Efficient Load balancing to serve Heterogeneous Requests in Clustered Systems using Kubernetes

1st Amit Dua
Department of Computer Science and Information Systems
BITS, Pilani
Pilani, India
amit.dua@pilani.bits-pilani.ac.in

2nd Sahil Ranadive
Department of Computer Science and Information Systems
BITS, Pilani
Pilani, India
f2016097@pilani.bits-pilani.ac.in

3rd Aditi Agarwal
Department of Computer Science and Information Systems
BITS, Pilani
Pilani, India
f2016095@pilani.bits-pilani.ac.in

4th Neeraj Kumar
Computer Science & Engineering Department
Thapar University
Patiala (Punjab), India
neeraj.kumar@thapar.edu

***Abstract*—Load balancing is an important part of a distributed computing environment which ensures that all devices or processors perform the same amount of work in an equal amount of time. Most load balancing algorithms assume similar processing power and workload for all the processors. However, now systems have become more complex and can have processors of different capabilities, workload, and configurations. In this paper, we propose an alternative algorithm for scheduling tasks. We configure the clusters dedicated to a particular type of task(real-time, dataintensive, etc.). Labels have been defined for each job to classify them into these categories. Then we modify the algorithm to introduce load balancing techniques using task migration.**

***Index Terms*—clustering, kubernetes, load balancing**

I. Introduction

Load Balancing techniques for distributed computing system assume that the processors have similar processing power and similar workload. This study proposes a load balancing algorithm for heterogeneous tasks. We compare the performance of the proposed algorithm, with that of Kubernetes, a container orchestration system for automating deployment, scaling and management. Given some jobs first we assign them a label denoting the type of task they can be classified into(ex, real time, data intensive, others). We also try to approximate the size of the tasks. Using these labels and size of tasks we devise an efficient load balancing and scheduling algorithm for reducing the response time of the service.

II. Related Work

Various algorithms are discussed by S. Aslam et al.such as round robin, Min-Min, Max-Min and Ant colony algorithm, Throttled load balancing and Carton [1]. Improving response time for user requests is a critical part of cloud computing(also for distributed computing) and bandwidth from servers to clients is often a bottleneck, so a Throttled Modified Algorithm is proposed by N. X. Phi et al. [2]. The next three sections of the paper describe the system model, load balancing algorithm and performance evaluation.

III. System Model

The system is modelled as stream of incoming jobs and set of clusters which process the jobs. Jobs have been classified into three categories: 1) Data intensive jobs: These are jobs which request large portions of the RAM for processing. 2) Real time jobs: These jobs are required to be processed quickly. 3) Other jobs. On the basis of the above three categories, three clusters have been made. Each of these clusters have been assigned a set of resources(fast memory or RAM, processing power and disk space). Let j_1,..,j_n is the incoming stream of jobs. Timestamps of their start time are denoted as s_1,s_2,.....,s_n respectively. The RAM/memory consumed by the jobs are denoted as d_1,..,d_n respectively .

$$\forall i \in \{1, 2, ...n\}, d_i > 0,\ s_i >= 0 \tag{1}$$

Through the algorithm we aim to obtain wait times w_1,w_2,.....,w_n, run times r_1,r_2,.....,r_n and end times e_1,e_2,.....,e_n for all the jobs. We know that simply, for any job j_i,end time can be calculated as

$$e_\mathrm{i} = s_\mathrm{i} + w_\mathrm{i} + r_\mathrm{i} \tag{2}$$

Note that end time for a job is a timestamp, just like start time.

For any job j_i running in cluster C_k, runtime r_{ik} can be calculated as:

$$r_\mathrm{ik} = d_\mathrm{i}/P_\mathrm{k} \tag{3}$$

To keep a track of jobs which are being processed in a cluster we maintain arrays A_1,A_2 and A_3 for each of the cluster. These arrays keep track of jobs being currently

processed in the cluster. We can say that at any time t, for cluster C_k

$$\sum_{i \in q_{it}} d_i <= D_k \tag{4}$$

IV. Proposed Load Balancing Algorithm

input : An array *jobs* of size n
output: An efficient assignment of jobs to clusters

```
for i ← 1 to n do
    label ← checkLabel(jobs[i]);
    if label == 0 then
    flag ← jobToCluster(cluster0,jobs[i]);
    else if label == 1 then
    flag ← jobToCluster(cluster1,jobs[i]);
    else
    flag ← jobToCluster(cluster2,jobs[i]);
    // check if job is not scheduled
    if flag == 0 then putAtEnd(jobs,i);
end
// now perform migration
waitTime0 ← compTimeLastJob(cluster0);
waitTime1 ← compTimeLastJob(cluster1);
waitTime2 ← compTimeLastJob(cluster2);
i ←min(waitTime0,waitTime1,waitTime2);
j ←max(waitTime0,waitTime1,waitTime2);
diff ←diffInWaitTimes(i,j);
while diff >threshhold do
    // put last job of clusterj in
       clusteri
    migrate(clusterj , clusteri);
    // recompute the wait times of
       jobs in all clusters
    waitTime0 ← compTimeLastJob(cluster0);
    waitTime1 ← compTimeLastJob(cluster1);
    waitTime2 ← compTimeLastJob(cluster2);
    i ←min(waitTime0,waitTime1,waitTime2);
    j ←max(waitTime0,waitTime1,waitTime2);
    diff ←diffInWaitTimes(i,j);
end
```

Algorithm 1: Scheduling and Migration Algorithm

The initial scheduling requires iterating through all the n jobs which yeilds a time complexity of $O(n)$. The combined time complexity of scheduling and migration is $O(n)$ as well.

The space complexiy of the algorithm is $O(n)$.

V. Experiments and Results

The study evaluates the scheduling algorithm using makespan, throughput and response time metrics. The mathematical presentation of makespan is

$$Makespan = \max_{\forall j \in \{1,2,3\}} T_j \tag{5}$$

Throughput is defined as

$$Throughput = \frac{n}{Makespan} \tag{6}$$

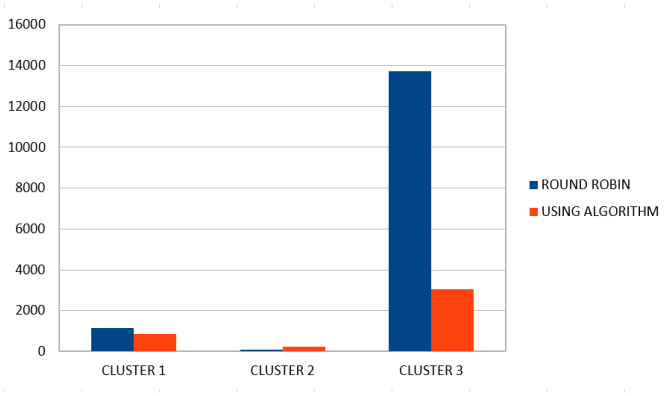

Fig. 1. Comparison of round robin scheduling with the proposed scheduling.

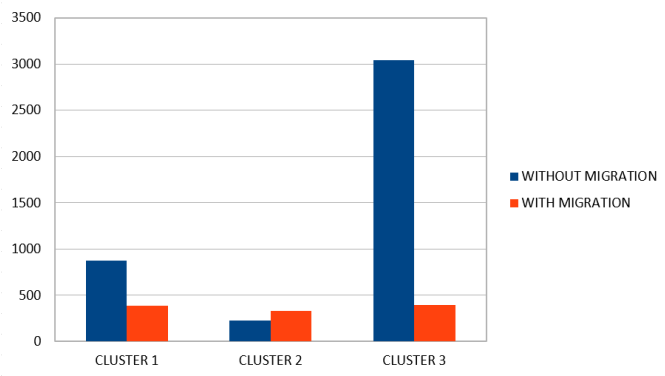

Fig. 2. Performance results with and without task migration.

Response time is the time taken to respond to a request for service, i.e., response to a job.

Average response time can be calculated as

$$R = \sum_{i=1}^{n} r_i \tag{7}$$

The experiment has been carried out on a list of 1000 jobs with varying expected running time and data.

As shown in graph1 the processing time for cluster1 reduces from 1152s to 872s, reduces from 13721s to 3045s for cluster3 whereas it increases for cluster2 from 88s to 230s. The overall throughput has increased from 0.0668 to 0.2411.

Graph2 shows the comparison between the proposed algorithm, with and without migration.

The processing time for cluster1 reduces from 872s to 387s, reduces from 3045s to 397s for cluster3 whereas it increases for cluster2 from 239s to 332s.

The overall throughput has further increased from 0.2411 to 0.896.

References

[1] S. Aslam and M. A. Shah, *Load Balancing Algorithms in Cloud Computing: A Survey of Modern Techniques*, 2015 National Software Engineering Conference, 2015.

[2] N. X. Phi, C. T. Tin, L. N. K. Thu, T. C. Hung, *Proposed Load Balancing Algorithm To Reduce Response Time And Processing Time On Cloud Computing*, International Journal of Computer Networks and Communications (IJCNC) Vol.10, No.3, 2018.