An Enhanced Client-Server Assignment for Internet Distributed Systems

Rajesh D. Bharati[*1], Ishan Naidu[#2], Anurag Kiran[#3], Kalpesh Khune[#4], Chirag Vyas[#5]

[*]*Faculty, Department of Computer Engineering, Padmashree Dr. D. Y. Patil Institute of Engineering & Technology, Pimpri, Pune, India*

[#]*Student, Department of Computer Engineering, Padmashree Dr. D. Y. Patil Institute of Engineering & Technology, Pimpri, Pune, India*

***Abstract*— Internet is a collection of several distributed systems consisting of various clients and servers. These clients communicate with each other with the help of intermediate servers. An optimized performance for such a system needs proper client-server assignment. Such a system can be achieved by mainly considering the factors like total communication load and load balancing of the servers. We propose an approach based on an algorithm which obtain an approximately optimal solution for the client-server assignment problem. The simulation experiments will show that our approach has better efficiency than the existing client-server assignment.**

***Keywords*— Distributed systems, client-server systems, graph clustering, load balancing, communication overhead, optimization.**

I. INTRODUCTION

An Internet distributed system consists of a number of nodes (e.g., computers) that are linked together in ways that allow them to share resources and computation. An ideal distributed system is completely decentralized, and that every node is given equal responsibility and no node is more computational or resource powerful than any other. The clients and servers collectively form the distributed systems. Typical examples of such systems are e-mail, instant messaging, e-commerce, etc. For sending a mail from node A to another node B, the data first flows to email server of node A. Then the data flows towards the email server of node B and finally it reaches node B. Hence, it is the responsibility of email servers to send receive emails for the clients assigned to it. Clients do not communicate with each other directly. Servers communicate on behalf of their clients. Similar kind of process can be used for instant messaging services and e-commerce services.

Client server assignment can be designed based on the following observations:

1. If two clients are assigned to the same server, the server will receive message from one client and forward to another one. If they are on different servers, the sender client first sends to its server. The sender's server will receive data and forward it to the receiver's server. The receiver server will receive data and forward it to receiver client. Thus the total communication between two frequently interacting clients increases if they are assigned to two different servers. Assigning these two clients to a single server makes all information exchange locally. It is more efficient to have all the clients assigned to few servers to minimize total communication.

2. On the contrary having fewer servers, results in those servers being heavily loaded while others are left underutilized. If a server is heavily loaded it results in low performance due to excessive resource usage on that server. Thus it is necessary to consider load balance on the server while assigning clients.

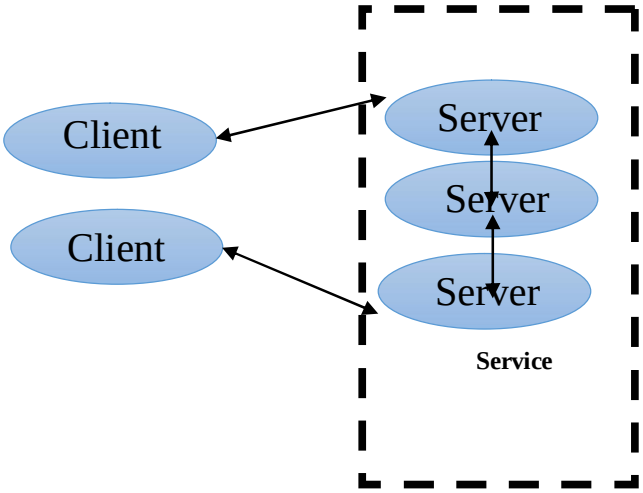

Figure 1: Service across multiple servers.

From the above observations it is clear that total communication load and load balancing are two contradicting features. Hence equilibrium must be maintained between overall communication load and load balancing of the servers.

In this paper, we obtain a solution for optimal client-server assignment based on the total communication load and load balancing on the servers. The pattern that defines the communication between the clients is considered as input and the required output is an approximately optimal client-server assignment for a pre-specified tradeoff between total communication and load balancing.

Taking in consideration the current world scenario, client-server assignment problem can be used for a number of applications from social networking, online auctioning to e-commerce.

II. Related Work

1. Clustering Algorithms:

Client Server assignment problem is an instance of clustering problem. Clients and their communication pattern can be denoted as a graph, with vertices representing the clients and the edges between two vertices representing the communication between the respective clients. Communication frequency between two clients can be represented by the weight of the edge between the corresponding vertices. Clustering algorithm forms fixed number of clusters of clients based on a given objective. Objective of the clustering algorithm in this paper is to minimize the amount of inter group communication and balance the sum of weights of all edges in the group. Normalized cuts (NC) is a clustering algorithm that partitions an undirected graph into two disjoint partitions such that

$$F_{nc} = \frac{E_{1,2}}{E_{1,1,} + E_{1,2}} + \frac{E_{2,1}}{E_{2,2} + E_{2,1}}$$

where $E_{i,j}$ is the sum of weights of all edges that connects the vertices in group i and j. In order to solve F_{nc} efficiently, NC uses Eigen values of adjacency matrix. But NC tends to cause imbalance in the volumes of the groups by isolating the vertices that do not have strong connection to others. A heuristic algorithm based on relaxed convex optimization is used for finding the approximate solution to the client server assignment problem.

2. MapReduce: Simplified Data Processing on Large Clusters

MapReduce is a programming model and an associated implementation for processing and generating large data sets. The users specify a map function that processes a key/value pair to generate a set of intermediate key/value pairs, and a reduce function that merges all intermediate values associated with the same intermediate key. In this model, many real world tasks are expressible, as shown in the paper.

Programs written in this functional style are automatically parallelized and executed on a large cluster of commodity machines. The details of partitioning the input data, scheduling the program's execution across a set of machines, handling machine failures, and managing the required inter-machine communication is taken care of by the run-time system. Due to this, the resources of a large distributed system can be easily utilized by the programmers that do not have any experience with parallel and distributed systems.

Our implementation of MapReduce runs on a large cluster of commodity machines and is highly scalable: a typical MapReduce computation processes many terabytes of data on thousands of machines. Programmers find the system easy to use: hundreds of MapReduce programs have been implemented and upwards of one thousand MapReduce jobs are executed on Google's clusters every day.

3. Distributing Link Stress for Internet TV Systems

There has been an in-depth investigation on the issue of distributing link stress. It is a basic problem in popular Internet TV systems because the service quality at the receiving end is determined by network congestion and bandwidth usage of the links. Less congestion level accommodate more number of subscribers and service channels, which are the most important concerns for service providers. However, earlier works mostly focus on user-centric performance metrics, such as the downloading time, round-trip time, packet loss rate of individual client, but tend to give no attention to the network load caused by the concurrent clients. There exists a graph theoretical approach to distribute link stress and also a scheme that reduces and distributes link stress. With the help of experiments on an actual Internet testbed, PlanetLab, it is shown that the scheme has the remarkable advantages over existing schemes in reducing and balancing the link stress. We expect the results and the scheme can be applied to various Internet TV systems.

III. Optimal Client-Server Assignment

As we know, the load balance and the total communication load are two opposing metrics. Thus, for different applications there will be different tradeoff between these two quantities.

Algorithm for load balancing:

1. Maintain a matrix A_{N*N} which stores the rate of data exchanged between clients. Rows and columns represent the clients in system.
2. Maintain a matrix X_{N*M} denoting which client is using which servers for communication. 1 denotes client using the server and 0 denotes not using the server.
3. Maintain matrix L_{N*M} storing total load on server due to client.
4. Calculate total load on all the servers i.e. F_c.
5. Calculate approximate average load that should be given to each server i.e. (F_c/M).
6. Calculate the load balance factor for each server i.e. $\int_{i=1}^{N}(Li + Fi) \leq \left(\frac{Fc}{M}\right)$
7. Split the number of servers in two groups with m = [M/2] and M-m = [M/2] servers in each group.
8. First perform intra-server load balance by comparing load balance factor 'Fc' of each server then perform inter-server load balance by comparing load balance factor of server from one group and servers from other group.

A threshold needs to be setup in order to avoid congestion of requests from the clients. This can be achieved with the help DOS (Denial of Service) attack.

Algorithm for DOS Attack Detection:

1. Accept the randomly distributed requests by the servers.

2. The server accept request protocol works for a period of time T.
3. Check for the threshold value, i.e., if number of source IP > threshold value.
4. If number of source IP is greater than threshold value then confirm the DOS attack.
5. If DOS attack detected then stop responding to clients in the system.

Consider an example

$A_{i,j}$ =

	c1	c2	c3	c4
c1	0	8	4	0
c2	8	0	10	2
c3	4	10	0	0
c4	0	1	0	0

$X_{i,j}$ =

	s1	s2	s3
c1	1	0	1
c2	0	1	1
c3	1	0	1
c4	0	1	0

$L_{i,j}$ =

	s1	s2	s3
c1	6	0	6
c2	0	10	10
c3	7	0	7
c4	0	1	0

where Matrix $A_{i,j}$ represent rate of data exchanged between clients,
Matrix $X_{i,j}$ represents server assignments,
Matrix $L_{i,j}$ represents load on each server due to client.
Here, s_i and c_i are servers and clients respectively.

Load on s1 = 6+0+7+0 = 13
Load on s2 = 0+10+0+1 = 11
Load on s3 = 6+10+7+0 = 23

Total load F_c = 13+11+23 = 47

Approximate average load on each client,

$$\frac{Fc}{M} = \frac{47}{3} = 15.6 \approx 16$$

Load balance factor, $\int_{i=1}^{N}(Li + Fi) \leq \left(\frac{Fc}{M}\right)$

Load balance factor for server,

f1 = 3, f2 = 5, f3 = -7

Before Load Balance:

Group 1 Group 2
s1 s2 s3

13 11 23

After Intra-server Load Balance:

Group 1 Group 2
s1 s2 s3

13 16 18

f1 = 0, f2 = 1, f3 = -1

After Inter-server Load Balance:

Group 1 Group 2
s1 s2 s3

14 17 20

f1 = 3, f2 = 0, f3 = -3

Advantages of proposed system:

The advantages of the proposed system are as follows:

1. The two groups have different number of servers, a server within a group with fewer servers will likely to have a higher load than a server in the group with more servers. This reduces the load balance.
2. We describe a number of emerging applications that have the potential to benefit from the client-server assignment problem.
3. It optimizes the performance of a class of distributed systems over the Internet.
4. Reduction in the overall communication load.
5. Increases the load fairness among the servers, i.e., the load balance.
6. Performance of a number of emerging applications that have the potential to benefit from the client-server assignment problem can be improved.
7. Higher efficiency than existing systems.

Applications:

1. **Facebook:** It allows circles of friends to exchange messages assigning them to the same server which will reduce the inter-server communication and will result in reducing the overall communication load.
2. **Twitter:** Like Facebook, same operation is performed in twitter for optimizing the overall performance.
3. **eBay:** In this case, it lets a user log in to the server that is likely to have contents of interest to that user which will raise the efficiency of the system.

The client-server assignment problem is also relevant to a host of emerging applications ranging from social network applications such as Facebook and Twitter to online distributed auction systems such as eBay. Facebook is a

system that allows circles of friends to exchange messages and pictures among themselves. Since friends are likely to communicate with each other than non-friends, assigning friends to the same server will reduce the inter-server communication and will result in reducing the overall communication load. At the same time, it is preferable to balance the communication load. This is exactly the client-server assignment problem encountered in the IMS. Online distributed auction systems is another candidate for applying the client-server assignment. If a user logged in a server which has contents that are mostly not of interest the user, then on average, every item search by a user will generate a larger communication overhead, as the search must be done across multiple servers. Therefore, letting a user log in the server that is likely to have contents of interest to a user will raise the efficiency. In this case, the types of contents can also be viewed as clients. The client-server assignment also has the potential to be applicable to distributed database systems, such as MapReduce. Assigning the search keywords which are often queried together to the same servers will reduce the inter-server communication.

IV. CONCLUSION

The major sections that are used in this paper for optimizing client-server assignment are optimizing the total communication cost and load balance. Also DOS attack is used to check the threshold of the requests generated by the clients. The algorithm used provides results that are always approximately optimal and is a better and enhanced way for client-server assignment in Internet distributed systems.

REFERENCES

[1] H. Nishida and T. Nguyen, "Optimal client-server assignment for Internet distributed systems," IEEE Trans. Parallel and Distributed Systems, vol. 24, no. 3, Mar. 2013.

[2] J. Dean and S. Ghemawat, "MapReduce: Simplified Data Processing on Large Clusters," Comm. ACM, vol. 51, pp. 107-113, Jan. 2008.

[3] I. Dhillon, Y. Guan, and B. Kulis, "Weighted Graph Cuts Without Eigenvectors a Multilevel Approach," IEEE Trans. Pattern Analysis and Machine Intelligence, vol. 29, no. 11, pp. 1944-1957, Nov. 2007.

[4] Graclus, "Graclus Software," http://www.cs. utexas .edu/users/ dml/Software/graclus.html, 2012.

[5] S. Balakrishna and Dr. Ling Ding, "An efficient client-server assignment for Internet distributed systems," Dept. Comp. Sc. & Sys., Univ. of Washington, Tacoma.

[6] S. C. Han, "Distributing link stress for Internet TV systems," Dept. Comput. Eng., Myongji Univ., Yongin, South Korea.

[7] T. Duong-Ba, T. Nguyen, "Distributed client-server assignment," Sch. of EECS, Oregon State Univ., Corvallis, OR, USA.