A Keyword-Set investigate System for Peer-to-Peer Networks

[1] Sivaram Rajeyyagari
Assistant Professor, Department of Information Technology, EVM College of Engineering and Technology, Narasaraopet, Guntur, Andhra Pradesh

Abstract
The Keyword-Set Search System (KSS) is a Peer-to-Peer (P2P) keyword look for sys-tend that uses a circulated wrong side up index. The main challenge in a distributed index and search system is sentence the right proposal to partition the index across the nodes in the network. The most understandable scheme would be to partition the index by keyword. A keyword partition index requires that the list of index entries for each keyword in a search be retrieved, so all the lists can be joined; only a few nodes need to be contact, but each sends a potentially great amount of information, The document list for each set of keywords is then fetched from the network. The lists are intersected to compute the list of similar documents. The list of index entry for each set of words is smaller than the list of entrance for each word. Thus look for using KSS results in a slighter query time over head.
Keyword: *Peer-to-Peer,KSS,Napster,network*

1.Introduction

Peer-to-peer (P2P) systems that enable music file sharing have turn out to be popular among the Internet users. harmony file sharing P2P systems characteristically have millions of users[21]. high and mighty each user contributes a few documentation to the network by *sharing* music files, there are nine of millions of files obtainable in the network. The user is effectively pooling their luggage compartment and network resource to make a large number of files available to each other. A large number of files make a music file sharing system attractive to a lot of users, but it also makes it harder to find the file that a customer wants. Keyword search scheme allows users to search files by specify a few of the words from the desired metadata fields. User enters the search words, then the system communicate with supplementary nodes and fetch the matching results. In the composition file sharing systems, keyword search is done using centralized as well as spread algorithms

In a centralized scheme, a single node is accountable for maintaining a centralized index that maps words to files in a particular peer. The entire query are routed to the central server. Since, the server maintain s the index for the entire network, the member of staff serving at table can tell a client what files are matching and where to find persons files. Napster[17]is an example of such a centralized system. Napster users connect to the Napster folder of online songs, search for songs and get a stick to the peers that are sharing those files. This architecture however creates a innermost point

of failure. Since all the queries are being answer from a central server, the organization responsible
When a swelling receives a query, the inflammation runs the uncertainty in its local manifestation and sends a reply with matching filenames. Thus, Gnutella a voids centralization, but at the expense of network flooding during a query. Broad casting the query to all neighbors is necessary in the basic Gnu tells a procedure because the system does not be acquainted with what files are life form shared in the system to the front of time. Hence, each time a query is made, the classification needs to ask other nodes if they have matching files. One way to make a disseminated search system efficient is by the theater some pre-computation earlier than queries are made. The system then uses data structures build and inhabited during pre-computation to do less meting out and announcement while answering queries. Construction a search index is a well-liked pre-computation in a search system. When query are made, these indices can be used to specially locate the nodes serving the corresponding documents without have to exchange a few expressions by means of all the nodes.

2.Distributed Inverted Indexing

An Inverted Index is a data bargain that maps words to files and hence helps quickly find the article in which a known word appears. construction and update an reverse Index is a pre-computation step in which searchable terms are extract from a paper. Once language are extracting, an associative list is shaped that connections each statement with a document dial (and optionally the position in which the word appear in that document).In the propose obtainable in this thesis, the index does not surround the setting for the remark.A wrong way up fact list can be built nearby and served from a on its own attendant or site. Web study engines such as Google inch the web, construct the file in the surrounding area and use it to suddenly find the consequences for query.P2P file allocation structure like Napster also use a curbed index that get updated when a user sticks in to the Napster server and shares a file. The slender index is used by the attendant to answer query.
A spread indexing system, instead of storing the entire index in one attendant (perhaps simplemented as a server bunch), distributes the index to multiple nodes in the net. The brave is finding the right scheme to partition the in crosswise the nodes in the intricate. The straight forwar story is to screen the file by keywords in a paper and hoard all index entries for a given keyword on a specific node. In this proposal, to method a query, the system needs to fetch index entries for each speech in the query from the network and cumulative the results. If a client searches for *you*, the system downloads file entries for *love* and *you* and intersects the lists. Since near are hundred thousands of songs with the word *worship* in the title with hundreds of thousands of songs with the appearance *you* in the designation, the system transmits a large number of guide entries (potentially a few Mega bytes in size) for both of these words. This is in correctly large

band width for query in a P2P classification because crowd width available to most nodes in the internet is rather diminutive [21].
We recommend incremental penetrating which addresses the subject of system traffic caused by multi-word query, by incrementally recurring documents sort by pagerank. The search format works as follows: at what time a Boolean multiword uncertainty is conventional, the first term in the uncertainty is examined and is routed to the peer which owns the part of the index that contains this term. The alphabetical listing is access and a inventory of credentials that the index entry point to are examined and sort by the pagerank. Instead of forward all these documents to the peer answerable for the next term, only the top x% of hits are forward. So, the next look closely receives only a small fraction, albeit surrounding the most important documents. The second examine also follow the same practice. It finds the credentials that match the succeeding term and performs the boolean operation on the two sets of documents.

3.Napster

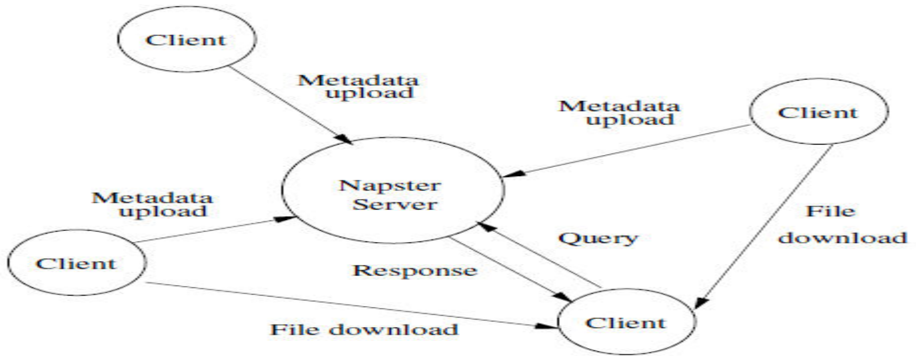

Figure 1: consumers close to a middle Napster server for content location. Files are downloaded beginning peers.

Napster is system that allows citizens to share music. The search catalog is centralized; the luggage compartment and serving of les is distributed. When a bump joins the system, the node sends a list of all the les to the Napster server. A purchaser looking for a musicle goes to the Napster and issues a query. The member of staff serving at table searches its index, which is built by means of the metadata sent by the peers. The end result for a query is a list of IP addresses of the peers that are distribution the ¯le. The customer then downloads the in a without stopping line from the peer. From distribution standpoint, Napster is a P2P bargain. on the other hand, from the indexing end of view, Napster is a countrywide understanding. completion To simplify the psychoanalysis the following assumption are made: A consistent network in which all peers are always in attendance. A network transport model where the peers bring together all the page rank mail for each other generate during one pass into a single message, and broadcast this message in one network call per peer. every one peer serializes sending of these communication to other peers instead of sending them at the same time as to all peers. The IP speak to caching scheme proposed in Section 3.2 is used, hence the pagerank bring up to date messages can be directly exchanged between peers lacking having to be running scared every time. The computational employment to be done is constant .

4.Conclusion

During in this paper, a prototype of a KSS system and keywordwas built. The current version of KSS software has a command line interface to upload documents, download documents, and search for documents using query keywords. Ultimately, a statically linked application with a GTK interface should be built and an alternative web-proxy based UI should be provided to the users

5.Bibliography

[1] E. Adar and B. Huberman. Free riding on gnutella, 2000.
[2] Anurag Singla and Christopher Rohrs. Ultrapeers: Another Step Towards Gnutella Scalability, December 2001.
[3] Christopher Rohrs. Query Routing for the Gnutella Network, December 2001. http://rfc-gnutella.sourceforge.net/Proposals/QRP/query routing.htm.
[4] Arturo Crespo and Hector Garcia-Molina. Routing indices for peer-to-peer systems. Technical Report 2001-48, Stanford University, CS Department, November 2001.
[5] Frank Dabek, M. Frans Kaashoek, David Karger, Robert Morris, and Ion Stoica. Wide-area cooperative storage with CFS. In Proceedings of the 18th ACM Symposium on Operating Systems Principles (SOSP '01), Chateau Lake Louise, Banff, Canada, October 2001.
[6] FastTrack. http://www.fasttrack.nu/. [7] FreeDB. http://www.freedb.org/. [8] Gnutella. http://gnutella.wego.com/.
[9] Gordon Mohr. Hash/URN Gnutella Extensions (HUGE) v0.92, October 2001.
[10] Grokster. http://www.grokster.com/.
[11] Jordan Ritter. Why Gnutella Can't Scale. No, Really. http://www.darkridge.com/ jpr5/gnutella.html
[12] D. Karger, E. Lehman, F. Leighton, M. Levine, D. Lewin, and R. Panigrahy. Consistent hashing and random trees: Distributed caching protocols for relieving hot spots on the World Wide Web. In Proceedings of the 29th Annual ACM Symposium on Theory of Computing, pages 654–663, El Paso, TX, May 1997.
[13] Kazaa. http://www.kazaa.com/.
[14] L F Mackert and G M Lohman. R* optimizer validation and performance evaluation for distributed queries. In Proceedings of the 12th International Conference on Very Large Data Bases, pages 149–159, Kyoto, Japan, 1986.
[15] Mark A. Sheldon and Andrzej Duda and Ron Weiss and David K. Gifford. Discover: a resource discovery system based on content routing. Computer Networks and ISDN Systems, 27(6):953–972, 1995.

[16] James K Mullin. Optimal semijoins for distributed database systems. In IEEE Transactions on Software Engineering, pages 558–560, May 1990.

[17] Napster. http://www.napster.com/.

[18] Scott Shenker Qin Lv, Sylvia Ratnasamy. Can heterogeneity make gnutella scalable? In First International Workshop on Peer-to-Peer Systems (IPTPS) 2002, Cambridge, MA, March 2002.

[19] Sylvia Ratnasamy, Paul Francis, Mark Handley, Richard Karp, and Scott Shenker. A scalable content addressable network. In Proceedings of SIGCOMM 2001, San Diego, August 2001.

[20] Patrick Reynolds and Amin Vahdat. Efficient peer-to-peer keyword searching. Technical Report 2002, Duke University, CS Department, February 2002.

[21] Stefan Saroiu, P. Krishna Gummadi, and Steven D. Gribble. A measurement study of peer-to-peer file sharing systems. In Proceedings of Multimedia Computing and Networking 2002 (MMCN ’02), San Jose.

22] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan. Chord: A scalable peer-to-peer lookup service for internet applications. In Proceedings of the ACM SIGCOMM ’01 Conference, San Diego, California, August 2001.

[23] Sumeet Thadani. Meta Information Searches on the Gnutella Network, 2001. http://rfcgnutella.sourceforge.net/Proposals/MetaData/meta information searches.htm.

[24] Steve Waterhouse. Jxta search: Distributed search for distributed networks. http://search.jxta.org/JXTAsearch.pdf.

[25] Beverly Yang and Hector Garcia-Molina. Efficient search in peer-to-peer networks. Technical Report 2001-47, Stanford University, CS Department, November 2001.

[26] Ben Zhao, John Kubiatowicz, and Anthony Joseph. Tapestry: An infrastructure for fault-tolerant wide-area locat ion and routing. Technical Report UCB/CSD01-1141, Computer Science Division, U. C. Berkeley