Research Paper Engineering

Avoiding Loss of Data While Sharing in Structured Peer to Peer Network Overlay Modeling

* Mr. S. Venu Gopal ** Dr. N. Sambasiva Rao
*** Mr. S. K. Lokesh Naik **** Ms. T. Jagadeeswari

*,**,***,**** Department of Computer Science and Engineering, Vardhaman College of Engineering, Shamshabad, Hyderabad, India.

ABSTRACT

The peers are computer systems which are connected to each other via the Internet. Peer to Peer systems are distributed systems in that files can be shared directly between systems on the network without the need of a central server. Improving data availability by caching in structured overlay peer to peer systems. Caching is always used to achieve load balance in structured overlay peer to peer systems currently, but none of the current caching algorithm takes system load into consideration. I propose a caching model for improving data availability in structured peer to peer systems. So we can get the same data from other peers as the data of the leaving peer. The aim of my framework is to maintain a threshold level of availability of data at all times. Identify a set of factors that the data availability and propose a model that decides when more replication is necessary. We evaluate the accuracy and performance of the proposed model using simulations. Our preliminary results show that the model is effective in predicting the required number of replicas in the system. Peer uses a passive file requested statistical algorithm to evaluate the reducible query load caused by caching a file to a neighbor, and discusses how to compute the updating overhead under different updating algorithms. Each peer determines whether caching a file to a neighbor is worthwhile is based on the relations between the query load reduced and the updating overhead caused by the caching. In peer to peer systems, replication (or caching) may have the advantage of improving data availability.

Keywords : Peer to Peer, Replication, and Structured Overlays, Caching

1. Peer to Peer network:

In this paper I focused, on a p2p network such as Freenet & Gnutella, here server managing directory of peers and data is not required. Here we show the procedure of how the data exchange between peers in the system. Here I present some existing query forwarding methods and replication methods and problems.

The popularity of peer to peer file sharing application such as Gnutella and Napster has created a flurry of recent research activity into peer to peer architectures.

In order to evaluate a proposed peer to peer system, the characteristics of the peers that choose to participate in the system must be understood and taken into account. For example in some peers in a file sharing system has low-bandwidth, high-latency bottleneck network connections to the internet. The system must be careful to avoid delegating large or popular portions of the distributed index to those peers.

In this paper, we focusing on a peer to peer network such as Freenet and Gnutella, where a server managing a directory of peers and their data is not required. In this section, we show the procedure of data sharing between peers in that system In addition, we present some existing query forwarding methods and replication methods.

In peer to peer system each system is acting as a server as well as client. So if any system is added to the peer network it requests the data. The requested data may be available near-by the peer based on the neighbor peer it will share the data.

What I suggest is that if any peer requesting data that should be secure.

Here there is no centralized control of systems each system has his own control.

In future trends peer to peer plus webRTC (Web Real-Time-Communications). While communicating or sharing of data to each of the system we use some of the file sharing algorithms.

So here we apply replication methods 1. Path replication and 2. Owner Replication.

Path Replication method: The requested data is replicated on all the peers along the data transmission path between the peer requesting the data and the peer having the data.

Owner Replication: the query stores a replica of the requested data by the sender peer. Based on the requested data if peer1 having the data. Peer5 requested to peer1 so in this case peer1 will send the data to peer5. But there may be a hacking problem. To avoid this type of problem we use **break, store and make** a replica method.

Replication of requested data in peer's storage.

```
Peer.replication(DATA)

{

If(hold<capacity){

Storage.push (DATA);

}

Else{

Storage.pop();

Storage.push(DATA);

}}
```

Here what we are doing is the replicated data of the form of break and store in the peer1. If any neighbor peer5 requesting data peer1 applies **release and send method** is applied at the end of the peer5 it will receive 2 files one is (key,file). Key contains secure key of the senders details and logic data and file contains peer5 requested data.

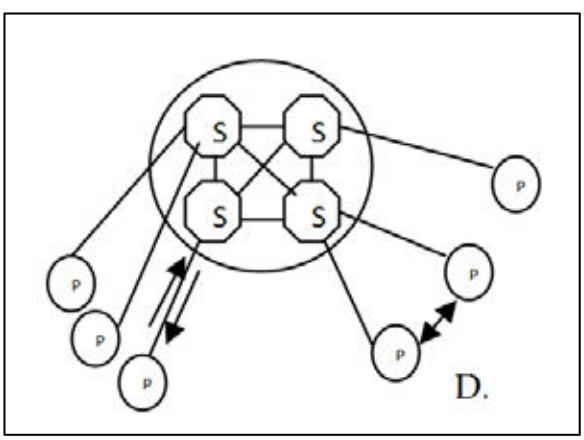
First peer5 unlock the key and store the data and apply break and store method. So here there may be a replica will store in neighbor peers this replica also uses the break and store method. If any peer requested same data the requested peer again apply release and send method.

Caching is not sufficient: store and balancing in DHTs where uniform hashing is used to achieve storage load-balancing in expectation. Caching is popularly used to deal with hotspots and query load imbalances.

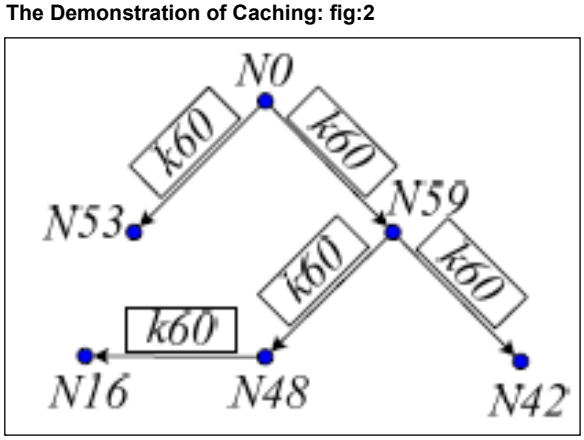
Measurement study of peer to peer file sharing system. If some peers in a file sharing system have low bandwidth, high latency bottleneck network connections to the internet.

Performing a detailed measurement study of the two most popular peer to peer file sharing systems, namely Gnutella and Napster. The hosts are will going to choose to participate in these systems are typically end-users office or home machines, located at the edge of the internet.

Napster.com: fig: 1



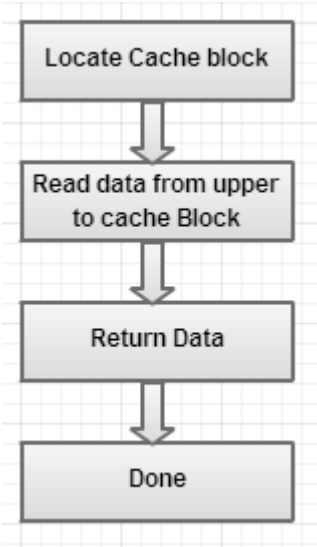
The Demonstration of Caching: fig:2



The Demonstration of Caching File requested statistical algorithm:

Peer stores two file list: local file and cache file (file_list,cache_list).

Node N0 caches the K60 to Node N53 and Node N59, Node



N59 caches to N48 and N42 and N48 caches N16 and so on. According to the cache destination of file K60 stores by the nodes N0 is N59 and N53, the cache destination of file K60 stored by N59 is N48 and N42.

Path replication replicates the data on all peers along the query forwarding route between the peer requesting the data and the peer having the data. The service providing peer receives the query which contains information about the sequence of peers that forwarding the query, then the service provider peer sends a reply and replica in the reverse direction of the query forwarding route.but random replication is difficult to implement because peers only know their neighbors, they do not know the overlay topology.

While sharing data using above caching methods there may be theft of data. To avoid this I proposed the following method.

First find out neighbor peer in the network whatever the cached file forwarding to neighbor each cached file assigned to id that we call it as cached_file_id. It contains (source_node_id, cached _file_id, destination_node_id). Next it moves to destination_node_id to next destination_node_id.

The neighbor peers stores these parameters (source_node_id, cached _file_id, destination_node_id).

A flow chart of locating cache from upper peer to neighbor peer:

Based on the above flow chart the cache block is going to update every time. If the same file is requested by the newly added node the file is fetch from the neighbor peer, but it will not move to first node.

In this case if requested file is not present in the neighbor peer search to another peer like this we can get requested file from the neighbor peer to our node.

Conclusion:

In peer to peer network each system act as a server as well as client. Any peer can send request to it's neighbor and can get data from without any loss of information. If any system crash in middle of processing without any loss of data same data can download from neighbor by using caching. So we can achieve the goal.

REFERENCES

[1]. A. Rowstron and P. Druschel. Pastry: Scalable, distributed object location and routing for large-scale peer-to-peer systems.In IFIP/ACM International Conference on Distributed Systems Platforms (Middleware), 2001. [2]. I. Stoica, R. Morris, M. F. Kaashoek, and H. Balakrishnan, “Chord: Ascalable peer-to-peer lookup service for internet applications,” in Proc.of ACM SIGCOMM, 2001. [3]. V. Padmanabhan, H. Wang, P. Chou, and K. Sripanidkulchai, “Distributing Streaming Media Content Using Cooperative Networking,” in Proc. of the 12th International Workshop on Network and Operating Systems Support for Digital Audio and Video (NOSSDAV 2002), Miami Beach, Florida, May 2002. [4]. S. Saroiu, P. Gummadi, and S.D. Gribble, "A Measurement Study of Peer-to-Peer File Sharing Systems," Proceedings of International Conference on Distributed Computing Systems, 2002 [5]. L. Garces-Erce, E. Biersack, P. Felber, K.W. Ross, G. Urvoy-Keller, "Hierarchical Peer to-Peer Systems," 2003, to appear in Euro-Par 2003, Klagenfurt, Austria. [6]. A. Datta, W. Nejdl, and K. Aberer. Optimal caching for firstorder query load-balancing in decentralized index structures. In The 4th International Workshop on Databases, Information Systems and Peer-to-Peer Computing (DBISP2P), 2006. [7]. Fang Wang, Yamir Moreno, Yaoru Sun “The Structure of Peer-to-Peer Social Networks, Phys. Rev. E 73, 036123 (2006).