ISSN:0975-9646

Efficient File Transmission and Load Balancing using Multicast over Dynamic Bandwidth Weighted Links

Onkar Habbu, Jayant Golhar, Chinmay Nalawade, Sunil Mane

Department Of Computer Engineering,
College Of Engineering, Pune-411005

***Abstract*— The proposed paper presents the algorithm for Efficient transmission of files having very large size from server to each of the client who request for that file. The technique is ideal for data sensitive networks. This project reduces the time required for the transfer of file compared to traditional client-server transmission due to addition of intermediate steps. The file is split in number of parts is determined by the algorithm. Then, the dynamic ratio of the bandwidth of different links between server and requesting clients is taken. The algorithm decides how many parts of the split file has to be sent via links connected to server depending upon the bandwidth of the links. The remaining parts of the file are taken from other clients having the same file. This results in substantial decrease in the network time required for the transmission of all the parts. Even after considering the overhead of splitting and joining the file at both the ends, this technique shows considerable improvement over traditional method. The load on the server is also substantially reduced depending upon the dynamic values given by the algorithm.**

I. INTRODUCTION

Today, on the web, one of the most common research area is to increase the speed we get while file transfer. In industries or many other networks, we come across situation where we have to send a large file from the server to the clients. Given that, there may be the situation where same file will be required by number of clients adding to redundancy on server side by sending them again and again. This will not only result in huge traffic on server links but also consume server resources to much greater extent. In this paper, we propose the solutions that will improve the situation considering both the above said parameters. By providing these solutions we also argue that these are the best solutions comparing the outputs of traditional methods and by using this solution. Though this solution work in all types of networks, it is ideal for data intensive networks where file size is considerably large.

The main idea behind the solution we are proposing is the optimum consumption of network bandwidth by Parallelization of data transfer over different bandwidth weighted links at run time. The file has to be sent from server and received by the client but the intermediate steps in our case are different. First, file is split into different parts. The algorithm we propose decides the size of each part and number of parts. Then, we calculate the dynamic bandwidth ratio of all the clients requesting the same file to the server. Then, ideal ratio in which the number of parts are sent to different parts is calculated by the algorithm. Then, clients communicate with each other and on the basis of bandwidth ratio each client gets the remaining parts from other clients. In this way, the parallelization of file transfer takes place reducing the time required for actual transmission of file. This also performs the important task of load balancing as server is ready to serve other requests with its full capacity after sending only some of the parts to each client. Technically, server has to send the complete file only once irrespective of the number of requests and client.

II. RELATED WORK

a. Traditional file transfer

If there is server and hundreds of clients then traditional file transfer would require a file to send from server to each and every client requesting that particular file, no matter what the file size is, whether the file is same or not, the time of request and availability of bandwidth and resources. If the file required by number of client is same, it introduces unnecessary redundancy in network traffic. This leads to extravagant load on server bandwidth and resources. In this case the file transfer is complete only when each and every client downloads the complete file. So time required by each client to download the same file depends on the bandwidth of the link between each client and server.

III. PROPOSED SYSTEM

a. Working

The system we are proposing is dynamic, time-efficient and load balancing in nature. If a client requests for a particular file, our server waits for some arbitrary time rather than immediately transferring the file to that client. If during this window of time another few clients request for the same file, server will store all the requests along with their IP's, Bandwidth of the intermediate link which is calculated dynamically. Then, after the waiting time is over, server calculates the ratio of the bandwidths between all the links to clients who requested the same file during window time. Then, the algorithm (explained in ahead section) calculates the ideal ratio of the bandwidth and maps it to the ratio of the parts of file to be sent over these different bandwidth weighted links. This ratio gives the number of parts in which file needs to be split. Then, file is split into these many parts. Server multicasts the different parts of the file to different requesting clients as decided by the aforementioned

algorithm. Server also multicasts all the information like which client possesses how many and which parts of the file, what is the bandwidth of the intermediate links, IP's of all the clients to all the clients. The job of server ends here.

Now, it is the responsibility of client to ask other clients for the remaining parts of the file. This is possible as client has all the information about other clients requesting for the same file as mentioned before. Client request other clients for the remaining parts and get them simultaneously from number of clients. Finally, client checks with the information it has about the parts of the file and if the number of parts received by clients through server and all other clients is equal to the total number of parts, client combines these parts of file to get original file.

b. **Advantages**

1. The parallelization of the data transfer.
   For example Suppose, server S has two clients – client A and client B and ratio of the bandwidth between links SA and SB is "1/3". Now, if both clients A and B request file F from server having size 100 MB and the algorithm splits the file into 4 parts having size 25 MB each. Then, server sends part I to A and next 3 parts II, III, IV to B. But, after sending parts I and II to A and B respectively, instead of remaining idle, A sends the part I and receive parts II, III and IV from B at the same time when server is sending remaining parts to B.
2. Dynamic Load Balancing.
   Consider the above mentioned case in advantage 1. Here, the total data transfer server does is actually equal to the size of single file (Sum of all the parts of file). It is independent of the number of clients requesting the same file. This incredibly reduces the load on the server side in terms of bandwidth and resources. It shifts this load to client side which were otherwise idle.
3. Scalability
   The system we are proposing is highly scalable. In fact, more the number of clients, more is the improvement in the performance. This is possible as clients share the load of the server. Hence, more the number of clients, less is the load on the server and thereby more is the performance.
4. Time efficient file transfer.
   As explained in point 1 and point 2, the load of data transfer is optimally shared by server and clients. This results in the optimum use of network bandwidth resulting in drastic reduction in time required for file transfer.

IV. ALGORITHM

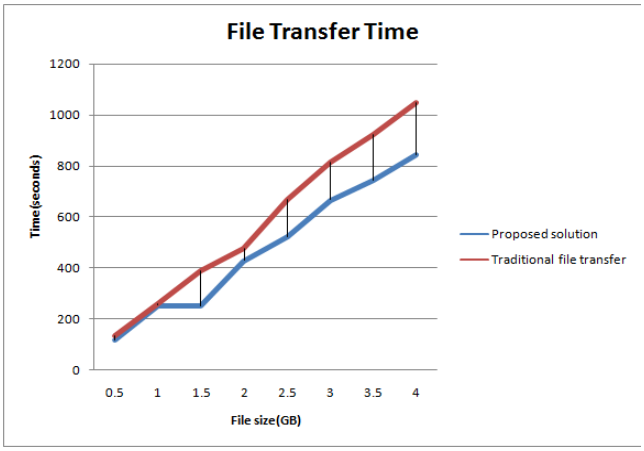
1. First client requests for a particular file.
2. Server will wait for arbitrary window time.
3. Form a group of clients which are requesting same file.
4. Measure the Bandwidth of the links between server and each client that fall within the group.
5. Find minimum bandwidth in that group.
6. Take the ratio of bandwidth of each client with respect to minimum bandwidth
   E.g. let {B1, B2, B3.....Bn} be the bandwidth of clients {C1, C2, C3,.....Cn}
   Let Bm be the minimum bandwidth.
   So the list formed by taking the ratio of bandwidth of each client to minimum bandwidht is {B1/Bm, B2/Bm, B3/Bm,....Bn/Bm}
7. Calculate the minimum roundup loss and corresponding ideal multiplier.
   In order to calculate the minimum roundup loss we multiply every item of the ratio list by number from 1 to 10 and also the nearest integer by number from 1 to 10 and calculate the total round up loss by adding the round up losses for each multiplier for all elements in the ratio list.
   Ex. Now {B1/Bm, B2/Bm, B3/Bm,....Bn/Bm} is the ratio list.
   The total round loss for any multiplier is calculated as follows
   Roundup_loss(for p as multiplier) = ((B1/Bm * p - int(B1/Bm) * p) + (B2/Bm * p - int(B2/Bm) * p) + (B3/Bm * p - int(B3/Bm) * p) + ...+ (Bn/Bm * p - int(B1/Bm) * p))
   In this way the value of p is varied from 1 to 10 and corresponding round up losses for each multiplier be {R1, R2, R3,...., R10}.
8. We calculate the minimum round up loss by taking minimum from {R1, R2, R3,...., R10}.
9. The ideal multiplier is found by finding the index of the minimum round up loss in the {R1, R2, R3,...., R10}
10. Each item in the ratio list is multiplied by the ideal multiplier.
11. Sum all the items in the ratio list to find the ratio total.
12. The ratio total is multiplied by the ideal multiplier to calculate the number of parts in which the file should be split at the server.

12.1 Number of parts = ratio total * ideal multiplier

13. To find the number of parts that should be sent to each client we divide each item in the ratio list by the ratio total and multiply by Number of parts.
    13.1 Let {B1/Bm, B2/Bm, B3/Bm,....Bn/Bm} be the ratio list obtained after step 10
    13.2 Rt be the ratio total obtained in step 11
    Let No be the number of parts obtained in the step 12.1
    The no of parts that will be sent to 1 client will be p1 = B1/Bm / Rt * No
14. In this way the number of parts is calculated for each client to get the list = {p1, p2, p3, ..., pn}
15. Then, the number of parts as decided by above list are sent to different clients.
16. Then, client uses the information received by the server and get the remaining parts from other clients simultaneously.
17. Once all the parts are received, each client combines all the parts to get the original file.

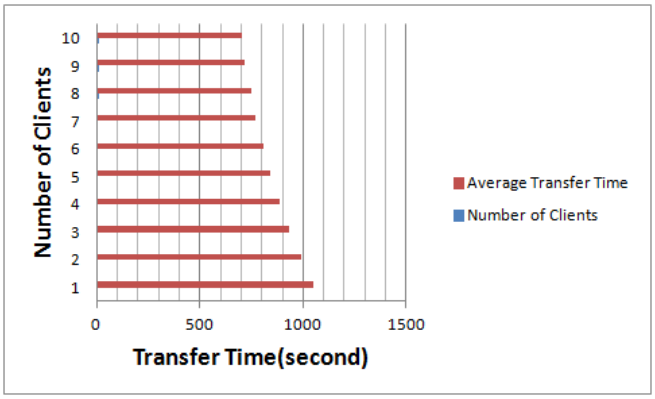
V. RESULTS

a. Time verses File Size.



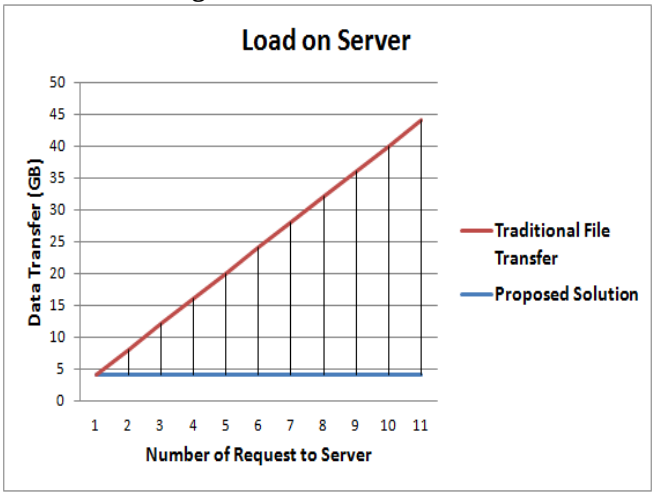
This graph shows the improvement of system by applying the solution we propose over traditional file transfer system. As the file size increases, the margin of improvement also increases making it ideal for data intensive networks.

b. Time verses Number of clients



This graph shows how the performance of system improves with the increase in the number of clients. This is really contradictory to the traditional file transfer system as here, the average file transfer time decreases with the increase in number of clients. The prime reason behind this is that clients share the load of server in this system. Hence, more the number of clients, more the resources and bandwidth available.

c. Load Balancing



This graph shows how load on server side is reduced drastically as the number of request increases. In our proposed system, the clients share the load of server which were otherwise idle.

VI. FUTURE SCOPE

If we only consider the network transfer time i.e. time required to transfer the parts of file and not combining them, the result would have been much better. The splitting and combining of files at both the ends add the extra overhead. There is a scope in future if we can reduce this extra overhead so that system improvement would be surprisingly high. The combining of files could be done at run time as one of the approach towards achieving this.

VII. CONCLUSION

To conclude, we would like to describe the our proposed system as an efficient alternative to the traditional file transfer system. It not only improves the performance but also balances the load on server and clients. This system is highly scalable, in fact higher the number of clients, higher is the system performance. This system is ideal for the data intensive networks where servers has to serve for many requests at the same time.

REFERENCES

[1] Tatsuhiro Chiba, Mathijs den Burger, Thilo Kielmann and Satoshi Matsuoka, "Dynamic Load-Balanced Multicast for Data Intensive applications," 2010 10th IEEE/ACM International Conference on Cluster, Cloud and Grid Computing.

[2] Stefan Saroiu, P. Krishna Gummadi, Steven D. Gribble, "A Measurement 'Study of Peer-to-Peer File Sharing Systems" Dept. of Computer Science and Engineering, Univ. of Washington, Seattle, WA, 98195-2350.

[3] Chao Gong, Ovidiu Daescu, Raja Jothi, Balaji Raghavachari, Kamil Sarac, "Load Balancing for Reliable Multicast," Department of Computer Science, University of Texas ar Dallas Richardson, TX, USA.

[4] M. Matsuda, T. Kudoh, Y. Kodama, R. Takano and Y. Ishikawa, "Efficient mpi collective operations for clusters in long-and –fast networks," in IEEE International Conference on Cluster Computing(cluster 2006), 2006.

[5] K. Takahashi, H. Saito, T. Shibata and K. Taura, "A stable broadcast algorithm," in 8th IEEE International Symposium on Cluster Computing and the Grid(CCGrid), 2008, pp.392-400.

[6] T. chiba, T. Endo and S. Matsuoka, "High-performance mpi broadcast algorithm for grid environments utilizing multi-lane NICs," in 7th IEEE International Symposium on Cluster Computing and the Grid(CCGrid), 2007, pp.487-494.