The International Journal Of Engineering And Science (IJES)
||Volume||2 ||Issue|| 7 ||Pages|| 26-34||2013||
ISSN(e): 2319 – 1813 ISSN(p): 2319 – 1805

Improve Multi-Dimension Data lookup in P2P DHT-base System

1,Iyas Abdullah, 2,Mohammad Alodat.

ABSTRACT

The Decentralized, Resource Sharing Distributed Environments Such As Computational Grids And Peer-To-Peer (P2P) Storage And Retrieval Systems, A P2P Information Discovery System That Supports Flexible Queries Using Partial Keywords And Wild-Cards, And Range Queries. Users Looking Up Resources Stored In Peer-To-Peer Systems Have Only A Little Information For Identifying Of Resources. 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.

Date of Submission: 16July. 2013 Date of Publication: 30.July 2013

I. INTRODUCTION

The Recent Years Have Seen An Increasing Interest In Two Resource Sharing Environments: Peer-To-Peer (P2P) Computing And Grid Computing. In The Peer-To-Peer (P2P) Computing Paradigm, Entities At The Edges Of The Network Can Directly Interact As Equals (Or Peers) And Share Information, Services And Resources Without Centralized Servers. Key Characteristics Of These Systems Include Decentralization, Self-Organization, Dynamism And Fault-Tolerance, Which Make Them Naturally Scalable And Attractive Solutions For Applications. Similarly Grid Computing Is Rapidly Emerging As The Dominant Paradigm For Wide Area Distributed Computing [6].A Fundamental Problem In These Large, Decentralized, Distributed Resource Sharing Environments Is The Efficient Discovery Of Information, In The Absence Of Global Knowledge Of Naming Conventions. For Example A Document Is Better Described By Keywords That By Its Filename, Or A Computer By A Set Of Attributes Such As CPU Type, Memory, Operating System Type Than By Its Host Name. The Heterogeneous Nature And Large Volume Of Data And Resources, Their Dynamism (E.G. CPU Load) And The Dynamism Of The Sharing Environment (With Nodes Joining And Leaving) Make The Information Discovery A Challenging Problem. An Ideal Information Discovery System Has To Be Efficient, Fault-Tolerant, Self-Organizing, Has To Offer Guarantees And Support Flexible Searches (Using Keywords, Wildcards, Range Queries). P2P Systems, By Their Inherent Properties (Self-Organization, Fault-Tolerance, Scalability), Provide An Attractive Solution.

The Distributed Hash Table Paradigms (Chord [17], Pastry [3], Tapestry [12] And CAN [14]) Are 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.

II. BACKGROUND

Range partitioning designates each node responsible for one contiguous range of attribute values, and thus provides good locality. The amount of meta-data is fairly small, requiring just the attribute values at the partition boundaries. However, ensuring load balance across partitions as data evolves is a non-trivial problem.

All data elements are described using a sequence of keywords (common words in the case of P2P storage systems, or values of globally defined attributes - such as memory and CPU frequency - for resource discovery in computational grids). These keywords form a multidimensional keyword space where the keywords are the coordinates and the data elements are points in the space. Two data elements are "local" if their keywords are lexicographically close or they have common keywords. Thus, we map documents that are local in this multi-dimensional index space to indices that are local in the 1-dimensional index space, which are then mapped to the same node or to nodes that are close together in the overlay network. This mapping is derived from a locality-preserving mapping called Space Filling Curves (SFC) [2, 16].

In [13] exhibit benefit support multi-dimensions query by compare the load-balances between database solution by compares the effectiveness Partitioning Multi-Dimensional Data When data is multi-dimensional, one could still partition it based on just one dimension; this approach becomes very expensive when queries involve ranges on a non-partitioning attribute, since the query would have to be forwarded to a large number of nodes.The architecture of the presented P2P information retrieval system is similar to data-lookup systems [14, 17], Existing information storage/discovery systems can be broadly classified as unstructured or structured. Unstructured systems (such as Gnutella [21]) are based on flooding techniques and process queries by forwarding them to neighboring peers. While unstructured systems are relatively easy to maintain and can support complex queries, they do not guarantee that all matches to a query in a practical-sized system will be found. Furthermore, overheads of flooding can be significant, and a number of techniques have been proposed to reduce these overheads [4, 8, 11]. Structured data lookup systems (e.g. CAN [14], Chord [17]) use structured overlay networks and consistent hashing to implement Internet-scale distributed hash tables

Support single-attribute range queries. Andrezejaket.al [5] employ Hilbert Space Filling Curves (SFC) [16] to reversely map one-dimensional data space to CAN's d-dimensional Cartesian space, with data locality preserved. Gupta, Agrawal et.al[7] attempt to hash ranges instead of key words to nodes, and locality sensitive hashing is used to ensure that similar ranges are mapped to same node with high probability. Skip Graphs [9] and SkipNet [10] are based on skip-list, both of which support single-attribute range queries but have load-balancing problems.

At [1] Dak, for supporting multi-dimensional range queries in P2P networks. Working as a scalable indexing platform, Dak can simultaneously support multiple indexes with various dimensionalities. For each index scheme, Dak employs a uniform locality-preserving hashing mechanism to partition and map the multi-dimensional data space to a 1-d key space. The distinct feature of Dak is that it does not need to physically maintain or dynamically generate any search trees. Instead, it utilizes the trees embedded in the underlying DHT to refine and deliver queries. Exploring DHT links can eliminate the cost of maintaining additional in-network data structures for query delivery. Moreover, the maintenance messages for DHT links can be piggybacked on to the query delivery messages, so as to reduce the maintenance cost. In [20], proposes techniques to using result cache to avoid duplicate work and data movement, and used to efficiently store and retrieve such prior results by viewing tree.We can see to at [15] Partitioning Single-Dimensional Data Hash partitioning can be used to distribute tuples across a set of disks. When using a relational key as the hash attribute, this approach ensures load balance, and minimal meta-data (just the hash function). However, hashing destroys data locality, and range queries are very expensive.

Desiderata for supporting multi-dimensional queries should ideally have certain properties. First, a good partitioning strategy should have the following characteristics:

(1) Locality The cost of executing a query in a P2P system is often proportional to the number of nodes that need to process the query; hence, each query should ideally execute at as few nodes as possible. For multi-dimensional range queries, this implies that the partitioning must have locality, i.e., tuples nearby in the data space should ideally be stored at the same node.

(2) Load Balance The amount of data stored by each node should be roughly the same, to ensure load balance. This load balance should be ensured even as (a) data evolves with tuple insertions and deletions, and (b) nodes join and leave the system.

(3) Minimal Metadata define partition metadata as a directory that maps each data point to the node managing the partition containing the point. In a P2P system, there is no central site that can maintain this directory, and metadata will be distributed across the participant nodes themselves. More metadata works that need to perform in updating it across multiple nodes when join and leave the system. Therefore, we wish to keep the metadata as small and simple as possible.

III. DISTRIBUTED HASH TABLE PROPERTY

In DHT any node can use it to determine the list of node that is currently in live node. A DHT systems map keys to node in p2p system infrastructure. We assume underlying DHT-Base P2P data storage system maybe each data mapped in several peer nodes.We will use the example of a bibliographic database system that stores scientific articles. Files are identified by descriptors, which are textual, human-readable descriptions of the file's content. Let h(descriptor) be a hash function that maps identifiers to a large set of numeric keys. The peer node responsible for storing a file f is determined by transforming the file's descriptor d into a numeric key. This numeric key is used by the DHT substrate to determine the node responsible for f. In order to find f, a node n has to know the numeric key or the complete descriptor.

3.1 Description of Data

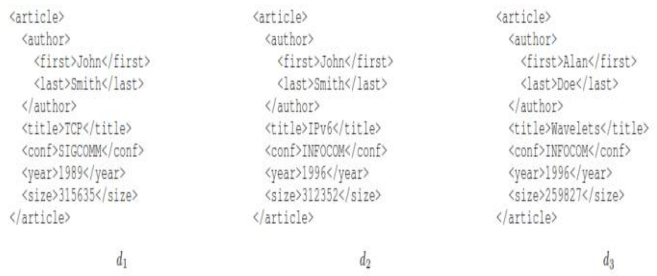
In our work the description of data taken from large of data for example (DBLP [22]) used by many publicly-accessible databases disappear as XML. By XML Path Language (XPath) is syntax and a data model for addressing parts of an XML document. It includes some features of a general-purpose expression language and is designed to be a little language that can be used for application-neutral processing within XML systems. For example, could use XPath to locate all the section-title elements in a document.XPath processes anything as a sequence. A sequence is an ordered heterogeneous collection of items. The items can be either nodes from an XML document or atomic values. Atomic values can be any type defined in the XML Schema recommendation, including complex types. To declare a sequence in an XPath, just separate the items with commas and enclose the whole sequence in parentheses: (2, 'declencheur', 5.10)A path is made of location steps separated by a forward slash (/), such as:/po:PurchaseOrder/po:ProductList/po:Name.However, the location steps in XPath identify items in a sequence (those items might be XML nodes) instead of nodes in a tree.

3.2 Order of data

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 find the data (in node) then the aim of work that to 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 then 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.

3.3 The 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 Figure4. 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>
  <author>
    <first>John</first>
    <last>Smith</last>
  </author>
  <title>TCP</title>
  <conf>SIGCOMM</conf>
  <year>1989</year>
  <size>315635</size>
</article>
```

d_1

```
<article>
  <author>
    <first>John</first>
    <last>Smith</last>
  </author>
  <title>IPv6</title>
  <conf>INFOCOM</conf>
  <year>1996</year>
  <size>312352</size>
</article>
```

d_2

```
<article>
  <author>
    <first>Alan</first>
    <last>Doe</last>
  </author>
  <title>Wavelets</title>
  <conf>INFOCOM</conf>
  <year>1996</year>
  <size>259827</size>
</article>
```

d_3

Figure 1: Description XML files

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 3 illustrate our file queries in our system, and we observed algorithm for search query in nodes at Figure 2.

```
1:  Let $\{n' : p(n') = n\} = \{c_1, \dots, c_n\}$
2:  $q_w \leftarrow q$
3:  $r_w \leftarrow \emptyset$
4:  **loop**
5:    **if** $q_w = \epsilon$ **then**
6:      **return** $r_w$
7:    **else**
8:      Let $q_w = a_1 \dots a_m$
9:      **if** $\nexists i, j, s_1, s_2 : s_1 \| a_i \| s_2 = r(c_j)$ **then**
10:       **return** $r_w$
11:     **else**
12:       $(i^*, j^*) \leftarrow \min\{(i, j) : \exists s_1, s_2 : s_1 \| a_i \| s_2 = r(c_j)\}$
13:       $r_w \leftarrow r_w \cup \{r(c_{j^*})\}$
14:       $r' \leftarrow S(c_{j^*}, a_{i^*+1} \dots a_m)$
15:       $q_w \leftarrow q_w \ominus r'$
16:       $r_w \leftarrow r_w \cup r'$
17:     **end if**
18:   **end if**
19: **end loop**
```

Figure 2: Search for Query q in node n, S(n, q)

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 3: File Queries.

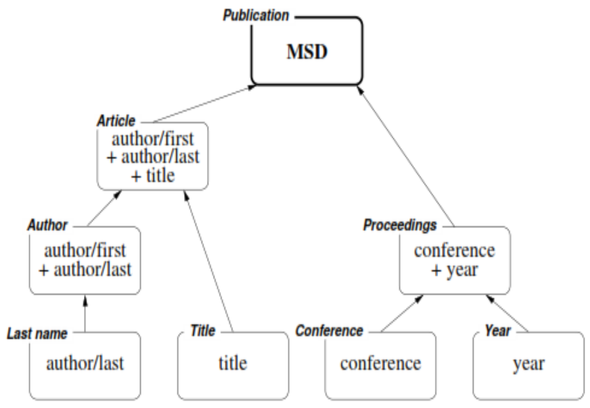


Figure 4: Index scheme for bibliography database

3.4 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 found the specific query.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 however 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.

Approach of indexing available data objects in the system : 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 [14] 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 for-warded 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.

3.5 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 the 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 [16].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 [18]. 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/specializati on 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 able to arrive directly to the file by following the shortcuts stored in the caches.

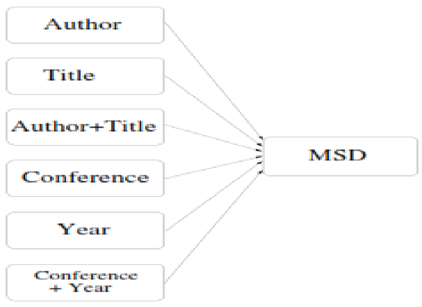


Figure 5: Simple: each of queries depends of point that forms a set of pair to return it. Each of pair point to MSD

IV. 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 [22] 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.

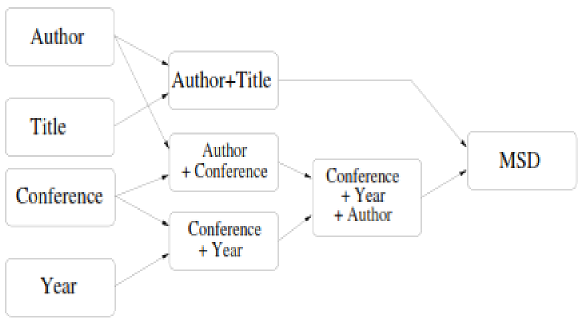


Figure 6 Flat: All queries that appear in simple are possible point directly to give us the MSD.

4.1 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.

4.2 Index Schema

Index can acting as three types when we build the chain of query in our system: simple, flat and complex as shown in Figure 5, 6, 7.

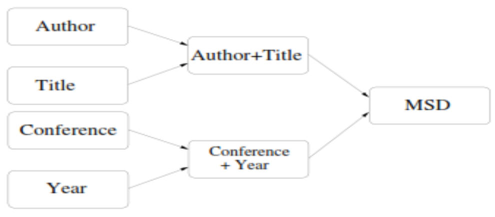


Figure 7 Complex: the query in the simple split to specific query to avoid long result list

4.3 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.

V. 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.The studied parameters are: number of user-system interactions required to find data, efficiency of shortcuts, and storage dedicated to caching.Interactions required to find data when user send a query and the list of result are appear of specific queries. The user selects one of articles that match of target. However, that the number of iterations is expected to increase when the user initiates the search with a generic query.Simulation results are shown in Figure 8. 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.

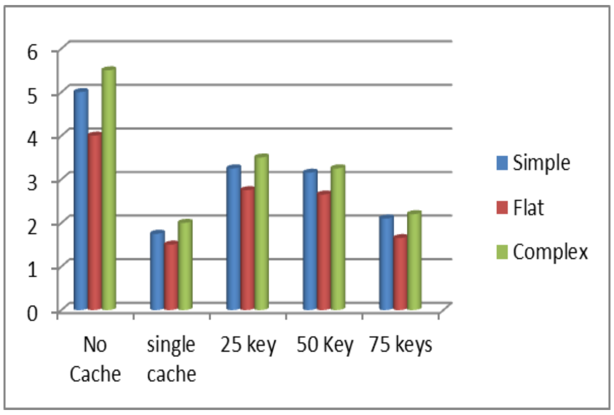


Figure 8: Number interaction to find data

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 9 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 this perform the user to an index chain

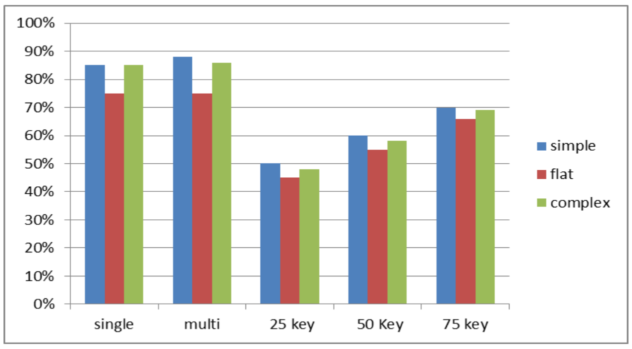


Figure 9: Cache Hit Ratio.

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.

VI. 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

[1] by Xiaoyu Yang , Yiming Hu, "A scalable index architecture for supporting multidimensional range queries in peer-to-peer networks" (2006) , in Proceedings of the International Conference on Collaborative Computing: Networking, Applications and Worksharing (CollaborateCom).
[2] T. Bially. A class of dimension changing mapping and its application to bandwidth compression . PhD thesis, Poly-technic Institute of Brooklyn, June 1967
[3] A.Rowstronand P. Druschel,"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, Nov. 2001.
[4] A. Crespo and H. Garcia-Molina. Routing indices for peer-to-peer systems. In Proceedings of the 22nd International Conference on Distributed Computing Systems (ICDCS),Vi-enna, Austria, July 2002.
[5] A.Andrzejakand Z.Xu,"Scalable, Efficient range queries for grid information services,"in Proceedingsofthe SecondIEEE International Conference on Peer-to-PeerComputing(P2P2002), (Sweden), pp.33 40,Sep.2002.
[6] I. Foster and C. Kesselman. The GRID - Blueprint for a New Computing Infrastructure. Morgan Kaufmann Publishers, Inc., San Francisco, CA, 1999.
[7] A.Gupta,D.Agrawal, and A.El Abbadi,"Approximate range selection queries in peer-to-peer systems, in Proceedings of the First Biennial Conference on Innovative Data Systems Research (CIDR), (Asilomar , CA),January 2003.
[8] A. Iamnitchi, I. Foster, and D. Nurmi. A peer-to-peer approach to resource discovery in grid environments. Technical Report TR-2002-06, University of Chicago, 2002.
[9] J.Aspnes and G.Shah,"Skip graphs,"in Fourteenth Annual ACM-SIAM Symposium on Discrete Algorithms, pp.384–393,Jan 2003.
[10] N.J.A.Harvey, M.B.Jones, S.Saroiu, M.Theimer, and A.Wolman, Skipnet: A scalable overlay network with practical locality properties, in the Fourth USENIX Symposiumon Internet Technologies and Systems (USITS '03), (Seattle, WA),2003.
[11] Q. Lv, P. Cao, E. Cohen, K. Li, and S. Shenker. Search and replication in unstructured peer-to-peer networks. In Proceedings of the 16th international conference on Supercomputing , June 2002.
[12] B.Y. Zhao, J. D.Kubiatowicz, and A.D.Joseph, "Tapestry: An infrastructure for fault-tolerance wide-are a location and routing," Tech.Rep.UCB/CSD-01-1141,ComputerScienceDivision, U.C.Berkeley, Apr. 2001.
[13] Prasanna Ganesan, Beverly Yang, Hector Garcia-Molina "One Torus to Rule them All :Multi-dimensional Queries in P2P Systems" Proceedings of the 7th International Workshop on the Web and Databases: colocated with ACM SIGMOD/PODS 2004
[14] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker. A scalable content-addressable network. In Proceedings of ACM SIGCOMM , 2001.
[15] A.Silberschatz,H.F.Korth,andS.Sudarshan."Database SystemConcepts", chapter17.McGraw-Hill,1997.
[16] H. Sagan. Space-Filling Curves. Springer-Verlag, 1994
[17] I. Stoica, R. Morris, D. Karger, F. Kaashoek, and H. Balakr-ishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of ACM SIGCOMM, 2001.
[18] J.L.Bentley, "Multidimensional binary search trees used for associative searching," Commun.ACM, vol.18,no.9,pp.509–517,1975

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

[20] Bobby Bhattacharjee Sudarshan , Sudarshan Chawathe , Vijay Gopalakrishnan , Pete Keleher , Bujor Silaghi "Efficient Peer-To-Peer Searches Using Result-Caching" (2003) In Proc. of the 2nd Int. Workshop on Peer-to-Peer Systems

[21] Schmidt, C.; Parashar, M.;Rutgers Univ., Piscataway, NJ, USA ." Flexible information discovery in decentralized distributed systems " High Performance Distributed Computing, 2003. Proceedings. 12th IEEE International Symposium on

[22] DBLP. http://dblp.uni-trier.de/.