A Secure Load Balancing Approach For Peer-To-Peer Networks

Vishakha Patange
Department of Computer,
SCOE,
Pune, Maharashtra

Mrs. D. D. Gatade
Department of Computer,
SCOE,
Pune, Maharashtra

***Abstract*- Load balancing among application layer peer-to-peer (P2P) networks is critical for its effectiveness but, is considered to be the most important development for next-generation internet infrastructure. Most structured P2P systems rely on ID-space partitioning schemes to solve the load imbalance problem. P2P system harnesses the resources of large populations - networked computers in a cost-effective manner such as the storage, bandwidth, and computing power. In structured P2P systems, data items are spread across distributed computers (nodes), and the location of each item is determined in a decentralized manner.**

I. INTRODUCTION

Each peer in peer-to-peer network acts as a client and server. Peer as client requests for the data and as server it fulfills the requests for other peers. In this process if a most popular data is been requested by many peers to a specific peer at a same time, then the peer gets a lot of load at the same time, hence called as overloaded. This overloaded peer is unable to fulfill the requests of other peers, hence it stuck unless the load is reduced or it gets under loaded.
In this Peer to Peer Load Balancing project the peer is managed by allocating its storage as set of virtual server. A peer which is overloaded by the requests of other peers is balanced by sending extra requests and associated virtual servers to other peer. The load of the peer is calculated in terms of number of requests it has in queue.

The requests are then send to such peers who are under loaded. If more than one peer is under loaded then a peer is chosen by checking its trustworthiness to fulfill these requests. The trust of a peer is calculated in terms of number of requests it satisfies successfully. In order to this process the requests are fulfilled without waiting in queue for such long time. This resulting in global load balancing in structured P2P systems.

i. Trustworthiness of the peer

In this Peer to Peer Load Balancing project a peer which is overloaded by the requests of other peers is balanced by sending extra requests to other peer. The load of the peer is calculated in terms of number of requests it has in queue. The requests are then send to such peers who are under loaded. If more than one peer is under loaded then a peer is chosen by checking its trustworthiness to fulfill these requests. The trust of a peer is calculated in terms of number of requests it satisfies successfully. In order to this process the requests are fulfilled without waiting in queue for such long time.

II. MATHEMATICAL MODEL

N is set of number of nodes in the network as
N= {n1, n2, n3,n4,n5,n6,…….,nn}

P represent set of storage available at each peer as
P= {p1, p2,p3,p4,p5,p6,…….,pn}

1) VS represent set of storage of virtual server available at each virtual peer at Peer in the network as
VS={vs1,vs2}
Note that in peer to peer each peer has two virtual peers which are having their own space limit set by each peer separately.
2) SDL is set of Sampled Destination List is list of network nodes those are under loaded and can accommodate upcoming load
SDL={underloadednode1, underloadednode2, underloadednode3,..,underloadednoden }

3) FS is a set of File Size uploaded by each peer
FS={fs1,fs2,fs3…….fsn}

4) US represent set of Upload Speed while file uploaded by particular peer
US={us1,us2,us3,…….,usn}

5) DS represent set of Download Speed while file downloaded by particular peer
DS={ds1,ds2,ds3,…….,dsn}
6) TRU and TRD are set of values representing Transfer Rate while Upload and Transfer Rate while Download activity of particular peer which will be calculated as follows

TRU={fs1/us1,fs2/us2,fs3/us3….,fsn/usn}
TRD={fs1/ud1,fs2/ud2,fd3/ud3….,fsn/udn}

7) LV represent the set of Load Value on the peer representing occupied space on the peer which will be calculated as

LV=$\sum Psl, Vs1sl, Vs2\, sl$-$\sum Pus, Vs1us, Vs2\, us$

Where,
sl is space limit of particular peer and their virtual servers.

us is occupied space of particular peer and their virtual servers.

8) LVPS represent Load value of complete Peer to Peer System which will be the summation of space available at all peer and their virtual servers.

LVPS =$\sum_1^n psl\ , vs1sl\ , vs2sl$ -$\sum_1^n pus\ , vs1us\ , vs2us$

Where,
sl is space limit of particular peer and their virtual servers.
us is occupied space of particular peer and their virtual servers.

A. Upload Activity :

1) Pas represents available space at requested peer which will be calculated as
Pas=Psl-Pos

Where,
Psl is size limit of requested peer.
Pos is occupies space by requested Peer

2) If Pas <Flv i.e if available space at requeated peer is less than space required to accommodate upcoming file load value i.e Flv available space is calculated.
Pas[1…n]= Psl[1…n]-Posn[1…n]
VSas[1…n]= VSsl[1…n]- VSosn[1…n]

If Pas[1…n]>= Flv or VSas[1…n]>= Flv
Add Pas or VSas to UDL

Where,
Pas[1…n] represent actual storage space of all peers in the network.
Psl[1…n] represent space limits of peer storage.
Posn[1…n] represent occupied space from peer storage.
VSas[1…n] represent actual storage space of all virtual server in the network.
VSsl[1…n] represent space limits of virtual server storage.
VSosn[1…n] represent occupied space from virtual server storage.
UDL is list of under loaded destination which can accommodate upcoming file.

3) If UDL (Under loaded Destination List) have more than one entry i.e. if more than one under loaded destination is present with same available space UPF is calculated i.e. upload performance Factor load will get transferred to peer or virtual server based on this UPF

UPF=$\sum FileSize / \sum TimeTakenToUpload$

B. Download And Simulate Load Activity :

1)Current download requests are calculated at requested destination RC i.e. Recuest Count as

RC=$\sum DownloadRequestCount$

Where,
DownloadRequestCount is and integer which represents count of request at requested destination.

2)If RC > MRC where MRC is Max Request Count Which is predefined, another peer is found having same file and migrate current downloaded request to that peer.

3)If more than one peer is having requests to the file download Performance Factor DPF is calculated and destination having high DPF is selected. DPF will be calculated as
DPF=$\sum FileSize / \sum TimeTakenToDownload$

4) In case of Simulate load request are send to all files available at peer selected to be overloaded and repeat above 3 steps for each file available at selected peer.

III. ALGORITHMS USED

A. Load Balancing While Upload Activity:

In "Peer To Peer Load Balancing System" dedicated space is provided to all peers in network to hold uploaded data. Also each peer is having two virtual servers that balance load of original peer when it will become bottleneck. While uploading files in peer to peer network, a file is uploaded to request destination peer if appropriate space is available at destination peer, If appropriate space is not available at requested peer "REALLOCATION" algorithm is applied on network node to find least loaded peer or virtual server and upcoming load i.e. file is stored on least loaded basis on peer or virtual server which is having large space limit and available space.

Algorithm: REALLOCATION UPLOAD
Input: filesize fs, destinationpeer dp, destinationpeervsavailableSpace das, destinationvsspace dvss, sampleddestinationList sdl
Output: sampleddestinationList sdl

```
1.Switch Load(fs)
        Case Load(fs)<=das
                Accommodate Load on das;
Break;

Case Load(fs)>das
        Find DestinationpeerAvailableSpace>=Load(fs)
        If DestinationpeerAvailableSpace>=Load(fs)
                Add destinationpeer To sdl;

        If destinationvsspace>=Load(fs)
                Add destinationpeer To sdl;
Break:
2.return sdl
```

Algorithm : DISTRIBUTION WITH SECURE PERFORMANCE FACTOR
Input : sampleddestinationsList sdl, filesize fs
Output : securehighperformancedestination hpd

```
1 if sdl != null
Find LeastLoaddestination hpd from sdl
        By comparing Available space at each node;
          Find peer availablespace as;
        If peer have same availablespace;
          Check for peer from secureperformanceratio;
          Add peer with secureperformanceratio to hpd


        Return hpd;
 Else return NoSpaceInNetwork;
2 If fs <= spaceavailableatlld
Allocate space to upcoming load in hpd;
 Else return NoSpaceInNetwork
```

In above DISTRIBUTION algorithm process of finding least loaded node i.e. a node that have large available capacity to hold upcoming load in proportional to remaining sampled nodes in the network along with virtual severs is done . Least loaded peer or virtual sever is selected and finally upcoming load is stored to this least loaded node.

B. Algorithm for Download Activity:

While downloading files from peer to peer network a file information table is provided on personal and global basis. In personal view files uploaded by particular peer is shown on his interface which is generated by clicking “BrowseMyFiles” button on user interface. In global view all file present on the peer to peer network irrespective of the owner of file is listed so that any user can request any file in the network. In file information catalog, file information along with file id is shown. To download any file peer will need to provide file id that is available in file information catalog and click “Download” button to download required file. To balance load on peer to peer network and a requested peer for providing files an approach is used where the migrates request to another peer if currently requested peer get bottleneck with many download request.

To achieve above scenario a request queue is implemented which holds six requests at a time. A parallel request processing is also implemented so that each request gets dedicated request handler and process separately. If request queue of requested peer get full with six requests and getting seventh request RANDOM DESTINATION GENERATOR algorithm is applied to find another destination which is having requested file. Firstly the destinations having requested file is found and then randomly selecting one of them to migrate download request to it.

If any of remaining node do not having requested file or are busy to process other download requests, System will generate a message saying all peers are busy in processing other requests and no other peer is having requested file so please try after some time.

Algorithm : DESTINATION GENERATOR WITH SECURE PERFORMANCE FACTOR

```
Input  : requestcount c, filename f, requestingpeer rp
Output : sampleddestination sd, secureperformanceratio spr, highperformancesecurepeer hsp

1 if requestcount<=5
        Identify filesize,filelocation, fileowner, finallocation
Where filename=filename
Identify requestingpeer p
Process download request for p

2 else
        For each destination
        If(destination have file f)
                Add destination to sampleddestination sd
Else
        No other peer have requested file
        Please try after some time

3 for each peer in sampleddestination sd
        Calculate secureperformanceratio spr
        Identify max of spr
        Identify respective highperformancesecurepeer hsp have max of spr

4 migrate request to highperformancesecurepeer  hsp
```

C. Algorithm for Simulate Load:

In simulate load activity request is send to selected node as many as possible to bottleneck that node to demonstrate load balancing while download activity. To implement maintained scenario a interface is developed that take name of node which want to make overloaded with download request and then migrate request if selected peer get bottleneck.

Algorithm : SIMULATE LOAD SECURE PERFORMANCE FACTOR

```
Input : nodetounderload ntd, file f, requestcount c
Output : securehiperformancepeer

1 if(ntd have file)
Select all files belong to ntd

2 request to download each file f
       If(requestcount<=5)
          Identify filesize,filelocation, fileowner, finallocation
          Where filename=filename
          Identify requestingpeer p
          Process download request for p
3 else
      For each destination
         If(destination have file f)
            Add destination to sampleddestination sd
        Else
        No other peer have requested file
        Please try after some time

4 for each peer in sampleddestination sd
    Calculate secureperformanceratio spr
    Identify max of spr
    Identify respective highperformancesecurepeer hsp have max of spr

5 migrate request to highperformancesecurepeer hsp
```

IV. RESULTS

A. System Upload Activity

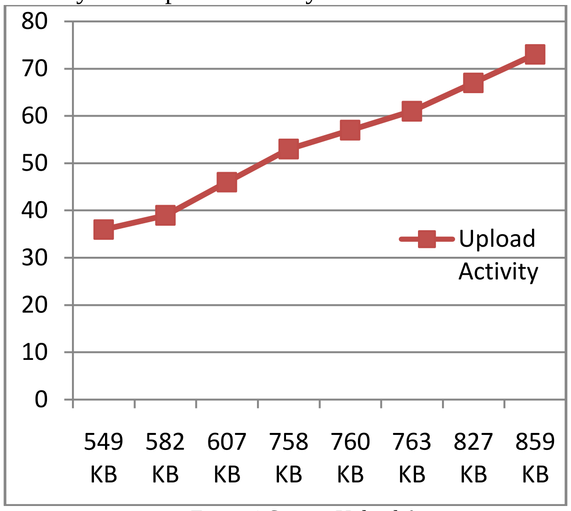

Figure 1 System Upload Activity

Figure 1 shows the upload activity of whole system. Whatever data is uploaded by the peers in the network, a record is maintained and a graph is plotted in respected to file size and time required.

B. System Download Activity

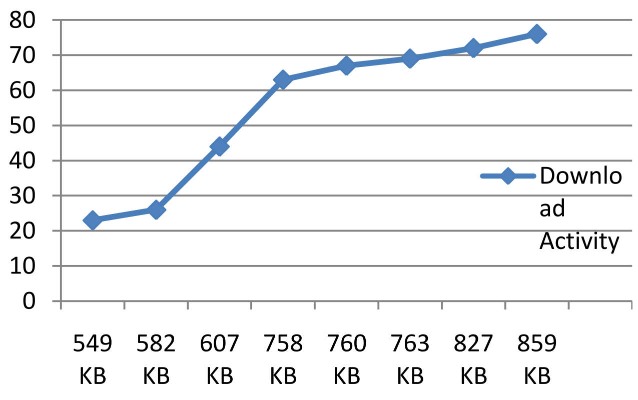

Figure 2 System Download Activity

Figure 2 shows the download activity of whole system. Whatever data is downloaded by the peers in the network, a record is maintained and a graph is plotted in respected to file size and time required.

C. System Storage

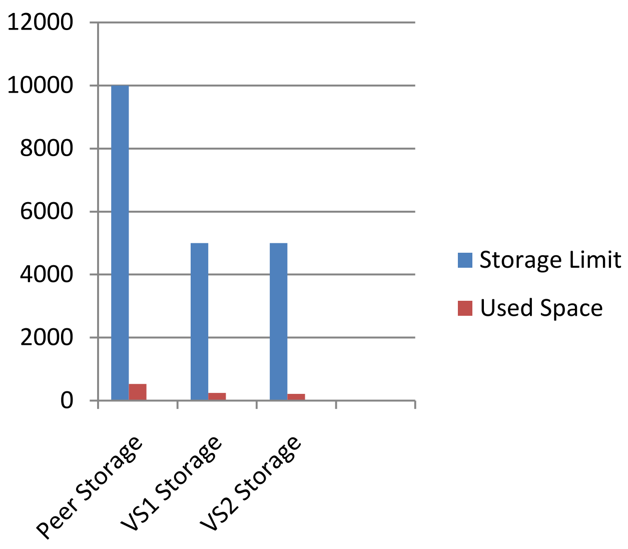

Figure 3 System Storage

Figure 3 shows the storage of a specific peer. Here, the storage limit as well as storage used is recorded and maintained.

V. CONCLUSION

A load balancing algorithm for DHTs with virtual severs is studied which represents the system state. With the DHT used, each peer identifies whether it is under loaded and then reallocates its loads if it is overloaded.

Unlike existing solutions that often rely on global knowledge of the system, each peer in our proposal independently estimates the capacities of participating peers and the loads of virtual servers based on partial knowledge of the system. Each peer identifies whether it is over-loaded and then reallocates the load. Our system is secure and based on reliable network protocol. Our system is also light weighted, can be easily getting installed on network. It also reliable in terms of network performance while uploading and downloading information. The probability of peer getting overloaded is minimized by finding least loaded peer in network to store upcoming load on the network.

VI. FUTURE SCOPE

In this project, a secure access mechanism to different peer can be provided so that one can access limited amount of information from network. Making decentralize database can also lead to decrease in work over head of single peer and improves performance. Use of various upgraded technologies and provided libraries in future can also improve data transfer rate through network.

Implementing encryption and decryption algorithm to store data in coded format can also be done and provide key using key distribution so that in case of security leak one cannot understand the coded data unless hi have security key to decrypt the information.

REFRENCE

[1]. Daniel Warneke, Christian Dannewitz, "Load Balancing in P2P Networks: Using Statistics to Fight Data and Execution Skew," IEEE Trans., Oct. 2009.

[2]. Hung-Chang Hsiao, Member, Hao Liao, Ssu-Ta Chen, and Kuo-Chan Huang, "Load Balance with Imperfect Information in Structured Peer-To-Peer Systems," IEEE Trans. Parallel and Distributed Systems, vol. 22, no. 4, Apr. 2011.

[3]. Leonidas Lymberopoulos, Symeon Papavassiliou, Vasilis Maglaris, "A Novel Load Balancing Mechanism for P2P Networking," ICST, October 2007.

[4]. Quang Hieu Vu, Member, Beng Chin Ooi, Martin Rinard, and Kian-Lee Tan, " Histogram-Based Global Load Balancing in Structured Peer-To-Peer Systems," IEEE Trans. On Knowledge and Data Engineering Vol.21, No.4, April 2009.

[5]. S. Ayyasamy, S. N. Sivanandam, " A Cluster Based Replication Architecture for Load Balancing in Peer-To-Peer Content Distribution, "International Journal of Computer Networks and Communications (IJCNC) Vol.2, No.5, September 2010

[6]. S. Surana, B. Godfrey, K. Lakshminarayanan, R. Karp, and I. Stoica, "Load Balancing in Dynamic Structured P2P Systems," Performance Evaluation, vol. 63, no. 6, pp. 217-240, Mar. 2006.

[7]. S. S. Patil, S.K. Shirgave, " Load Balancing in Structured P2P Systems using Server Reassignment Technique," International Journal of Computer Applications (0975-8887) Volume1- No.4.