Global Journal of Computer Science and Technology: G Interdisciplinary
Volume 14 Issue 3 Version 1.0 Year 2014
Type: Double Blind Peer Reviewed International Research Journal
Publisher: Global Journals Inc. (USA)
Online ISSN: 0975-4172 & Print ISSN: 0975-4350

A Tool based Edge Server Selection Technique using Spatial Data Structure

By Debabrata Sarddar, Sandip Roy & Rajesh Bose
Kalyani University, *India*

Abstract- Space partitioning is the process of dividing a Euclidean space into a non-overlapping regions. Kdimensional tree is such space-partitioning data structure for partitioning a Euclidean plane like the surface of earth. This paper describes a tool-based logically partitioning technique of earth surface using K-dimensional tree to segregate the edge servers over the earth surface into a non-overlapping regions for the particular Content Delivery Network. Consequently selecting an edge server based on Least Response Time lo ad balancing algorithm is introduced to improve end-user response time and fault tolerance of the host server.

Keywords: *content delivery network, K-d tree, least response time, load balancing, nearest neighbor search, spatial data structure.*

GJCST-G Classification: *E.1*

Strictly as per the compliance and regulations of:

A Tool based Edge Server Selection Technique using Spatial Data Structure

Debabrata Sarddar [α], Sandip Roy [σ] & Rajesh Bose [ρ]

Abstract - Space partitioning is the process of dividing a Euclidean space into a non-overlapping regions. K-dimensional tree is such space-partitioning data structure for partitioning a Euclidean plane like the surface of earth. This paper describes a tool-based logically partitioning technique of earth surface using K-dimensional tree to segregate the edge servers over the earth surface into a non-overlapping regions for the particular Content Delivery Network. Consequently selecting an edge server based on Least Response Time load balancing algorithm is introduced to improve end-user response time and fault tolerance of the host server.

Keywords: content delivery network, K-d tree, least response time, load balancing, nearest neighbor search, spatial data structure.

I. Introduction

Content Delivery Network (CDN) is a large distributed network of multiple data centers scattered over the earth surface [1] [3]. Today CDNs deliver a huge number of internet content including text, scripts and images and also on-demand streaming media files. Content providers pay CDN operators (e.g. Akamai, Mirror Image Internet etc.) for delivering the aforesaid contents to their customer to improve the overall network performance [2].

In this paper we have introduced a tool for partitioning earth surface using K-d Tree and also a closest edge server is selected based upon proposed least response time load balancing strategy.

Figure 1 : An example of Content Delivery Network over the earth surface.

Author α: Department of Computer Science & Engineering, University of Kalyani, Kalyani, India. e-mail: dsarddar1@gmail.com
Author σ: Department of Information Technology, Brainware Group of Institutions, Kolkata, India. e-mail: sandiproy86@gmail.com
Author ρ: Senior Project Engineer, Simplex Infrastructures Ltd. Kolkata, India. e-mail: bose.raj00028@gmail.com

II. Background Studies

a) Content Delivery Network (CDN)

Increasing the global availability of the internet content, improving the page load time and reducing the bandwidth cost CDN edge servers are scattered over the earth surface. When users from different location are requesting for a particular web content which is algorithmically direct to the nearest edge server to achieve the goal. In this paper we have instigated a technique for partitioning earth surface using the K dimensional tree (K-d tree) and select the nearest edge server using least response time load balancing method which is discussed below.

b) K-dimensional Tree (K-d tree)

K-d Tree is space partitioning data structure for arranging coordinate points (latitude, longitude) over the earth surface. It can be sub-divided the earth surface into a non-overlapping regions [4] [5] [7]. In this context we have described an efficient edge server searching technique using K-d tree.

c) Least Response Time

The Least Response Time is a one of the most popular load balancing technique is used in this context [13] [14]. Using the aforesaid load balancing algorithm, to regulate how to dispense load among the edge servers. This paper we have used the network "ping" command to get average response time of the edge servers which are scattered over the earth surface [10].

III. Proposed Algorithm

In our proposed algorithm we have prepared an efficient tool for CDN provider which is supervising a CDN to select low latency edge server. The set of edge servers are considered as the set of coordinate points P (e.g. latitude and longitude) scattered over geographical region and here we build a K-d tree using P which is scattered over the earth surface as shown in figure 2 and logically partition the edge servers into a non-overlapping region as like figure 3 [6]. Using function kd_closestpointsearch, we have found nearest edge servers of the current location of end-user (e.g. Kolkata) [8] [9] [11]. Then we can calculate accurate network latency using "ping" command over the closest edge servers' IP address to find the minimum average latency time for delivering web content of a particular host server [10]. Executing "ping" command we get status

and result information of the edge server, if the status value is 0 means server is active otherwise 1 signifies the server is dead.

Our proposed algorithm is developed using Matlab R2012b which is described below [11] [12]. In figure 3, the black maker is depicted that the current location of end-user and the closest edge server, among different edge servers, is waiting to send the web content to the end user that is our primary challenge.

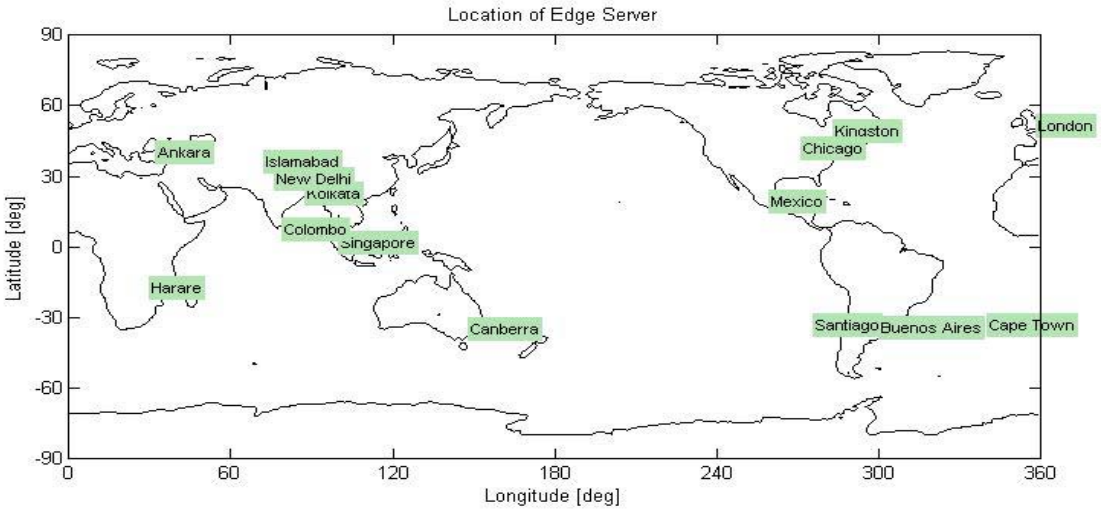

Figure 2 : Location of edge servers over the earth surface

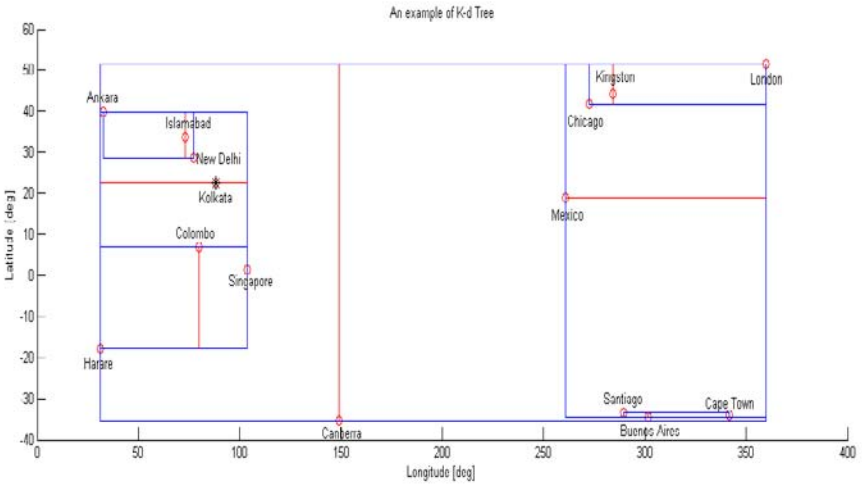

Figure 3 : Location of edge servers over the earth surface using K-d Tree

a) Algorithm for selecting edge server using K-d Tree

1. plot_stuff ←1
2. if (plot_stuff)
   2.1 close all; end
3. A set lat = {lat_1, lat_2, lat_3, …, lat_M} of latitudes are assigned in [1 × M] array
4. A set lon = {lon_1, lon_2, lon_3, …, lon_M} of longitudes are assigned in [1 × M] array
5. sz = size(lon)
6. for i ←1 to sz
   6.1 if lon(i) <= 0
      6.1.1 lon(i) = lon(i)+360
   6.2 end
7. end
8. for i ←1 to sz
   8.1 X (:, 1) = lon
   8.2 X (:, 2) = lat
9. End
10. mylon = 88.3697200 // Longitude of current location (e.g. Kolkata)
11. mylat = 22.5697200 // Latitude of current location (e.g. Kolkata)
12. if mylon <= 0
   12.1 mylon = mylon+360
13. end
14. point = [mylon mylat]

15. tree = **kd_buildtree(X, plot_stuff)** // Build K-d Tree
16. **[index_vals,vec_vals,node_number]=** **kd_closestpointsearch (tree, point)** // Finding the closet point using K-d Tree

b) Function for finding closest edge server of the end-user's current location

function [index_vals, vector_vals, final_node] = kd_closestpointsearch (tree, point, node_number)

// Initialize the global variable

1. global tree_cell
2. global safety_check
3. A set ipaddr = {ipaddr1, ipaddr2, ipaddr3, …, ipaddrM} of IP addresses in [1 × M] string array
4. if(nargin==2)
   - 4.1 safety_check=0
   - 4.2 node_number=1
   - 4.3 tree_cell=tree
   - 4.4 final_node=node_number
   - 4.5 clear tree
5. end

//if the current node is a leaf then output its results

6. if(strcmp (tree_cell (node_number).type, 'leaf'))
7. index_vals=tree_cell(node_number).index
8. vector_vals=tree_cell(node_number).nodevector
9. final_node=node_number
10. [status, result] = dos (['ping -n 1 ' ipaddr (index_vals,:)])
11. Return
12. End

// if the current node is not a leaf

//check to see if the point is to the left of the split dimension if it is to the left then recurse to the left

13. If(point(tree_cell(node_number).splitdim)<=tree_cell(node_number).splitval)
    - 13.1 if (isempty (tree_cell (node_number).left))

// in case the left node is empty, then output current results

- 13.1.1 index_vals =tree_cell (node_number).index
- 13.1.2 vector_vals =tree_cell (node_number).nodevector
- 13.1.3 final_node=node_number;
- 13.1.4 [status, result] = dos (['ping -n 1 ' ipaddr (index_vals,:)])
- 13.2 Return

14. else
    - 14.1 index_vals=tree_cell (node_number).index
    - 14.2 vector_vals =tree_cell (node_number).nodevector
    - 14.3 final_node=node_number
    - 14.4 [status, result] = dos (['ping -n 1 ' ipaddr (index_vals,:)])
    - 14.5 [index_vals, vector_vals, final_node] =kd_closestpointsearch(0,point,tree_cell(node_number).left)
    - 14.6 End

15. else

// as the point is to the right of the split dimension recurse to the right

16. if (isempty(tree_cell(node_number).left))

// In case the left node is empty, then output current results

- 16.1 index_vals=tree_cell(node_number).index
- 16.2 vector_vals =tree_cell (node_number).nodevector
- 16.3 final_node=node_number
- 16.4 [status, result] = dos (['ping -n 1 ' ipaddr (index_vals,:)])
- 16.5 Return

17. else
    - 17.1 index_vals=tree_cell (node_number).index
    - 17.2 vector_vals =tree_cell (node_number).nodevector
    - 17.3 final_node=node_number
    - 17.4 [status, result] = dos (['ping -n 1 ' ipaddr (index_vals,:)])
    - 17.5 [index_vals, vector_vals, final_node] =kd_closestpointsearch(0,point,tree_cell(node_number).right);
18. end
19. end

IV. Simulation Analysis

Step 1 : Latitude and Longitude value of different edge servers are assigned in lat and lon array variables, which are enlisted in table 1 and negative value of longitude are transformed by adding 360° which are listed in table 2.

Table 1 : Latitude and Longitude of different edge servers over the earth surface

Location of edge server	Latitude	Longitude
Kolkata	22.5667°N	88.3667°E
Singapore	1.3000°N	103.8000°E
Colombo	6.9344°N	79.8428°E
London	51.5072°N	0.1275°W
Chicago	41.8819°N	87.6278°W
New Delhi	28.6139°N	77.2089°E
Ankara	39.9300°N	32.8600°E
Islamabad	33.7167°N	73.0667°E
Santiago	33.4500°S	70.6667°W
Mexico	19.000°N	99.1333°W
Kingston	44.2333°N	75.6919°W
Buenos Aires	34.6033°S	58.3817°W
Harare	17.8639°S	31.0297°E
Cape Town	33.9253°S	18.4239°E
Canberra	35.3075°S	149.1244°E

Table 2 : Latitude and Modified Longitude of different edge servers over the earth surface

Location of Edge Servers	Latitude	Longitude	Modified Longitude
Kolkata	22.5667	88.3667	88.3667
Singapore	1.3000	103.8000	103.8000
Colombo	6.9344	79.8428	79.8428
London	51.5072	-0.1275	359.8725
Chicago	41.8819	-87.6278	272.3722
New Delhi	28.6139	77.2089	77.2089
Ankara	39.9300	32.8600	32.8600
Islamabad	33.7167	73.0667	73.0667
Santiago	-33.4500	-70.6667	289.3333
Mexico	19.0000	-99.1333	260.8667
Kingston	44.2333	-75.6919	284.3081
Buenos Aires	-34.6033	-58.3817	301.6183
Harare	-17.8639	31.0297	31.0297
Cape Town	-33.9253	-18.4239	341.5761
Canberra	-35.3075	149.1244	149.1244

Table 3 : IP Address and Domain name of different edge servers over the earth surface

Location of Edge Servers	IP Address	Domain Name
Kolkata	203.197.118.81	www.jaduniv.edu.in
Singapore	137.132.21.27	www.nus.edu.sg
Colombo	192.248.17.88	www.cmb.ac.lk
London	212.113.11.22	www.lse.ac.uk
Chicago	198.101.129.15	www.uchicago.edu
New Delhi	103.27.9.20	www.du.ac.in
Ankara	80.251.40.153	www.ankara.edu.tr
Islamabad	61.5.158.124	www.islamabadairport.com.pk
Santiago	158.170.64.116	www.udesantiago.cl
Mexico	128.123.3.2	www.nmsu.edu
Kingston	130.15.126.136	www.queensu.ca
Buenos Aires	190.224.163.234	www.buenosairesherald.com
Harare	196.201.17.237	www.caaz.co.zw
Cape Town	41.72.141.237	www.capetown.travel
Canberra	137.92.97.88	www.canberra.edu.au

Table 4 : Status information of edge server and Average Network Latency time of closest edge server

Location of Edge Servers	IP Address	Status	Average time(ms)
Canberra	137.92.97.88	Dead	Request timed out
Kolkata	203.197.118.81	Active	260
Colombo	192.248.17.88	Active	Destinationhost unreachable
Singapore	137.132.21.27	Active	238

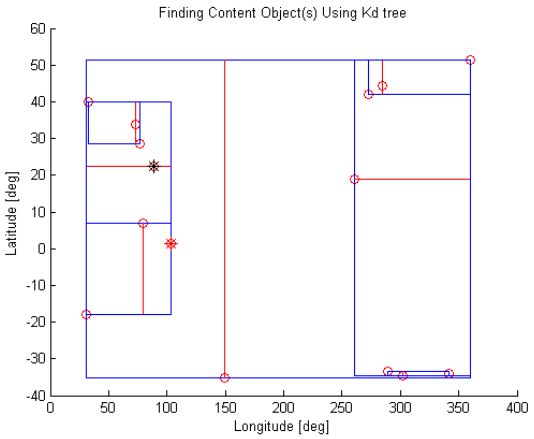

Figure 4 : Nearest edge server selection by our proposed methodology, example Selected edge server Singapore(Red Marker) & User's current location Kolkata (Black Marker)

Step 2 : The IP Address of different location of edge servers are assigned in ipaddr variables and the IP Address of edge servers along with domain names are listed in table 3.

Step 3 : The edge servers are decomposed using K-d tree as shown in figure 3.

Step 4 : Using kd_closestpointsearch function we have search closest edge server of the current location of end-user and consequently find out the accurate network latency time using "ping" command.

Step 5 : The connection is established between least average response time active edge server located at Singapore and end-user from Kolkata for sending the web content as shown in figure 4.

V. Conclusion

Our proposed tool and simulation results proclaim minimum network latency and minimum packet loss in selection of closest edge server over the earth surface. In this paper we have used K-d tree algorithm for decomposing the earth surface. Usage of kd_closestpointsearch method helps us to find the nearest edge server of the end-user. Our proposed tool can be used in wireless network and wired network for delivering the web content efficiently to improve the throughput of total network.

References Références Referencias

1. Nygren, E., Sitaraman, R. K., and Sun, J. (2010). The Akamai Network: A Platform for High Performance Internet Applications. *ACM SIGOPS OSR,* 44(3), 2–19.
2. Parikh, J., Prokop, H., Sitaraman, R., Dilley, J., Maggs, B., and Weihl, B. (2002). Globally Distributed Content Delivery. *IEEE INTERNET COMPUTING,* 50-58.

3. Repantis, T., Cohen, J., Smith S., and Wein, J. (2010). Scaling a Monitoring Infrastructure for the Akamai Network. *ACM SIGOPS Operating Systems Review,* 44(3), 20-26.
4. Bentley, J. L. (1975). Multidimensional binary search trees used for associative searching. *Communications of the ACM,* 18(9) 509. doi:10.1145/361002.361007.
5. Chandran, S. Introduction to kd-trees. *University of Maryland Department of Computer Science.*
6. Rosenberg, J. B. (1985). Geographical Data Structures Compared: A Study of Data Structures Supporting Region Queries. *IEEE Transactions on Computer-Aided Design of Integrated Circuits and Systems,* 4(1), 53-67. doi:10.1109/TCAD.1985.1270098.
7. Moore A. An introductory tutorial on KD trees.
8. Clarkson, K. L. (1983). Fast algorithms for the all nearest neighbors problem, 24th *IEEE Symp. Foundations of Computer Science,* (FOCS '83), pp. 226–232. doi:10.1109/SFCS.1983.16.
9. Vaidya, P. M. (1989). An O(n log n) Algorithm for the All-Nearest-Neighbors Problem. *Discrete and Computational Geometry,* 4(1), pp.101–115. doi:10.1007/BF02187718.
10. Sarddar, D., Roy, S. and Bose, R. (2014). An Efficient Edge Servers Selection in Content Delivery Network Using Voronoi Diagram. *IJRITCC,* 2(8), 2326-2330.
11. Friedman, J. H., Bentely, J., and Finkel, R. A. (1977). An Algorithm for Finding Best Matches in Logarithmic Expected Time. *ACM Transactions on Mathematical Software 3,* 209-226.
12. Vemulapalli, P. (2010). Kd tree implementation in Matlab. Retrieved from MATLAB CENTRAL website:http://www.mathworks.in/matlabcentral/fileexchange/26649-kdtree-implementation-in-matlab
13. Patel, V. P., Patel, H. D., and Patel, J. P. (2012). A Survey on Load Balancing in Cloud Computing. *IJERT*, 1(9).
14. Kherani, F. F. and Vania, J. (2014). Load Balancing in cloud computing, *IJEDR,* 2(1), 907-912.
15. Mata-Toledo, R. and Gupta, P. (2010). Green data center: how green can we perform. *Journal of Technology Research, Academic and Business Research Institute,* 2(1), 1-8.

This page is intentionally left blank