Data retrieval for lookup in p2p DHT-base System

IYAS ABDULLAH,[a] MOHAMMAD ALODAT[b]
[a] University of Craiova , Romania
eyasao@yahoo.com
[b] University of Craiova , Romania
systemout7@yahoo.com

Abstract. Grid computing and Peer-to-peer (P2P) systems are emerging as new paradigms for managing large scale distributed resources across wide area networks. While Grid computing focuses on managing heterogeneous resources and relies on centralized managers for resource and data discovery, P2P systems target scalable, decentralized methods for publishing and searching for data. In large distributed systems, a centralized resource manager is a potential performance bottleneck. Decentralization can help avoid this bottleneck, as is done in P2P systems. In This paper we describe techniques for indexing data stored in peer network base in distributed hash table DHT system can look up for specific data to give it to the users.

Keywords: peer to peer, overlay systems, dht-based system.

1 INTRODUCTION

Peer-to-peer (P2P) networks are among the most quickly-growing technologies in computing. However, the current technologies and applications of today's P2P networks have (at least) two serious limitations.

Poor Scaling: From the centralized design of Naspter, to the notoriously inefficient search process of Gnutella, the scalability of P2P designs has always been problematic. While there has been significant progress in this regard, scaling is still an issue in the currently deployed P2P systems.

Impoverished query languages: P2P networks are largely used for file sharing, and hence support the kind of simplistic query facility often used in file system \search" tools: Find all files whose names contain a given string. Note that \search" is a limited form of querying, intended for identifying (\finding") individual items. Rich query languages should do more than \find" things: they should also allow for combinations and correlations among the things found. As an example, it is possible to search in Gnutella for music by J. S. Bach, but it is not possible to ask specifically for all of Bach's chorales, since they do not typically contain the word \chorale" in their name.

The distributed hash table paradigms (Chord [7], Pastry [1], Tapestry [2] and CAN [12]) has addressed some of the scalability and reliability problems that plagued earlier peer-to-peer overlay networks such as Napster[10] and Gnutella[5]. Appropriate for building large-scale distributed applications due to their scalability, fault-tolerance and self-organization. However, these DHTs are designed for exact key lookup. Range queries cannot be efficiently supported since consistent hashing mechanisms de-story data locality (nearby data points in the multi-dimensional data space are mapped to the same node or to nodes that are close together in the overlay network). The challenges of extending current DHTs to efficiently support range queries on multi-dimensional data include: (1) the design of a dimension reducing scheme which can effectively partition and map the multi-dimensional data space to nodes, while preserving the data locality; (2) the design of a light-weighted routing algorithm to efficiently deliver queries to the corresponding nodes;(3) the design of load-balancing

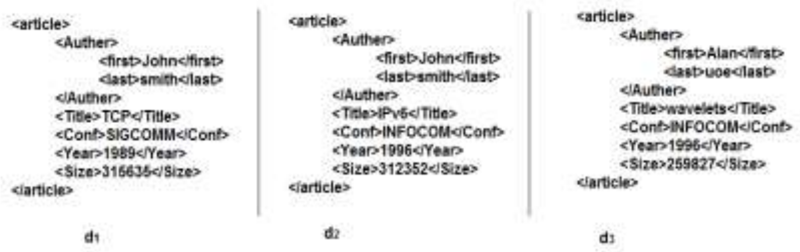
mechanisms to ensure uniform distribution of load among nodes.
A major limitation of P2P DHT systems is that they only support exact-match lookups: one needs to know the exact key (identifier) of a data item to locate the node(s) responsible for storing that item. Propose to augment P2P DHT systems with mechanisms for locating data using incomplete information. Note that we do not aim at answering complex database-like queries, but rather at providing practical techniques for searching data in a DHT.
Indexing techniques can be layered on top of an arbitrary P2P DHT infrastructure, and thus benefit from any advanced features implemented in the DHT (e.g., replication, load-balancing). We have conducted a comprehensive evaluation that demonstrates their effectiveness in realistic settings.

2 DISCRIPTION DATA TYPS

When the several query that have the same description file is known the goal of our work to be more flexible when search the of key to finding data (in node) then the aim of work that use less specific query that discover of description .
The principle to generate multiple keys for given descriptor, and to store these keys in indexes maintained by the DHT infrastructure. Indexes contain key-to-key mappings or can say a query-to-query service. By iteratively querying the index service, a user can traverse upward the partial order graph of the queries and discover all the indexed files that match his broad query.
In order to manage indexes, the underlying DHT storage system must be slightly extended. Each node should maintain an index, which essentially consists of query-to-query mappings. Then when we want to insert the query must be slightly with node (key) that responsible for query. And for look up not be more specific query, return all the list of query that be related of key of that query.
To store files and construct indexes as follows:
Given a file f and its descriptor d, with a corresponding most specific query q, we first store f at the node responsible for the key k = h(q). then generate a set of queries Q = {q1,q2 ..... ,ql} then may that each query related in that node (key). We compute the numeric key ki= h(qi) for each of the queries, and we store a mapping (qi; q) in the index of the node responsible for ki in the DHT. Iterate the process shown for q to every qi, and continue recursively until all the desired index entries have been created.

3 HOW INDEX WORK

In the Figure1 we can see the list of description of file, and then we can illustrate this description as tree of hierarchical indexing is shown in Figure 3 (We can see index key in top of box). The index at the origin of an arrow stores mapping between its indexing key and the indexing key of the target. For instance, the Last name index stores the full names of all authors that have a given last name; the Author index maintains information about all articles published by a given author; the Article index stores the descriptors (MSDs) of all publications with a matching title and author name.



```
<article>
    <Auther>
        <first>John</first>
        <last>smith</last>
    </Auther>
    <Title>TCP</Title>
    <Conf>SIGCOMM</Conf>
    <Year>1989</Year>
    <Size>315635</Size>
</article>
```

d1

```
<article>
    <Auther>
        <first>John</first>
        <last>smith</last>
    </Auther>
    <Title>IPv6</Title>
    <Conf>INFOCOM</Conf>
    <Year>1996</Year>
    <Size>312352</Size>
</article>
```

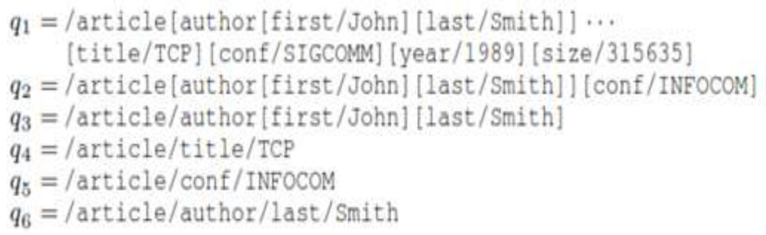
d2

```
<article>
    <Auther>
        <first>Alan</first>
        <last>uoe</last>
    </Auther>
    <Title>wavelets</Title>
    <Conf>INFOCOM</Conf>
    <Year>1996</Year>
    <Size>259827</Size>
</article>
```

d3

Figure1. Description XML File

The top of index Publication corresponds to the entries stored in the underlying DHT-based storage system. The complete keys provide direct access to the associated files. The other indexes mappings that enable the user to iteratively search the database and locate the desired files use query-to-query, Figure 2 illustrate our file queries in our system.



q_1 = /article[author[first/John][last/Smith]] ···
[title/TCP][conf/SIGCOMM][year/1989][size/315635]
q_2 = /article[author[first/John][last/Smith]][conf/INFOCOM]
q_3 = /article/author[first/John][last/Smith]
q_4 = /article/title/TCP
q_5 = /article/conf/INFOCOM
q_6 = /article/author/last/Smith

Figure 2. File Queries.

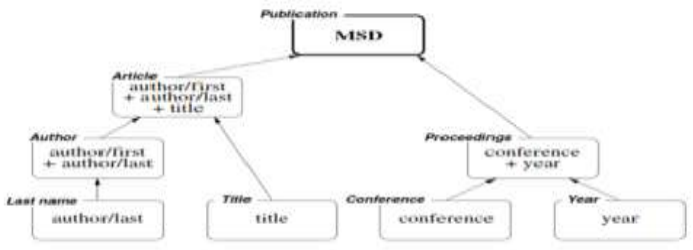


Figure 3. Index scheme for bibliography database

4 INDEXING AVAILABLE DATA OBJECTS

The peers maintain routing information about other peers at logarithmically increasing distance in the ring. A querying peer hashes the name of requested object and then uses the routing information to forward the query to an appropriate peer. CAN [12] hashes the objects into a d-dimensional coordinate space, where parts of the space are owned by peers. The peers maintain routing information about the 2d neighbors in the coordinate space. When a peer asks for an object, the object name is hashed and then the peer holding the desired hash key is located taking advantage of the structure of the hash space. The query is forwarded to this

peer via neighbors and the peer can send the requested object to the querying peer. Advantages of these schemes are that they are completely distributed and highly scalable. Moreover they do not flood the network and direct the request toward a peer that holds the relevant information. These approaches are classified as highly structured P2P systems.
Since P2P systems provide scalable storage and efficient retrieval (at least for exact-match queries), database researchers have begun to ponder if P2P systems can be designed to provide complex query facilities on top of these DHT-based P2P systems.

4.1 Construct and Processing Index

The file is discover by index entries, the file be more likely to located rapidly if the indexed enough time likely under names. The index path is effect of lookup file but lead to given file arbitrary. To reduce bandwidth requirements and reduce space the index maybe use the deeper index hierarchal.
A key component of a data-lookup system is defining index space and deterministically mapping data elements to this index space. To support complex keyword searches in a data lookup system, to associate each data element with a sequence of keywords and define a mapping that preserves keyword locality. The keywords are common words in the case of P2P storage systems, and values of globally defined attributes of resources in the case of resource discovery in computational grids.
To efficiently support range queries and queries using partial keywords and wildcards, the index space should preserve locality and be recursive so that these queries can be optimized using successive refinement and pruning. Such an index space is constructed using the Hilbert SFC in [6] and The SHA-1 is used by both the transmitter and intended receiver of a message in computing and verifying a digital signature [4].
Scheme that partitions and maps the multi-dimensional data space to the 1-d key space. The mechanism is based on the technique of k-d tree [9]. The uniformity can be achieved by a space transforming mechanism that maps the original data space to a virtual data space which has data items uniformly distributed. For most real applications, the data distribution functions are monotonically increasing and can be well modeled in advance, thus they can be used to guide the data space transforming.
In a read/write system, when the file is deleted we have to find all the index refer of the description to that file and delete all mappings.
Index entries can also be created dynamically to adapt to the users query patterns. For instance, a user who tries to locate a file f using a non-indexed query q0, and eventually finds it using the query generalization/specialization approaches.
When build an adaptive cache in the DHT to speed up accesses to popular files. Assume that each node allocates a limited number of index entries for caching purposes. A peer can create "shortcuts" entries in the caches of the indexes traversed during the lookup process. When another user looking for the same file via the same path will be capable of arrive directly to the file by following the shortcuts stored in the caches.

5 LOOKUP FILE

At first we must contract with node responsible of query that we have to look up. The node maybe return list of query that mapped at the responsible of query then we can choose one of query and repeat process until find the specific query.
As skip graph [8] and skipNet [11] used skip list, the node seeking a key and it proceeds along the same level without overshooting the key, continuing at a lower level if required, until it reaches level 0. Either the address of the node storing the search key, if it exists, or the

address of the node storing the key closest to the search key is returned.
When we look up the file using query not exist in the index and the file does exist in peer to peer system in this case looking to query have some index path that lead of file.
Examples: We can find q3, such that q3 contain in q0 and there exists an index path from q3 to d1. Therefore, the file associated to *d* can be located using this generalization/specialization approach, although at the price of a higher lookup cost.

6 IMPLEMENTATION

The index interacts with the end user, who expects to find data they want using partial information. The information returned by the system should be as concise and relevant as possible. From the system point of view, the search process should be simple, the amount of network traffic should be minimized, and the storage space dedicated to the indexing should remain within reasonable limits.
The bibliographic database contains articles published in journals and conference proceedings. We simply used that the underlying DHT is able to find a node n responsible for a given key k, where n stores data associated with key k. We used the publicly-available DBLP [3] archive, which consists of an XML-formatted list of publications. The archive being in XML format, entries are pretty similar to those in Figure 1.

6.1 Index Building

In our system we build chain of query where each of query cover the next one each chain correspond to the path from the leaf in the tree.
When the DBLP don't share and store query log then the user can search the paper by the author name, title, conference or journal, and year of publication when have special interface. Our queries correspond of chain to make our search more flexible for give specific or related papers to the user.

6.2 Index Schema

When we build the chain of query Index can acting in our system complex as shown in Figure 4.

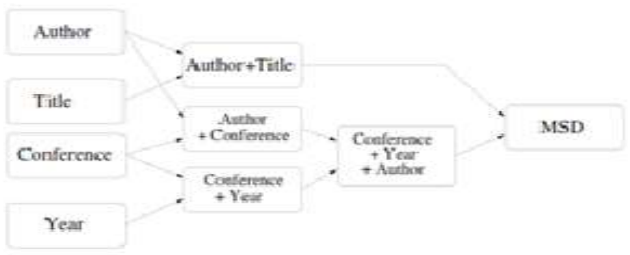


Figure4. Complex: the query in the simple split to specific query to avoid long result list

6.3 Caching

For improve the look up to some articles when this articles more popular than other and need to arrive it frequently we produce the cache entry for this articles to reduce load balances and improve time when look up in the system. We have three different cache policies:

- Multi-cache: the cache entry are created on each node along of look up path
- Single-cache: The cache entries are created on the first node are connected

- LRU (least-recently used): only a limited number of cache entries can be stored on each node Similar to the single-cache.

7 SIMULATION

In our simulate P2P network generation the node of DHT by Python and our query exist as XML using XPath on the top distributed the database of bibliographic stored of implemented of articles then measure the Index schemes and cache policies in our simulation. On each attempt sequential of indexing network from our query generator.
Simulation results are shown in Figure 5. In our result the flat indexing scheme, which creates the shortest query chains, also requires the fewest interactions to locate data.
The number of lookup steps be reduced by caching, which becomes smaller as the cache capacity increases. The multi-cache policy it presents the same characteristics as the single-cache policy. More complex data representations would need longer index chains, where the effect of shortcuts is more important.

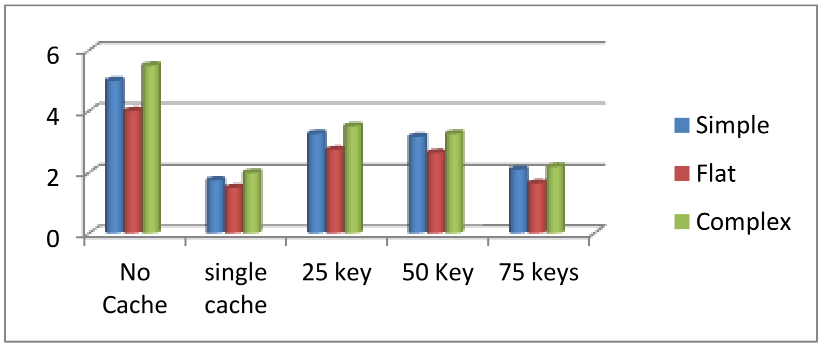


Figure5. Number interaction to find data

7.1 Efficiency of shortcut

In measure of hit ratio observe adaptive cache when the fraction of request data already in cache then don't need to go to full search path Figure 6 shows the results for the different policies tested. When most hit ratio occur in first node in the chain we will see multi-cache policy more efficient than the single-cache policy. And cause the user query is very simple and clearly these perform the user to an index chain.

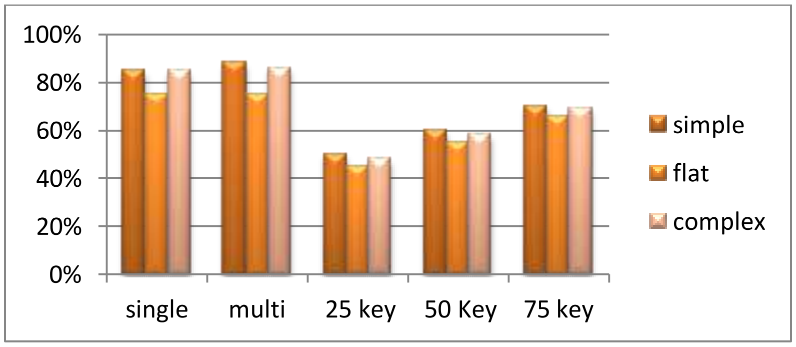


Figure6. Cache Hit Ratio.

7.2 Locating non-indexed data

It's happened when user using a query that combination of many fields (author, year, etc.) that has not been indexed. The data can be located by generalize the original query to find index entry then specialized it by follow. We can observe that the cache reduces the number of errors, because an index entry is created automatically after the first lookup. The system can still adapt to the user querying habits by creating shortcuts dynamically.

8 CONCLUSION

Our works discover specific data when their complete keys are known in advance. We index data store for lookup the data in DHT when discover matching query, we effectiveness index in P2P bibliographic database, although our data support exact-match lookups: one needs to know the exact key of a data item to locate the node responsible for storing that item but they depend on the exact matching facilities of the underlying DHT.

REFERENCES

A.Rowstronand P. Druschel, (2001) *Pastry: Scalable, decentralized object location, and routing for large-scale peer-to-peer systems,* ” in Proceedings of the 18th IFIP/ACM International Conference on Distributed System Platforms(Middleware), (Heidelberg, Germany), pp. 329– 350.

B.Y. Zhao, J. D.Kubiatowicz, and A.D.Joseph, (2001) *Tapestry: An infrastructure for fault-tolerance wide-are a location and routing,*Tech.Rep.UCB/CSD-01-1141,ComputerScienceDivision, U.C.Berkeley.

DBLP. http://dblp.uni-trier.de/.

FIPS180-1,Secure hash standard. Spring field ,VA:U.S.(April1995) Department of Commerce/NIST, National Technical Information Service,.

Gnutella webpage: http://gnutella.wego.com/.

H. Sagan.(1994), *Space-Filling Curves*. Springer-Verlag,

I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakr-ishnan.(2001) *Chord: A scalable peer-to-peer lookup service for internet applications*. In Proceedings of ACM SIGCOMM.

J.Aspnes and G.Shah,“ Skip graphs,(2003)” in Fourteenth Annual ACM-SIAM *Symposium on Discrete Algorithms*, pp.384– 393.

J.L.Bentley, (1975) *Multidimensional binary search trees used for associative searching,*” Commun.ACM, vol.18,no.9,pp.509– 517,

Napster. http://www.napster.com/.

N.J.A.Harvey, M.B.Jones, S.Saroiu, M.Theimer, and A.Wolman, , (2003*): Skipnet A scalable overlay network with practical locality properties*, in the Fourth USENIX Symposiumon Internet Technologies and Systems (USITS '03), (Seattle, WA).

S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker.(2001*) A scalable content-addressable network*. In Proceedings of ACM SIGCOMM ,.