Past-based Search Function in Pastry

Wang-Cheol Song
Department of Computer Engineering, RIAT
Jeju National University
Jeju, South Korea
philo@jejunu.ac.kr

Seung-Chan Lee
Department of Computer Engineering
Jeju National University
Jeju, South Korea
this.dreamzinn@gmail.com

***Abstract*—The P2P technology is very popular to share several kinds of contents such as mp3 and video in the Internet. Recently the structured P2P has been studied widely, and there are many P2P algorithms such as Pastry, CAN, Chord, Tapestry and so on based on the Distributed Hash Table (DHT). As these structured P2P provides the context-aware routing, content can be shared among user-peers without help of any servers. However, the general users are used to utilize the keyword-based search engine in the Internet for searching content, and the structured P2P supports only the exact matching. Hence, a keyword-based searching algorithm is required. In this paper we propose a search function in Pastry – one of the structured P2Ps. The search function is designed using PAST which is the file storage system of Pastry.**

Keywords-Search Function; Structured P2P; Pastry; PAST; DHT

I. INTRODUCTION

The P2P technology is very popular to share several kinds of contents such as mp3 and video in the Internet. From year 2000 various structured P2P technologies have been announced to access the contents without the server [1-3]. Most of them are based on the Distributed Hash Table (DHT), so that if the name of the contents is known, the location of the contents can be found without help of any servers.

When we want to use services in the Internet, we are used to find something using a couple of keywords through search engine in the web. Because of that, as you know, there are many search engines in the Internet nowadays. So, some people may guess the structured P2P does not need the search function because if we only know the name of contents, we can be routed to a location of the service that we want. That is one of the advantages of the structured P2P over the unstructured P2P. But, the general users are not familiar with such way. That is, users generally do not know the exact name of contents they want. They ordinarily know a couple of keywords or remember only some parts of the name.

As the structured P2P operates with no servers, the user-peers cannot ask to search something to a search engine. With the exact name of the content, they just trust the structured P2P to take users to where the content is without searching operation. In other words, if you do not know the exact name, you never can go to the node having the content. If we want to provide the search function to users, the existing server based search algorithms to provide location information of content cannot be applied because the structured P2P assumes no server. Therefore, as the nodes of a peer-to-peer network cannot rely on a central server coordinating the exchange of content location, they are required to actively participate by independently and unilaterally performing tasks such as searching for other nodes, locating content.

There have been recent works related with the search function in P2P networks. [11] introduces several structured P2P technologies and describes required functions and issues for them. [4] outlines a research agenda for building complex query facilities on top of the DHT-based P2P systems. Lundgren et al. [5] propose a search engine called SCAN on Pastry. [6, 7] says P2P keyword searching based on DHT. Also, there are other approaches to design new DHTs for peer to be routed to the content without searching function [12] [13]. Although these works verify need of the search function in the structured P2P network, no one has proposed the keyword-based search function without the server.

Pastry [2] used in this paper is a structured P2P overlay network as the location and routing substrate that is efficient, scalable, fault resilient, and self organizing. It represents a second generation of peer-to-peer routing and location schemes along with Tapestry [8], Chord [1] and CAN [3]. Pastry assigns the unique identifier from a circular 128-bit namespace to every node and every object as a nodeId and key, respectively. When a message and a key are given, the message can be efficiently routed to the node with the nodeId numerically closest to the key. It guarantees a definite answer to a query in a bounded number of network hops. Also, Pastry assumes an existing infrastructure at the network layer, and the emphasis is on self-organization and the integration of content location and routing. In the viewpoint of scalability, it can be noticed that Pastry nodes only use local information. The global information exchange in the routing algorithms limits the scalability, necessitating a hierarchical routing architecture like the one used in the Internet. In addition, Pastry achieves network locality so that the entries in the routing table of each Pastry node are chosen to be close to the present node according to the proximity metric. With these merits, we have chosen Pastry as the platform to develop P2P based applications in the further research.

PAST [9] is an application of Pastry as the P2P storage system. Replicas of an object are stored at the k nodes whose nodeIds are the numerically closest to the object's key as in Figure 1. PAST also maintains the invariant that the object is replicated on k nodes, regardless of node addition or failure. Since nodeId assignment is random, these *k* nodes are unlikely to suffer correlated failures.

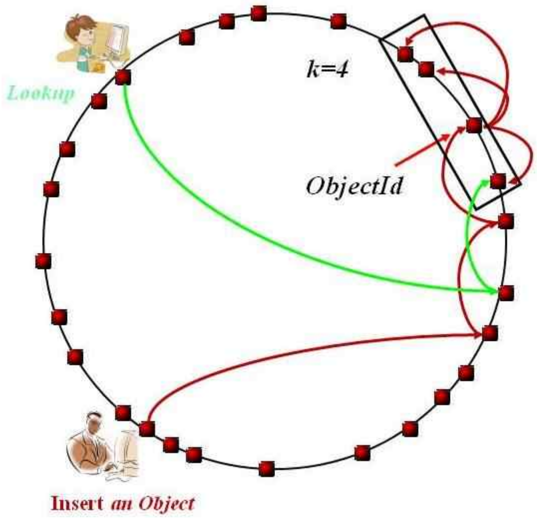

Figure 1. PAST operation in the Pastry ring

In order to develop the search function in Pastry, first we need to summarize the characteristics of the DHT in the aspect of search function.

- When the id of an object is known, we can access the node having the object directly and get it. That is, Pastry is content-addressable. It is because Pastry uses a name space for both nodes and objects and the object is stored in a node with nodeId identical to the object id.
- The id is uniquely generated by the hash function. As a similar keyword does not generate a similar hash code, an object' id is independent from id of other objects with the similar name.
- Pastry supports DHT based exact matched searching, but provides no facilities to search objects or nodes with related keywords.

We can easily imagine that when a user uses network based applications, he or she may want to begin with searching some contents by related keywords. As the exact content names are not usually known in most of cases and some people may want to know a list of more things than the exact content as what they want, keyword-based search function must be supported, although Pastry provides the content-addressability.

The rest of this paper is organized as follows. Section 2 describes the related work. Section 3 presents the design of search function in Pastry and Section 4 briefly explain it with a scenario. Section 5 shows performance evaluation and we conclude in Section6.

II. Related Work

There have been recent works for search function in P2P networks. A system to support complex queries over structured peer-to-peer systems is proposed in [4]. This approach relies on the underlying peer-to-peer system for both indexing and routing, and implements a parallel query processing layer on top of it.

[5] proposes a search engine called SCAN to effectively perform distributed user lookup based on Pastry. This paper has the same approach as the keyword-based search engine, but it encodes Content meta-data e.g., keywords using ASCII table, into a set of Pastry keys for NodeIds that are inserted into the network. [6, 7] says peer-to-peer keyword searching and are based on DHT. But they propose the mechanisms for collaboration among peers as the search engine servers sharing indices.

[13] is the searching algorithm for a structured P2P - CAN in [3]. It proposes a searching algorithm named Recursive Partitioning Search (RPS) and develops the DHT properly modified for the searching function. This approach is intended to build the modified DHT and perform the blind searching. [12] proposes a peer-to-peer information retrieval system to support content- and semantic-based full-text searches. It also modifies the DHT of CAN, and presents resources and queries as vectors so that documents in the network are organized around their vector representations using a ranking algorithm. [14] is a extension from [12] to use a two-phase distributed semantic indexing method. [15] also proposes to modify the DHT, but it takes the Ontology approach.

Compared to the above works, we intend to design a keyword-based searching function. Usually such a search function is supposed to need the server operation, but we are sure that if we utilize PAST - the P2P storage of Pastry we can develop an efficient and scalable searching function specified to Pastry.

III. Design of Search Function in Pastry

The structured P2P uses the DHT to find the requested contents by using the content's exact name without servers such as DNS. It supports only the exact matching. It is because as the DHT is based on the hash function, similar keywords produce entirely different results. However, users usually want to search what they want by using a couple of interesting keywords and get the list of the available contents.

We have designed the search function in the Pastry. We assume that every Pastry object has one or more keywords and the owner of the object register the keywords when he joins the Pastry ring. A couple of points should be considered for design of the search function as follows: As the node id and the object id are hashed in the Pastry, every id is independent each other. So, object ids hashed with keywords related to the object must be independent from an object id hashed with object's name. But the general users want to search objects by using one or more related keywords. Secondly, we should find where the owner of the object can store the keywords. This question arises because we do not want to lean on any server storage.

In this paper we have resolved them by using the P2P storage – the PAST. When an object joins the Pastry ring, it stores every keyword in the PAST as follows: *n* hashed keywords and the object's id are stored as *n* pairs of the content's names and the content itself like (h_1, N), (h_2, N) … (h_n, N) through a PAST command – **Insert** as in Figure 2. Then, when a peer wants to find the object by one or more hashed keywords, he can try to find the object's id through a PAST command like **Lookup**(h_3) and get the object's id, N. As we can expect, the more popular the keyword is, the more objects' ids the **Lookup** function returns. Then, this search function returns a list of objects and the most overlapped ids in the **Lookup** results are ahead listed up.

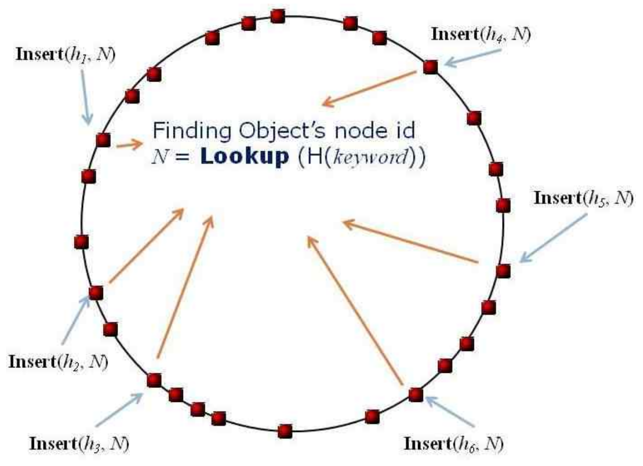

Figure 2. Storing and Searching keywords in PAST

PAST itself provides the robust P2P storage system by storing the replicas in the leaf set nodes. And, the objects stored in PAST are maintained as replicated on k nodes, regardless of node addition or failure. Therefore, the proposed search function based on the PAST is expected to be robust, too.

IV. A SIMPLE SCENARIO

If we think a *rose* as a Pastry content, it may have some keywords such as *flower*, *red* and *thorn*. And, if codes of the keywords – *flower*, *red* and *thorn* can be hashed as h_1, h_2 and h_3, the pairs of (h_1, *rose*), (h_2, *rose*) and (h_3, *rose*) can be stored using PAST command – **Insert**. Also, for keywords - *star*, *red*, *energy* and *center* hashed as h_4, h_2, h_5, h_6 in case of *sun*, the pairs of (h_4, *sun*), (h_2, *sun*), (h_5, *sun*) and (h_6, *sun*) can be stored in PAST.

Then, when a user want search an object with two keywords – *flower* and *red*, by using the hashed id h_1 of *flower* and the hashed id h_2 of *red* we can search the object like **Lookup**(h_1) and **Lookup**(h_2). Two lookup commands may return results as (*rose*) and (*rose*, *sun*) respectively. From these results the user can get a list as the search result in a way that the most overlapped object is first displayed. So, the user gets *rose* as the most related object.

V. PERFORMANCE EVALUATION

As the proposed search function is based on PAST, we think it could be robust and scalable like the characteristics of PAST. In order to evaluate the performance, we have simulated it on the open source code of the FreePastry [10].

For the simulation environment, we have run the FreePastry version 2.1 with JAVA JDK 6 on a Microsoft Window XP machine. The FreePastry is implementation of the Pastry as well as PAST. It could operate on the real network, but we have done the simulation on the Network Simulator of the FreePastry. We have selected the EuclideanNetwork as the network topology in the simulator. As we use the virtual clock in this simulation environment, we assume there is neither error nor delay in the network.

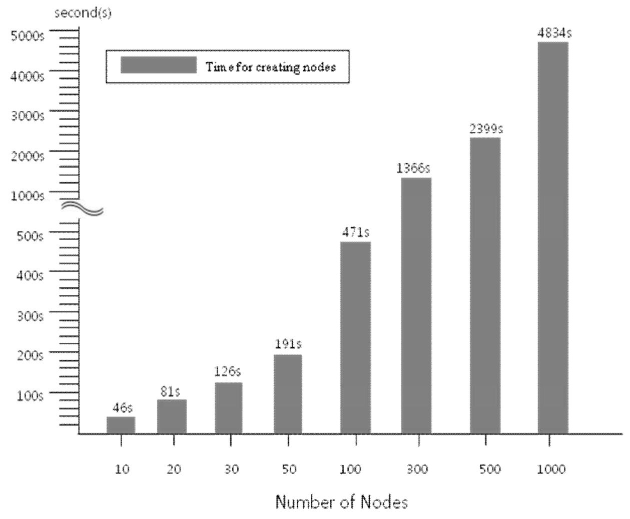

Figure 3. Time for Creating nodes in Pastry ring

As the starting point, we have measured time for creating nodes in the Pastry ring as in Figure3. We have measured time for the following number of nodes: 10, 20, 30, 50, 100, 300, 500 and 1000. Although measured time increases according to the number of nodes, we can know the averaged time per a node is almost constant near 4 seconds.

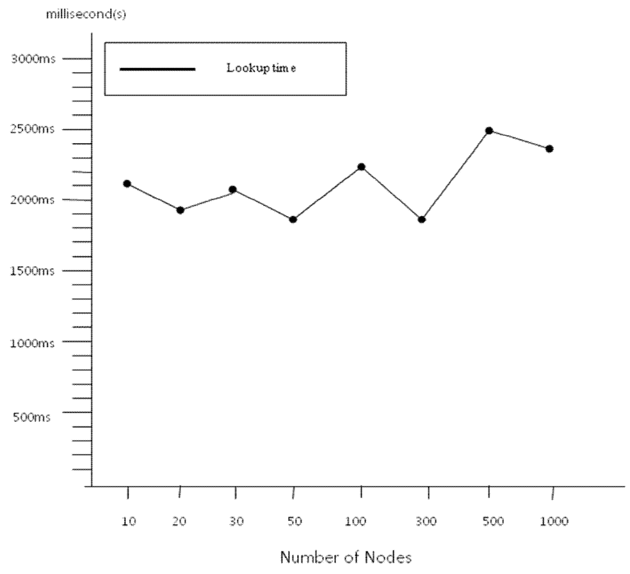

Figure 4. Time to retrieve five keywords

We have measured time to retrieve keywords as the Lookup time. We have assumed each node has five keywords. Every node simultaneously puts the **Lookup** command to retrieve a keyword five times, and measures time for a node to retrieve all of five keywords in cases of the following number of nodes: 10, 20, 30, 50, 100, 300, 500 and 1000. Figure 4 shows the measured time for the Lookup. The Lookup time for querying five keywords is almost flat near 2 seconds regardless of the number of nodes. From this figure we can say this search function is scalable.

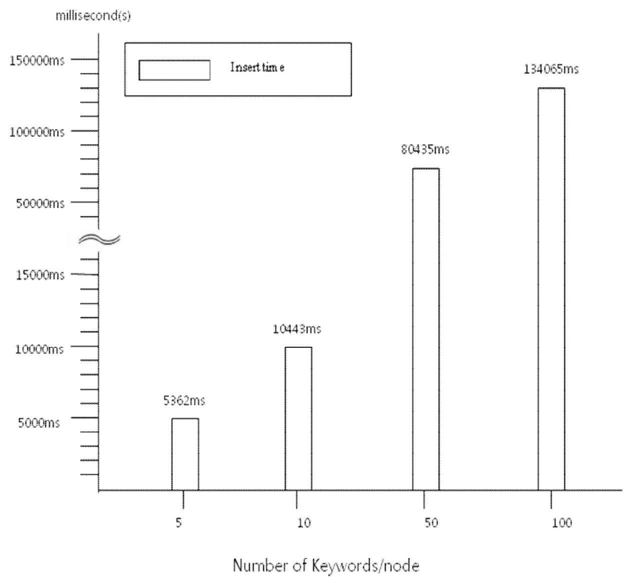

Figure 5. Insert time in 100 nodes

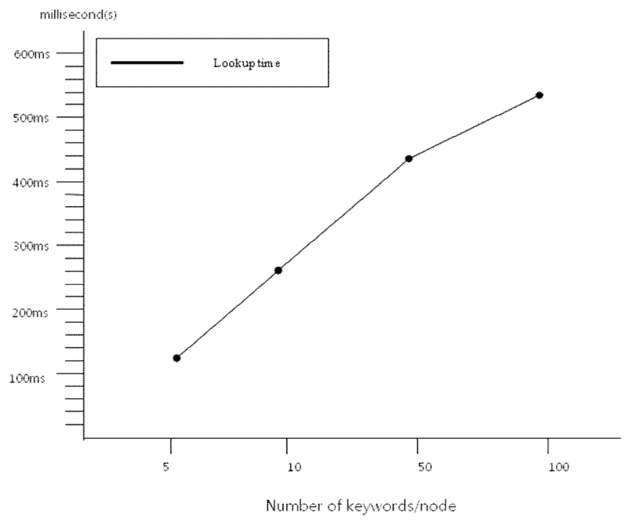

Figure 6. Lookup time in 100 nodes

Figure 5 and Figure 6 show the Insert time and the Lookup time when varying number of keywords for 100 nodes. In Figure 5 every node simultaneously issues the **Insert** commands of the number of keywords to store a keyword, and measures time for a node to store all keywords in cases of the following number of keywords: 5, 10, 50 and 100. The Insert time is shown to increases according to the number of keywords. However, the measured time per a keyword is 1.07, 1.04, 1.61 and 1.34 seconds in each case. These values look rather flat.

Figure 6 is the same case as the Figure 5, except the **Insert** command is replaced with **Lookup**. The Lookup time increases according to the number of keywords. But, the Lookup time per keyword is just 24, 26, 8.4 and 5.4 milliseconds, and it looks decreasing. We think it is due to parallel processing of retrieving keywords. When the number of keywords is small, its effect is a little shown, but as the number of keywords increases the parallel processing gets the effect. Therefore, we think the Lookup time get no effect from the number of keywords.

With the results in the above, we can conclude our search function is scalable. Therefore, we can say it could be valuably utilized in the Pastry based applications.

VI. CONCLUSION

Since many structured P2P algorithms are proposed and developed recently, there are so many tries to use one of them to develop various P2P systems. Our team has also tried to develop a system in Pastry, but we realize we need a search function. We wanted to use some keywords to discover content object, but we could not. As we have described, the structured P2P provides only the exact matching.

We have proposed a search function and evaluate the performance to be scalable. We can see it gets little effects from the number of nodes as well as the number of keywords. Pastry is a popular structured P2P platform and PAST is provided as an open source as the well matched system. If the DHT should be developed in other way for the search function, applying the Pastry to a system development is difficult. But, as we just use the original Pastry DHT with PAST, we think we can keep the advantages of Pastry DHT to apply Pastry to develop a system. PAST itself is already proven robust and scalable. Therefore, we think our system can be scalable. Also we expect it could be robust. We expect the proposed search function could be applied to various developments in the near future.

ACKNOWLEDGMENT

"This research was supported by the MKE(The Ministry of Knowledge Economy), Korea, under the ITRC(Information Technology Research Center) support program supervised by the NIPA(National IT Industry Promotion Agency)" (NIPA-2010-(C1090-1011-0009))

REFERENCES

[1] Ion Stoica, Robert Morris, David Karger, M. Frans Kaashoek, and Hari Balakrishnan, "Chord: A Scalable Peer-to-Peer Lookup Service for Internet Applications,"Proc. of the 2001 ACM SIGCOMM Conf., pp. 149-160, 2001.

[2] A. Rowstron and P. Druschel, "Pastry: Scalable, Decentralized Object Location and Routing for Largescale Peer-to-Peer Systems," in IFIP/ACM Int'l Conf. on Distr. Systems Platforms (Middleware), 2001, pp.329-350.

[3] S. Ratnasamy, P. Francis, M. Handley, R. Karp, and S. Shenker, “A Scalable Content-Addressable Network,” Proc. of ACM SIGCOMM, pp. 161-172, 2001.

[4] M. Harren, J. Hellerstein, R. Huebsch, B. Loo, S. Shenker, and I. Stoica, “Complex queries in DHT-based peer-to-peer networks,” LNCS Vol. 2429, pp. 242 – 259, In IPTPS 2002.

[5] H. Lundgren, R. Gold, E. Nordström, M. Wiggberg, “A Distributed Instant Messaging Architecture based on the Pastry Peer-To-Peer Routing Substrate,” In Proc. of Swedish National Computer Networking Workshop, Stockholm, Sept. 2003.

[6] Hanhua Chen, Hai Jin, Jiliang Wang, Lei Chen, Yunhao Liu, Lionel M. Ni, “ Efficient multi-keyword search over p2p web”, In WWW pp. 989-998, 2008.

[7] Patrick Reynolds and Amin Vahdat, “Efficient Peer-to-Peer Keyword Searching”, In Middleware, pp. 21-40, 2003.

[8] B. Y. Zhao, J. D. Kubiatowicz, and A. D. Joseph, “Tapestry: An infrastructure for faultresilient wide-area location and routing,” Technical Report UCB//CSD-01-1141, U. C. Berkeley, April 2001.

[9] Peter Druschel and Antony Rowstron, “PAST: A large-scale, persistent peer-to-peer storage utility,” In Proc. HotOS VIII, Schloss Elmau, Germany, pp. 75-80, May 2001.

[10] http://www.freepastry.org/FreePastry/, May 2010.

[11] Androutsellis-Theotokis, S. and Spinellis, “A survey of peer-to-peer content distribution technologies,” *ACM Comput. Surv.* 36, 4, pp. 335-371, Dec. 2004.

[12] Tang, C., Xu, Z., and Mahalingam, M., “pSearch: information retrieval in structured overlays,” SIGCOMM Comput. Commun. Rev. 33, 1, pp. 89-94, Jan 2003.

[13] Vishnevsky, V., Safonov, A., Yakimov, M., Shim, E., and Gelman, A. D., “Scalable Blind Search and Broadcasting in Peer-to-Peer Networks,” In Proceedings of the Sixth IEEE international Conference on Peer-To-Peer Computing P2P. IEEE Computer Society, Washington, DC, pp. 259-266, 2006.

[14] Y.Chen, Z. Xu, C. Zhai, “A Scalable Semantic Indexing Framework for Peer-to-Peer Information Retrieval,” in ACM SIGIR 05 Workshop on Heterogeneous and Distributed Information Retrieval, 2005.

[15] C. Sangpachatanaruk, T. Znati, “A P2P Overlay Architecture for Personalized Resource Discovery, Access, and Sharing over the Internet,” in CCNC’05, pp. 24-29, 2005.