Improving Lookup Performance and Consistency in DHT

Aye Moe Aung
University of Computer Studies, Yangon
Email: ayemoeaung@gmail.com

Abstract

During recent years, Distributed Hash Table (DHTs) have been extensively studied through simulation and analysis. As a fundamental problem in DHT based P2P system, lookup consistency and load balancing is important to avoid performance degradation and guarantee system fairness. In this paper, three important aspects: lookup consistency, group communication and lookup replication are involved to improve DHT lookup performance and load balancing. Lookup consistency maintains a ring structure and guarantees consistent lookup results in the presence of node join and leave, regardless of where the loop is initiated. Broadcast algorithm for group communication avoids redundant messages. Lookup replication places replicas in a DHT called Symmetric replication that enables parallel lookup results. Parallel lookups are known to reduce latencies. A new system will be implemented by integrating the three algorithms in a middle ware.

1. Introduction

Many organizations and companies are facing the challenge of simultaneously providing an IT services to millions of users. A few search engines are enabling millions of users to search the Web for information. Every time a user types the name of an Internet host, the computer uses the global domain name system (DNS) to find the Internet address of that host.

The provision of services, such as the global domain name system has many challenges. In particular, the system which provides such large-scale services needs to have several essential properties. The design needs to be scalable, not relying on single points of failure and bottlenecks. And a large-scale system needs to be self-managing, as new servers are constantly being added and removed from the system. The system also needs to be fault-tolerant, as the larger the system, the higher the probability that a failure occurs in some component. The distributed hash table, which encompasses many of the above mentioned properties.

Peer-to-Peer systems have gained tremendous popularity in recent years due to characteristics of scalability, fault tolerance and self management. Among various P2P systems, DHT system is taken as a promising approach to build a simple and yet efficient infrastructure for P2P applications.

A DHT is said to construct an overlay network [1], because its nodes are connected to each other over an existing network, such as the Internet, which the overlay uses to provide its own routing functionality. The existing network is then referred to as the underlay network. If the underlay network is the Internet, the overlay routes requests between the nodes of the DHT, and each such reroute passes through the routers and switches which form the underlay. The structured overlay network is therefore used to distinguish overlay networks created by DHTs from other overlay networks.

Structure Overlay Network (SONs) are a major class of these peer-to-peer system, examples of SONs include DKS [1], Chord [2] and Chord# [3]. SONs provide lookup services for internet scale applications. DHTs use a SON's lookup service to provide a put/get interface for distributed systems with eventually stronger consistency guarantees [4]. In contrast, many distributed systems require stronger consistency guarantees, relying on services such as consensus and atomic commits [6].

In this paper, we perform the study on three important aspects on improving DHT lookup performance in overlay network.

1.1 Lookup Consistency

Most DHTs construct a ring by assigning an identifier to each node and make nodes point to each other to form a sorted linked list, with its head and tail pointing to each other. The provide algorithms to maintain a ring structure which guarantees atomic or consistent lookup results in the presence of joins and leaves, regardless of where the lookup is initiated. Put differently, it is guaranteed that lookup results will be the same as if no joins or leaves took place. No routing failures can occur as nodes are joining and leaving. There is no bound on the number of nodes that may simultaneously join or leave the system [1]. The provided algorithms do not depend

on any particular replication method, and hence give a degree of freedom to the type of replication used in the system. Furthermore, the algorithm shows how ring maintenance can be augmented to handle arbitrary additional routing pointers.

1.2 Group Communication

Group communication can be used as a basic building block to provide overlay multicast [1]. It enables any nodes to efficiently send a message to all nodes in a specified set of identifiers, e.g. broadcast a message to all nodes with identifiers in the specified set. This approach will create one DHT per multicast group, and whenever a node requests to multicast information to a group, it broadcasts the information to all the nodes within the DHT that represents the multicast group. The provide algorithms for efficiently broadcasting a message to all nodes in a ring-based overlay network in $O(\log n)$ time steps using n overlay messages, where n is the number of nodes in the system. It shows that how the algorithms can be used to do overlay multicast.

1.3 Replication

Structure peer-to-peer system relies on replication as a basic means to provide fault-tolerance in presence of high churn [6]. The way to place replicas in a DHT called symmetric replication, which makes it possible to do parallel recursive lookups. Parallel lookups have been shown to reduce latencies. Moreover, joins or leaves only require exchanging $O(1)$ message, while other schemes require at least $\log(f)$ messages for a replication degree f. The scheme is used to do load balancing, end-to-end fault tolerance. It is a general end-to-end scheme and can be applied to all structure peer-to-peer system[1]. Furthermore, each join and leave operation only requires sending 1 message to maintain the replication degree. Moreover, nodes can make concurrent requests to any specified replicas. It is more secure as multiple requests to different replicas so not need to pass through the same node.

2. Related Work

DHTs have been the subject of much research in recent years, with substantial amount of work on resilience of overlays to churn. While these studies show that overlay network can be violated consistent lookup result in the presence of node joining and leaving. While nodes join and leave, routing tables can fail. Therefore, they also show how lookups are affected on DHT.

Daniel Stuzbach and Reza Rejaie[4] have explored the performance of lookup by analytically deriving the benefits of different ways to increase the richness of routing tables in DHT. Their results show that efficiency and consistency of lookup in DHT can be improved by performing parallel lookup and maintaining multiple replicas.

Ye Tian et. al [5] analytically study three important aspects on improving DHT lookup performance under churn such as lookup strategy, parallelism and lookup key replication. Their study is based on the different churn levels and how to select the optimal degree of parallelism.

Alexander Reinefeld et. al [6] studies various failure detection algorithms in Overlay Networks. They use the majority based algorithm the evaluate data consistency in structure overlay network.

3. System Model

This system consists of nodes, which communicate by message passing. The system employs following assumptions.

Asynchronous system: This means that there is no known upper bound on the amount of the time it takes to send a message or to do a local computation on a node.

Reliable communication channels: a channel is reliable if every message sent through it is delivered exactly once, provided that the destination node has not crashed.

FIFO communication channels: This means that messages sent on a channel between two nodes are received in the same order that they were sent.

Figure.1 shows the proposed system architecture. It consists of two main sites such as client and server site. At client site, a client has a key for which it wishes to find the associated value. The client provides the key to any one of the nodes, which then performs the lookup operation and returns the value associated with the provided key. At server side, DHT performs the lookup operation, which returns the value associated with any given key. DHT maps file names to the URLs representing the current location of the file. Each node has a routing table that contains pointers to point other nodes. A query is routed through the nodes and its neighbors such that it eventually reached the node responsible for the provided key. Most applications that use a DHT need to store more than one type of information in the DHT.

4. Work flow of DHT

DHT is a hash table which is distributed among a set of cooperating computers or nodes. It contains key/value pairs, which refer to as items. DHT constructs a ring by assigning an identifier to each node and make nodes point to each other to form a

sorted linked list, with its head and tail pointing to each other. DHT also has operations for managing items, such as inserting and deleting items. Each node is responsible for part of the items, which it stores locally. Every node able to lookup the value associated with any key.

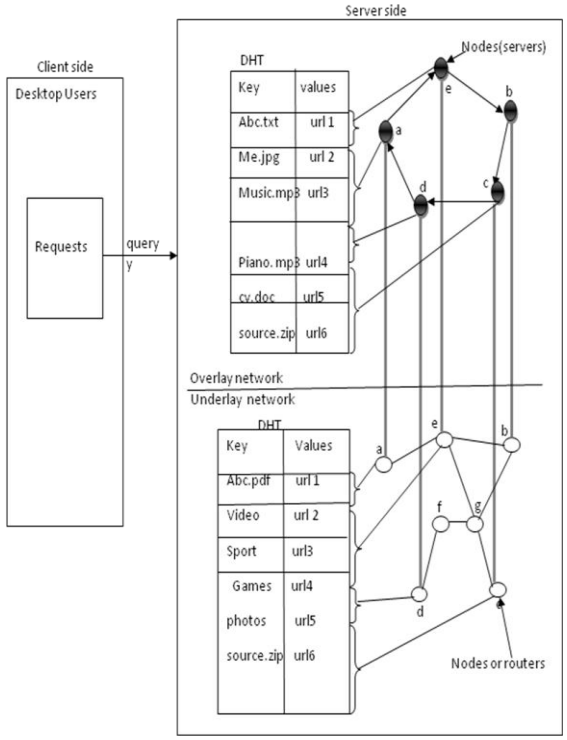

figure 1. Proposed system architecture

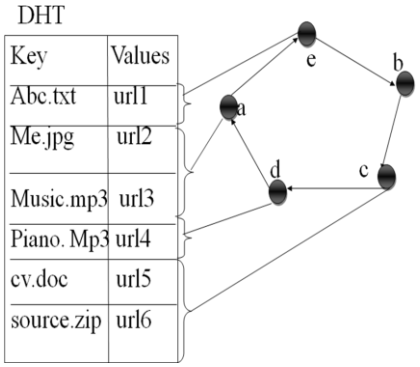

Figure 2 . Example of DHT mapping file name to the urls.

Figure 2 describes an example of a DHT mapping file names to the URLs, which represent the current location of the files. The items of the DHT are distributed to the nodes *a*, *b*, *c*, *d*, and *e*, and the nodes keep routing pointers to each other. If an application makes a lookup request to node *d* to find out the current location of the file abc.txt, *d* will route the request to node *a*, which will route the request to node *e*, which can answer the request since it knows the URL associated with key abc.txt. Not every node needs to store items, e.g. node *b*.

5. System architecture

Nodes (Servers) are connected to each other over an existing network, such as the internet, which overlay uses to provide its own routing functionality. DHTs dynamically decide which node is responsible for which items. Node can continuously join and leave the system. If nodes join, DHT will ensure that the routing table is updated to reflect departure of nodes and items are redistributed before a node leave. If nodes fail, the system will automatically replace whenever a node fail, some other node actively starts replicating the items of the failed node to restore the replication degree.

5.1 Handling Joins and Failures

Each node maintains a successor-list consisting of node's c immediate successors, where c is typically set to $\log_2$ (n), n being the network size. Each node periodically checks to see if its successor and predecessor are alive. If successor is found to be dead, it is replaced by the closet alive successor in the successor-list. If predecessor is found to be dead, node set predecessor = nil. Joins are also handled periodically. A joining node makes a lookup to fine its successor s on the ring, and set successor = s. Each node periodically asks for its successor's predecessor pointer, and updates its successor pointer if it gets a closer successor. Thereafter, the node notifies its current successor about its own existence, such that the successor can update its predecessor pointer if it finds that the notifying node is a closer predecessor. Hence, any joining node is eventually properly incorporated into the ring.

5.2 Failure Detectors

Failure detectors are modules used by a node to determine if it neighbors are alive or dead. Failure detector can only provide probabilistic results about the failure of a node. A node sends a ping to its neighbors at regular intervals. If it receives an acknowledgment within a timeout, the neighbor is considered alive. Not receiving an acknowledgment within the timeout implies the neighbor has crashed. The timeout is chosen to be much higher than the round-trip time between the two nodes.

Therefore the algorithm (lookup consistency) is used to maintain and guarantee consistent lookup result in the presence of node joining and leaving. In this system, the nodes in that network are interconnected as a group. Group information is stored in the DHT using the group name as a key and group information as a value. Group communication

is used to efficiently broadcast a message to all nodes in a ring based overlay network.

Structure peer to peer system relies on replication to provide fault-tolerance in DHT. Symmetric replication which only needs O (1) message for every join and leave operation to maintain any replication degree.

6. Implementation

In the implementation of this work, an interface system will be constructed using the popular programming language Java where a distributed hash table is available. Virtual nodes used for a single machine to join an overlay with multiple identities. It used for load balancing purpose, where nodes with more resources can assume several identities to relief other nodes. This system facilitates the use of multiple identifiers by providing a single communication manager, on top of which any number of virtual nodes can be registered. Only one IP/Port address is consumed per communication manager, regardless of the number of virtual nodes. Every node will have its own routing table, but at most one connection is open between any pair of machine. Communication between virtual nodes on the same machine does not have to go through the network.

7. Evaluation

The evaluation of system has the following structure:

Successively joined nodes into the system until the system had a network with 1024 nodes. The simulator uses an exponential distribution for the failures, join and leave under different node level. If failure detector detect as a dead node, failure detector module automatically replace that dead node. In evaluation, will be examine three metric: Hops, Latency and Message sent.

Hops : The number of hops from the source to the destination.

Latency : The duration from the start of the lookup to when a response is received by the final destination, which is a function of the number of hops and the time spent waiting for response and timeouts.

Message Sent: The overhead used to perform the lookup.

8. Conclusion

A number of approaches have been proposed for DHT lookup performance and load balancing. DHT is a peer-to-peer application that is used by hundreds of thousand simultaneous desktop users, each being part of the DHT. This may caused lookup failure or data key loss. That is used to have an efficient and decentralized replacement for common file sharing applications. This paper presents three important aspects in handling DHT lookup performance. The lookup consistency algorithm can examine ring maintenance and differentiated between leaves and failures and are able to give strong guarantees while joins and leaves are happening. The group communication algorithms are suitable for stable environments. It provides an overlay broadcast algorithm, which avoids redundant messages. It provide fault tolerant in an end to end type since the failure of a peer along the path of one request does not require repeating the request as another one of the concurrent requests succeeds. By applying replication the handling of joins and leaves can be simplified.

9. References

[1] A. Ghodsi. “Distributed k-ary System: Algorithms for Distributed Hash Tables. PhD Dissertation”, KTH Royal Institute of Technology Oct, 2006.

[2] I. Stocia, R. Morris, D. Liben-Nowell. Chord: “A Scalable Peer-to-Peer Lookup Protocol for Internet Applications. New block IEEE/ACM Transactions on Networking (TON)”, 11(1):17.32,2003.

[3] T. Schutt, F. Schintke and A. Reinefeld. “Structure Overlay without Consistent Hashing: Empirical Results”. In Proceeding of GPAPA,06 2006.

[4] D. Stutzbach and R.Rejaie. “Improving lookup performance over a widely deployed DHT”, 10 Apr 2007.

[5] D. Wu, Y. Tian, and K. W. Ng. “Analytical study on improving lookup performance of DHT systems under churn”, 2006.

[6] M. Tallat . Shafaat, Monika Moser, Ali Ghodsi, Thorsten Schutt, Seif Haridi, Alexander Reinefels. “On Consistency of Data In Structured Overlay Network”, April 4, 2008.