Container Scheduling Algorithm In Docker Based Cloud

Jigna Acharya1*, Anil C. Suthar2

1Research Scholar, Gujarat Technological University, Gujarat, India
2Director, New L J Institute of Technology, Gujarat Technological University, Gujarat, India

Abstract
In last few years, Docker container is used as light weight virtualization for building infrastructure as a service of cloud computing. Container placement on machines is traditional scheduling problem in docker based cloud. Docker supports default scheduling strategy called spread strategy. Its main objective is to place containers on each and every machine available in docker cluster. However, spread strategy does not consider load of machines. So, it leads to overloading or under loading of machines as well as not proper utilization of resources. In this paper, we proposed new scheduling algorithm that consider CPU usage of every machine in the docker cluster. To choose the most appropriate machine to place containers needed to be allocated in scheduling process, author define method for CPU utilization of every machine and select machine with minimum CPU utilization value for container placement. The experiment results shows that our scheduling algorithm use minimum number of machines in docker cluster as well as proper utilization of resources as compared to spread strategy.

Keywords: Docker, Container, Cloud computing, Spread Algorithm, Scheduling.

Introduction
Now a day, Container based virtualization is becoming more popular in cloud computing rather than virtual machine based virtualization. Virtual machine technology is totally different from container technology in cloud computing. Virtual machine technology virtualizes hardware and runs OS on every machine, which runs required application on the system. While in container technology, it runs application direct on the host machine. Therefore, container is light weight technology. As an emerging virtualization technology, Docker container has many advantages over traditional virtualization technologies.[1]

http://www.webology.or
Moreover, there are various service providers trying to mitigate their services from conventional clusters to container clusters for lesser costs and other benefits brought by containerization.[2] Docker containers are very fast to start and deploy in cloud computing environment. Kubernetes, docker swarm, apache and mesos are various frameworks available in cloud computing to place containers on machines. Various scheduling strategy is supported by these frameworks. Spread strategy of docker swarm executes a container on the node having the least number of containers. [3] However, large machines are involved to run services in cloud center. So, management and scheduling of containers on to physical machines are required to accommodate dynamically distributed workloads.[4]

To formulate scheduling of containers effective and fulfill the requirements of industry better, we did study on existing scheduling strategy of docker swarm. We find that spread strategy have some disadvantages when placing containers on machine in the cluster. For example, spread strategy waste storage space of machine and does not fully utilized resources of machines.

Based on docker swarm, we proposed novel scheduling algorithm that consider CPU usage of every node in the docker cluster before placing containers on machine. This strategy also utilizes maximum resources of machine so that we can overcome the problem that we find in spread strategy. The experiment outcome explain that the algorithm utilizes maximum resources of machine in terms of CPU usage of machine before placing of containers and also minimize machines in the docker cluster.

The reminder of paper is arranged as follows. Section 2 reviews Related Works. Section 3 represents problem description and formulation. Section 4 describes proposed algorithm. Experiments and results are represented in Section 5 and Section 6 represents Conclusion and Future work.

Related Work

In recent few years, there have been many research work done on container scheduling problem in docker based cloud computing. Yanghu Guo et al. [5] Propose container scheduling policy based on neighborhood division in micro service (CSBND). It works of load balancing and system response time to optimize system performance. Lianwan LI et al. [6] suggets a Particle Swarm Optimization-based container placement algorithm of Docker platform, which to have solved the problem of inadequate resource consumption and load balance. M.Suresh kumar et al. [7] Creates energy optimal model that can save energy of machine and automatically shut down the container if there is no process to run. Yanal Alahmad et al. [8] Suggests a novel Availability-Aware container scheduling strategy that aims to increase the availability level of the application service in the cloud container-based platform. Ruiting Zhou et al [9] presents scheduling algorithm that achieves computational and economic efficiency.

Jingze Lv et al. [10] Proposes container scheduling algorithm based on machine learning that lies in replacing the eigenvectors of the original random forest regression.
algorithm with those in micro services. With the help of this algorithm services give quick responses. Chanwit Kaewkasi et al. [11] represents ACO based container scheduling algorithm that distributes container on host machine in such a way that balance resource usage. Tarek Menouer et al. [12] proposed scheduling strategy that is based on a combination of PROMETHEE and Kung multi-objectives decision algorithms in order to place containers. It considers multi objective criteria like waiting number of CPU and free memory size. Xin Xu et al. [13] Propose stable machine theory for resource scheduling approach of container in cloud to reduce response time of job and improve resource utilization rate. Feifei Chen et al. [14] propose a container scheduling method based on the Min-Min. Target function of the physical machine is created for energy conservation and also consider current resource utilization rate. Andrew Chung et al. [15] Represents new stratus cluster scheduler that dynamically allocate container in heterogeneous cloud environment. These schedulers estimates cost of job at run time and also avoid machines that are not frequently utilized. Zhenjiang Li et al. [16] Propose virtual migration strategy, focusing on the energy consumption differences of different physical machines, can optimize the energy consumption of container cloud computing centers and effectively reduce the overall energy consumption to target the optimal utilization rate. Yang Hu et al. [17] propose an Enhanced Container Scheduler (ECSched) for efficiently scheduling concurrent container requests on heterogeneous clusters with multi-resource constraints. It is graph-based scheduler, which can control the minimum-cost flow model to efficiently process parallel container requirements.

**Problem Description & Formulation**

Default spread scheduling strategy in docker swarm does not consider load of machine while distributing containers on machine in the cluster. It works like round robin manner to place container. Because of above situation, proper resource utilization of machine is not done as well as every machine is used in cluster still there is no need to run machine. To overcome above situation, we proposed novel container scheduling algorithm here that will place container on machine whose CPU usage is minimum every time.

Suppose we have \( T = \{ t_1, t_2, \ldots, t_n \} \) images read from the workload file to create and start containers in the cluster. Each node system in the cluster is represented by \( S = \{ s_1, s_2, \ldots, s_m \} \). The time at which we read image from file and starting container on node machine is defined as \( T \rightarrow [0, \infty] \)

So, \( \text{Time (} t_j \text{)} = \text{installation time of task } t_j \)

\( = \text{Time (} j \text{)} \)

To choose machine in cluster for placing container, our proposed scheduling algorithm calculate CPU utilization of every machine in cluster at particular time. It is represented as below,
CUO: \([0, \infty] \rightarrow \mathbb{R}^n\)

\[
\text{CUO (Time (i))} = (\text{CU}_1, \text{CU}_2, ..., \text{CU}_n)_{\text{Time}(i)}
\]

After finding CPU utilization of every machine, our proposed scheduling algorithm will find machine in cluster whose CPU utilization is minimum. It is defined using min function,

\[
\text{Min} : \mathbb{R}^n \rightarrow \mathbb{R}
\]

\[
\text{Min} : (x_1, x_2, ..., x_n) = x_i
\]

\[
\text{Min} : (\text{CU}_1, \text{CU}_2, ..., \text{CU}_n) = \text{CU}_i
\]

Min function will return minimum CPU utilization of machine. To find index of system whose CPU utilization is minimum, it can be represented as below

\[
\text{I} : \mathbb{R} \rightarrow \mathbb{S}
\]

\[
\text{I} (\text{CU}_i) = S_i
\]

After getting index of machine, now our scheduling strategy is placing the container on machine is represented below

\[
\text{System} : \mathcal{T} \rightarrow \mathbb{S}
\]

\[
\text{System} (\mathcal{T}) = \{ (\text{I} \circ \text{Min}) \circ \text{CUO} \circ \text{Time} \circ (\mathcal{T}_i) \}
\]

\[
= \{ (\text{I} \circ \text{Min}) \circ \text{CUO} \} \circ \{ \text{Time}(j) \}
\]

\[
= \text{I}(\text{CU}_i)
\]

\[
= S_i
\]

\[
\text{System}(\mathcal{T}_j) = s_i
\]

**Proposed Algorithm**

<table>
<thead>
<tr>
<th>Algorithm: Proposed Container Scheduling Algorithm</th>
</tr>
</thead>
<tbody>
<tr>
<td>Input: ( { S_i }, { T_i } )</td>
</tr>
<tr>
<td>Output: ( \text{System}(T_j) = s_i )</td>
</tr>
<tr>
<td>1 For each ( T_i ) do</td>
</tr>
<tr>
<td>2 For each ( S_i ) do</td>
</tr>
<tr>
<td>3 Calculate ( \text{CUO(Time(i))} )</td>
</tr>
<tr>
<td>4 End</td>
</tr>
<tr>
<td>5 For each ( \text{CU}_i ) do</td>
</tr>
<tr>
<td>6 Calculate ( \text{Min}(U_i) )</td>
</tr>
<tr>
<td>7 End</td>
</tr>
<tr>
<td>8 Place container ( T_i ) on ( S_i )</td>
</tr>
<tr>
<td>9 End</td>
</tr>
</tbody>
</table>

**Experimental Environment**

630

http://www.webology.or
In this section, we implemented proposed container scheduling algorithm for docker based cloud environment. We used 3 machines Intel Core I3, CPU of 1 core, 3.4 GHz having 5 GB RAM with Linux Ubuntu 18.04 OS to form docker swarm. In docker swarm one machine is manager and remaining two are worker machines. The experiment uses prime number image for testing. This image is used to generate workload. We installed Prometheus, grafana, cadvisor and node exporter for monitoring load of machines. Our proposed scheduling strategy read image from workload file one by one and before placing container on machine it calculates CPU utilization of every machine. Machine having less CPU utilization is selected to place containers.

**Results and discussion**

In our experiment, the client request to create application in the container to find prime number from 1 to \( n \) number. The experiments were carried out under the scheduling of the Spread algorithm and our proposed algorithm. The aim of the proposed algorithm is to distribute the containers on available machines, whose CPU utilization is less. The results are compared with existing algorithm in following figures. Here we use three machines for placing containers, so in results all three machines are indicated with labels and different color in Prometheus monitoring tool. This tool fetches data using node exporter. For comparing results we use Prometheus queries node_load1, node_load5 and node_load15. These queries were used to find load of three machines under different circumstances.

![Graph showing results](image-url)
Figure 1: Initial container placement using spread algorithm (a) Node_load15 usage query (b) Node_load1 usage query (c) Node_load5 usage query

Figure 1 (a), (b) and (c) shows load of three machines for starting prime number application from last 5 minutes using Prometheus query node_load15, node_load1 and node_load5 respectively. It uses existing spread algorithm for container placement.
Figure 2: Initial container placement using Proposed algorithm (a) Node_load15 usage query (b) Node_load1 usage query (c) Node_load5 usage query

Figure 2 (a), (b) and (c) shows load of three machines for starting prime number application from last 15 minutes using prometheus query node_load15, node_load1 and node_load5 respectively. It uses proposed scheduling algorithm for container placement.
Figure 3 Node Load Performance under the Existing Spread Algorithm and Proposed Algorithm (a) Node_load15 usage query (b) Node_load1 usage query (c) Node_load5 usage query

Figure 3 (a), (b) and (c) shows results of three machines under existing spread algorithm and proposed algorithm using prometheus query node_load15, node_load1 and node_load5 respectively. From the figure3, we can find that proposed algorithm has better load balance for each node. For example, proposed algorithm not used last third machine for placing container after applying all Prometheus query while in existing spread algorithm used all three machines for placing all containers. Therefore, the proposed algorithm's container placement method load balance as compared to existing spread algorithm.

Conclusions

635
http://www.webology.or
After detail study of the spread scheduling strategy, we find that spread algorithm have some disadvantages while placing containers. It wastes resources of machine. Therefore, we proposed new container scheduling algorithm, which is implemented in docker swarm. Firstly, proposed container scheduling algorithm find CPU utilization of every machine at particular time interval and then among all CPU utilization it finds the machine having less CPU utilization. Less CPU utilization machine is selected for placing container. In this way, proposed scheduling algorithm minimizes resource waste.

In forthcoming study, additional research will be focused on memory utilization as well as reservation of resources. Heterogeneous machines can also be taken to implement proposed container scheduling algorithm. We will also try to implement binpack and random strategy and compare with our proposed container scheduling algorithm.

Acknowledgements

I would like to express my special thanks of appreciation to my research guide Dr. Anil Suthar for their able guidance and support to do my research work.

References


http://www.webology.or


