Energy Efficient Genetic Algorithm for Container Consolidation in Cloud System

Dimple Patel
Dept. of Computer Science & Eng
National Institute of Technology
Rourkela, India
dimplepatel982@gmail.com

Manoj Kumar Patra
Dept. of Computer Science & Eng
National Institute of Technology
Rourkela, India
manojpatra.in@gmail.com

Bibhudatta Sahoo
Dept. of Computer Science & Eng
National Institute of Technology
Rourkela, India
bibhudatta.sahoo@gmail.com

***Abstract*—A lot of work has been done and extensive research is going on for reducing the energy consumption of large-scale cloud data centers along with the maximization of host level utilization. Because of on-demand service provisioning, the number of users' requests increases which leads to an increase in the number of physical machines and data center size. This results in increased energy consumption in the data center as energy consumption is directly proportional to the number of the active physical machine. The container is a lightweight approach to operating system virtualization, offers an opportunity to improve the efficiency in cloud data centers. Containerized cloud gaining widespread popularity in recent years, so an efficient container consolidation is an open research challenge. Container consolidation is used to optimize energy consumption, as the resource utilization by the containers is directly relates to energy consumption. Container consolidation is an NP-Hard problem which can be solved by heuristic and metaheuristic algorithm. Therefore, in this paper, we implemented the heuristic and metaheuristic algorithm for energy reduction in the cloud. We implemented the next fit, best fit, best fit decreasing, first fit, first fit decreasing and compare their result. The experimental result shows that the best fit decreasing and first fit decreasing gave the almost same result. Further, this result is compared with the proposed Energy Efficient Genetic Algorithm(EEGA). We got the best result with the proposed EEGA.**

***Index Terms*—Cloud Computing, Energy Consumption, Containerized Cloud, Virtualization, Virtual Machine.**

I. Introduction

Cloud computing is a utility-oriented delivery of computing services on a pay-as-you-go basis. It uses remote servers that are hosted on the internet for storing, managing, and processing data that are delivered on-demand. Cloud computing is defined as configuring, manipulating and accessing the software and hardware resources remotely. It offers online data storage, platform independence, on-demand scalability, ease of management, cost-effectiveness, high rate of reliability, continuous deployment, availability, etc. Such advantages of cloud computing change the way of deployment of the application. Companies require on-demand scalability for their enterprise application which can be achieved through cloud computing models.

Scaling of monolithic applications is difficult as it consists of a lot of services into a single code. This is why most of the companies are moving towards micro-service architecture. Application in micro-service architecture consists of many decoupled services independent of each other. Each of these services performs specific tasks independently of each other. The benefits of microservices application are it allows independent updating and redeployment of a small part of the application. One of the technologies enabling microservices architecture is containerization of application. Containerization is OS-level virtualization. The difference between container and Virtual Machine is as shown in Fig.1

The virtual machine is a software program that emulates the functionality of physical hardware. Hypervisors are the way to manage virtual machines. A hypervisor runs VMs that have their OS. The hypervisor sits between hardware and VM and is responsible for creating VM. Each VM has their operating system and on top of VM, applications are run. The container provides an isolated environment to an application. The container requires an underlying OS that provides the basic services to all the containerized applications. As each container run on the same host OS, container overhead is low as compared to virtual machines. Another advantages of the container over virtual machines are its ability to start a new container faster than starting new virtual machines, containers are very lightweight as compared to virtual machines, also resource management overhead is less than a virtual machine. The architecture model proposed in [3] for smart city using IoT can be containerized for better performance.

The numerous advantages of cloud computing encourage platform providers to increase the capacity of their data centers to meet the increasing demand of the customer. As a result of this, energy consumption increases as to power such a large infrastructure, cloud data center need more energy. It is difficult to define unique service for energy consumption in the data center due to its complexity. The author in [4] has identified four scenarios within a system where energy is not used efficiently- Network, Servers, Cloud Management System and Appliances. Reason for energy wastage in data centers are inefficiency in data center cooling systems [5] , network equipment [6], and server utilization [7]. Servers are the main reason for energy consumption in the data center [7]. Therefore we try to minimize energy consumption on the server-side.

A single cloud data center may consist of thousands of servers that are spread over 4500 square meters. A 500 square

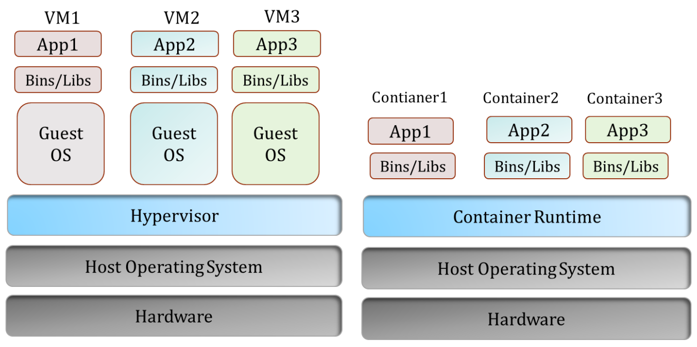

Fig. 1: Virtual Machine Vs Container

meter data center can consume the power of 38 megawatt-hours(MWh) per day, which is more than power consumption in 3500 households in the EU. Energy consumption also increases the total cost of ownership (TCO). This results in a decreasing return on investment (ROI) of the cloud infrastructure. Also, energy consumption leads to the emission of carbon dioxide (CO_2), which is estimated to be 2% of global emission [8].

The rest of the paper is structured as follows. Section II describes related work. In section III, System model is discussed. Section IV describes the proposed genetic algorithm. Section V describes experimental analysis and result. Section VI gives conclusion and future work.

II. Related Work

Claus Pahl et al. [1] describes an introduction to virtualization and need for the containerization. They discussed that PaaS cloud service can use a container to manage and orchestrate application. The paper describes containerization technologies and how the application can be deployed on PaaS platform. In paper [2], Ozmen Emre et al. compared the effects of Linux application containerization on power consumption to server virtualization and BareMetal environments. They used five different benchmarking tests namely timed Linux kernel compilation benchmark, aio-stress benchmark, RAM speed benchmark, loopback TCP network performance benchmark and apache benchmark on three different platforms Bare Metal, KVM and Docker. The author concludes that in the apache benchmark test, ram speed test, and timed Linux kernel compilation test, docker container has better performance with less energy consumption than KVM. However, in hard disk and network tests, KVM was having less energy consumption than a docker container. Baremetal had better results in all benchmarks tests.

Piraghaj et al. [9] find an efficient virtual machine size and host container on a virtual machine so that the wastage of resources is minimized while executing the workload. The main contribution is to find an approach for the efficient allocation of resources that matches closely the actual resource usage of deployed container. For this, they analyzed the cloud's trace log from Google cluster and clustered tasks based on their usage pattern. Then this task is mapped to a container, suitable virtual machine size for containers are calculated and these containers are hosted on their associated virtual machine types. The proposed method is compared against fixed VM size and the result shows improvement in energy consumption. The author hosted containers on VM but not directly on the physical machine.

In paper [10], Piraghaj et al. improve the energy efficiency of servers by proposing a framework that consolidates containers on virtual machines. The proposed framework decreases the number of running servers through the container migration algorithm. Three stages for container migrations are considered when to migrate containers, how many containers need to be migrated and where to migrate. In the first stage, if the host is found to be overloaded or under-loaded, then container migration is initiated. For the second stage, ”MCor” and ”MU” algorithms are considered. For the third stage, three bin packing algorithms are applied. In paper [11], Dong et al. proposed the most efficient server first (MESF) task scheduling scheme to minimize energy consumption in the data center. The proposed energy model name is VPC (Virtual Power Consumption). Parameter considered are CPU along with resource allocation. This is done on a virtual machine and not on containers.

Guan et al. in [12], design an AODC framework for minimizing application deployment cost and for supporting automatic scaling when the workload of cloud application changes. They minimize resource allocation and evaluated models using the CPLEX optimization tool but not considered real-world datasets. Dong et al. in [13], proposed a solution for the containerized application scheduling based on integer linear programming. They evaluated container image pulling cost, host energy conservation and network transition cost from the client to the container host.

Hanafy et al. in [14], demonstrate the association between container and host selection policy. Host selection policies such as ACO, Least Full, Most Full, Most correlated, least correlated, max variance and min variance are considered and for container selection, they considered max-correlated, max-variance, max-usage, and min-variance. They find a better result with max-variance host selection and most usage container selection policy. Chen Q. et al. in [15] proposed a utilization based migration algorithm for mapping VM to servers. The proposed algorithm is evaluated with real workload form CoMon. Tao Shi. et al. in [16] proposed Two-stage multitype PSO algorithm for energy saving. They first wrap the application into a container and allocated this container to VM using First Fit algorithm. Then this VM is allocated to a server using PSO algorithm.

III. System Model

This section describes an architectural model, problem statement and mathematical model for the proposed system.

A. Architectural Model

The proposed system mainly consists of 3 main component. (i) Application, which consists of different independent modules. Each of these modules is described by the amount of CPU required which is measured in MIPS and amount of memory required which is measured in MB. It is assumed that these two specifications of the modules are specified by the user. (ii) Container mapper, which maps each of these modules into a container and, (iii) Servers, on which these containers are running. These servers are specified with the amount of CPU and memory they possess. Servers can be homogeneous or heterogeneous.

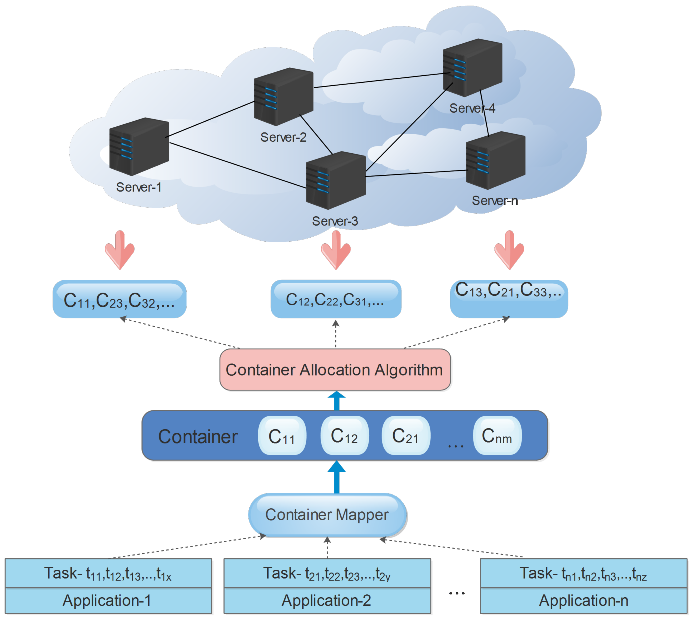

Fig. 2: Architecture Model

Fig 2 shows the proposed architecture model. There are multiple application each of which consists of multiple modules. These modules are sent to container mapper which is responsible to map each of these modules into a container. Then apply a container allocation algorithm that maps these containers to a minimum number of servers so that energy consumption can be minimized.

B. Problem Statement

For a given application, we have a container set $C = \{c_1, c_2, c_3, ...c_m\}$ where c_i, $i = \{1, 2, ..., m\}$ is a container of a modules of the application. Let host machine set be $H = \{h_1, h_2, h_3, ..., h_n\}$. Container runs on host machine h_k where $k = \{1, 2, 3, ..., n\}$. The objective is to find an efficient container allocation algorithm such that the overall energy consumption in data center can be minimized.

C. Mathematical Model

It is assumed that the application with its requirement and dependencies are specified. Each of these applications consists of an independent module and these modules are mapped to container. So, for each application, we have a set of container $\{c_1, c_2, \ldots, c_m\}$. The workload of server is represented as $(s_{max\_cpu\_capacity}, s_{max\_memory\_capacity}, s_{max\_power}, s_{idle\_power})$, where $s_{max\_cpu\_capacity}$ represents the servers maximum processing capacity in MIPS, $s_{max\_memory\_capacity}$ represents the server maximum memory in GB, s_{max_power} is the maximum power consumption for a give server and s_{idle_power} is the idle power consumption for the server. Container workload is represented as $(c_{cpu\_utilization}, c_{memory\_utilization})$ where, $c_{cpu\_utilization}$ represents the containers processing capacity in MIPS and $c_{memory\_utilization}$ represents the containers memory requirement in MB. The correlation between server utilization and the electric power consumption proposed in [17], i.e. for a given server k,

$$P_k(u) = (P_{k,max} - P_{k,idle}) \cdot u_k + P_{k,idle} \tag{1}$$

where, u_k is utilization rate of server k.

Utilization rate of a server is calculated by utilization of both CPU and Memory i.e.

$$u_k = u_{k_{CPU}} + u_{k_{memory}} \tag{2}$$

where $u_{k_{memory}}$ rate for a given server k is given by,

$$u_{k_{memory}} = \frac{u_{k_{memory\_utilized}}}{u_{k_{max\_memory\_capacity}}} \tag{3}$$

and $u_{k_{CPU}}$ rate for a given server k is given by,

$$u_{k_{CPU}} = \frac{u_{k_{cpu\_utilized}}}{u_{k_{max\_cpu\_capacity}}} \tag{4}$$

The total power consumption of data center is given by,

$$P = \sum_{k=1}^{n} P_k(u) \tag{5}$$

Let $x_{i_{memory}}(k)$ and $x_{i_{cpu}}(k)$ denotes the memory and cpu required by container i on k^{th} server. Objective is to $minimize \sum_{k=1}^{n} P_k(u)$, Subjected to,

$$\sum_{i=1}^{n} x_{i_{memory}}(k) <= memory\_capacity(k)$$

$$\sum_{i=1}^{n} x_{i_{cpu}}(k) <= cpu\_capacity(k)$$

IV. Proposed Algorithm

The genetic algorithm is a metaheuristic algorithm based on Darwins's theory of evolution. It is a randomized algorithm in which random changes are applied to the current solution to find the new solution. The basic workflow of the proposed energy efficient genetic algorithm is as shown in Algorithm 1.

A. Encoding Candidate Solution

Length of chromosome is taken equal to the number of container taken in which each gene represent a positive integer between 1 to number of servers. The chromosome representation scheme is as shown in diagram 3.

Figure 3 shows an example of representation of chromosome in which 10 containers are considered. Container number $C1, C2, C3, C4, C5, C6, C7, C8, C9, C10$ are placed on servers $S1, S2, S3, S1, S2, S2, S1, S3, S3, S2$ respectively.

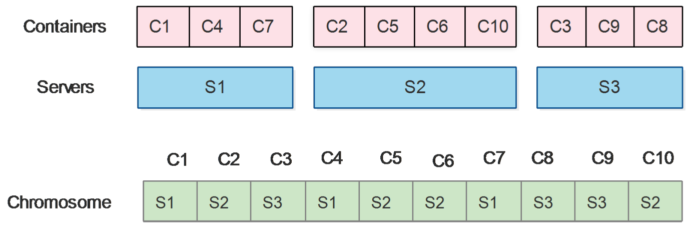

Fig. 3: Chromosome Representation

```
Algorithm 1 EEGA: Energy Efficient Genetic Algorithm
1: Determine the number of chromosomes and generation.
2: Initialize each chromosome with random value.
3: Process step 4-7 until the number of generation met.
4: Calculate the fitness value for each chromosome in the population.
5: Select two individual from the chromosome according to the selection method.
6: Perform crossover on the selected two individual to generate new child.
7: Perform mutation on newly generated child.
8: Return the best solution.
```

B. Initial Population

An initial population of the chromosome is generated randomly. While generating random value, we also take care that the container is not mapped to a server that is overloaded or may go for overload.

```
Algorithm 2 Initial Population Generation
Input: chromosome length($c\_length$), population Size($p\_size$), number of servers($no\_of\_servers$),number of container ($no\_of\_containers$) ,container CPU utilization list($c\_cpu\_utilization$),container memory utilization list ($c\_memory\_utilization$),server cpu utilization list ($s\_cpu\_utilization$),server memory utilization list ($s\_memory\_utilization$) , maximum server CPU utilization ($max\_server\_cpu\_utilization$), maximum server memory utilization ($max\_server\_memory, \_utilization$).
Output: initial population $P$.
1: for $i = 1$ to $p\_size$ do
2:     for $j = 1$ to $c\_length$ do
3:         $rnd$ = random.randint(0,$no\_of\_servers$)
4:         while $s\_cpu\_utilization)[rnd] + c\_CPU\_utilization[j] >= max\_server\_CPU\_utilization$ and $s\_memory\_utilization)[rnd] + c\_memory\_utilization[j] >= max\_server\_memory\_utilization$ do
5:             $rnd$ = random.randint(0, $no\_of\_servers$)
6:         end while
7:         $p(i, j) = rnd$
8:     end for
9: end for
10: return $P$
```

C. Fitness Function

Fitness function finds the energy consumption of each server.

```
Algorithm 3 Fitness Function Algorithm
Input: server list $server\_list$.
Output: total energy consumption ($E$).
1: Initialize energy consumption($E$) = 0
2: for each server $S_i$ in $server\_list$,calculate the energy consumption do
3:     $E = \sum_{i=1}^{n} (P_{i,max} - P_{i,idle}).u + P_{i,idle}$
4: end for
5: return $E$
```

D. Selection Method

The elitism method is used for selection. When creating a new population, there may be chances of losing the best chromosome. Therefore this method first copies the best chromosome to the new population. First, we sort all the chromosomes according to their fitness value. Then, we select the first 20% of the total population for the new population and for the rest of 80%, we randomly selected two individuals and perform crossover.

E. Crossover Method

The proposed Energy Efficient Genetic Algorithm(EEGA) performs a uniform crossover in which random number is generated if the generated random number is less then 0.5 then select a gene from parent1 otherwise select a gene from parent2. While selecting the gene number, guarantee that the assigned container will not exceed the total workload on the server.

F. Mutation

Randomly select a gene from a newly generated child and change the value of a gene by taking care that the change value of gene will not exceed server utilization.

```
Algorithm 4 Mutation Method
Input: a newly generated child $child$.
Output: a mutated child $child$.
1: Generate a random container id ($rnd_1$).
2: Generate a random server id ($rnd_2$) between ($0, no\_of\_servers$).
3: if the generated container id does not exceed capacity of generaed server id then
4:     child[$rnd_1$] = $rnd_2$.
5: else
6:     Generate random server id until the container id exceed the server id capacity.
7: end if
8: return $child$.
```

V. Experimental Analysis and Result

The experiment was performed by considering heterogeneous servers where server maximum processing capacity considered from 2000 to 8000 MIPS and memory capacity taken is 4GB, 8GB, 12GB, and 32GB. The number of servers considered is equal to the number of containers. The number of container taken are 10, 20, 30, 40, 50. Container processing capacity is taken between 500 to 3000 MIPS and memory capacity is between 400 to 1000 MB. The maximum power consumption for server considered between 80 to 100W and idle power between 10 to 20 W. This system model description is as shown in the table I.

TABLE I: Parameters Considered for the System Model

Server Type	Homogeneous/ Heterogeneous
Servers maximum processing capacity	Between 2000 to 8000 MIPS
Servers maximum memory capacity	4GB, 8GB, 16GB, 32GB
Containers Processing Capacity	Between 500 to 3000 MIPS
Containers Memory Capacity	Between 400 to 1000 MB
Maximum Power	Between 80 to 100 W
Idle Power	Between 10 to 20 W

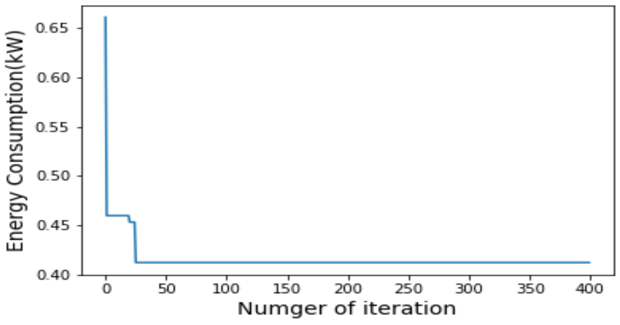

Fig. 4: 10 Container with 400 Iteration

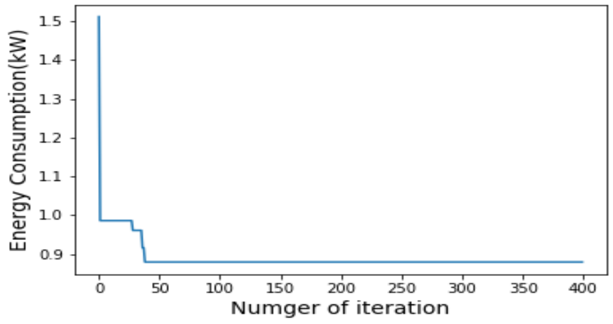

Fig. 5: 20 Container with 400 Iteration

The Stopping condition for the proposed genetic algorithm is decided by experimenting with the 2000 iteration on 10, 20, 30, 40, 50 container as shown in Fig. [4], Fig.[5], Fig.[6], Fig.[7] and Fig.[8]. The experimental result shows that almost after 200 iterations, energy consumption is constant. Therefore we decide stopping condition to 200 iterations.

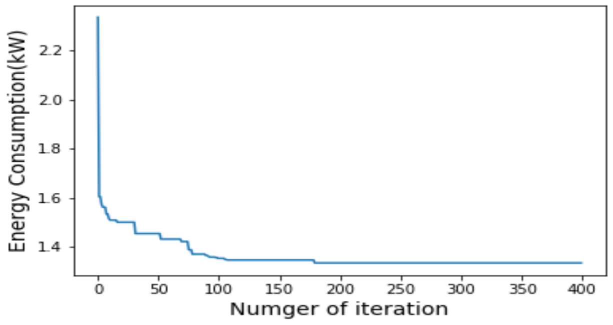

Fig. 6: 30 Container with 400 Iteration

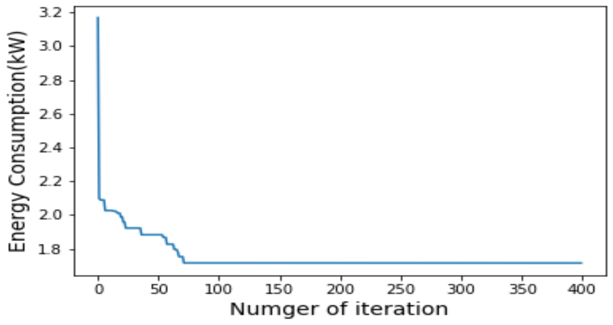

Fig. 7: 40 Container with 400 Iteration

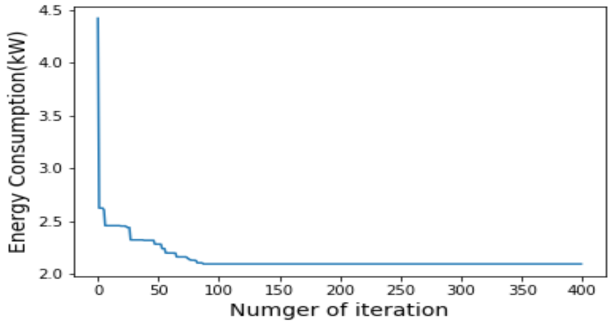

Fig. 8: 50 Container with 400 Iteration

Similarly, we decided population size by considering population size from 10 to 2000 and 10, 20, 30, 40, 50 containers. We got energy consumption constant after 1000 population size. So 1000 population size is fixed for the proposed genetic algorithm. For the selection method, we use the elitism method in which some percent of the chromosome are directly selected for the next generation. For deciding the percentage value, we perform experiment with $10\%, 15\%, 20\%, 25\%$ and 30% chromosome and we got less energy consumption with 20%. The result of the experiment is as shown in Fig. 9.

So, the parameter considered for genetic algorithm are listed in table II. Now, first, we experiment with some simple heuristic algorithms like best fit, best fit decreasing, next fit, first fit and first fit decreasing. The number of the container taken were 10, 20, 30, 40 and 50. The result is compared and found that the performance of first fit decreasing and best fit decreasing is almost similar. Then, the result of EEGA, the proposed genetic algorithm is compared with best fit decreasing and found that the EEGA performs better than best fit decreasing(BFD) algorithm. The result is shown in Fig.10.

TABLE II: Parameter of Genetic Algorithm

Population size	1000
Number of generation	200
Chromosome size	Equal to number of container
Selection method	Elitism (20%)
Crossover method	Uniform crossover
Mutation	randomly changing one gene
Fitness function	$E = \sum_{i=1}^{n} (P_{i,max} - P_{i,idle}).u + P_{i,idle}$

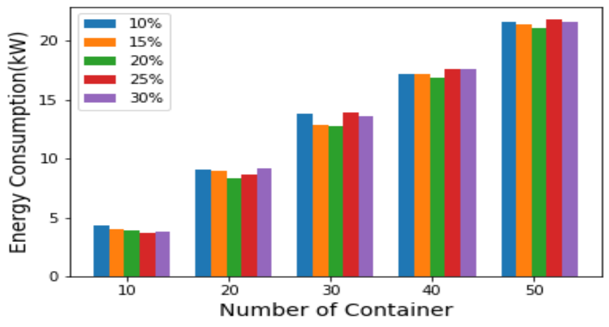

Fig. 9: Selection Method

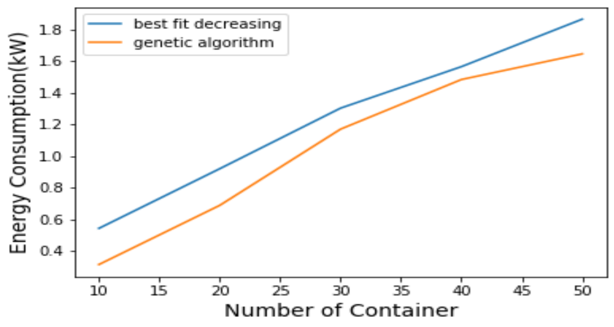

Fig. 10: Energy Consumption of EEGA Vs BFD

VI. Conclusion

In this paper, an energy-efficient genetic algorithm has been proposed to minimize energy consumption in the cloud data center. Multiple containers are running in a single server present in the data center. Several heuristic algorithms such as next fit, best fit, best fit decreasing, first fit, and first fit decreasing are implemented for minimization of energy consumption and it is found that the energy consumption in best fit decreasing and first fit decreasing is almost same and less than other algorithms. The result of the proposed energy efficient genetic algorithm(EEGA) and best fit decreasing are compared. The consumption of energy in EEGA is found to be lesser than best fit decreasing.

References

[1] Pahl, Claus. ”Containerization and the paas cloud.” IEEE Cloud Computing 2, no. 3 (2015): 24-31.

[2] DEMİRKOL, ÖZMEN EMRE, and AŞKIN DEMİRKOL. ”Energy efficiency with an application container.” Turkish Journal of Electrical Engineering & Computer Sciences 26, no. 2 (2018): 1129-1139.

[3] Patra, Manoj Kumar. ”An architecture model for smart city using Cognitive Internet of Things (CIoT).” In 2017 Second International Conference on Electrical, Computer and Communication Technologies (ICECCT), pp. 1-6. IEEE, 2017.

[4] Mastelic, Toni, and Ivona Brandic. ”Recent trends in energy-efficient cloud computing.” IEEE Cloud Computing 2, no. 1 (2015): 40-47.

[5] Greenberg, Steve, Evan Mills, Bill Tschudi, Peter Rumsey, and Bruce Myatt. ”Best practices for data centers: Lessons learned from benchmarking 22 data centers.” Proceedings of the ACEEE Summer Study on Energy Efficiency in Buildings in Asilomar, CA. ACEEE, August 3 (2006): 76-87.

[6] Heller, Brandon, Srinivasan Seetharaman, Priya Mahadevan, Yiannis Yiakoumis, Puneet Sharma, Sujata Banerjee, and Nick McKeown. ”Elastictree: Saving energy in data center networks.” In Nsdi, vol. 10, pp. 249-264. 2010.

[7] Greenberg, Albert, James Hamilton, David A. Maltz, and Parveen Patel. ”The cost of a cloud: research problems in data center networks.” ACM SIGCOMM computer communication review 39, no. 1 (2008): 68-73.

[8] Buyya, Rajkumar, Anton Beloglazov, and Jemal Abawajy. ”Energy-efficient management of data center resources for cloud computing: a vision, architectural elements, and open challenges.” arXiv preprint arXiv:1006.0308 (2010).

[9] Piraghaj, Sareh Fotuhi, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya. ”Efficient virtual machine sizing for hosting containers as a service (services 2015).” In 2015 IEEE World Congress on Services, pp. 31-38. IEEE, 2015.

[10] Piraghaj, Sareh Fotuhi, Amir Vahid Dastjerdi, Rodrigo N. Calheiros, and Rajkumar Buyya. ”A framework and algorithm for energy efficient container consolidation in cloud data centers.” In 2015 IEEE International Conference on Data Science and Data Intensive Systems, pp. 368-375. IEEE, 2015.

[11] Dong, Ziqian, Wenjie Zhuang, and Roberto Rojas-Cessa. ”Energy-aware scheduling schemes for cloud data centers on google trace data.” In 2014 IEEE Online Conference on Green Communications (OnlineGreencomm), pp. 1-6. IEEE, 2014.

[12] Guan, Xinjie, Xili Wan, Baek-Young Choi, Sejun Song, and Jiafeng Zhu. ”Application oriented dynamic resource allocation for data centers using docker containers.” IEEE Communications Letters 21, no. 3 (2016): 504-507.

[13] Zhang, Dong, Bing-Heng Yan, Zhen Feng, Chi Zhang, and Yu-Xin Wang. ”Container oriented job scheduling using linear programming model.” In 2017 3rd International Conference on Information Management (ICIM), pp. 174-180. IEEE, 2017.

[14] Hanafy, Walid A., Amr E. Mohamed, and Sameh A. Salem. ”Novel selection policies for container-based cloud deployment models.” In 2017 13th International Computer Engineering Conference (ICENCO), pp. 237-242. IEEE, 2017.

[15] Dasgupta, Gargi Sharma, Amit Verma, Akshat Neogi, Anindya Kothari, Ravi. (2011). Workload Management for Power Efficiency in Virtualized Data Centers. Commun. ACM. 54. 131-141. 10.1145/1965724.1965752.

[16] Shi, Tao, Hui Ma, and Gang Chen. ”Energy-aware container consolidation based on PSO in cloud data centers.” In 2018 IEEE Congress on Evolutionary Computation (CEC), pp. 1-8. IEEE, 2018.

[17] Feller, Eugen, Louis Rilling, and Christine Morin. ”Energy-aware ant colony based workload placement in clouds.” In Proceedings of the 2011 IEEE/ACM 12th International Conference on Grid Computing, pp. 26-33. IEEE Computer Society, 2011.