Kubernetes' performance, a Petri net-based performance model has been employed, and the results showed that the overheads associated with resource deployment, termination, and maintenance had a major influence on both Kubernetes' performance and that of the underlying cloud environment. The problem has been given consideration for the optimization of energy consumption and makespan. This has demonstrated that the amount of energy used by resources in a cloud/edge environment may be reduced by using a limited number of nodes for application processing and shutting off the rest.

## 2.3 Brownout in Cloud/Fog Computing Survey

In Klein et al. (2014), optional code has been turned on or off based on control theory, a brownout model for cloud applications. To demonstrate how this paradigm may be simply integrated with existing applications, it has been tested with two well-known

| Reference Paper | Objectives | Containertechnology | Limitations |
|---|---|---|---|
| Zhang et al. (2017) | Cost and Network | Docker | Resources are same type |
| Wan et al. (2018) | Utilization, Cost, Network | Docker | Complex computation |
| Mendes et al. (2019) | Energy,Utilization | Docker-Fog/Edge computing | Decion process is more complex |
| Chen, Zhou and Shi (2019) | Energy | Use min-min algorithm for container scheduling in edge computing | Optimize single objective |
| Kaur et al. (2019) | Energy efficient scheduler KEIDS | | N/A |
| . Medel et al. (2018) | Model to analyze performance of Kubernetes;Energy utilization of resources has been reduced | | |

Table 2: Summary of Literature Review of Containers Related works in Cloud/Fog/edge

benchmark cloud apps, RUBiS and RUBBo. The outcomes of the experiment indicate that this method enables the apps to function more efficiently or with a greater number of users. This method does not require extra capacity since it manages unforeseen loads or breakdowns, which strengthens the applications.

Xu and Buyya (2017), presented the brownout-based system concept, in which auxiliary apps or microservices are momentarily turned off. Authors also suggested a method, known as BMDP, which was based on brownout and an approximation of the Markov Decision Process and used to choose which components needed to be deactivated. When the suggested method (BMDP) was implemented, experimental findings from simulations on actual trace showed that the energy consumption had been decreased to 20 percent compared to non-brownout ones and that the discount amount had also been saved compared to baselines of brownout.

In Xu et al. (2018), an integrated strategy for container-based clouds to handle brownouts and energy has been developed and this strategy contains many policies so that the system's performance can be assessed based on identifying the appropriate containers to deactivate. The method has been evaluated in actual test environments, and results have proven that it performs better than baselines in terms of energy usage, reaction time, and SLA breaches.

Xu and Buyya (2019b), created a software system paradigm for energy-efficient clouds called BrownoutCon, which is built on containers and brownout. By using brownout-based algorithms to evaluate the suggested model on Grid'5000 infrastructure, it was discovered that both energy usage and QoS were improved. The suggested approach, however, does not account for network congestion.

Chen and Zhou (2019), presented a container selection policy based on resource usage equilibrium, brownout and penalty, and both. When the edge data center is no longer overcrowded, the containers are deactivated depending on their resource consumption and penalty ratios, with the lesser ratios being removed first.The Xu and Buyya (2019b) and Chen and Zhou (2019) brownout approach has been used in both , however, in Chen and Zhou (2019) there was no thought given to reducing the data center's overload time.

In the aim of reducing the energy usage of cloud data centers, Xu et al. (2016),brownout-based methodology has been suggested .This design enables or disables optional application components dynamically to save energy consumption, and algorithms are utilized to determine which containers should be eliminated. The trial's results showed a 20% decrease in energy use.

## 2.4 Research Niche

Optimizing Energy Consumption and improving Quality of Service in fog has been achieved in different ways as mentioned above in the related works.Many algorithms and techniques have been suggested by many authors in both cloud and Fog . The algorithms and techniques were related to reducing energy consumption , reducing response time, and low latency has been achieved as summarized in Table 1.Though some works are done in Fog related to reducing energy consumption, still there is a scope to do research without affecting QoS such as latency,response time, network usage and execution cost.So the proposed research aimed to reduce energy consumption in fog computing by maintaining QoS.

# 3 Methodology

This paper aimed to reduce the energy consumption in fog computing for the container-based microservice applications in cloud-fog/edge system using brownout paradigm .Also, to measure other metrics such as CPU utilization, Network usage, Response Time, Execution cost.

To tackle the challenges presented by the integration of cloud and IoT, fog/edge computing, which extends cloud computing, has been created. Fog/edge computing, however, has limited storage capacity and resource availability. It is crucial to make optimum use of these resources in order to reduce the energy consumption of fog devices, which in turn reduces the energy consumption of cloud servers.

## 3.1 Brownout Paradigm

A self-adaptive paradigm called brownout uses optional components, services, or containers to enable or disable depending on the system's unanticipated demands. The components that are optional can be turned off to better use the available resources. In order to enable adaptivity and prevent overloads, the brownout paradigm has been employed in cloud computing systems with shifting workloads. This makes the applications more responsive. 2 displays the many strategies Brownout has used in cloud computing.

The stages of the Brownout paradigm, which was developed from MAPE-K feedback loop control, are shown in 3.

- **Knowledge Module**:The system's abstract level has been gathered by collecting the key features and status information. Based on the information that has been

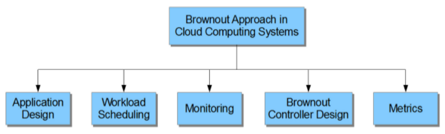

Brownout Approach in Cloud Computing Systems
Application Design
Workload Scheduling
Monitoring
Brownout Controller Design
Metrics

Figure 2: Brownout Approach- Cloud computing system from Xu and Buyya (2019a).

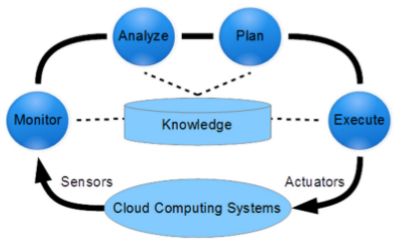


Figure 3: Model of Brownout Paradigm from Xu and Buyya (2019a).

recorded, the desired adjustments are activated.

- **Monitor module**:Continuous system status monitoring and data collection from the sensors are performed. Then, these data are submitted to the Analyze module for additional analysis.
- **Analyze module**:Data gathered from the Monitor module will be provided to the Plan module for reference, and other kinds of analysis will also be used.
- **Planning module**:This module develops a system modification strategy in order to be adaptable to changes in workload.
- **Execution module**:Plans will be put into action, and their primary goal will be to make sure the system requirements are met. Actuator also tracks recent modifications.

Based on the above model from paradigm, the proposed frame work has the Fog-Brownout Controller developed has mentioned in the 2.

## 3.2 Eclipse IDE

Eclipse IDE, which is one of the most popular Java Integrated Development where the iFogSim has been executed as this supports Java and so this IDE has been used in this research for creating classes for the framework.

## 3.3 iFogSim

iFogSim is used to simulate the proposed environment by building virtual environments. In cloudsim, there is a simple simulation tool that enables them to talk to one another. It is used to simulate and model the internet of things, cloud, and fog computing networks. With the help of the simulation capabilities of Cloud Sim,Mahmud and Buyya (2019).

There are five main classes in iFogSim: application, actuator, sensor, tuple, and fog-device. The other emulated services in iFogSim are the resource management service and the power monitoring service. IoT-related simulations of the size that are envisioned are supported by iFogSim. The following are the essential iFogSim elements needed for the

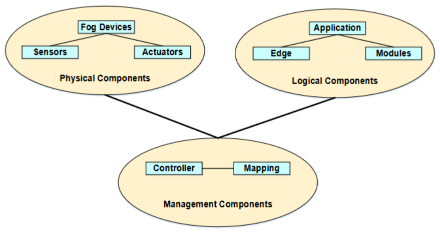


Figure 4: High-level view of interactions among iFogSim components from  Mahmud and Buyya (2019).

proposed work:

- **Fog Device**: May comprises of up-link and down- link bandwidths,processors, storage which represents the communication capacity of fog devices
- **Sensor**: It serves as an Internet of Things sensor that recognizes the tuple arrival rate and describes the sensor's output properties.
- **Actuator**: It shows how to do a certain action when a tuple from the application class arrives.
- **Tuple**: This is main component to do the communication within fog levels.
- **Application**: For scheduling or placement of the application modules in the fog device.

The aim of this research is that optimizing energy consumption in Fog by maintaining the QoS using brownout approach.So to implement the goal in the research , I have used the Brownout Paradigm in Fog computing which has been simulated using iFogSim.After simulation, energy consumption has been measured with various no.of containers.

## 4 Design Specification

One of the major problems in Fog computing is that energy consumption which affects the QoS.In order to maintain QoS , it is necessary that to optimize the energy consumption which in turn reduce the energy consumption in cloud, and by using Brownout Approach in Fog Computing the framework by referring Xu and Buyya (2019b)

### 4.1 Proposed Architecture

The proposed framework has the following major components:

- **IoT-Enabled Devices**:These devices can be autonomous vehicles, smart homes, smart phones etc.

| Parameters | Specifications |
|---|---|
| System Architecture | x86 |
| Operating System | Linux |
| Virtual Machine | Xen |

Table 3: Host(Cloud) Specifications

- **Brownout Controller**: The controller has the ability to disable optional containers if the holistic fog data center is overburdened, and it manages the application These containers are classified as mandatory or optional in order to handle containers with brownout as mentioned in Figure 5. The mandatory container is always operating, and the Optional container is activated or deactivated based on status of the system.
- **Fog-Data Center**:Resources are provided by these to fulfill the application request.

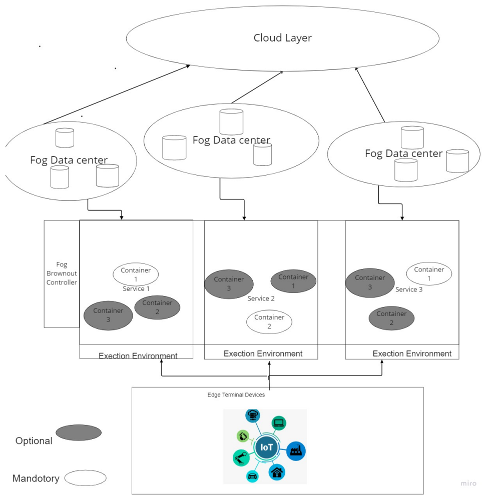


Figure 5: Proposed Architecture Overview

## 4.2 Flow of Proposed Model

Figure6 shows the flow of the proposed model.In this model, the users' requests from IoT-enabled devices are dispatched to containers and hosts.Then, information about the energy and utilization are collected from hosts with the help of system monitors.These information are sent to FogBrownout Controller which in turn refers the utilization models or power consumption models of the hosts to calculate the energy / utilization needs to be reduced.At the end the FogBrownout Controller deactivates the containers based on selection policies .

The system architecture of test program which calls the simulation has the following configuration:

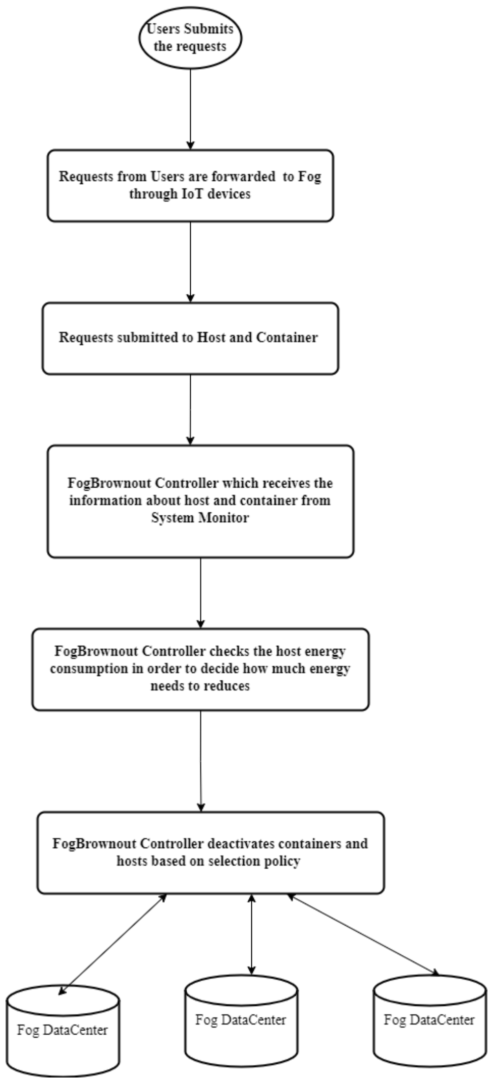

Users Submits the requests
Requests from Users are forwarded to Fog through IoT devices
Requests submitted to Host and Container
FogBrownout Controller which receives the information about host and container from System Monitor
FogBrownout Controller checks the host energy consumption in order to decide how much energy needs to reduces
FogBrownout Controller deactivates containers and hosts based on selection policy
Fog DataCenter
Fog DataCenter
Fog DataCenter

Figure 6: Flow of Proposed Model

| Parameters | Specifications |
|---|---|
| RAM | 2048 MB |
| MIPS | 1000 |
| Bandwidth | 10000 |
| Storage | 1000000 GB |

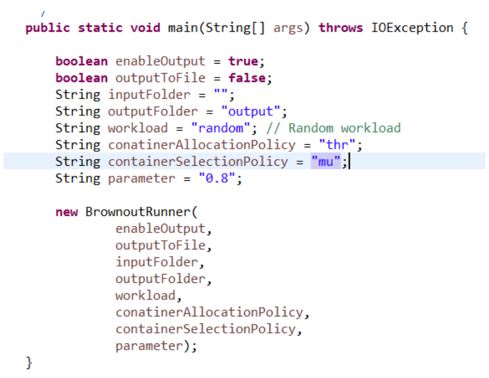
Table 4: Fog Specifications

# 5 Implementation

Since the setting up a real-time fog environment requires more time and cost, the proposed model has been developed in Simulator tool 'iFogSim'. iFogSim is suitable for simulating container-based applications of Fog Computing.The proposed model has been extended from the [1] and also from [2]. The code in BrownoutPrototype is not using iFogSim but where as I have used iFogSim simulator to implement the proposed model, and CloudSimBrownoutV4 was built on cloudsim.My_Demo.java class is the sample simulation class for Microservice Application.This class first has the configuration setting to create the fog-cloud infrastructure, then the proposed model has been called to activate or deactivate the optional components. The main classes as part of the proposed model implementation are as follows:

## 5.1 FogBrownOutMain Class

This is the main class which calls the BrownoutRunner Class where the containerAllocation policy, containerSelectionPolicy , workload, and parameters are passed. BrownRunner class extends the RunnerAbstract class . For this research , I have comparing both 'MinimumUtilization' policy and 'MaximumCorrelation' policy for containerselection policy.



```
public static void main(String[] args) throws IOException {

    boolean enableOutput = true;
    boolean outputToFile = false;
    String inputFolder = "";
    String outputFolder = "output";
    String workload = "random"; // Random workload
    String conatinerAllocationPolicy = "thr";
    String containerSelectionPolicy = "mu";
    String parameter = "0.8";

    new BrownoutRunner(
            enableOutput,
            outputToFile,
            inputFolder,
            outputFolder,
            workload,
            conatinerAllocationPolicy,
            containerSelectionPolicy,
            parameter);
}
```

Figure 7: Minimum Utilization ContainerSelectionPolicy

## 5.2 FogBrownoutController Class

This class has the methods to get dimmer value and overloaded hosts which are retrieved from containerlist information. So the formula for calculating the dimmer value is
d = 1-(numberOfOverloadedHosts)/total number of Hosts

## 5.3 Container Class

This class has the properties and method to get the information about the container.

[1]BrownoutPrototype: https://github.com/xmxyt900/BrownoutPrototype

[2]CloudSimBrownoutV4:https://github.com/xmxyt900/CloudSimBrownoutV4/tree/master/cloudsim-3.0.3/examples/com/cloudbus/cloudsim

## 5.4 BrownoutHelper Class

This class creates the cloudlet depends on the number of hosts and broker parameters which are passed from BrownoutRunner class.This has parameter to pass whether brownout is enable or not. For this study, I have kept brownout enabled is true.

## 5.5 BrownoutRunner Class

The class has bee extended from RunnerAbstract which is resposible for calling the createCloudletList,createVmList,createHostList methods.I have used createVmList method as it is inbuilt but with container's configuration.

## 5.6 Fog Infrastructure

In iFogsim, the simulation environment has been created for fog-cloud interaction of microservice applications.The cloud, proxy-layer, fog layers are simulated, before this the BrownoutRunner has been called.

## 5.7 Entity Interaction

8 shows the overview of the entity interaction in the proposed model in which FogBrownoutController is the main entity which works on the Brownout approach.

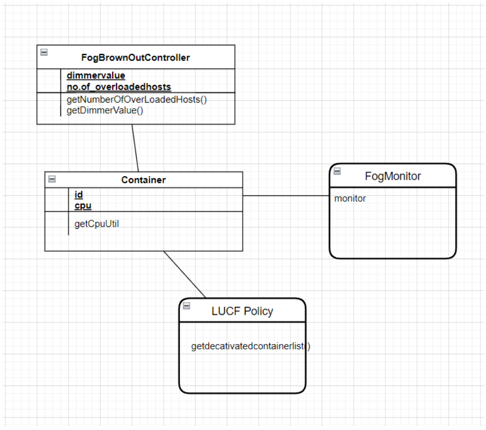


Figure 8: Entity Interaction of Proposed Model

# 6 Evaluation

The proposed model has been tested series of time using iFogSim in terms of exection time, energy consumption, and SLA violation by changing the no.of hosts and no.of containers in the BrownoutConstants.java class.

## 6.1 Case Study 1

**Energy Consumption(kWh)** This metric helps in knowing the total energy consumption in fog data centers which involves in aggregating the energy consumption of all the servers.To know this metric , I have take three scenarios where No.of.Containers has been changed from 20, 80, 200 as depicted in 9.From the graph , it is evident though the no.of.containers has been increased, there is no much increase in the energy consumption for the 200 containers, it is 249 kWh.

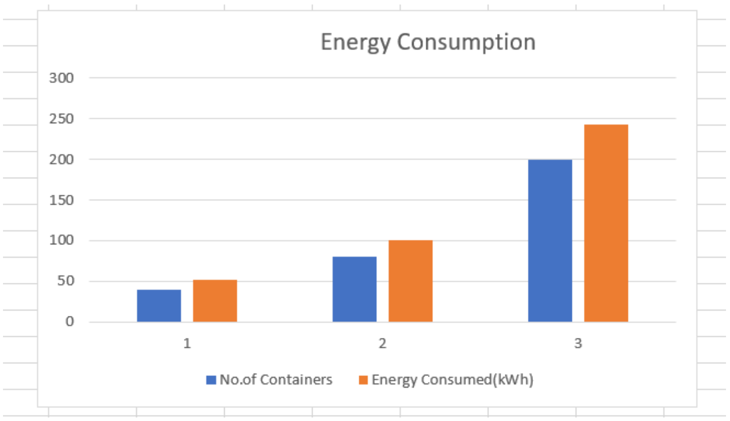


Figure 9: Energy Consumption

## 6.2 Case Study 2

**SLA Violation**

When the hosting container is not getting its requested CPU share, then the SLA violation is considered.For this metric, I have repeated the same experiments where no.of containers and hosts were changed.From the 10, it is clear that there is a decrease in SLA violations though the no.of containers .For example , if the no.of containers is increased to 800 containers, SLA violation is 0.00251%

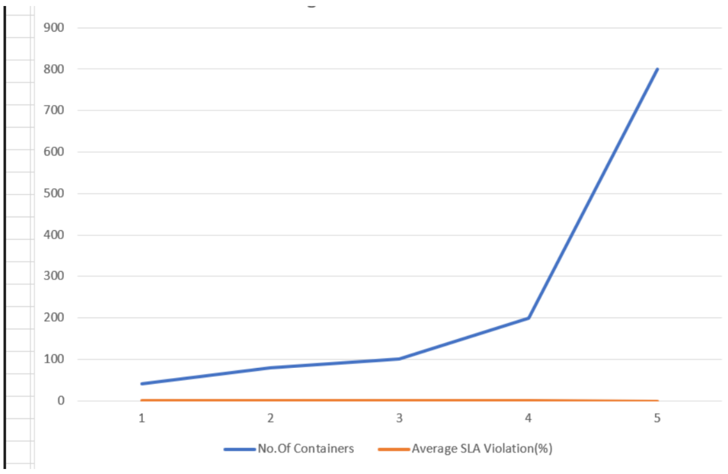


Figure 10: SLA Violation

### 6.3 Case Study 3

**Execution Time** This metric is the total time take to execute all the tasks icluding all the migration , deactivating the containers, and restarting, shutting down.For this metric, no.of.containers has been changed 100. 200,800, the execution was 0.03057, 200-0.06836,0.12330 in seconds which is not optimized using the proposed model.

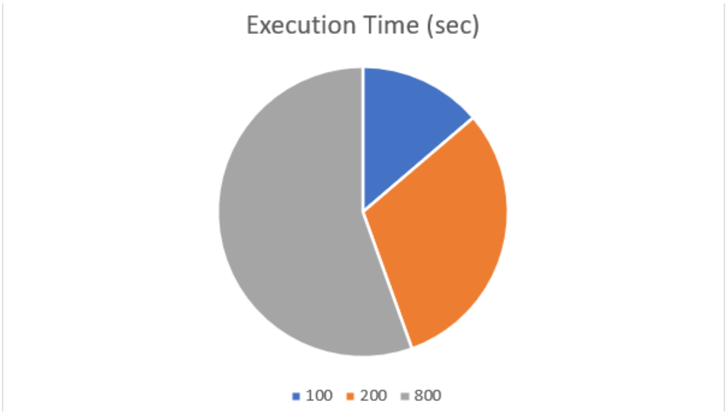


Figure 11: Execution Time

### 6.4 Discussion

From the above experiements, it is clear that the proposed model is satisfying the reduction of energy consumption, SLA violation but not execution time.But still, the experiment has to be compared with other models as well some real datasets

## 7 Conclusion and Future Work

With the increase in IoT devices,the cloud is facing challenges related to latency, network-congestion.In order to overcome this, Fog computing has been introduced.So Fog computing has advantages, still it faces issues related to energy consumption , and in terms of QoS as well. Though some works have been carried in optimizing the energy consumption and QoS, brownout paradigm is one of the promising model in terms of energy consumption optimization.So in this research, I have used brownout paradigm for energy consumption, and QoS optimization in Fog computing with the help of iFogSim.The iFogSim has been executed in Eclipse IDE, and series of experiments done to evaluate energy consumption, SLA, Execution time metrics. And, from the results it is evident that energy consumption and SLA has positive impact but not the Execution time. So in Future, real-time datasets can be used to evaluate this model in Fog.Also, the experiments needs to compared with the other models which is not done as part of this study.

## References

Baccarelli, E., Naranjo, P. G. V., Scarpiniti, M., Shojafar, M. and Abawajy, J. H. (2017). Fog of everything: Energy-efficient networked computing architectures, research challenges, and a case study, *IEEE access* **5**: 9882–9910. JCR Impact Factor 2021: 3.367.

Chen, F. and Zhou, X. (2019). The container selection policy based on brownout in edge computing, *2019 11th International Conference on Intelligent Human-Machine Systems and Cybernetics (IHMSC)*, Vol. 2, IEEE, pp. 97–100.

Chen, F., Zhou, X. and Shi, C. (2019). The container scheduling method based on the min-min in edge computing, *Proceedings of the 2019 4th International Conference on Big Data and Computing*, pp. 83–90. CORE2021 Rank: C.

Chen, S., Zheng, Y., Wang, K. and Lu, W. (2019). Delay guaranteed energy-efficient computation offloading for industrial iot in fog computing, *ICC 2019-2019 IEEE International Conference on Communications (ICC)*, IEEE, pp. 1–6. CORE2018 Rank: B.

Gai, K., Qin, X. and Zhu, L. (2021). An energy-aware high performance task allocation strategy in heterogeneous fog computing environments, *IEEE Trans. Comput.* **70**(4): 626–639. JCR Impact Factor 2021: 2.663.

Ilager, S., Ramamohanarao, K. and Buyya, R. (2019). Etas: Energy and thermal-aware dynamic virtual machine consolidation in cloud data center with proactive hotspot mitigation, *Concurrency and Computation: Practice and Experience* **31**(17): e5221. JCR Impact Factor 2021: 1.536.

Kaur, K., Garg, S., Kaddoum, G., Ahmed, S. H. and Atiquzzaman, M. (2019). Keids: Kubernetes-based energy and interference driven scheduler for industrial iot in edge-cloud ecosystem, *IEEE Internet of Things Journal* **7**(5): 4228–4237. JCR Impact Factor 2021: 9.471.

Kim, Y. G., Kim, J. I., Choi, S. H., Kim, S. Y. and Chung, S. W. (2019). Temperature-aware adaptive vm allocation in heterogeneous data centers, *2019 IEEE/ACM International Symposium on Low Power Electronics and Design (ISLPED)*, IEEE, pp. 1–6.

Klein, C., Maggio, M., Årzén, K.-E. and Hernández-Rodriguez, F. (2014). Brownout: Building more robust cloud applications, *Proceedings of the 36th International Conference on Software Engineering*, pp. 700–711. CORE2021 Rank: A*.

Mahmud, R. and Buyya, R. (2019). Modelling and simulation of fog and edge computing environments using ifogsim toolkit, *Fog and edge computing: Principles and paradigms* pp. 1–35.

Medel, V., Tolosana-Calasanz, R., Bañares, J. Á., Arronategui, U. and Rana, O. F. (2018). Characterising resource management performance in kubernetes, *Computers & Electrical Engineering* **68**: 286–297. JCR Impact Factor 2021: 3.818.

Mendes, S., Simão, J. and Veiga, L. (2019). Oversubscribing micro-clouds with energy-aware containers scheduling, *Proceedings of the 34th ACM/SIGAPP Symposium on Applied Computing*, pp. 130–137. CORE2021 Rank: B.

Mishra, B. S. P., Das, H., Dehuri, S. and Jagadev, A. K. (2018). *Cloud computing for optimization: foundations, applications, and challenges*, Springer.

Pierson, J.-M., Stolf, P., Sun, H. and Casanova, H. (2020). Milp formulations for spatio-temporal thermal-aware scheduling in cloud and hpc datacenters, *Cluster Computing* **23**(2): 421–439. JCR Impact Factor 2021: 1.809.

Wan, X., Guan, X., Wang, T., Bai, G. and Choi, B.-Y. (2018). Application deployment using microservice and docker containers: Framework and optimization, *Journal of Network and Computer Applications* **119**: 97–109. JCR Impact Factor 2021: 6.281.

Wu, H.-Y. and Lee, C.-R. (2018). Energy efficient scheduling for heterogeneous fog computing architectures, *2018 IEEE 42nd annual computer software and applications conference (COMPSAC)*, Vol. 1, IEEE, pp. 555–560. CORE2021 Rank: B.

Xiao, Z., Jiang, J., Zhu, Y., Ming, Z., Zhong, S. and Cai, S. (2015). A solution of dynamic vms placement problem for energy consumption optimization based on evolutionary game theory, *Journal of Systems and Software* **101**: 260–272. JCR Impact Factor 2021: 2.829.

Xu, M. and Buyya, R. (2017). Energy efficient scheduling of application components via brownout and approximate markov decision process, *International Conference on Service-Oriented Computing*, Springer, pp. 206–220. CORE2021 Rank: C.

Xu, M. and Buyya, R. (2019a). Brownout approach for adaptive management of resources and applications in cloud computing systems: A taxonomy and future directions, *ACM Comput. Surv.* **52**(1). JCR Impact Factor 2021: 10.282.
**URL:** *https://doi.org/10.1145/3234151*

Xu, M. and Buyya, R. (2019b). Brownoutcon: A software system based on brownout and containers for energy-efficient cloud computing, *Journal of Systems and Software* **155**: 91–103. JCR Impact Factor 2021: 2.829.

Xu, M., Dastjerdi, A. V. and Buyya, R. (2016). Energy efficient scheduling of cloud application components with brownout, *IEEE Transactions on Sustainable Computing* **1**(2): 40–53. JCR Impact Factor 2021: 5.40.

Xu, M., Toosi, A. N. and Buyya, R. (2018). Ibrownout: an integrated approach for managing energy and brownout in container-based clouds, *IEEE Transactions on Sustainable Computing* **4**(1): 53–66. JCR Impact Factor 2021: 5.40.

Yang, B., Li, Z., Chen, S., Wang, T. and Li, K. (2016). Stackelberg game approach for energy-aware resource allocation in data centers, *IEEE Transactions on Parallel and Distributed systems* **27**(12): 3646–3658. JCR Impact Factor 2021: 2.687.

Zhang, D., Yan, B.-H., Feng, Z., Zhang, C. and Wang, Y.-X. (2017). Container oriented job scheduling using linear programming model, *2017 3rd International Conference on Information Management (ICIM)*, IEEE, pp. 174–180.

Zu, Y., Shen, F., Yan, F., Shen, L., Qin, F. and Yang, R. (2019). Smeto: Stable matching for energy-minimized task offloading in cloud-fog networks, *2019 IEEE 90th Vehicular Technology Conference (VTC2019-Fall)*, pp. 1–5. CORE2021 Rank: B.