Efficient Query Processing in Peer-to-Peer Environment

Htar Htar Aye, Kyaw Zar Zar Phyu
University of Computer Studies, Yangon
htarhtaraye2010@gmail.com, kyawzarzarphyu@gmail.com

Abstract

Peer to Peer system extends ideas about distributed computing and client server computing, blurring the distinction between clients and servers. The P2P paradigm was recently adopted in the database community to overcome the limitations of distributed system, namely the static topology and the heavy administration work. XML databases are widely used for P2P system. Query processing over the P2P database faces many challenges and problems. This paper presents design and implementation of the distributed XQuery processing over P2P system. The system can dynamically execute distributed XQuery over the P2P system. Query processing is dynamic and designed to reduce network traffic because it only forwards partial results that are needed by the neighbor peer instead of every result. Each peer in the system has the query processing capabilities and its own catalog. This paper presents the design and implementation of the Peer–to-Peer query processing system using XML database. This system uses hotel and resort information system as a case study.

Keywords: Peer-to-Peer, Query processing, XML Database

1. Introduction

Peer to Peer computing offers new opportunities for building highly distributed data systems. Unlike client server computing, P2P is a very dynamic environment where peers can join and leave the network at any time. This yields important advantages such as operation without central coordination, peers autonomy, and scale up to large number of peers. Using XML database in P2P can benefit some advantages because XML offers an ideal mechanism to transfer short, structured messages between peer applications. XML can be easily customized for specific P2P systems and readily transmitted over today's Internet protocols. XML data can be encrypted using existing technologies, making it an ideal candidate for secure messages. There are already several implementations of XML-based messaging schemes, including SOAP and XML-RPC.

Utilizing XML to cache application data locally in P2P systems offers several advantages. Caching data in XML allows for more flexibility and easier retrieval than custom or unstructured formats, and it has a much smaller overhead than installing a relational database on each peer. Developers can take advantage of XML handlers to search, validate, retrieve, and manipulate the data needed to support the peer application. This approach will reduce the overall complexity of the P2P system. In many cases, XML stores are easier to implement than storing unstructured data directly in the file system and require less system resources to operate than relational databases. In this system, the distributed query processing with XML database is implemented as a hotel and resort information system.

The remaining of this paper is organized as follows. Section 2 comments on some related works. Section 3 presents P2P system. Section 4 describes distributed XQuery processing. Overview of the system is presented in section 5. Section 6 is the conclusion of the paper.

2. Related Work

Distributed database and distributed query processing techniques for relational database is described in [1]. Those ideas can also be extended for XML database. Many of the approaches described in this paper are same for distributed XQuery processing such as query decomposition and query rewriting.

Distributed query processing techniques for XQuery based on the mediator with adaptor that is connected to the remote database are presented in [2].

Query routing for P2P infrastructure is discussed in [3], they focus query routing for digital libraries base on P2P model.

Optimization of distributed XQuery is presented in [4]. The authors use query algebra on the active XML document and perform optimization based on the algebra.

Processing XML twing query pattern which are frequently executed operation of the XQuery is described in [5].

Distributed query processing on P2P system is described in [6], they use DHT (Distributed Hash Table) for their P2P infrastructure.

Distributed database extension called XRPC that used P2P infrastructure and XML for distributed P2P application is presented in [7].

3. Peer-to-Peer System

In a P2P, every participating node acts both as a client and as a server and pays its participation by providing access to some of its resources, most frequently, processing power and/or disk space. Although this idea is common to all P2P system, they differ considerably in their underlying architecture. Distributed query processing in P2P faces many challenges due to the nature of P2P systems. The system needs to manage its own catalog for distributed query processing and query routing for each peer and accumulating the final query result.

3.1 Types of P2P System

- **Structured P2P networks** employ a globally consistent protocol to ensure that any node can efficiently route a search to some peer that has the desired file, even if the file is extremely rare. Such a guarantee necessitates a more structured pattern of overlay links. By far the most common type of structured P2P network is the distributed hash table (DHT). Distributed hash tables (DHT) are a class of decentralized distributed systems that provide a lookup service similar to a hash table, (*key*, *value*) pairs are stored in the DHT, and any participating node can efficiently retrieve the value associated with a given key. Responsibility for maintaining the mapping from keys to values is distributed among the nodes, in such a way that a change in the set of participants causes a minimal amount of disruption.
- **Unstructured P2P network** is formed when the overlay links are established arbitrarily. Each peer knows its neighbors, but doesn't know the resources they have. Query routing is typically done by flooding the query to the peers that are hop distance from the query originator. In pure P2P networks, peers act as equals, merging the roles of clients and server. In such networks, there is no central server managing the network, neither is there a central router. Some examples of pure P2P Application Layer networks designed for file sharing are Gnutella and Freenet.

The system presented in this paper uses unstructured P2P system.

4. Distributed XQuery Processing

The use of XML has exploded in recent years. Even more information is passed between systems as transitory XML documents.

XQuery is a query language designed by the W3C to address these needs. It allows selecting the XML data elements of interest, reorganizing and possibly transforming them, and returning the results in a structure of the user wish [8].

XQuery has a rich set of features that allow many different types of operations on XML data and documents, including: selecting, filtering, searching, joining data, sorting, grouping, and aggregating data, transforming and restructuring XML, performing arithmetic calculations on numbers and dates, manipulating strings to reformat text.

Query processing in this system performs according to the following steps:

- **Query Parsing**: Distributed XQuery to be executed must be first parsed by query parser to form the query parse tree. Query parsing involve checking the syntax of the query, annotating the required semantic information for the query plan tree and gathering the other information for query plan. The result of the query parsing is the query parse tree node which can be further processed by the query plan tree generator.
- **Construction of Query Plan Tree:** query tree produced by the query parsing step is used to construct the query plan by the query plan generator. Query tree consists of query operator nodes such as Join, And, Simple node etc. Query Plan defines a set of nodes and operations for the distributed query, each node can be executed with its own XML database.
- **Query Evaluation:** query evaluation step finds the unevaluated portion of the query plan tree. When the query node is executed by the current query processor, it transforms the logical query in the query plan into the physical XQuery statement, and it executes the physical XQuery and caches the result. The evaluated query tree node is modified to reflect changes such as marking the query tree node.
- **Query Routing:** if no node can be further evaluated by the current query processor, it must route the query plan to its neighbor. If the whole query plan tree is evaluated, the resulting query result is sent back to the query initiator.

5. Overview of the System

The system uses hotel and resort information system as a case study. There are three sites in the system: Hotel site, Resort site and Hotel Ranking site. Each site has its own query processor capable of performing local XQuery operation. Each site can process their corresponding XQuery given by other sites via query plan. When a user posts queries to server by accessing a user interface posed by peer, the system constructs textual XQuery statement for the user's query. The execution of XQuery statement may be distributed because some of the required information cannot be satisfied by the local site. In order to successfully execute distributed XQuery statement, the system parse the textual XQuery into query plan, query plan is an internal representation shared by different XQuery processors in the system. It consists of query operator, query variable and query operator tree. Query operator defines operation for the node that is "Join" operation or "And" operation and query variable is the variable used in XQuery and the corresponding catalog document. And query plan is formatted as an XML file so that it can be transmitted over network easily. At any given site, a site accepts query plan from its neighbor, constructs the logical query plan from XML file, and processes the query plan.

Processing of the query plan consists of checking each query node in the query plan tree to see whether the local query processor that can fulfill the resource required by the query plan tree node. For example, if the query plan tree node requires some resources held by the local query processor, then it can process that query tree node. After the query tree node is processed, the resulting XML file is cached in the local query processor. When there is no other node that can be fulfilled by this local query processor, the query plan is then routed to the next neighbor. When the whole query plan tree is evaluated, the resulting XML file is sent back to the query initiator.

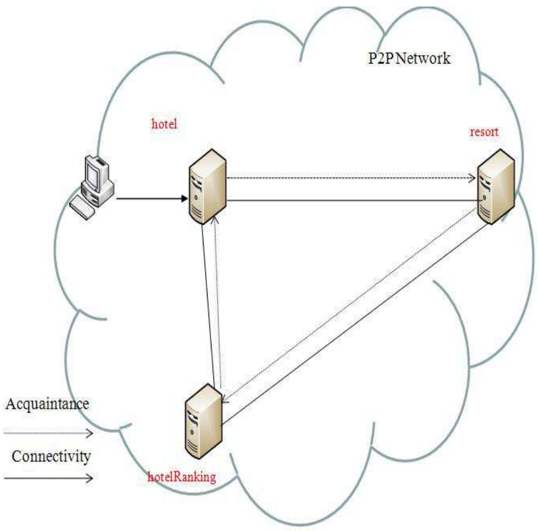


Figure 1. System Architecture

Figure 1 shows the overall system architecture. The figure describes the state of the query that is initiated at hotel site. However, query can be initiated at any site in the system. . Figure 2 shows the flow diagram of the system.

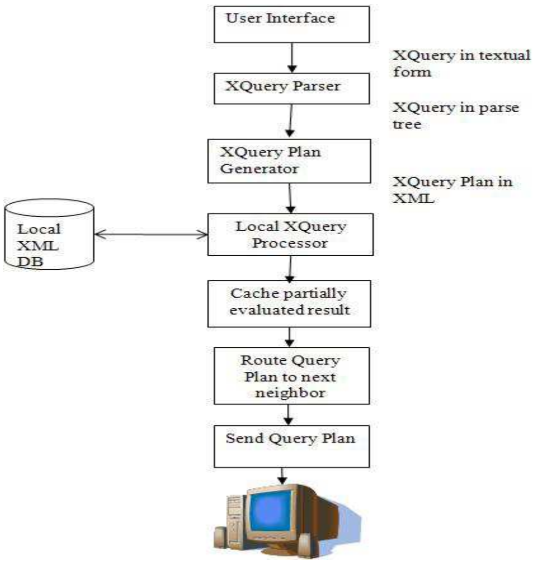


Figure 2. System Flow Diagram

For example, when a user wants to know all hotels' names in the rank between 2 and 4 and location of resort which type is golf in same location with hotel and names of resort, the system will construct the following XQuery.

```
for $a in  doc('Resort.xml')/Resorts/Resort[Type='Golf'],
     $b in doc('Hotel.xml')/Hotels/Hotel,
     $c in
doc('HotelRanking.xml')/HotelRanking/Ranking[Rank>=2
and Rank<=4]
     where $a/Location=$b/Location and
$b/HotelName=$c/RankName
      return
<result>
      {$b/HotelName}{$a/Location}{$a/ResortName}
</result>;
```

5.1 Query Parser

The task of the query parser is to parse the textual XQuery statement given by a user into query plan. Query parser simplifies the latter task of query processing, eg., it defines which expression should be executed locally, and which operation should be performed with which kind of node. Query plan consists of query variable, query operator, and query plan tree. Query variables are variables that are used in the original XQuery textual statement, query operator are Join, Select etc. Query plan tree is a binary tree consisting of various query tree node. Query tree node can be primary, "Join" node and "And" node. Primary node is the simplest form of node that specifies the query that can be executed by only one query processor, query that can be executed locally without other query processor.

5.2 Query Plan Generator

The task of query plan generator is to consume the query plan tree produced by the query parser and construct the query plan. Query plan is the logical query execution step decomposed into a set of tree node so that any node can be executed by a single XQuery statement. In other word, Query plan generator simplifies the query processing step so that each query tree node can be executed by individual query processor. Figure 3 shows the query plan tree for example XQuery described in section 5.

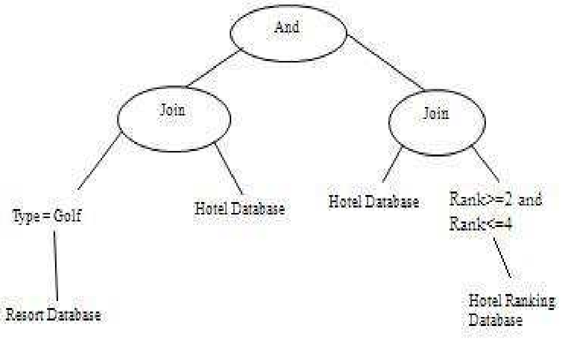
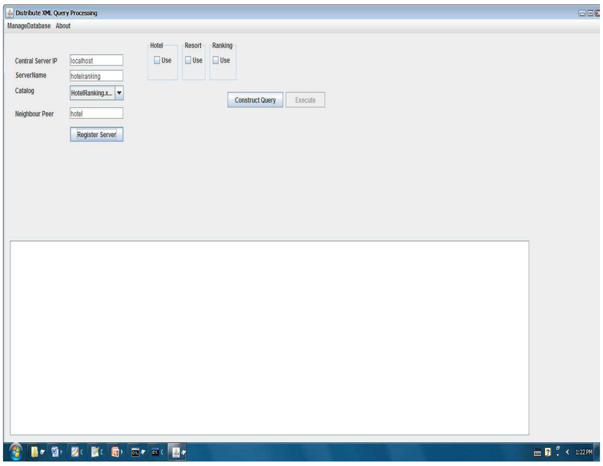


Figure 3. Query Plan Tree

5.3 Local XQuery Processor

The local XQuery processor maintains a catalog. Catalog is a set of XML document which are stored in the XML database of the current site. This component is responsible for execution of the local query on the site, and storing the partial result of the local query.

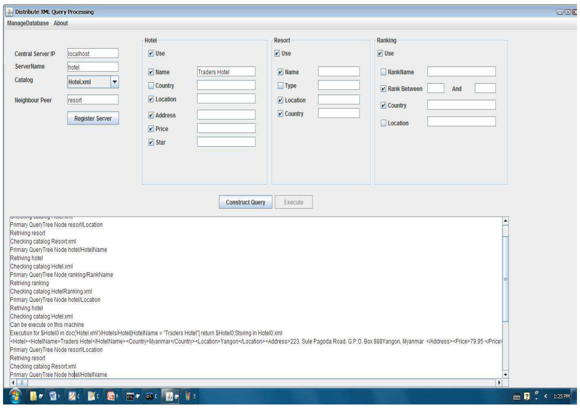


Central Server IP: localhost
ServerName: hotelranking
Catalog: HotelRanking.x...
Neighbour Peer: hotel

Hotel: □ Use
Resort: □ Use
Ranking: □ Use

Figure 4. Main User Interface for Each Peer

Figure 4 shows main user interface for each peer. User can receive the desired hotel and resort information by filling the user interface form. Peer can manage its own catalog, e.g, adding or deleting data and must identifies itself with name and must also give the peer name.



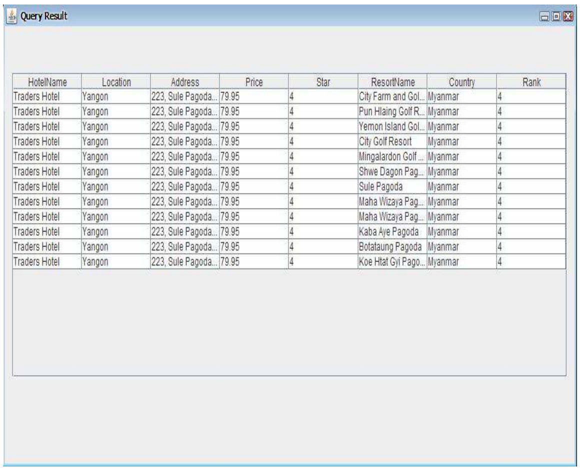
Central Server IP: localhost
ServerName: hotel
Catalog: Hotel.xml
Neighbour Peer: resort

Hotel: ☑ Use; ☑ Name: Traders Hotel; □ Country; ☑ Location; ☑ Address; ☑ Price; ☑ Star
Resort: ☑ Use; ☑ Name; □ Type; ☑ Location; ☑ Country
Ranking: ☑ Use; □ RankName; ☑ Rank Between … And …; ☑ Country; □ Location

```
Primary QueryTree Node resort/Location
Retriving resort
Checking catalog Resort.xml
Primary QueryTree Node hotel/HotelName
Retriving hotel
Checking catalog Hotel.xml
Primary QueryTree Node ranking/RankName
Retriving ranking
Checking catalog HotelRanking.xml
Primary QueryTree Node hotel/Location
Retriving hotel
Checking catalog Hotel.xml
Can be execute on this machine
Execution for $Hotel0 in doc('Hotel.xml')/Hotels/Hotel[HotelName = "Traders Hotel"] return $Hotel0;Storing in Hotel0.xml
<Hotel><HotelName>Traders Hotel</HotelName><Country>Myanmar</Country><Location>Yangon</Location><Address>223, Sule Pagoda Road, G.P.O. Box 888Yangon, Myanmar  </Address><Price>79.95 </Price>
Primary QueryTree Node resort/Location
Retriving resort
Checking catalog Resort.xml
Primary QueryTree Node hotel/HotelName
```

Figure 5. The Intermediate Result Executed by the Peer.

Figure 5 shows the peer processing the query and the intermediate results in the display.



HotelName	Location	Address	Price	Star	ResortName	Country	Rank
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	City Farm and Gol...	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Pun Hlaing Golf R...	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Yemon Island Gol...	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	City Golf Resort	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Mingalardon Golf ...	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Shwe Dagon Pag...	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Sule Pagoda	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Maha Wizaya Pag...	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Maha Wizaya Pag...	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Kaba Aye Pagoda	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Botataung Pagoda	Myanmar	4
Traders Hotel	Yangon	223, Sule Pagoda...	79.95	4	Koe Htat Gyi Pago...	Myanmar	4

Figure 6. Final Query Result

Figure 6 shows the final query result after the query is fully evaluated.

6. Conclusion

This paper presents design and implementation of distributed XQuery processing system for a Peer-to-Peer architecture. The system can execute query dynamically across peers. This system uses both query shipping and data shipping approach; it is more efficient than method that only uses data shipping approach. This system does not forward every data resulted from the execution of the local query on the peer. It forwards a query result to the desired peer if it is needed by that peers. So, traffic for unnecessary routing of all partial results is avoided. The limitation of this system is that there is no user access control for the system. So, anyone can add or remove data record in the system. This system uses Java programming language and Qizx for XML database.

7. References

[1] Donald Kossmann, “The State of Art in Distributed Query Processing”, ACM Computing Surveys, Vol.32, No.4, December2000, pp.422–469.

[2] Guilherme Figueiredo, Vanessa Braganholo, Marta Mattoso, “A Methodology for Query Processing over Distributed XML Database”.

[3] George Kokkinidis, Lefteris Sidirourgos, Theodore Dalamagas, and Vassilis Christophides, “Semantic Query Routing and Processing in P2P Digital Libraries”, Hellenic Data Management Symposium (HDMS'08), July 2008.

[4] S.Abiteboul, I.Manolescu S.Zoupanos, “OptimAX: optimizing distributed continuous queries”, in NationalConference, Bases de Données Avancées, 2007.

[5] Lu Jiaheng, “Efficient Processing of XML Twing Pattern Matching “, PhD Thesis, National University of Singapore, August 2006.

[6] Reza Akbarinia, Esther Pacitti, Patrick Valduriez ,”Query Processing in P2P System”, Technical Report, INSTITUT NATIONAL DE RECHERCHE EN INFORMATIQUE ET EN AUTOMATIQUE, Janauray 2007.

[7] Ying Zhang, “Towards P2P XML Database Technology”, Conference on Very Large Database (VLDB) 07, September23-28, 2007, Vienna, Austria.

[8] Priscilla Walmsley, *XQuery***,** Oreilly, April 2007.