IJESRT
INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY

ENHANCEMENT IN NETWORK ENGINEERING THROUGH SKIP LIST DATA STRUCTURE FOR FEDERATED CLOUD DATA CENTER

Dr. Vibhakar Pathak*, Shabana Patel, Er. Vishal Shrivastava, Rahul Sharma
[*] Professor: Department of Computer Engineering Arya College of Engineering and IT Kukas, Jaipur, Rajasthan
M.Tech Scholar: Department of Computer Engineering Arya College of Engineering and IT Kukas, Jaipur, Rajasthan
Professor: Department of Computer Engineering Arya College of Engineering and IT Kukas, Jaipur, RajasthanAss.Professor: Department of Computer Engineering Arya College of Engineering and IT Kukas, Jaipur, Rajasthan

DOI: 10.5281/zenodo.573554

ABSTRACT

In digital century cloud computing is most use and fastest growing platform for making resources available to the users. Data can be available to the user easily with their credential like name password etc users have to memorize these things. But for programmer/developer there is much more to do. Cloud computing is based on internet there are large number of servers where user can access and store data. In terms of cloud computing the background network is known as DCN (Data Center Network). In this paper we have explain Skip List Topology because this topology in the account as the size of list is dynamic and how we can reduce the data traverse time and traffic of network. It is very easy to insert and search data in the list. It is apparent that Skip List is better in term of efficiency then B-tree where both are based on DCN.

KEYWORDS: Skip list, DCN, Traffic Engineering, Cloud Computing, B-Tree, DNS.

INTRODUCTION

Cloud Computing aims to deliver reliable, secure, fault-tolerant, sustainable and scalable infrastructure for hosting Internet-based application services. it is a kind of Internet-based computing system, where different services are available such as access data, storage, servers, data and applications that are delivered to an organization's computers and devices through the Internet .

Data centers a kind of core repository where data is stored managed and distributed. A data center is a facility that centralizes an organization’s IT operations data and equipment, where it stores, manages and circulates its data. Data centers house is a network’s most critical systems and are vital for the continuity of daily operations and activities.

Traffic engineering is the way that improves the network performance by work or operate with the hand flow of data in the network. Traffic based performance measures include delay variation, packet loss and throughput. An important aim of Internet traffic engineering is to facilitate reliable network operations.

Various data structured are used:

Fat is a tree, and processors are connected to the bottom layer. The distinguish advantage/feature of a fat-tree is that for any switch, the number of links going down to its siblings/switch is equal to the number of links going up to its parent in the upper level. Therefore, the links get “fatter” towards the top of the tree, and switch in the root of the tree has most links compared to any other switch.

B-trees are basically a self balanced search trees that has designed to work well on magnetic disks or other direct-access secondary storage. B-trees are very similar to red-black trees, but they are better at minimizing disk I/O operations.

Skip lists are a interesting data structure: very simpleyet have the same asymptotic efficiency as much more complicated AVL trees and red-black trees. A skip list is a data structure that is used for storing a sorted items list with a help of hierarchy of linked lists that connect increasingly sparse subsequences of the items.

TOOLS: CLOUDSIM is a library for simulation of cloud scenarios. It provides important classes to describe data centers, virtual machines, computational resources, applications, users and policies for the management of various parts of the system such as scheduling and provisioning.

CloudSim: It is an extensible simulation toolkit that allow modeling and the simulation of Cloud computing systems and application provisioning environments. [14]

PREVIOUS WORK

The findings were that skip lists are a very easier and efficient algorithm to implement as compare to self-adjusting tree and balanced search tree algorithms [13]. They have basically focused on link list algorithm to perform Skip list algorithm SL data structure consists of linked lists formed in layers, which were linked in a pyramidal way [12]. Deployed the Data Center Network on the concept of B-Tree Topology,They have taken this topology in the account as the size of tree remains constant and it is very easy to estimate general behavior of the data flow [7]. Author has explained both fat tree and Ethernet (WAN) and comparison and In this paper they have used Prim's MST algorithm for Fat-tree based routing [10].There is an urgent need for effective and efficient methods to extract unknown and unexpected information from spatial data sets of unprecedentedly large size, high dimensionality, and complexity author has use B-tree method for storing spatial data and search for geographical search [9].Content-Centric Networking (CCN) is a new network architecture aiming to solve many fundamental problems of existing IP networks [11]. In this paper author has describe some technologies that which are in use today to control of packets in the global network first he described the current organization of net and the key role played by Border gateway protocol (BGP) [13]. With the approach of the Cloud, deployment and hosting became cheaper and easier with the use of pay-per-use flexible elastic infrastructure services offered by Cloud providers [5].

DATA STRUCTURE AND ALGORITHM

A skip list S for a sorted dictionary D made of a series of sequences that we can note as $\{S_0, S_1,...,S_h\}$. Every sequence S_i stores a subset of the items of D classified by a non-decreasing key plus items with two special keys, denoted as $-\infty$ and $+\infty$, where $-\infty$ is less than every possible key that would be inserted in D and $+\infty$ is greater than every possible key that can be inserted in D. Moreover, the sequences in S satisfy the following conditions:

- Sequence S_0 Possess every item of dictionary D (plus the special items with keys $-\infty$ and $+\infty$).
- For $i = 1,..., h - 1$, sequence S_i posses(in addition to $-\infty$ and $+\infty$) a randomly generated subset of the items in sequence S_{i-1}.
- Sequence S_h contains only $-\infty$ and $+\infty$.

It is conventional to visualize a skip list S with sequence S_0 at the lower level and sequences $S_1,..., S_{h-1}$ above it. Also, we can denote to h as the height of skip list S. The sequences are set up so that S_{i+1} possess more or less every other item in S_i. As can be seen next in the insertion method/technique, the items in S_{i+1} are chosen at random from the items in Si by choosing each item from Si to also be in S_{i+1} with possibility of 1/2. Essentially, we flip a coin for each item in Si and place that item in S_{i+1} Now if the coin comes up as "heads." Then, we expect S1 to get about n/2 items, S2 to have about n/4 items, and, in general, S_i to have about (n/2) i items. As In other words, we expect the height h of S to be about log (n). We consider a skip list as a two-dimensional collection of positions which are set horizontally into levels and vertically into towers. Each level refers to a sequence S_i and each tower contains positions storing the same item across consecutive sequences. The locations in a skip list can be traversed using the below operations:
Next (p): Return the position succeeding p on the same level.
Prev (p): Return the position preceding p on the same level.
Below (p): Return the position under w p in the same tower.
Above (p): Return the position over p in the same tower.
Searching Operation in Skip list
Algorithm: SkipSearch (Node):
    Input: A search node

```
        Output: Position p in the bottom list S_0 such that the entry at p has the largest key less than or equal to
node
        While below (p) = null do
                p ← below (p) [drop down]
                while
                        key (after (p)) ≤ Node  do
        Let p ← after (p)
                {scan forward/ahead}
                        end while
        end while
        return p
```

Insertion Operation in Skip list

```
Algorithm: SkipInsert (Node)
        Input:  Add Node
        Output:  the entry inserted in to the skip list
                                        p ← SkipSearch (Node)
                                                q ←  null,
                                                r ←  (Node)
                                                i ←  -1
                                                repeat
                                                i ←  i+1
                                                if  h<= I then
                                        h←  h+1 {new level add}
                                                i ←  next(s)
        s ← insertAfterAbove (null, s, (-∞, null))
           insertAfterAbove(s, t, (+∞, null))
        while above(p) = null do
         p ←  prev(p) {scan forward}
          p ← above(p) {jump up to higher level)
       q ← insertAfterAbove(p,q, r)
    {add a position to the tower of the new entry}
    end while
        until coin Flip() = tails
                n  ← n+l
                  return q
```

OVERHEAD

We have added the Ethernet header and Interframe gap.
1500 - 20b (IPv4) -20b (TCP + checksum) = 1460b DATA (and 40b Overhead)
Add 40b + 14b (Ethernet) + 4b (FCS) + 12b (Interframe gap) + 8b (preamble) = 78b Overhead
78 / 1460 * 100 = 5.34% overhead --------------------------- (1)
1460 / (1460 + 78) * 100 = 94.93% Throughput/ Good put --------- (2)
Taking consideration of all overheads and switching structure applied for skip list we conclude to following throughput formula.
Total overhead = $(X* 1.053)^{1.0609}$
Where X is distance in time (Milliseconds)
$(1.03)^n$ = switching overhead for **n** level of switch
$(1.03)^2$ =1.0609 (2 level of switching is used in skip list)
0.53 Fast Ethernet overhead. (From 1)

EXPERIMENTAL SETUP

To proceed for implementing the work in following steps:

1. For developed a program we had first imported CloudSim in JVM.
2. We have developed a Program for Distribution of Data Center Network with the logic of SkipList topology.
3. In our program there is associated IP Address(first three digits) as Primary Key of the Distribution

4. Associated Geographical Distance of that particular DSS represented by IP Address.
5. Data send time (Round Trip time) is measured in Milliseconds.
6. We have compared the round trip time of Skip list based on DCN and B-Tree round trip time based on DCN.
7. Observe and proved that Skip List based more efficient than B-Tree.

RESULT S AND ANALYSIS

In order to make work more visible we have shown a tabular comparison of time taken by both architectures sending data India to other DNS.

In these figures we can observe that B –TREE based on DCN takes more time than SKIP LIST based on DCN. In B-TREE with increase in distance, time to traverse data goes high and it takes more time on the other hand data traverse in SkipList takes less time with increase in distance which means Skip list is sending data faster than B-Tree. In the table, we have mentioned the comparison between time in milliseconds taken by SkipList architecture and the time that was mentioned earlier taken by the B-Tree architecture.

We can observe the last Colum of the table percentage changes based on time taken to traverse data with change in distance. With increase in distance data traverse time in B-tree is higher than SkipList and this difference has been shown in percentage change using above graph.

After doing several simulations we got following data:

Table – 1:-Comparison of SkipList and B-Tree and difference in percentage change

IP / DNS	Geographical Distance	Data Center with Skip List	Data Center with B-Tree	Percentage Change
181.118.0.0	16000	71.82	84.38	17.48816486
182.160.96.0	1200	4.9	4.97	1.428571429
187.41.0.0	15600	70.65	82.08	16.17834395
188.154.0.0	6900	30.28	33.66	11.16248349
190.107.0.0	14500	64.75	66.8	3.166023166
191.80.0.0	14600	65.14	67.29	3.300583359
192.248.127.255	1429	5.54	5.71	3.068592058
193.188.127.255	2960	12.78	12.38	-3.129890454
194.224.255.255	7950	34.31	35.31	2.914602157
195.202.64.0	5000	21.5	21.59	0.418604651
196.202.0.0	5000	21.26	21.59	1.552210724
197.159.31.255	7298	31.4	32.24	2.675159236
198.104.96.0	15000	67.89	69.254	2.00913242
199.7.90.255	15000	67.09	61.213	-8.759874795
200.129.0.0	14700	66.14	59.95	-9.358935591
201.220.128.0	15700	70.57	64.15	-9.097350149
202.70.112.0	12000	53.84	48.64	-9.658246657
203.2.127.255	11300	50.28	45.72	-9.069212411
204.12.143.255	7572	32.83	30.27	-7.797745964
205.255.0.0	13900	62.38	63.879	2.403013786
206.99.153.71	11000	49.172	49.836	1.350361995
207.184.199.8	12900	57.281	59.014	3.025436008
209.212.96.0	8264	36.6	36.79	0.519125683
210.80.31.255	9479	42.06	42.55	1.165002378
211.60.64.0	4100	16.93	17.491	3.313644418
212.154.0.0	4600	19.53	19.76	1.177675371
213.167.128.0	8300	36.5	36.96	1.260273973
214.255.255.0	13600	61.32	62.41	1.777560339
217.29.128.0	4300	18.62	20.07	7.787325456
218.172.80.0	7700	33.48	37.94	13.3213859
219.73.127.255	8600	37.34	42.82	14.67595072

Result has been generated in order to observe that SKIP LIST architecture based Data Center Network on the B-Tree architecture based Data Center Network.

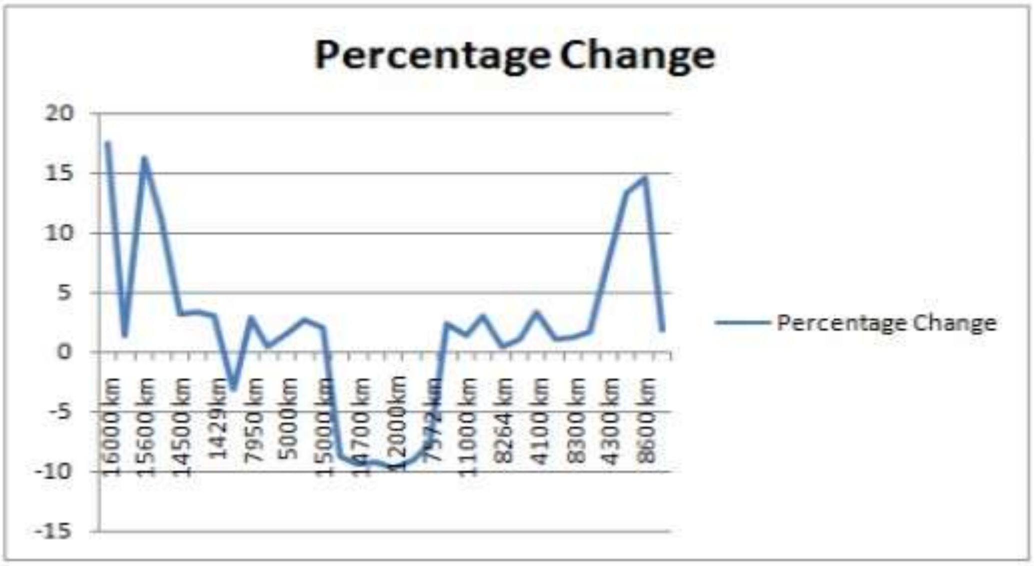

Fig 1. Comparison of Skip List and B-Tree routing delay in milliseconds (as per TABLE -1 below)

CONCLUSION

By observing tables we concluded that:

1) Skip list is more efficient data structure for traffic engineering than B-Tree and its subsequence predecessor like MST and Tree Ethernet.
2) For 32 node data centre with skip list is efficient by 1.944 milliseconds on overall data transfer in all nodes.
3) For less 20 node data centre with skip list is not efficient as compare to B-tree based data centre.
4) The performance advantage in also due to random nature of skip list.
5) Skip list can provide hot swappable nodes in Data Centre.

REFERENCES

[1] Pugh, W. Skip Lists: A Probabilistic Alternative to Balanced Trees. Algorithms and Data Structures: Workshop WADS '89, Ottawa, Canada, August 1989, Springer-Verlag Lecture Notes in Computer Science 382, 437-449. (Revised version to appear in Comm. ACM).
[2] Goodrich, M. and Tamassia, R., Simplified Analyses of Randomized Algorithms for Searching, Sorting, and Selection.
[3] A. Weiss, "Computing in the Clouds," net Worker, vol. 11, Dec. 2007, pp. 16-25.
[4] http://opensourceforu.com/2014/03/cloudsim-framework-modelling-simulating-cloud-environment/
[5] CloudAnalyst: A CloudSim-based Visual Modeller for Analysing Cloud Computing Environments and Applications (Bhathiya Wickremasinghe , Rodrigo N. Calheiros2, and Rajkumar Buyya)
[6] http://www.dot.state.mn.us/metro/trafficeng/traffic_impact_management_data.html
[7] Deployment of Data Center Network using B-Tree Methodology (Dr. Vibhakar Pathak, Suman Saurabh Sarkar).
[8] Comparison of Skip List Algorithms to Alternative Data Structures (David N. Etim)
[9] B-Tree Data Structure - Storing Spatial Data and Efficient Search of Geographical Locations (Ajay sood , Er.Charanjeet singh)
[10] Comparison of fat tree and Ethernet (WAN) routing in cloud data center.(Methodology (Dr.Vibhakar Pathak,Sweta Agrawal)
[11] Multiple Tree-Based Online Traffic Engineering for Energy Efficient Content-Centric Networking (Ling Xu, Tomohiko Yagyu)
[12] Skip List Data Structure Based on New Searching Algorithm & Its Applications: Priority Search (Mustfa Akshu , Ali Karci)
[13] Inter domain Traffic Engineering with BGP.

[14] CloudSim: a toolkit for modeling and simulation of cloud computing environments and evaluation of resource provisioning algorithms (Rodrigo N. Calheiros, Rajiv Ranjan, Anton Beloglazov, C´esar A. F. De Rose and Rajkumar Buyya)

CITE AN ARTICLE

Pathak, V., Dr, Patel, S., Shrivastava, V., Er, & Sharma, R. (2017). ENHANCEMENT IN NETWORK ENGINEERING THROUGH SKIP LIST DATA STRUCTURE FOR FEDERATED CLOUD DATA CENTER. *INTERNATIONAL JOURNAL OF ENGINEERING SCIENCES & RESEARCH TECHNOLOGY*, *6*(5), 481-486. doi:10.5281/zenodo.573554