LOAD BALANCING IN PUBLIC CLOUD COMBINING THE CONCEPTS OF DATA MINING AND NETWORKING

Priyanka R

M. Tech Student, Dept. of Computer Science and Engineering, AIET, Karnataka, India

Abstract

Load balancing in the cloud computing environment has an important impact on the performance of whole system. A Good load balancing method makes cloud computing more efficient with increased user satisfaction. The combined concepts of networking, data mining and cloud computing technology are used to achieve a good load balancing strategy. Cloud partitioning helps to simplify the load balancing problem. The algorithms from networking and data mining are used to achieve cloud partitions. A cloud partitioning concepts introduced here is used in load balancing model for the public cloud with a switch mechanism to choose different strategies for different situations. The algorithm applies the round robin and game theory after cloud partitioning as the load balancing strategy to improve the efficiency in the public cloud environment.

Keywords: *Load balancing, public cloud, Cloud partition*

---***---

1. INTRODUCTION

Cloud Computing, the long-held dream of computing is similar to utility computing. It is expected to accelerate innovation and business agility of the IT industry, enabling software to become more attractive as a service and shaping the way IT hardware is designed and purchased [1]. Cloud computing provides a distributed computing environment that focuses on providing a wide range of users with distributed access to virtualized, scalable hardware and/or software infrastructure over the internet. Cloud computing provides hardware and software packages which are delivered as a service to clients over an outsized scale network [2].

1.1 Load balancing concepts

Load balancing is a process of distributing the total load on to the individual nodes of the collective system to maximize throughput, resource utilization and to minimize the response time, along with removing a condition in which some of the nodes are heavily loaded while some others are light. [3]

Load balancer is a software program which receives connection request from clients and forwards it to one of the backend server which replies accordingly. Because of this the client will be unaware of where the data is stored. This separation also has security benefit which hides the structure of internal network and helps to prevent attacks.

In cloud computing environment, there is random arrival of jobs with random CPU utilization. Such requirements can load a specific resources heavily, while the other resources are less loaded.

2. RELATED WORK

There are many studies being conducted in this stream. However, load balancing in the cloud is still a new problem that needs new architectures which can adapt to changing needs. Soumya Ray and Ajanta De Sarkar [15] proposed a brief review of existing load balancing algorithms. K. Ramana, A. Subramanyam and A. Ananda Raohave [16] have put forth that Load balancing algorithm tries to balance the total system load by transferring the workload from heavily loaded nodes to lightly loaded nodes and also presented the performance analysis of various load balancing algorithms based on different qualitative parameters, considering static and dynamic load balancing approaches. In the existing system the partitioning of cloud is based on area which might not always be well suited in all situations. Hence, there is a need of new partitioning method using which we can achieve a better performance.

3. SYSTEM MODEL

The load balancing strategy involves creating cloud partitions. A cloud partition is a sub area of public cloud. Here the divisions are based on the VDBSCAN algorithm. Once the public cloud is partitioned, then the load balancing starts when a job arrives at the system. The main controller decides which cloud partition should receive the job. The load balancer assigned for each partition then decides how to assign the jobs to the nodes. When the load status of a cloud partition is idle or normal, this task can is accomplished locally. If the cloud partition is overloaded, this job should be transferred to another partition. The whole process is shown in Fig.1.

We are using following techniques.

1. Dijkstra's algorithm to find the shortest distance between two nodes
2. VDBSCAN Cluster based Cloud Partition
3. Round-Robin Technique
4. Game Theory Technique.

Following algorithm shows the procedure

Algorithm 1

```
Begin
    Apply Dijkstra's algorithm to find the shortest path
    Apply VDBSCAN algorithm to find clusters and
    make each cluster as a partition
    while job do
        searchBestPartition (job);
        if (partitionState == idle || partitionState ==
        normal) then
            Send Job to Partition;
        else
            search for another Partition;
        end if
    end while
end
```

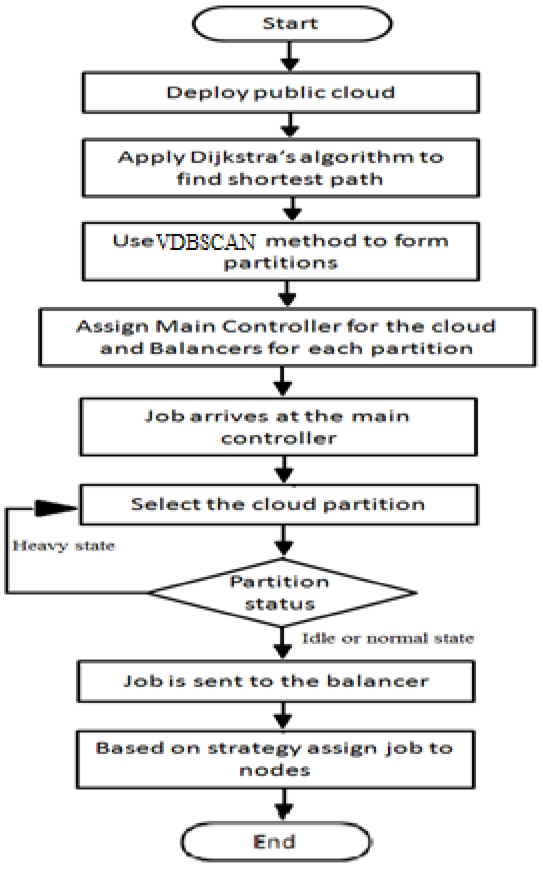
Start
Deploy public cloud
Apply Dijkstra's algorithm to find shortest path
Use VDBSCAN method to form partitions
Assign Main Controller for the cloud and Balancers for each partition
Job arrives at the main controller
Select the cloud partition
Heavy state
Partition status
Idle or normal state
Job is sent to the balancer
Based on strategy assign job to nodes
End

Fig -1: Job assignment strategy

3.1 Cloud Partition

A cloud partition is a sub area of public cloud. Partitioning will simplify the process of load balancing.

3.1.1 Dijkstra's Algorithm

The use of networking concept will determine the distance at which each server is placed. For this, here, we use distance formula to find weights and then use the dijkstra's algorithm [4]. Dijkstra's algorithm solves the single-source shortest-paths problem on a directed graph G= (V,E) and all edges have non-negative weights. In this section, therefore, we assume that w(u,v)>=0 for each edge for each edge E(u,v) [5].

Dijkstra's algorithm maintains a set S of vertices whose final shortest-path weights from the source S have already been determined. The algorithm iteratively selects the vertex u ∈ V-S with the minimum shortest-path estimate, adds u to set S , and relaxes all edges leaving u. Here, we use a min-priority queue Q of vertices, keyed by their values of d. The following algorithm is referred from [5].

```
DIJKSTRA(G,w,s)
 INITIALIZE -SINGLE-SOURCE(.G, s)
 S=φ;
 Q=G.V
while Q!=φ
    u=EXTRACT-MIN(Q)
    S=SU{u}
    for each vertex v ∈G.Adj[u]
        RELAX (u,v,w)
```

Thus, using the Dijkstra's algorithm we first find the shortest path and then we use the shortest path concept in VDBSCAN method to develop clusters.

3.1.2 DBSCAN

In the pervious step, the shortest distance between any two points is calculated. This value is used in next step to form clusters. Here we use VDBSCAN(Varied Density Based SCANning) method to form clusters.

DBSCAN can find clusters of arbitrary shape. However, clusters that lie close to each other tend to belong to the same class. Its computing process is based on six rules or definitions, creating two lemmas [6][13].

Definition 1: (The Eps-neighbourhood of a point) The $N_{Eps}(p)$ represents the Eps-neighbourhood of a point p and is defined by

$$N_{Eps}(p) = \{q \in D | dist(p,q) < Eps\} \quad (1)$$

Where D is set of given points. For a point to belong to a cluster it needs to have at least one other point that lies closer to it than the distance Eps.

Definition 2: (Directly density-reachable) There are two kinds of points belonging to a cluster; there are border points and core points [6].

“The Eps-neighborhood of a border point tends to have significantly less points than the Eps-neighborhood of a core point”.

The border points will still be a part of the cluster and in order to include these points, they must belong to the Eps-neighborhood of a core point q.

$$p \in N_{Eps}(q) \qquad (2)$$

In order for point q to be a core point it needs to have a minimum number of points within its Eps-neighborhood

$$|N_{Eps}(q)| \geq MinPts \text{ (core point condition)} \quad (3)$$

Definition 3: (Density-reachable)“A point p is density-reachable from a point q with respect to Eps and MinPts if there is a chain of points $p_1...,p_n$, $p_1=q$, $p_n=p$ such that p_{i+1} is directly density-reachable from p_i.”

Definition 4: (Density-connected)
There are cases when two border points will belong to the same cluster but where the two border points don’t share a specific core point. In these situations the points will not be density-reachable from each other. There must however be a core point q from which they are both density-reachable.

“A point p is density-connected to a point q with respect to Eps and MinPts if there is a point o such that both, p and q are density-reachable from o with respect to Eps and MinPts.”

Definition 5: (cluster)
If point p is a part of a cluster C and point q is density-reachable from point p with respect to a given distance and a minimum number of points within that distance, then q is also a part of cluster C.

1) "$\forall$ p, q: if $p \in C$ and q is density-reachable from p with respect to Eps and MinPts, then $q \in C$.

Two points belongs to the same cluster C, is the same as saying that p is density-connected to q with respect to the given distance and the number of points within that given distance.

2) $\forall p, q \in C$: p is density-connected to q with respect to Eps and MinPts.

Definition 6: (noise)
Noise is the set of points, that don’t belong to any of the clusters.
Lemma 1: A cluster can be formed from any of its core points and will always have the same shape.
Lemma 2: Let p be a core point in cluster C with a given minimum distance (Eps) and a minimum number of points within that distance (MinPts). If the set O is density-reachable from p with respect to the same Eps and MinPts, then C is equal to the set O.

VDBSCAN

VDBSCAN[13] algorithm detects cluster with varied density as well as automatically selects several values of input parameter Eps for different densities.Peng Liu, Dong Zhou and Naijun Wu have proposed a detailed description in [14].

Description of the Algorithm In general the algorithm has two steps, choosing parameters Eps_i and cluster with varied densities. The algorithm [13] is as follows,

(i) It calculates and stores k-dist for each project and partition the k-dist plots.
(ii) The number of densities is given intuitively by k-dist plot.
(iii) The parameter Epsi is selected automatically for each density.
(iv) Scan the dataset and cluster different densities using corresponding Eps_i
(v) Display the valid cluster with respect to varied density.

Algorithm:

```
Partition k-dist plot.
Give thresholds of parameters $Eps_i(i=1,2,…..n)$
For each $Eps_i(i=1,2,…..n)$
    a) $Eps = Eps_i$
    b) Adopt DBSCAN algorithm for points that
       are not marked.
    c) Mark points as $c_i$
Display all the marked points as corresponding
clusters.
```

By using this VDBSCAN algorithm we can form clusters of nodes and make each cluster as a partition.

3.1.3 Assigning Jobs to the Cloud Partition

When a job arrives at the public cloud, the first step is to choose the right partition. Job assignment strategy is presented from [7]. The cloud partition status can be divided into three types:

(1) Idle: When the percentage of idle nodes exceeds α, change to idle status.
(2) Normal: When the percentage of the normal nodes exceeds β, change to normal load status.

(3) Overload: When the percentage of the overloaded nodes exceeds γ, change to overloaded status.

The parameters α, β and γ are to be set by the cloud partition balancers [7].

When the load status of a cloud partition is idle or normal, this partitioning can be accomplished locally [7]. If the cloud partition load status is not normal or idle, this job should be transferred to another partition. The partition load balancer then decides how to assign the jobs to the nodes. Server load status is divided into three types. If one cloud server is overloaded and it again getting new client request while other servers are in Idle or Normal state then following algorithms are used.

- Idle: If it is in idle status, this job should be transferred to another partition by using Round Robin algorithm.
- Normal: If it is normal, this job should be transferred to another partition by using Game theory algorithm.
- Overload: If it is overload, this job should be transferred to another partition. That partition selected using above two algorithms.

Step 1 Define a load parameter set: $F = \{F_1, F_{2,..}F_m\}$with each $F_i(1 <= i <= m, F_i \in [0,1])$ parameter being either static or dynamic. m represents the total number of the parameters.[7]

Step 2 Compute the load degree as:

$$\text{Load degree}(N) = \sum_{i=0}^{m} \alpha_i F_i \qquad (4)$$

$\alpha_i (\sum_{i=0}^{n} \alpha_i = 1)$ are weights that may differ for different kinds of jobs. N represents the current node.[7]

Step 3 Define evaluation benchmarks. Calculate the average cloud partition degree from the node load degree statistics as:

$$\text{Load degree}_{avg} = \frac{\sum_{i=0}^{n} \text{Load\_degree}(N_i)}{n} \qquad (5)$$

The bench mark $\text{Load\_degree}_{high}$ is then set for different situations based on the Load_degree_{avg}.[7]

Step 4 Three nodes load status levels are then defined as[7]:

- Idle When

  $$\text{Load\_degree}(N) = 0 \qquad (6)$$

  There is no job being processed by this node so the status is changed to Idle.
- Normal For

  $$0 < \text{Load degree}(N) <= \text{Load\_degree}_{high} (7)$$

  The node is normal and it can process other jobs.
- Overloaded When

  $$\text{Load degree}_{high} <= \text{Load\_degree}(N) \qquad (8)$$

The node is not available and can not receive jobs until it returns to the normal.[7]

The load degree results are input into the Load Status Tables created by the cloud partition balancers. Each balancer has a Load Status Table and refreshes it each fixed period T. The table is then used by the balancers to calculate the partition status. Each partition status has a different load balancing solution. When a job arrives at a cloud partition, the balancer assigns the job to the nodes based on its current load strategy. This strategy is changed by the balancers as the cloud partition status changes.

3.1.4 Round Robin Algorithm

In the regular Round Robin algorithm, every node has an equal opportunity to be chosen. However, in a public cloud, the configuration and the performance of each node will be not the same; thus, this method may overload some nodes. Thus, an improved Round Robin algorithm is used, which called “Round Robin based on the load status”.

```
Step 1:Job arrives at the main controller
Step 2: Job is assigned to balancer based on request location, balancer status
Step 3:In particular partition P
        Set i=0
        Set s[n] as no. of servers arranged in increasing order
        of jobs  in P for all n=1,..,n
        Set s[c] as no. of idle servers in P from s[n]
        for all c=1,..,n
Step 4:When job arrives
        If s[c] != NULL
                Then, send the connection to s[i]
                i= i+1;
                If i == c
                        Then i=1
        Else
                Go to game theory
        End if
Step 5:go to step 1
```

3.1.5 Game Theory

The players in the game are the nodes and they contend for jobs. Suppose there are n nodes in the current cloud partition with N jobs arriving, then define the following parameters:
μ_i : Processing ability of each node, i = 1,…, n.
Φ_j : Time spent for each job.
$\Phi = \sum_{j=1}^{N} \phi_j$
Φ_j :Time spent by the entire cloud partition, $\Phi < \sum_{j=1}^{N} \mu_i$
S_{ij}: Fraction of job j that assigned to node i ($\sum_{j=1}^{N} S_{ij} = 1$ and $0 <= S_{ij} <= 1$).

In this model, the most important step is finding appropriate value of S_{ij}. Here “the best reply” method proposed by Grosuet al.[12]can be used to calculate S_{ij} of each node. The Nash Equilibrium here is to minimize the response time of each job.

4. CONCLUSIONS

Load balancing in public clouds is a difficult task with cloud division problem being the major one. Using clustering algorithms is one way of doing it, and there may be several other ways that get the job done. This problem can be easily solved by combining the concepts of networking and data-mining into the load balancing problem of public cloud

FUTURE WORK

Since this work is conceptual framework, more work is needed to implement the framework and resolve new problems.

(1) Cloud division rules: It can be further enhanced by devising other methods and comparing performance. The division rule should not simply be based on the geographic location.
(2) Set the refresh period: Refresh period to communicate the load status should be properly set. It should neither be too short nor too long. Tests and statistical tools can be used to set a reasonable refresh periods.
(3) Other load balance strategy: Other load balancing strategies may also be used, based on tests comparing different strategies. Many tests are needed to guarantee system availability and efficiency.

REFERENCES

[1].Michael Armbrust, Armando Fox, Gunho Lee and Ion Stoica (2009) “Above the clouds: a berkeley view of cloud computing,” University of California at Berkeley Technical Report No. UCB/EECS-2009-28

[2]. Soumya Ray and Ajanta De Sarkar “Execution analysis of load balancing algorithms in cloud computing environment,” International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.2, No.5, October 2012

[3]. K. Ramana, A. Subramanyam and A. Ananda Rao, “Comparative analysis of distributed web server system load balancing algorithms using qualitative parameters,” VSRD-IJCSIT, Vol. 1 (8), 2011, 592-600

[4]. Network Working Group J. Moy, Editor Request for Comments: 1245, “OSPF protocol analysis,” Editor Request for Comments: 1245 Proteon, Inc. July 1991

[5]. T. H. Cormen et al, “Introduction to algorithms,” 3rd edition, Prentice-Hall of India,2010

[6]. Henrik Bäcklund, Anders Hedblom andNiklasNeijman, A density-based spatial clustering of application with noise,” Data Mining TNM033 2011-11-30 LinköpingsUniversitet - ITN

[7].Gaochao Xu, Junjie Pang, and Xiaodong Fu, “A Load Balancing Model Based on Cloud Partitioning for the Public Cloud,” IEEE transactions on cloud computing year 2013

[8].Suriya Begum, Dr. Prashanth C.S.R, “Review of load balancing in cloud computing,” IJCSI International Journal of Computer Science Issues, Vol. 10, Issue 1, No 2, January 2013

[9]. P. Mell and T. Grance, “The NIST definition of cloud computing,”NIST Special Publication 800-145 September 2011

[10]. Martin J. Osborne, “An introduction to game theory 1995–2000”: Text book, University of Toronto, Canada, Oxford university press 2000, pg 1-8

[11]. Jeffrey M. Galloway, Karl L. Smith and Susan S. Vrbsky, “Power aware load balancing for cloud computing,” World Congress on Engineering and Computer Science 2011Vol I WCECS 2011, October 19-21, 2011, San Francisco, USA

[12].Daniel Grosu and Anthony T. Chronopoulos, “Noncooperative load balancing in distributed systems,” Journel of parallel and distributed computing ELSEVIER Comput. 65 (2005) 1022 – 1034

[13].M.Parimala, Daphne Lopez, N.C. Senthilkumar, “ A survey on density based clustering algorithms for mining large spatial databases,” International Journal of Advanced Science and Technology Vol. 31, June, 2011

[14]. Peng Liu, Dong Zhou andNaijun Wu, “VDBSCAN: Varied density based spatial clustering of applications with noise,” 1-4244-0885-7/07/$20.00 ©2007 IEEE

[15]. Soumya Ray and Ajanta De Sarkar “Execution Analysis of Load Balancing Algorithms in Cloud Computing Environment,” International Journal on Cloud Computing: Services and Architecture (IJCCSA), Vol.2, No.5, October 2012

[16]. K. Ramana, A. Subramanyam and A. AnandaRao,“Comparative Analysis of Distributed Web Server System Load Balancing Algorithms Using Qualitative Parameters,” VSRD-IJCSIT, Vol. 1 (8), 2011, 592-600