International Journal of Innovation Sciences and Research
Vol.4, No, 11, pp.532-534, November- 2015

Available online at
http://www.ijisr.com

RESEARCH ARTICLE

LOAD BALANCING WITH TOKEN BUCKET ALGORITHM (LBTB)

***Enakshmi Nandi and Debabrata Sarddar**

Department of Computer Science & Engineering, University of Kalyani, Kalyani, India

Accepted 21st October, 2015; Published Online 30th November, 2015

ABSTRACT

Cloud Computing is new concept in internet technology which has become so popular to provide different services to client like as on-line office software, game and on-line storage facility, multi-media sharing etc. The main principal of cloud computing is offering computing, storage and "software as a service" facilities (Voorsluys et al., 2011). It provides different type of computing resources to make easy the execution of large scale tasks. The springiness of resources without paying a premium for large scale is an inevitable step in the history of IT industry. The progress in web traffic and several services are increasing day by day making load balancing a great research topic in cloud computing environment. Load balancers are used for allocating load to different virtual machines in such a way that none of nodes gets loaded heavily or lightly. If failure has occurred in load balancing process that creates unavailability of data. Our main aim is to maintain proper load balancing strategy with the help of token bucket algorithm with maintain congestion control properly.

Key Words: Load Balancing, Cloud computing, token bucket algorithm

INTRODUCTION

Cloud Computing is an eminent terms in this era which want to allow to access a large amount of computing power in a fully virtualized manner, by assembling resources and offering a single system view. In cloud computing environment clients can access the operational capability faster with the help of internet (Zeng and Veeravalli, 2004). Load balancing process is an important issue in cloud computing because overloading of a system may cause poor performance which can make the technology unsuccessful.

So necessity of efficient load balancing algorithm is an important factor in cloud computing for efficient utilization of resources. There are different types of algorithm existing for load balancing purpose those are RR, LBMM, OLB, Throttled etc. Our paper focus on a load balancing algorithm with the help of token bucket algorithm where we want balancing load properly with the help of each token respectively. From our study we get minimum completion time and more secure load balancing process through our approach. Here all tasks are generated per token and serially, so congestion cannot occur and all nodes perform their task with very efficiently.

Related works

Load balancing algorithm (Shimonski and Windows , 2000) resolves the effect of balancing the server workloads. Load balancing algorithm is divided into static algorithm and dynamic algorithm (R.X.T. and F.Z. A , 2010). The static algorithm does not take in to account the preceding state or nature of the node while distributing the node. The common static algorithms are Round-Robin Scheduling Algorithm.

****Corresponding author: Enakshmi Nandi,***
Department of Computer Science & Engineering, University of Kalyani, Kalyani, India.

- **Round-Robin Algorithm (RR)** : Round-Robin Scheduling Algorithm is the simplest one which could be most easily be carried out and selects the first node randomly, then assign jobs to all other nodes in round-robin manner. However, it is only applicable for cloud computing in that case when some nodes might heavily loaded and some are not. The good side of this algorithm is that if any node fails, it will not halt the system; it will only affect the system performance.

- **Load Balancing Max-Min and Max (LB3M)**: In this method it is to find that has maximum average completion time .For that purpose searching the unassigned node that has minimum completion time less than the maximum average completion time selects that node and respective task is dispatched. The minimum completion time is of an assigned node is the summation of minimum completion time of assigned task on this node and the minimum completion time of the current task .In this way this process will continue until all tasks have been completed properly (R.X.T. and F.Z. A , 2010) .

- So, the ideal load balancing algorithm should achieve the following targets:

- Leave the collections, computing of load node information for each node; prevent the front-end scheduler from being system bottleneck.

- Reduce the disturbances of load balancing algorithm as far as possible.

Token Bucket Algorithm

There are many applications where it is better to allow the output to speed up somewhat when a larger burst arrives than to lose the data. Token Bucket algorithm provides such a solution. In this algorithm leaky bucket holds token, generated at regular intervals. Main steps of this algorithm can be described as follows:

- In regular intervals tokens are hurled into the bucket.
- The bucket has a maximum capacity.
- If there is a ready packet, a token is taken out from the bucket, and the packet is send.
- If there is no token in the bucket, the packet cannot be send (http://www.slideshare.net/UmeshGupta3/leaky-bucket-algorithm)

Figure 1 shows the two scenarios before and after the tokens existing in the bucket have been consumed. In Fig. 1(a) the bucket holds two tokens, and three packets are waiting to be sent out of the interface, in Fig.1 (b) two packets have been sent out by consuming two tokens, and 1 packet is still left.

Our Proposed approach (LBTB)

We assume that all nodes are allotted by each task by token wise. Each task passed in the bucket according to one token with one single task, then that task assigned by a single node. There is counter for measuring each task or packet with its corresponding tokens, without token no tasks or packets will arrive through this bucket.

Even there is no token left but task are remain in schedule then it cannot passed through bucket. So there is regulated flow of data. There is no chance for data congestion in this process and no nodes are overloaded, load balancing maintain properly.

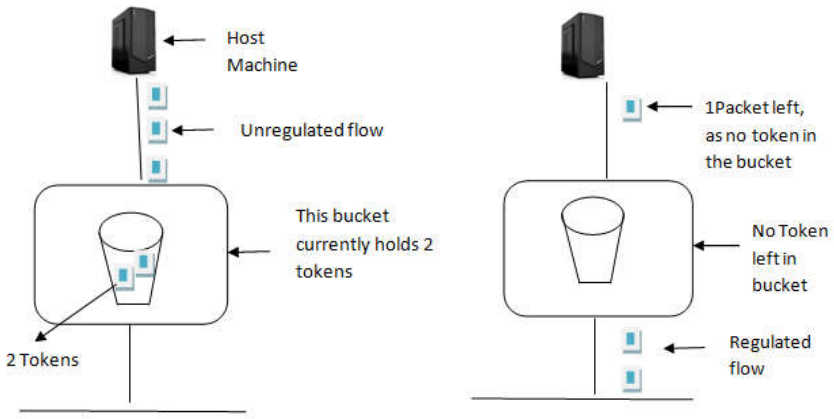

Figure 1. (a) Token bucket holding two tokens, before packets are sending out, (b) Token bucket after two packets are send, one packet still remains as no token is left

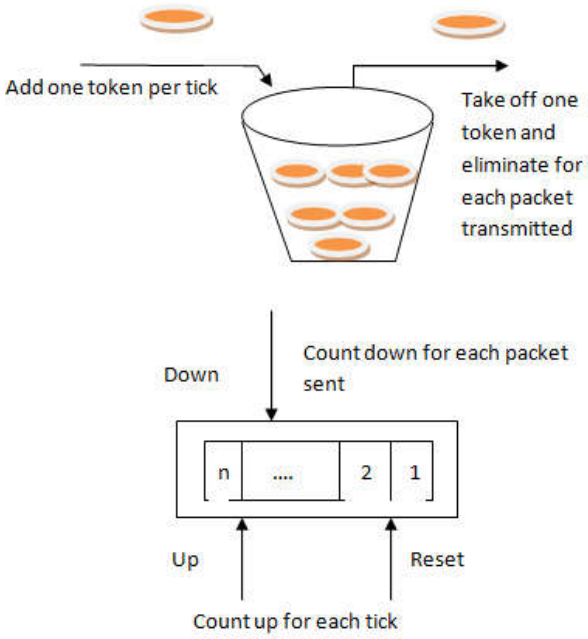

Figure 2. Implementation of the Token bucket algorithm

The token bucket algorithm limits of burst, which is restricted by the number of tokens available in the bucket at a specific instant of time. The implementation of basic token bucket algorithm is simple; a variable is used to count the tokens. This counter is incremented every *t seconds* and is decremented at the time of when a packet will sent. When this counter reaches zero, no further packet is sent out as shown in the Figure 2.

Algorithm

Step 1: A token is added at every **Δ**t time.

Step 2: The bucket can hold at most b-token. If a token has reached when the bucket is full then it is eliminated.

Step 3. When a packet of m bytes reached m tokens are detached from the packet is sent to the network.

Step 4. If less than n tokens are accessible, no tokens are removed from the bucket and the packet is considered to be inappropriate

The inappropriate packet may be enqueued for subsequent transmission when sufficient token have been assembled in the bucket (grabpage.info/t/ www.bing.com:80/ images/ search?q =Leaky.. by slide share).

Result and Analysis

Table 1 shows the total completion time or output time for each task at different computing nodes. The threshold is average of all completion tasks in t_i in all computing nodes. Where we also calculate the burst length. In figure.3 we compare with the completion time with some existing load balancing method of each node. We get better completion time with our approach.

Table 1. The total completion time or output time for each task at different computing nodes and threshold voltage

Node\task	N1	N2	N3	N4	Threshold	Completion time(sec)
T1	11	12	9	13	11.25	29
T2	14	22	12	24	18	26
T3	23	30	10	31	23.5	27.2

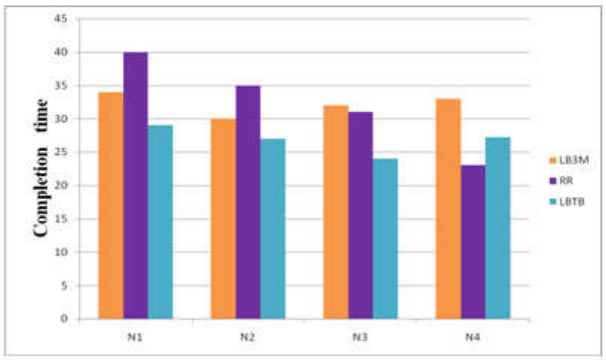

Fig. 3. Shows the comparison between completion time of different load balancing Scheduling

Calculate Total Completion time for token bucket process

$T_{o/p}$ = burst length +Total arrival time
Where $T_{o/p}$ is the total completion time

Burst length = S= bucket capacity (c) / ((data arrival rate (M) - token arrival rate (**ρ**)) * network arriving rate)

Total arrival time = bucket capacity /Token arrival rate (6)

Conclusion and Future work

Our researched load balancing scheduling shows the better completion time with the comparison with Round Robin and LB3M technique.For four different nodes we get less completion time i,e 29,27,24,27.2 sec .So we can say that with using LBTB algorithm we get less completion time and load balancing stargy is easily maintained and congestion control of data values are maintained properly.

Our future aim is to look after the drawback of our method ,i.e analyasis about response time.According to our method though the completion time is less with respect to some existing load balancing algorithm but response time is not less with comparative to some other load balancing technique.So our research will continue about that matter and will see the forward step that how to resolve this problem with very efficiently.

REFERENCES

Voorsluys, William, 2011. James Broberg, and Rajkumar Buyya. "Introduction to cloud computing." *Cloud computing: Principles and paradigms*: 1-44.

Shimonski, R. Windows 2000 and Windows server 2003 clustering and load balancing. Emeryville. McGraw-Hill Professional publishing,CA,USA(2003), p 2,2003.

R.X.T. and X. F.Z. 2010. A load balancing strategy based on the combination of static and dynamic, in database technology and applications (DBTA), 2010 2nd international workshops, pp.1-4

http://www.slideshare.net/UmeshGupta3/leaky-bucket-algorithm

grabpage.info/t/www.bing.com:80/images/search?q=Leaky.. by slide share

Zeng, Z. and Veeravalli, B. 2004, July. Rate-based and queue-based dynamic load balancing algorithms in distributed systems. In *Parallel and Distributed Systems, 2004. ICPADS 2004. Proceedings. Tenth International Conference on* (pp. 349-356). IEEE.

****