ISSN 2347 - 3983
Volume 8. No. 5, May 2020
International Journal of Emerging Trends in Engineering Research
Available Online at http://www.warse.org/IJETER/static/pdf/file/ijeter99852020.pdf
https://doi.org/10.30534/ijeter/2020/99852020

Load Balancing Analyzer: A Recommendation System using Machine Learning

Ravi Kumar Tata [1], Saiteja Mothe[2], Prabhat Koneru[3], Venkata Ramana N[4], B Sadhana[5]
[1]Koneru Lakshmaiah Education Foundation, India, rktata@kluniversity.in
[2]Koneru Lakshmaiah Education Foundation, India, saiteja.mothe6221@gmail.com
[3]Koneru Lakshmaiah Education Foundation, India, saiprabhat7@gmail.com
[4]Koneru Lakshmaiah Education Foundation, India, ramana@kluniversity.in
[4]Koneru Lakshmaiah Education Foundation, India, sadhana@kluniversity.in

ABSTRACT

The Parallel and distributed systems focus on the concept of load balancing. A tangle is split into a hard and fast number of processors that are to be executed in parallel. However, there may be a state that some processors will complete their tasks before other processes and reach idle state as the work is unevenly divided or perhaps some processors complete before the others. Ideally, we like all the processors to have the minimum wait time. Achieving the above goal by spreading the tasks is eventually named as load balancing. In the paper, we stress on clustering techniques. The basis of the join-idle-queue algorithm is seen by using clustering. The technique of load balancing uses Support Vector Machine (SVM) and clustering techniques (K-means, Hierarchical). A comparative study of the above techniques is done utilizing load balancing.

Key words: Load Balancing, Support Vector Machine, K-Means, Clustering Techniques, R Studio.

1. INTRODUCTION

An easy way to enhance and implement the load balance is network load balancing. Incoming client requests are load balanced by internet applications. It allows system administrators to make clusters. In-network load balancing, clients cannot differentiate the cluster from one server. Programs of the server also are not aware that a cluster is running. Any point of a network allows remote cluster management and greater overall control in Network load balancing (NLB). Clusters are given to services in Port-defined controls by administrators.

Other hosts' presence is often monitored by sending regular messages to all or any cluster members. Recovery and failure of the host are often handled automatically and quickly. To handle network traffic, network load balancing software requires low overhead. Excellent performance scaling is delivered by the process, limited by subnet bandwidth. The single policy does not integrate these criteria. Every policy concentrates on one or more criteria based on web server load balancing. Proper load balancing is achieved when all these criteria are load balanced by each policy across the farms of a web server. To provide optimization, each policy can be extended further when load balancing takes place.

Services of TCP/IP include services of the terminal, and also media services are streamed, which provides high scalability and availability for enterprising. Applications of transaction, databases (back-end) link clients and display internet protocol services like web applications that bring extraordinary value to network load balancing. During a cluster, hosts called (Network load balancing servers) communicate among themselves for providing key benefits, which include scalability and high availability.

2. TYPES OF LOAD BALANCING ALGORITHMS

The algorithms of Load balance are mainly classified into two types. They are static and dynamic load balancing. At the time of compile, tasks are distributed to a various number of processors in Static load balancing. At run time, different types of tasks are together collected in dynamic algorithms [1]. Structure of electronics and dynamics of molecular are intermittent problems that are solved using parallel techniques of genetic algorithms that support refinement mesh [2]. Rebalance is done by providing information on the way in which it has to reduce in dynamic.

There are three steps to balance a dataset they are evaluating, which includes imbalance, balancing way should be decided, and also redistribution is done [3]. Algorithms [4] are again distributed, and they include static algorithms, dynamic and adaptive algorithms. The workload is distributed based on the previous experience, and it characterizes the system in Static. State information is used in the execution of the program to make decisions in dynamic decisions. At last, adaptive

algorithms are a part of dynamic. Behavior is adapted dynamically, and parameters are also changed according to the requirements. Centralized and distributed are two different types of dynamic load balancing [4].

Decisions of balancing load are made in different strategies. All decisions are made in a single processor called centralized processor in the centralized scheme. Replication of load balancer is present on every processor in the scheme of distribution. Estimates of unpredictable load are essential and efficient in systems of highly parallel [5]. The workload of the CPU and the service of the multiprocessor are increasing more and more [6]. The problem of load balancing arises when all the processors are of different speeds. If one processor has high speed, it completes its work before other processors. With this problem, performance is degraded in a computer system [13]

3. PARAMETERS OF LOAD BALANCING ALGORITHMS

The different parameters that are required to present the differed load adjusting calculations are

3.1 Overload Rejection

The extra over-loaded dismissal measures are required if the balancing of the load is beyond imagination. The first over-burden dismissal measures come to a stop at the point when the over-load circumstances close. Moreover, load balancing is ceased after this instance occurs.

3.2 Fault-Tolerant

This parameter tells us that calculation is in a situation to experience the convoluted flaws. It also legitimizes a calculation to keep going accordingly in case of some obstacle or failure. On the off chance that the exhibition of the calculation declines, the abatement is proportionate to the intensity of disappointment. Even a little can cause absolute declination [12].

3.3 Normalizing Image Inputs

This technique ensures the distribution of the data similarly according to the input parameters that are set previously. It helps in faster training of the data. The normalization is performed by deducting the average from each one of the pixels and dividing the obtained result with standard deviation. [6]

3.4 Stability

This is generally defined as far as holdovers inside interchange of information among processors and along lines the add-overs inside heap adjusting calculation by attaining quicker execution by a foreordained measure of time.

3.5 Nature of Load Balancing Algorithms

Static burden adjusting doles out burden to hubs probably rudely occasions. It is very challenging to build assumptions of appearance times of burdens and handling required for future burdens. While the other hand, during a robust burden, adjusting heap dissemination is shaped at run-time upheld present preparing rates of system conditions. A data balancing arrangement is used in neighborhood and comprehensive data.

3.6 Cooperative

Sharing of data between processors is done to settle the assignment technique while other processors are not performing the execution. This parameter characterizes the autonomy degree so that every processor is in reasoning what manner it can utilize its autonomous assets. Inside favorable circumstances, many processors are responsible for holding their very self-bit of planning goals [9]. The processors need to cooperate with a higher-level objective.

3.7 Process Migration

Relocation type gives when the framework intends for the procedure. They can be implemented locally or remotely. The calculation is complicated to choose whether to make a change in load or in procedure execution.

3.8 Resource Utilization

This parameter usage is to incorporate programmed weight adjusting. The disseminated framework contains an infinite number of procedures that request all the more handling force. In the event that the calculation is proficient at using assets, they will be moved to underloaded processors more efficiently.

4. METHODOLOGY

4.1 SVM Classification

This technique of classification is mostly used as it gives correct accuracy with low power. SVM full form is a support vector machine and also used for various tasks (Classification, regression). However, examples of classifying are most probable. The main aim of this algorithm is to get a hyperplane that separately classifies the data points. Many hyperplanes can be produced to separate the two different types of data. Points of data are also called support vectors, which are dependent on the plane with respect to orientation. Margin can be maximized using these vectors. The position of a plane can be changed by deleting the points [10]

4.2 K-Means

A different set of observations are given, then we have to categorize into clusters. We use Euclidean distance as a

measurement in k-Means clustering. K groups of similarity are categorized by this algorithm [7]. Certain features and sets of data items and values of vectors are given in the input data. There are three points that are to be discussed in this algorithm. They are

a) Randomly points are initialized.

b) Updating of means coordinates are done by categorizing each item.

c) For any number of iterations, we repeat the process until we found the clusters.

The group of similar data is categorized and then formed into clusters. A single mean vector method is used for categorizing each cluster. In an unsupervised machine learning algorithm, there is unlabeled data that is divided into clusters. If one of the clusters has been failed, then other clusters take up the workload of this cluster [8].

a) Recognizing the two packs that are closest together.

b) Mix the two most similar gatherings.

This returns until all of the gatherings are joined. It appears in the frameworks underneath. In data mining and estimations, different leveled batching assessment is a methodology for bunch examination which hopes to fabricate a chain of the significance of bundles, for instance, tree-type structure subject to the movement [9].

4.3 Hierarchical Clustering

Tree type structure is shaped in progressive put together strategies based on respect to the chain of command. Various leveled bundling starts by viewing each recognition as an alternate gathering [11]. By then, it, again and again, executes the going with two phases:

a) Recognizing the two packs that are closest together.

b) Mix the two most similar gatherings.

This returns until all of the gatherings are joined. It appears in the frameworks underneath. In data mining and estimations, different leveled batching assessment is a methodology for bunch examination which hopes to fabricate a chain of the significance of bundles, for instance, tree-type structure subject to the movement [9].

4.4 Implementation

a) Collect data and load the dataset

b) Install the required packages

c) Pre-process the data

- Remove noise from the data.

4.4.1 For SVM Classification

a) Split the data

- Training (70%)
- Testing (30%)

b) Apply the formulae of SVM on training data.

c) Predict the graph.

d) Testing is done on the testing data.

```
install.packages("caTools")
library(caTools)
set.seed(123)
split=sample.split(dataset$priority,SplitRatio = 0.7)
training_set=subset(dataset,split==TRUE)
test_set=subset(dataset,split==FALSE)
split
View(training_set)
View(test_set)
install.packages('e1071')
library(e1071)
classifier = svm(formula = priority~ .,data = training_set,type = 'C-classification',kernel =
classifier
y_pred=predict(classifier,newdata=test_set)
y_pred
cm = table(test_set[,3], y_pred)
cm
install.packages("ElemStatLearn")
library("ElemStatLearn")
set = training_set
X1 = seq(min(set[, 1]) - 1, max(set[, 1]) + 1, by = 0.01)
```

Figure 1: Implementation of R-code for SVM Classification

4.4.2 For K-Means Clustering

a) Give the number of clusters

b) By using silhouette function, clusters are defined.

```
str(dataset)
View(dataset)
dataset$stability=ifelse(is.na(dataset$stability), ave(dataset$stability, FUN = function(stabi
dataset$stability
dataset$resource=ifelse(is.na(dataset$resource), ave(dataset$resource, FUN = function(resource
dataset$resource
dataset$faultolerant=ifelse(is.na(dataset$faultolerant), ave(dataset$faultolerant, FUN = funct
dataset$faultolerant
dataset$faccuracy=ifelse(is.na(dataset$faccuracy), ave(dataset$faccuracy, FUN = function(faccu
dataset$faccuracy
dataset$priority = factor(dataset$priority,levels = c('B', 'L', 'R'),labels = c(1, 2, 3))
dataset$priority
set.seed(123)
clustering <- kmeans(dataset, centers = 2, nstart = 20)
clustering
install.packages("cluster")
library(cluster)
install.packages("factoextra")
library(factoextra)
sil <- silhouette(clustering$cluster, dist(dataset))
fviz_silhouette(sil)
```

Figure 2: Implementation of R-code for K-Means Clustering

4.4.2 For Hierarchal Clustering

a) No. of clusters are defined

b) By using hclust function, a dendrogram is formed.

```
dataset$faultolerant=ifelse(is.na(dataset$faultolerant), ave(dataset$faultolerant, FUN = funct
dataset$faultolerant
dataset$faccuracy=ifelse(is.na(dataset$faccuracy), ave(dataset$faccuracy, FUN = function(faccu
dataset$faccuracy
dataset$priority = factor(dataset$priority,levels = c('B', 'L', 'R'),labels = c(1, 2, 3))
dataset$priority
a=dataset[,1:3]
a
View(a)
clusters=hclust(dist(a))
clusters
plot(clusters)
cut=cutree(clusters,4)
cut
table(cut,dataset$priority)
clusters=hclust(dist(a),method='average')
clusters
plot(clusters)
cut=cutree(clusters,3)
cut
table(cut,dataset$priority)
```

Figure 3: Implementation of R-code for Hierarchal Clustering

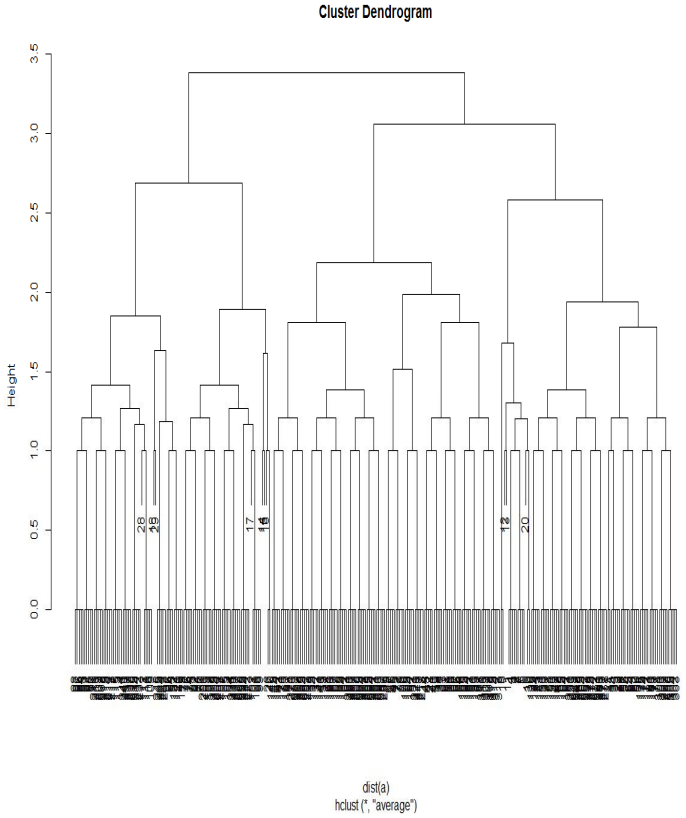

Figure 4: Dendrogram of the Hierarchal Clustering

5. RESULTS

The below-shown figures are the outputs of different Machine Learning techniques by using the load balancing dataset. Figure 3. represents the output of data analysis of the load balancing data by taking the consideration of parameters of fault-tolerant and Resource Utilization.

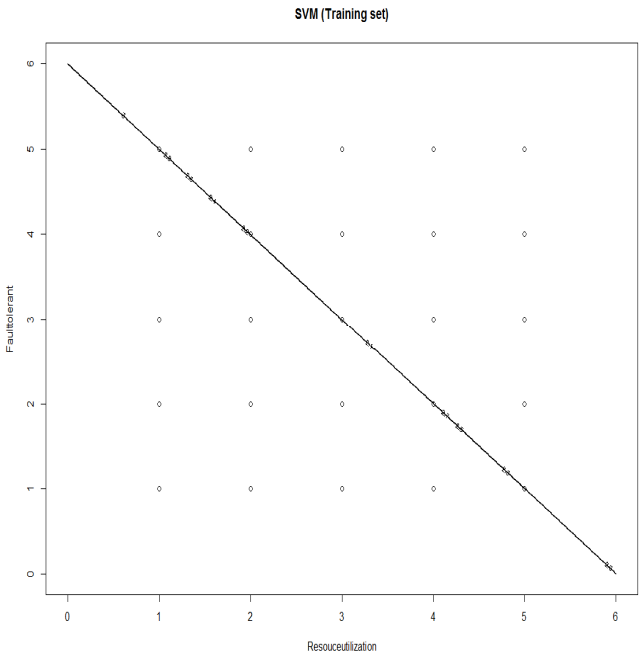

Figure 4: Plot of the data analysis of the dataset

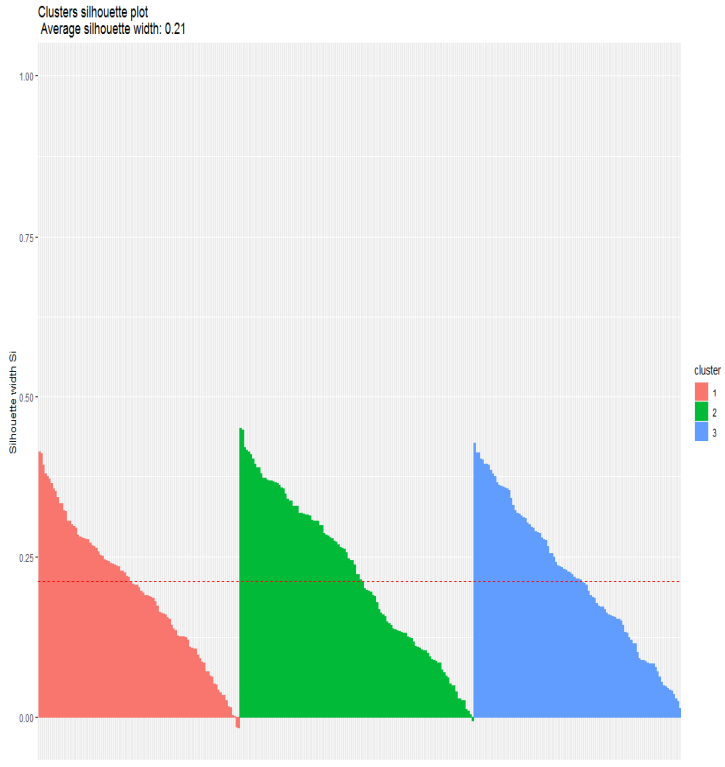

Figure 5: Silhouette Plot of the K-Means Clustering with 3 Clusters

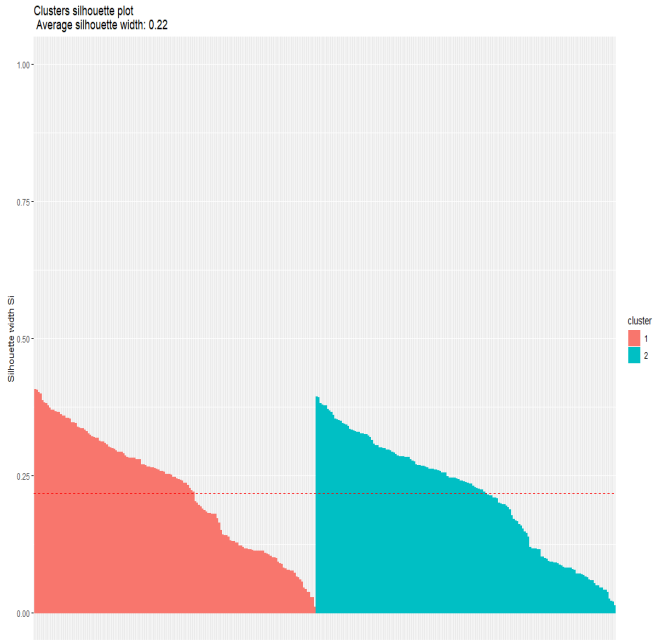

Figure 7: Silhouette Plot of the K-Means Clustering with 2 Clusters

6. CONCLUSION

Parallel computations have a significant problem with load balancing. If load had evenly distributed among all different processors' high efficiency is achieved. We have implemented SVM classification, K-means clustering, and hierarchical clustering for the load balancing dataset (table 1). SVM classification creates a hyperplane by separating into two classes by taking the consideration of parameters like fault-tolerant and forecasting accuracy. K-means is the clustering that divides data into the number of clusters that are most suitable by the dataset. Likewise, hierarchical clustering also segregates the data in the form of clusters. Moreover, the hierarchical comparison clustering and K-means clustering is done.

Table 1: Comparison table of Hierarchal and K-Means Clustering

	Hierarchal	K-Means
Clusters	The output is only tree	Exactly K no.of clusters
Running Time	Slower	Faster
Assumptions	Requires Distance Metric	Requires distance metric
Parameters	No.of clusters are not defined	No.of clusters are defined
Data	It can handle less data	It can handle more data
Time Complexity	$O(n^2)$	O(n)

6. FUTURE SCOPE

For more efficiency, Load balancing can also be implemented through Hadoop and spark using big data analytics. R-studio has many limitations, and it is not much efficient for the concept of load balancing. Spark supports clustering technology, and it has a faster speed. MapReduce model in Hadoop solves many problems of computation, queries have interacted, and processing of stream is done. The Spark of feature is in its memory cluster that is computing raises the processing of speed in the load balancing.

REFERENCES

1. Mandal, A. and Chandra, S. (2010) "**An Empirical Study and Analysis of the Dynamic Load Balancing Algorithms Used in Parallel Computing Systems**," Proceedings of ICCS-2010,19-20 Nov. West Bengal: University of North Bengal.
doi: 10.1109/IconDSC.2019.8816881
2. A. Roshdy, A. Gaber, F. Hantera and M. ElSebai, "**Mobility load balancing using machine learning with case study in live network**," 2018 International Conference on Innovative Trends in Computer Engineering (ITCE), Aswan, 2018, pp. 145-150.
doi: 10.1109/ITCE.2018.8316614
3. H. Yao, X. Yuan, P. Zhang, J. Wang, C. Jiang and M. Guizani, "**A Machine Learning Approach of Load Balance Routing to Support Next-Generation Wireless Networks**," 2019 15th International Wireless Communications & Mobile Computing Conference (IWCMC), Tangier, Morocco, 2019, pp. 1317-1322,
doi: 10.1109/IWCMC.2019.8766546
4. LeMair W.H. and Reeves P. A. (1993) "**Strategies for Dynamic Load Balancing on Highly Parallel Computers**," IEEE Transactions on Parallel and Distributed Systems. Vol. 4, No. 9
5. Pandey, S. K., and Tiwari, R. (2013). "**The Efficient load balancing in the parallel computer**," International Journal of Advanced Research in Computer Engineering & Technology (IJARCET) Volume 2, Issue 4, April 2013
6. Krishnamoorthy, S., Sadayappan P., Nieplocha P., and Krishnan M. (2005), "**Locality-aware Load Balancing for Dynamic and Irregular Computations**," 1st Workshop on Patterns in High-Performance Computing. Urbana-Champaign.
7. Pearce, O., Gamblin, T., Supinskiy, B., Schulzy, M., and Amato, M. (2012), "**Quantifying the Effectiveness of Load Balance Algorithms,**" US: Department of Energy.
8. Aldasht, M., Ortega, J. und Puntonet, C. (2007), "**Dynamic Load Balancing in Heterogeneous Clusters: Exploitation of the Processing Power**," 2nd Palestinian International Conference on Computer and Information Technology (PICCIT), Hebron, Palestine.

9. Mothe, Saiteja. (2020). "**A Model for Assessing the Nature of Car Crashes using Convolutional Neural Networks**." International Journal of Emerging Trends in Engineering Research. 8. 859-863. doi: 10.30534/ijeter/2020/41832020.
10. Tata, Ravi Kumar & Yasaswini, P & Rafi, G & Krishna, Dhulipalla. (2018), "**Comparative analysis on job prediction of students based on resume using data mining techniques.**" International Journal of Engineering & Technology. 7. 1100. doi: 10.14419/ijet.v7i2.7.12236.
11. Fouad, Amal. (2019). "**MRI Brain Cancer Diagnosis Approach Using Gabor Filter and Support Vector Machine**." International Journal of Emerging Trends in Engineering Research. 7. 907-914. doi:10.30534/ijeter/2019/297122019.
12. Satish Babu, Jampani & Tata, Ravi Kumar & Bano, Shahana. (2018). "**Optimizing webpage relevancy using page ranking and content based ranking"**. International Journal of Engineering & Technology. 7. 1025. doi: 10.14419/ijet.v7i2.7.12220.
13. NVS, Pavan. (2019). "**Mining Negative Associations between Regular and Frequent Patterns hidden in Static Databases**." International Journal of Emerging Trends in Engineering Research. 7. 54-67. 10.30534/ijeter/2019/04772019.