Optimizing the docker container usage based on load scheduling

M.SURESHKUMAR1st
Information technology
Sri Sairam engineering college
Chennai-44,India.
E-mail: sureshkumar.it@sairam.edu.in

P.RAJESH2nd
Information technology
Sri Sairam engineering college
Chennai-44,India
E-mail: srisairamrajesh@gmail.com

Abstract: In this project we introduce an energy-aware processing model used for balancing the load of docker container and job scaling using docker(1).The approach used is to create an energy optimal regime within which the containers operate.An important strategy for energy reduction is concentrating the load on the containers.When the load increases,then by api call a new container can be spawned which can the allocated with jobs to process.The container can be created automatically based upon threshold conditions.This ensures that the rest of the container is not over-loaded with jobs.When the load decreases that container can be killed to save energy.The Energy-Aware scaling algorithm is used which ensures that the largest possible number of containers operate within their respective operational domain.

Keywords: Load scheduling,Docker Container ,Docker Image.

I. INTRODUCTION

The concept of cloud computing is exponentially beginning to dominate every field from medicine to academic research.The main services are:Saas,PaaS,IaaS (1) which are provided by major service vendors for use by various communities.The idea of "Load balancing"deals with the ability to evenly distribute the workload among a set of docker containers to maximize the throughput, minimize the response time, and thus increase the ability of the system to survive faults by preventing them from getting over-loaded.The idea behind reduction in energy consumption is concentrating the load on a subset of docker containers and, if possible, switching the rest of them to a state with a low energy consumption. This observation implies that the traditional concept of load balancing in a large-scale system could be reformulated as follows: distribute evenly the workload to the smallest set of servers operating at optimal or near-optimal energy levels, while observing the Service Level Agreement (SLA) between the CSP and a cloud user. An optimal energy level is defined as the ability of the system to operate at maximized performance per watt of power.

A recent survey reports that idle containers contribute about 12.2% of unnecessary CO_2 emissions each year and it costs around $20 billion. According to Gartner research, reports suggest that average server utilization in large data-centers is 15%, while the utilization of containers is even lower, 11%. These results confirm earlier estimations that the average container utilization is in the 12-22% range(2). The approach of load scheduling involves scheduling the load among the containers concentrating the load on them and their capacity.

II. ENERGY AWARE SCALING ALGORITHM

The main purpose of this algorithm is to ensure that largest number of conatiners are actively processing load within their optimal energy regime(3).The actions implementing this policy are: (a) migrate a container operating in the undesirable-low regime and then switch the containers to a sleep state; (b) switch an idle containers to a sleep state and reactivate them in a sleep state when the cluster load increases; (c) migrate the containers from the sleep state when the load increases.For example,when there are some 100 containers operating in the regime,out of which only a 60% are allocated with the load,the remaining 40% are idle,then they can be switched to the sleep state and when the amount of load increases in the regime ,then those needed among 40% are switched to operating state.

The algorithm can be as follows:

1.Get the request on demand.
2.Calculate the processing capabilities of each of the container operating in the regime.
3.Allocate the job to each existing container if,

```
While
{
   CPU<Threshold and RAM<Threshold;
}
Else
{
   CPU>Threshold and RAM>Threshold;
Then
   Create a new container by calling RMI Service;
   Allocate the load equally to the spawned container;
}
```

III. EXISTING SYSTEM

The existing system will schedule the jobs only for the high loaded containers.It will ignore the medium and lightly-loaded containers.As a result of which they will be operating and consuming the energy.

IV. PROPOSED SYSTEM

The Proposed system will schedule jobs for the medium loaded containers.It will make the low loaded containers to be idle.When the amount of load to be processed increases,a new container can be created additionally by simple api call.This ensures that the existing machines are not over-loaded.When the amount of load falls below a threshold value,then that container can be killed to save the energy utilized by it.

Advantage of proposed system:

The proposed system will consider energy usage as one of the criteria while scheduling the load.Also less load containers will not be allocated with jobs and it will be in sleep state to save energy.And moreover spawning of a new container ensures that the existing containers are not over-loaded with too many jobs to process.This ensures a high degree of reliability in load balancing.

V. ARCHITECTURE DIAGRAM

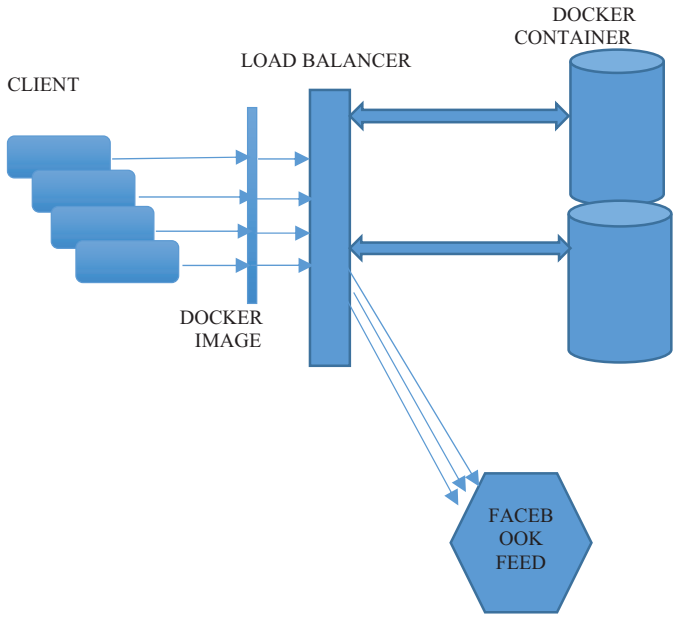

FIGURE1.1. ARCHITECTURE DIAGRAM

This model depicts the view about how the process works.When there is a request from the docker client to process the load(i.e,)the particular facebook feed,then the docker client sends a request to the docker daemon receives the request..Docker concept uses the client-server architecture.The Docker client and the daemon *can* be made to run on the same container,or a docker client can be connected to docker daemon.The Docker client and the docker daemon can communicate via a socket or they can do so via RESTful API(4)

It typically contains a docker image(3) which is pre-loaded with the available processing capabilities of the containers.Based on these values which it then compares with the threshold value,if it is within the optimal range,then those containers will be allocated with the jobs.When the threshold values exceeds,then a new container can be created within the cloud.This can be accomplished by the simple call to RMI service by the docker client.When there is a need to ensure that the newly created container is not needed,then it can be killed,thus releasing the memory and energy used by it.

VI. MODULE DESCRIPTION

A. *Retrieving container management status*

This module is a web application which will be deployed in a jboss server .The jboss server will be packed inside the docker container.This web application will retrieve and display the CPU and RAM used by the container.The parameters such as cpu load, Total Physical Memory Size can be determined.These parameters are considered when there is a need to create a new container.

B. *Implementation of Energy aware Scheduler*

This scheduler will schedule the jobs to the docker containers.The docker containers will have a web application which will receive the job to be processed.Then it executes the load allocated to it by the scheduler and sends the result back to the scheduler.

C. *Docker based container wake-up*

A high CPU load is noticed in the docker container based on the threshold value.Then the scheduler will spawn a new container and add the new container to load balance the jobs as shown in figure-1.2.If the load decreases,then the spawned container can be killed to ensure that amount of memory held by it is released as shown in figure-1.3.Thus,the remaining containers can process with high energy.

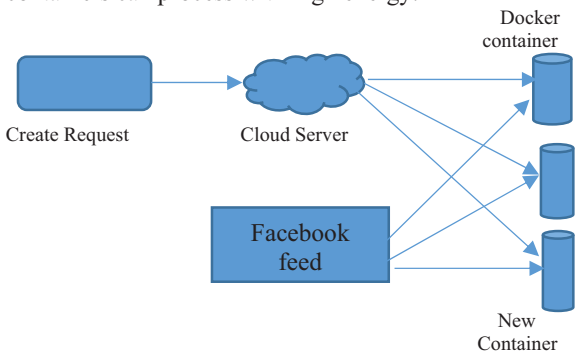

Figure1.2. Container Creation

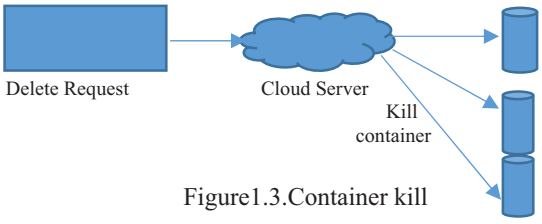

Figure1.3.Container kill

D. *Setting up the cloud and configuration*

Install docker and create docker image for the jboss server and the web application and start it in the cloud.This ensures that a view of container which are running can be effectively seen in the cloud.

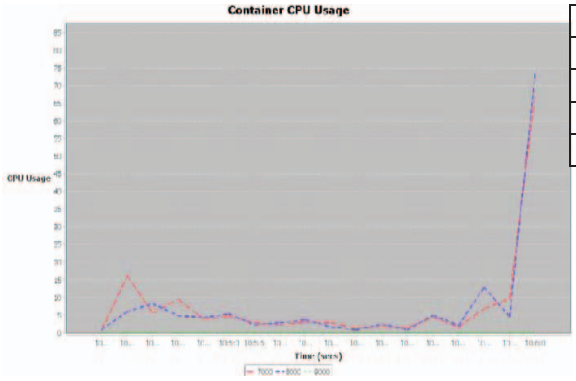

FIGURE1.4CPU USAGE OF CONTAINER-1 AND CONTAINER-2

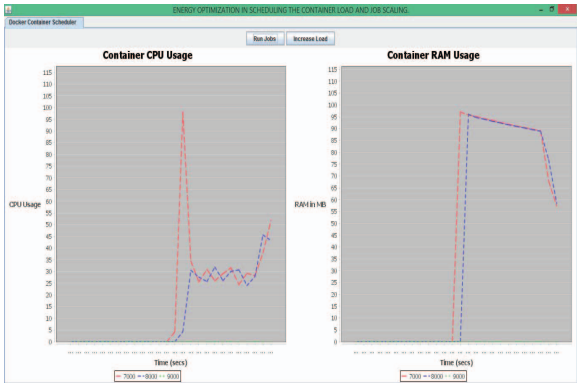

FIGURE1.5.CPU AND RAM USAGE BY CONTAINERS

TABLE1.1 CPU USAGE OF CONTAINER-1

S.No	Cpu usage(%)	Time(secs)
1	0	10:55:5
2	16	10:55:10
3	7	10:55:15
4	10	10:55:20
5	6	10:55:25
6	67	10:57:45
7	73	10:57:55

The table indicates the amount of cpu usage in % by the container-1.Intially it is created by the request of docker client to the docker daemon,which contains the docker image.The container is pulled up from the docker image by the scheduler.So initially it is not allocated with jobs,as time goes on it consumes the resources of the cpu to process the load efficiently.When a high load is to be processed by the container,then it consumes more memory.So a new containers can be additionally created which can be allocated with equal amount of load to be processed by the scheduler.

Table1.2 Cpu usage by container-2

S.NO	CPU USAGE(%)	TIME(secs)
1	0	10:55:56
2	3	10:56:01
3	10	10:56:06
4	7	10:56:11
5	4	10:57:16
6	63	10:57:31
7	77	10:57:41

VII. CONCLUSION AND RELATED WORK

The fact that power consumption of docker containers in data centers is significant and is expected to increase substantially.This motivates the interest of the research people to work in the energy optimization strategy when scheduling the jobs..Energy conservation is the major concern in cloud computing. By using this method we can save energy as much as possible with the help of docker container since it creates the container based on the load we processed. And it automatically shutdown the container if it has no jobs to process.Since everytime when the load is increased we are creating only instance of image not a new image so that we can save much time.

The containers operating in the undesirable low energy state need can be put into sleep mode.It is to ensure that they do not unnecessarily consume much of the resources that could otherwise be well utilized by the existing containers which are already in the process of executing the load allocated to them by the scheduler.

The future work involves creating mutilple containers to run in multiple machines.Now we have created multiple docker containers which can run only on single machine,therby utilizing the resources of that machine when it is processing the load.The current approach also involves automatic container creation to a certain amount.But in future it is expected that more number of container can be spawned and thus can efficiently process the load.

REFERENCES

[1] D. Ardagna, B. Panicucci, M. Trubian, and L. Zhang. “Energy-aware autonomic resource allocation in multitier virtualized environments.” IEEE Trans. on Services Computing, 5(1):2–19, 2012.

[2] J. Baliga, R.W.A. Ayre, K. Hinton, and R.S. Tucker. “Green cloud computing: balancing energy in processing, storage, and transport.” Proc. IEEE, 99(1):149-167, 2011.

[3] L. A. Barroso and U. H¨ozle. “The case for energyproportional computing.” IEEE Computer, 40(12):33– 37, 2007.

[4] A.Beloglazov and R. Buyya. “Managing overloaded hosts for dynamic consolidation on virtual machines in cloud centers under quality of service constraints.” IEEE Trans. on Parallel and Distributed Systems, 24(7):13661379, 2013.

[5] Ashkan Paya and Dan C. Marinescu,Energy-aware Load Balancing and Application Scaling for the Cloud Ecosystem,

10.1109/TCC.2015.2396059, IEEE Transactions on Cloud Computing.

[6] Google. “Google’s green computing: efficiency at scale.” http://static.googleusercontent.com/external content/untrusteddlcp/www.google.com/en/us/green/pdfs/google-green-computing.pdf (Accessed on August 29, 2013).