AN Effective Parallel XML Fuzzy Query Processing

K. Naresh Kumar
Research Scholar
Dept. of CS & SE
Andhra University

N.V.E.S Murthy
Dept. of Mathematics
Andhra University
Visakhapatnam, India

Ch. Satyanand Reddy, Ph.D
Dept. of CS & SE
Andhra University
Visakhapatnam, India

ABSTRACT
Representation and handling of inexactness in information has become the major issues in modern database system and next generation information systems. In order to deal with the information inexactness, fuzzy logic is integrated with various database model and theories. This paper presents a query processing model could coupled with fuzzy logic in XML database system. Our system is based on traditional XML databases, while permitting the storage of fuzzy data as well as crisp data. Crisp data are the usual precise data handled by the traditional databases whereas fuzzy logic gives the output in certain range. In this paper we are dealing with the concept of critical architectural component named fuzzy meta- knowledge base. The main aim of fuzzy meta- knowledge basis to keep the different types of fuzzy divisions for database attributes. Fuzzy meta- knowledge base defines and demonstrates data of fuzzy nature is stored in the fuzzy meta- knowledge base. The fuzzy query language is based on X-PATH. It can accept any type of fuzzy expressions in any condition in query part. For improving the performance of X-PATH, we are using Parallel Path Stack algorithm. Parallel Path Stack algorithm speed XML Query processing performance significantly.

Keywords
XML, Fuzzy logic, X-PATH, fuzzy query, Fuzzy meta-knowledge base.

1. INTRODUCTION
In the last three decades, there has been comprehensive research investigating how imprecise and variable data can be stored in databases given that it is extensive in most real-world scenario [1]. Imprecise data contains subjective views and conclusions in areas such as personnel assessment, policy choices and economic prognosis [2]. An immediate direction of investigation that is instantly suitable to numerous applications is how the traditional relational model can be expanded to include fuzzy data. Another highly researched domain concentrates on how relational data can be described in the Extensible Markup Language (XML), e.g., [3, 4, 5, 6]. Furthermore, Lee et al [7] and Turowski and Weng [8] depict examples of XML representation for fuzzy data modeling.

The Extensible Markup Language (XML) has a standard way to represent the data in a proper manner. With the regular progress in the XML data, the caliber to handle large set of XML data and to finding information from them becomes important for the Web-based knowledge systems [9, 10]. A proper solution is to cluster the resemble XML data based on their reference and structure. The grouping of XML data facilitates a number of standard applications such as developed knowledge reflow, data and schema integration, document classification analysis, structure summary and indexing, and query processing and optimization [11,12].

XML database systems are essential technique to handle strong XML data, permitting specification, storage, and querying of XML. Conventional database systems endorsement crisp data, where the data stored and queries given are accurate. In the current scenario, there become a lot of variable and unclear data. Human being obtains an innate brilliance to naturally analyzed various parameters and process all modes of data. For example, the approximation besmeared in determining the speed of a car get in from an aspect street to the road and its chances to pull up in front of other car are totally fuzzy. The essence to store obscure words motivates us to consolidate fuzzy techniques into database systems.

In this paper, we demonstrate a fuzzy XML database system that is used to manage the fuzzy XML data. The system was made on top of conventional XML systems, permitting the storage of both crisp data and fuzzy knowledge. The system is a generic system such that the perspective can be applied to any XML database system. Together with the incorporation of the fuzzy example in more phase of computer science, a simple and robust way of handling, manipulating and storing fuzzy data is becoming more essential. These facts make fuzzy database management systems (FDBMS) an essential part of difficult soft-computing systems, and encourage research in this area. This leads to various proposals of fuzzy database models and executions [14, 15, 16, 17, 18, 19, 20]. The system permits users to query the given database using fuzzy techniques. The query language is worked on the basis of XPATH mode permitting users to query fuzzy and crisp data over the database. Since XML database users are already familiar with XPATH commands, this characteristics much improves the usability of our query language. In rising to language specification, our system gives an environment helping fuzzy query execution at run time. The system also helps a fuzzy active rule language to handle integrity duress management and business point specification. This paper discussed the two essential characteristics of our fuzzy system; these are data storage and query management [13].

2. LITERATURE SURVEY
JianLiu·Z.M. Ma·XueFeng[21] proposed the incertitude of XML data found now as well as the ductility of demonstration given by XML emerge challenging themes for storing fuzzy XML data in conventional relational systems. In given paper, they have proposed a new mapping technique for collecting fuzzy XML data into a concerned database system based on an edge-based mapping technique. The exclusive characteristics of their technique were that no schema knowledge needed for the data storage. On the proposed basis, they have given a generic approach to interpret path expressions to SQL queries for processing XML queries. There were a number of routs for future research. They were currently working on a prototype giving the possibility of the approach for large practical usages. As future work, they planned to enhance their mapping technique by taking constraints during the data mapping, evolution query interpret techniques to translate complex XML queries such as XQuery queries into given SQL statements, and improving their technique by introducing query optimization methods.

R.D. Rodrigueset.al [22] described and presented the main features of the new fuzzy database Aliança, which incorporated

the main characteristics essential to treat knowledge of a wide range of likelihood. Peculiarly, imprecise knowledge was well handled. The given name showed the fact that the application was the union of fuzzy logic methods, a database relational management application and a fuzzy meta-knowledge database given in XML, in order to support and present obscure knowledge. The supplement gain was the use of XML to show the fuzzy meta-knowledge which made it was easy to maintain and consider the structure of imprecise knowledge. The fuzzy database techno tics Aliança approximates the interaction with databases to the generally way human beings reason.

Ying Jin, Sangeetha Veerappan [23] proposed a XML-based fuzzy application, permitting both crisp and fuzzy XML data. The application supported the three most famous applied distributions, namely trapezoidal, triangle, and interval distributions. The present application used permanent with given layer of fuzzy system, while supplementary functionality of fuzzy data storage and fuzzy queries have given by their system. The given paper proposed the execution of the data representation, fuzzy query syntax, and fuzzy query processing logic. The system was made on top of conventional XML databases, while permitted the storage of fuzzy data as well as crisp data. The system has been implemented in Java. Their future direction was the performance analyzing of the system.

Ting Yanget.al [24] has given expanded edge matching technique for comparing the structural resemblances between XML documents and clustering analysis. The given technique planned few demerits of edit distance based techniques and conventional edge matching based techniques. They have assigned various weights for the nodes in various levels and finding complete matching edges, EEM further detected topological edges and repeated substructures. The development had made structural resemblances comparison more reasonable and accurate. The clustering results have showed the advantage of EEM in comparison with other existing techniques.

Ying Jin, Hemal J Mehta[25] presented their active fuzzy XML database application, concerning on the support of adjunct events. A set of events have incorporated using composition factors to trigger active rules. The system implemented rules upon tapers of events automatically. Events have primitive that rose by a single database operation, such as insert, delete, and replace. Users have determined composite events that grouped by multiple primitive events. The formation of those events triggered one or more rules. In the given paper, they proposed their techniques of handling composite events in a fuzzy XML database system. Fuzzy logic has permitted people to specify obscure knowledge directly. Their research has made a fuzzy active rule-based database system on top of conventional XML databases, while permitted calculation of fuzzy data, as well as implementation of fuzzy active rules. A future direction was the performance evaluation on event processing, and further refinement to optimize the process.

Jian Liu et.al [26] demonstrated a fuzzy XML databases have been intensively analyzed, for instance. The proposed efforts were mainly built in representing fuzzy knowledge in XML databases; less satisfied the needs of managing fuzzy XML queries, particularly in fuzzy twig pattern queries. In order to processed the queries, a novel fuzzy labeling technique to hold the structural information of fuzzy XML databases, as well as some essential fuzzy XML algebraic operations have given in proposed paper. On the basis, they have given Algorithm "FTwig" for supporting fuzzy twig pattern queries.

3. PROPOSED METHODOLOGY

Here we are presenting our methodology based on concept of critical architectural component named fuzzy meta- knowledge base. We are describing fuzzy query language which is based on X-PATH. It can accept any type of fuzzy expressions in any condition in query part. The proposed methodology are describing below:

3.1 Fuzzy Meta-Knowledge base and data storage

In this paper Fuzzy Meta-Knowledge Base is used to store data related to the imprecise representation for data. In our current paper, we are dealing with four types of data types which are describing below:

Linguistic label with possibility distribution

The Linguistic variable used as a values which are in the form of a, b, c, d. We estimated the numerical value using trapezoidal distribution. When a property is related to an obscure value it gets a linguistic label with a possibility distribution. This kind of data is of "type" and it has a related trapezoidal possibility distribution and its description is saved in the fuzzy meta-base knowledge. Mostly trapezes are used to represent obscure values.

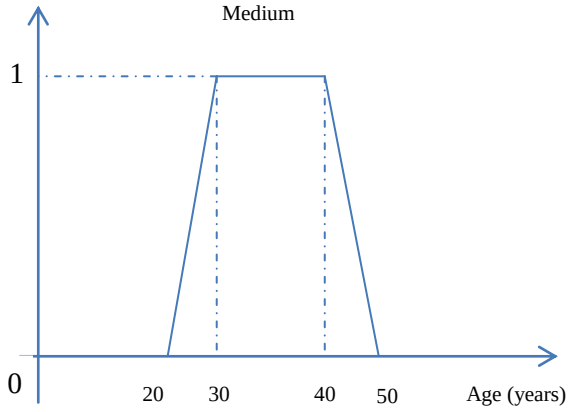

Figure 1: Linguistic label for the concept "Medium"

In the given in Figure1, for linguistic variable "popularity", the linguistic value "medium" is compatible to the trapezoidal delivery with a, b, c, d values of 20, 30, 40, 50 respectively. The values of threshold lie between 0 and 1.

In the FMKB, trapezoidal delivery is shown in the given XML format as labeled by the "type" attribute, as shown below:

<Linguistic Variable name="popularity">

<Type T="trapezoidal">

<Linguistic Term name="MEDIUM">

<alpha>20

<beta>30

<gamma>50</d>

</Linguistic Term>

Linguistic label with possibility interval

The Linguistic variable related to an interval possibility distribution. Figure 2 describes the interval possibility delivery for the price is from 80 to 100. The book store system contains the information related to book sold at every book store between the intervals [70,100].

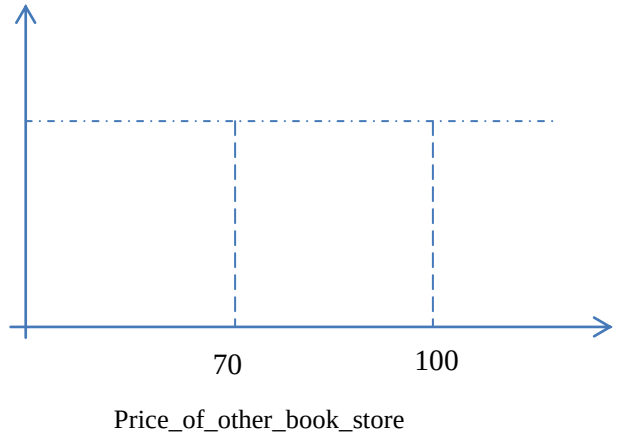

Figure 2: Distribution with possibility Interval

Approximate values

If the value d is in the field, the obscure concept almost d is named by a triangular potential distribution allocated around with a ledge as shown inFig.3.Here we are taking an example, in the bookstore system; we use quantity_in_demand to current future requirement on the book. We can state "quantity_in_demand is almost 90". Triangular delivery receives a median value (d) and a ledge value (m) with the threshold between 0.0 and 1.0. In a given database value with a median value d, the compatible range of value for the given linguistic variable generated by using the ledge and threshold.

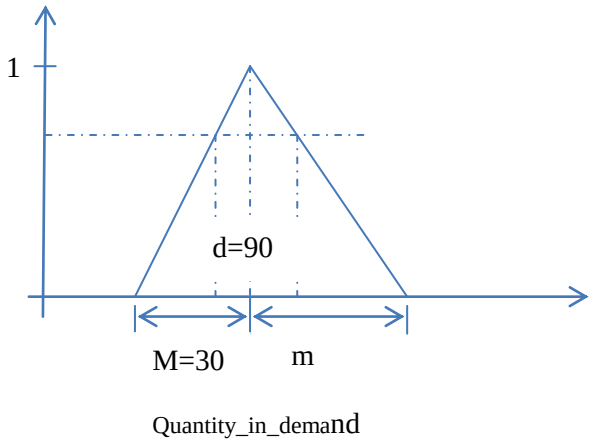

Figure 3: Distribution with Approximate Values

Threshold can also be applied to those two types of distribution.

…
<title> Flowers </title>
…
<Popularity>LOW</popularity>
<threshold_value>0.8</threshold_value>
…

Crisp data

Crisp data are the common exact data supported by the conventional databases. Crisp data receive the same resource as the imprecise data. It does not necessary require more information added to the fuzzy meta-knowledge base. Strings, real and natural numbers, dates are few paradigm of common crisp data formats.

Fuzzy XML XPath query

We have to satisfy users' fuzzy query motive, this paper applies fuzzy predicates (close to, at most, at least) to fill XPath query language. In our paper the concept of fuzzy is used to cluster the input data. The fuzzy cluster the data on the basis of properties. It does cluster on the basis of different properties of the data. One of the main characteristics of XML data is the disparity between content and structure. Here, we have taken the XML fragments:

<Employees>
<Employee emplid="1111" type="admin">
<firstname>John</firstname>
<lastname>Watson</lastname>
<age>30</age>
<email>johnwatson@sh.com</email>
</Employee>
<Employee emplid="2222" type="admin">
<firstname>Sherlock</firstname>
<lastname>Homes</lastname>
<age>32</age>
<email>sherlock@sh.com</email>
</Employee>
<Employee emplid="3333" type="user">
<firstname>Jim</firstname>
<lastname>Moriarty</lastname>
<age>52</age>
<email>jim@sh.com</email>
</Employee>
<Employee emplid="4444" type="user">
<firstname>Mycroft</firstname>
<lastname>Holmes</lastname>
<age>41</age>
<email>mycroft@sh.com</email>
</Employee>
<Employee emplid="4445" type="user">
<firstname>Mycroft1</firstname>
<lastname>Holmes1</lastname>
<age>40</age>
<email>mycroft@sh.com1</email>
</Employee>
<Employee emplid="4446" type="user">
<firstname>Mycroft6</firstname>
<lastname>Holmes6</lastname>
<age>45</age>
<email>mycroft@sh.com6</email>
</Employee>
<Employee emplid="4447" type="user">
<firstname>Mycroft7</firstname>
<lastname>Holmes7</lastname>
<age>34</age>
<email>mycroft@sh.com7</email>
</Employee>
<Employee emplid="4448" type="user">
<firstname>Mycroft8</firstname>
<lastname>Holmes8</lastname>
<age>35</age>
<email>mycroft@sh.com8</email>
</Employee>
<Employee emplid="4449" type="user">
<firstname>Mycroft9</firstname>
<lastname>Holmes9</lastname>
<age>45</age>
<email>mycroft@sh.com9</email>
</Employee>
<Employee emplid="5410" type="user">
<firstname>Mycroft10</firstname>

<lastname>Holmes10</lastname>
<age>46</age>
<email>mycroft@sh.com10</email>
</Employee>
<Employee emplid="4411" type="user">
<firstname>Mycroft11</firstname>
<lastname>Holmes11</lastname>
<age>23</age>
<email>mycroft@sh.com11</email>
</Employee>
<Employee emplid="4412" type="user">
<firstname>Mycroft12</firstname>
<lastname>Holmes12</lastname>
<age>22</age>
<email>mycroft@sh.com12</email>
</Employee>
</Employees>

These XML part check the common knowledge. In the given paper, we appropriate three fuzzy expanded expressions:

(A) When we will apply fuzzy predicate to a quality value of XML document, the fuzzy query picks nodes in which the quality has a value near to the value represented in the fuzzy query. The syntax is shown below:

Path[{@attribute name FuzzyPredicate compares value}]

(B)When the fuzzy predicate applied to label or to entity name, the query chooses nodes of with a name same to the explained in the fuzzy query, with the under mentioned syntax:

Path [{/tagname() Fuzzypredicate compare value}]

(C)When the fuzzy predicate is pour into an axis of a path expression, the collecting efforts to extract the all elements, attributes or text that are successors of the present node. The following syntax is used like:

Path1 {Fuzzypredicate} // …//path$_n$

In the next step we are performing P-Path Stack: Parallel-Path Stack Algorithm for improving the efficiency of XPath algorithm. Here we are describing the algorithm:

3.2 P-path stack: parallel path stack algorithm:

In this part, we first of all introducing even partition based technique, and then defining the algorithm P-PATHSTACK in the detail.

3.2(a) even partition

We assign region encoding <start, level, and end> to encode XML documents. Using region encoding, the relationship between two nodes, like as parent-child or ancestor-descendant relation. Like as the size of XML document gains, the space cost by region encoding gains similarly.

Suppose that AList or DList denote the ancestor attribute list and the descendant data list and they are in the document order. The dividing in a part of AList and DList into many buckets bucket (k=0,1,…, n_b), where bucket contains both AList and DList. Now we are introducing two Rules to partition DList and AList into different buckets.

RULE 1: Division DList

For k = 0 … (n_b -1)
*bucket$_k$.dstartpos = k * b_s*
if k < b_s – 1;
else
*bucket$_k$.dendpos = (k +1) * b_s -1;*
else
bucket$_k$.dendpos = |DList| -1;
end for

Regulation 1 that means DList is divided into n_b buckets, and each one buckets (except the last one) contains $b_s$descendant elements and the last contains the remaining elements. The variable dstartpos denotes the start position of bucket, while dendpos denotes shows the end position.

RULE 2: DIVISION AList

For k = 0 … (n_b -1)
bucket$_k$.dstartpos = min {p|a_p.end > bucket$_k$.minstart
0 ≤ p < |AList|, $a_p \epsilon$ AList}
bucket$_k$.aendpos = max {p|a_p.start < bucket$_k$.maxend
0 ≤ p < |AList|, $a_p \epsilon$ AList}
end for

bucket$_k$.minstart that means the minimal start value of region encode of descendant data in bucket$_k$, while *bucket$_k$.maxend* means the maximal end value of region encode <start, end, level>.

Rule 2 means that if one or more ancestor elements have descendants in *bucket$_k$* there is a start position and end position in AList. The elements between astartpos and aendpos are contained by bucket$_k$. If aendpos < astartpos, there are no ancestor elements in bucket$_k$, then the result of Path Stack algorithm in bucket$_k$ will be empty.

For an XPath root-to-leaf path a0//a1//a2//a3, we first divide leaf node a3 according to *RULE 1*, and partition a3's parent a2 according to *RULE 2;* then it based on the dividing outputs of a2, division its parent node a1 according to *RULE 2.* Again and again these steps, whenever until the root node a0.

In fact, the divide process can be reversed. We can divide the root-node elements (a0), and then divide its child (a1), repetition of these steps, until the leaf node (a3). Though the details are not completely the same with *RULE 2* when divided from the root to the leaf, some idea is the same.

We assume there aren't nested elements in an XML document element does not contain sub-elements which have contains the same name with the parent.

We can use binary search to get the first ancestor element whose the end value of region encode larger than *minstart* value of this bucket. We can also use binary search to get the last ancestor element whose start value of region encode is smaller than An Efficient Parallel Path Stack Algorithm for Processing XML Twig Queries *maxend* value of this bucket. Using binary search, the time in partition period is little, less than 5% of total elapsed time.

3.2(b) Work balances

The purpose of even division is to measure XPath in parallel. Thus we should make each bucket that contains the same number of elements as much as possible. It cannot get high speed up ratio of parallel algorithm for unbalanced division.

In *RULE 1* in section 3.2(a) makes the number of the leaf-node element in each bucket almost the same. The two rules cannot ensure the all number of elements from root to leaf in each bucket is the same, but for those XML documents with elements dispensed evenly, the total number in each bucket is close to each other. Because the number of leaf-node elements in each bucket is almost the same, and according to division rules, the number of its parent node elements assigned in each bucket will be close if the XML document with elements distributed evenly. We should also assume the balance between threads in form of parallel. So we should consider the number of threads used to determine the value of n_b (no of buckets) and b_s (the no of leaf-node elements in each bucket). We determine the values of *nband and bsas* mention in down:

// thread_number: number of threads used
*n_b = thread_number * 16;*
b_s = |DList|/ n_b;
/ sizehigh: the upper limt of number of leaf – node elements in each bucket */*
While b_s > sizehigh
*n_b * =2;*
b_s = |DList|/ n_b;
end while
/ sizelow: the lower of number of leaf – node elements in each bucket * /*
While b_s < sizelow
If (n_b == thread_number)
break;
else
n_b / = 2;
b_s = |DList|/ n_b;
end while

In the above pseudo codes, we divided DList into n_b buckets, and n_b is 16 times of number of threads used, then the same number of buckets there will be assigned to each thread. If b_s is too large, it will be make against some work balance, as the total number of elements contained in each bucket will many more; if b_s is too small, it will be make against the exertion of parallel characteristic, as parallel scheduling needs extra cost and partitioning data costs more time. In same, we can get the pseudo codes of division from the root node to the leaf node.

3.2(c) Element skips:

We have described the skipping function of even division below. The priority of skipping ancestor or descendant elements is much important in our parallel algorithm.

We can see that when doing division AList according to RULE 2 , we need to find the first ancestor element es whose end value of the region encode is larger than the minstartvalue of this bucket and the last one *el* whose start value of region encode is smaller than maxendvalue. In the different words, elements located before es or after el will be left.

The mentioned method consists of two steps. At first, we perform conversion from leaf node to root node. We divide DList into one bucket, and define which elements of its parent node belong to this bucket according to RULE 2. Repeat these steps, until the root node. At Second we perform division from root node to leaf node. We put the remaining elements of the root node into one bucket, and determine which elements of the child node belong to this bucket. After this procedure, many useless elements will be skipped. This approach can skip many useless elements at the head and at the end of the element list. As an example, the first step will leave the element a1, and the second step will skip the elements d7 and d8. The remaining elements are {a2, a3, a4} and {d1, d2, d3, d4, d5, d6}.

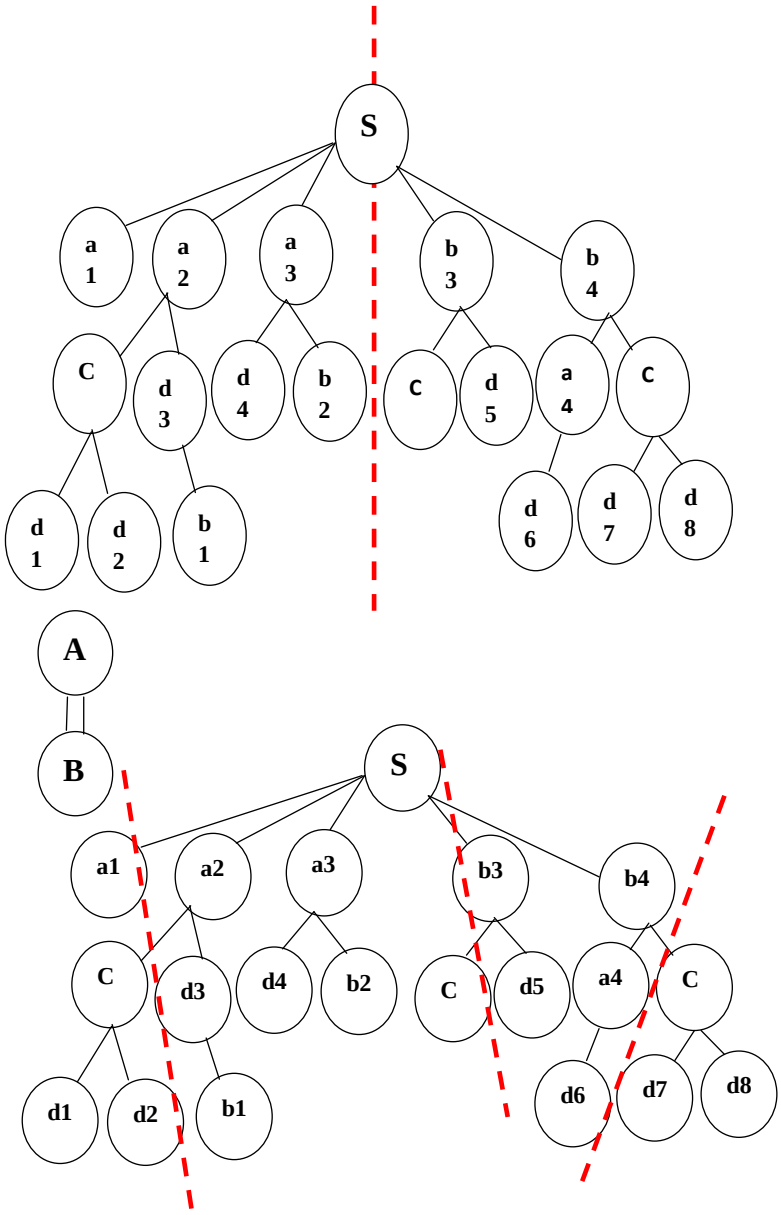

Figure 4: An XML tree (a) and (b) and their corresponding partition results

3.2(d) P-PATHSTACK: PARALLEL PATH STACK ALGORITHM:

In this section, we describe the Parallel PathStack algorithm, which is the key part of this paper. We propose Algorithm P-PathStack to compute answers of a query twig pattern in the algorithm. We first divide the data into various different buckets, and then for the data in each bucket.

Algorithm P-PathStack (q)

a. Define the value of n_{band} and b_{sas} expressed in 3.2 (b).

b. Division each root – to – leaf path into multiple buckets as in 3.2 (a).

c. For each root – to – leaf path, skip elements at the end of element list, as in section 3.2 (c).

d. call PathStack algorithm for all buckets in parallel

using OpenMP.

e. MergeAllPathSolutions().

We can see that the key idea of the parallel algorithm is data division and parallel execution. Line (a.) to Line (c.) partition data into several buckets, and the complexity is

$O(n_b(log|a_1List| + log|a_2List| + \ldots + log|a_nList|))$.

$|a_iList|$ denotes the number of elements with the name $a_i\ (i = 1, 2, \ldots, n)$ and a_i is a node contained by the root-to-leaf path. The complexity is much cheaper than the PathStack algorithm's linear complexity.

In line d we called the standard PathStack algorithm to figure out XPath result in each bucket. Notice that, we put all the buckets of the all paths into thread pool to effect in parallel. As a result, not only execute in parallel between different paths, also between buckets which belong to the in same path. We make bucket as the parallel unit, because we can reduce the granularity of parallel and make work load between threads more balanced, then we can add the efficiency of parallel algorithm. We use OpenMP to implement parallel execution. OpenMP uses thread pool technology; we can set the number of threads used in program, and OpenMP will assign buckets evenly to the threads and give to the threads to different CPU cores on multi-core systems.

For example, consider the XPath a [//b][//d]. There are two root-to-leaf paths, a//d and a//b. For path a//d, after Line 1 in Fig. 2 we get {a2, a3, a4; d1, d2, d3, d4, d5, d6}. Suppose we partition them into two buckets, bucket1 {a2; d1, d2, d3} and bucket2 {a3, a4; d4, d5, d6}. For path a//b, after Line 1 in Fig. 3 we get {a2, a3; b1, b2}; we partition them into two buckets, bucket3 {a2; b1} and bucket4 {a3; b2}. Now we get all four buckets and we use OpenMP to evaluate XPath result in the four buckets in parallel. The results of the four buckets respectively are {a2, b2, a2}, {a3}, {a2} and {a3}. Combine the result which belong to the same root to-leaf path, we get elements {a2, a2, a2, a3} and {a2, a3}, then merge them, and finally we get the final result {a2, a3}.

4. RESULTS AND DISSCUSSION

The proposed XPath query language using multi-core XML query processing system has been implemented in the working platform of JAVA (Net Bean 6.8). In our proposed methodology we calculated descendent node from XML tree. All the nodes divided into different buckets. We assigned equal number of bucket to every thread. Each thread executed separately, query the input and return output. The gradual results attained by the proposed methodology are given below:

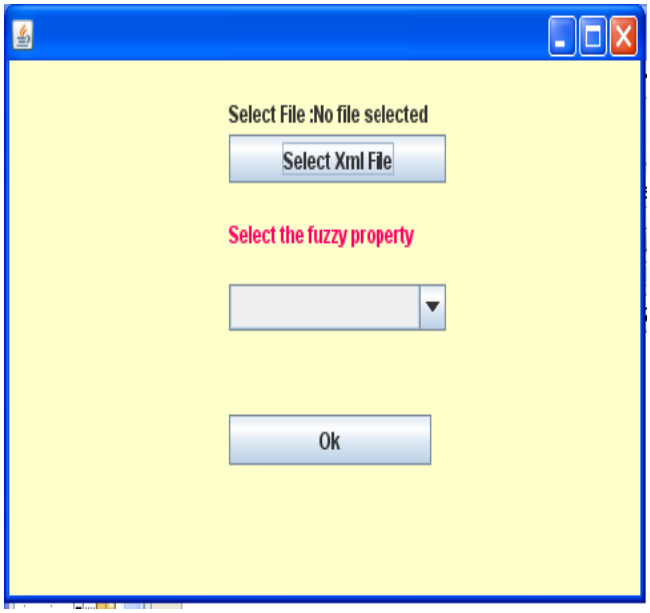

Figure 5: A Fuzzy input window clustering the input on the basis of property

Above figure shows the fuzzy input window which takes different input and clusters them to provide optimized output. This window takes the property of different application and provides optimized output after the process.

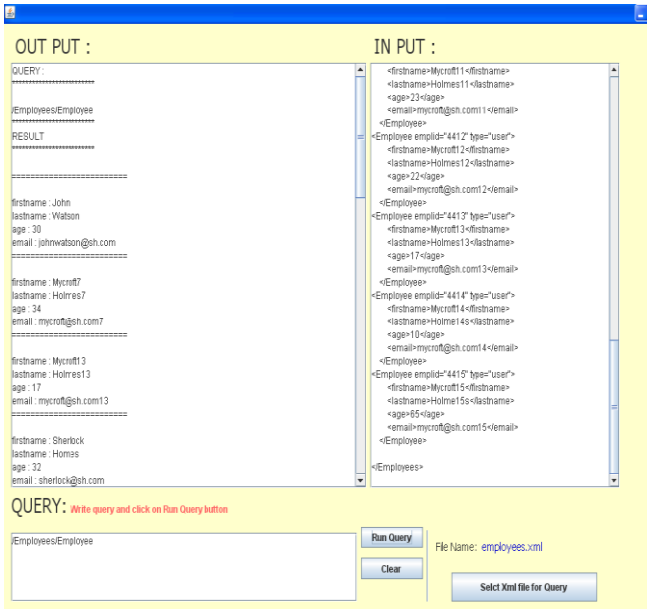

Figure 6: All employees' details from XML file

Fig. 6 showing the details of all the employees saved in XMl file. The above Fig. 6 has an employees.xml file in which all the operation performed. It searches all the information in the parallel way.

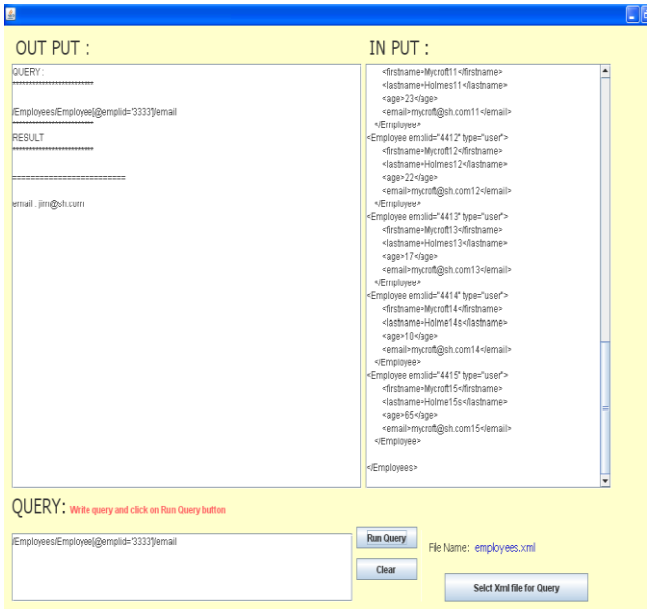

Figure 7: Result for particular employee e-mail id from given input file employees.xml

In the above Fig. 7, we wrote a query to obtain e-mail id for individual employee in the given condition.

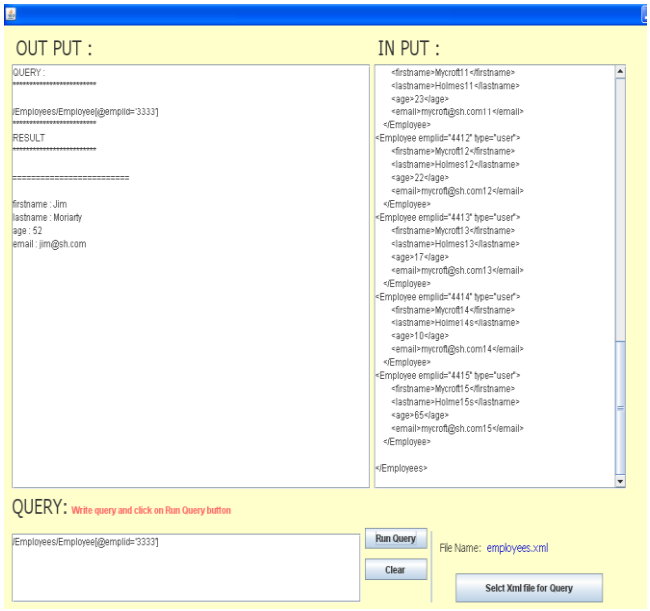

Figure 8: Result for particular employee details from input file employee.xml

In the above Fig. 8, it is showing the result generated from input file employee.xml. We extracted all the details of individual employee from employee.xml.

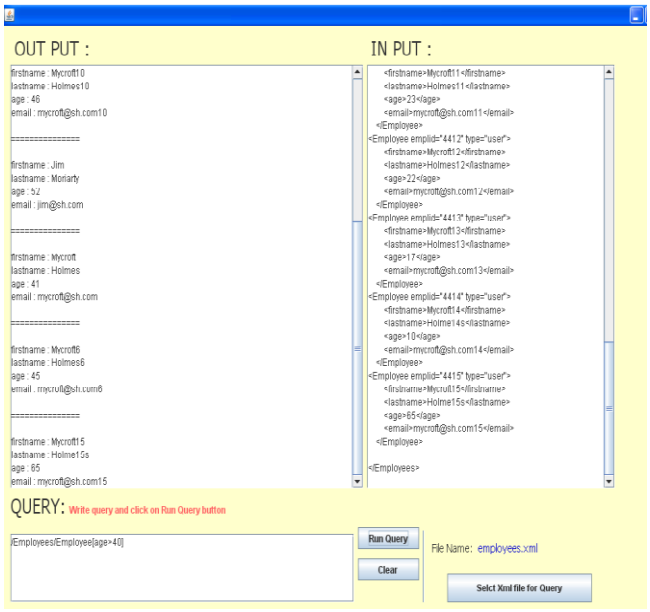

Figure 9: Showing the result of employees (age>40)

Above figure showing the result for employees whose age are greater than 40. All the result runs on parallel fashion. We calculated descendent node from XML tree. All the nodes divided into different buckets. We assigned equal number of bucket to every thread. Each thread executed separately, query the input and return output.

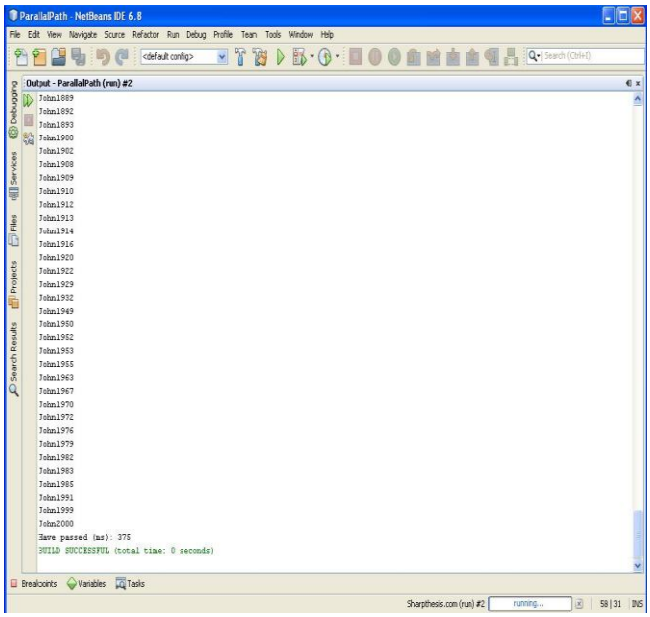

Figure 10: Showing the XPath output and its execution time

The above figure shows the result of XPath query and its execution time in completing the process. The XPath process on the fuzzy input to cluster the data and output of this is applied to the P-path query.

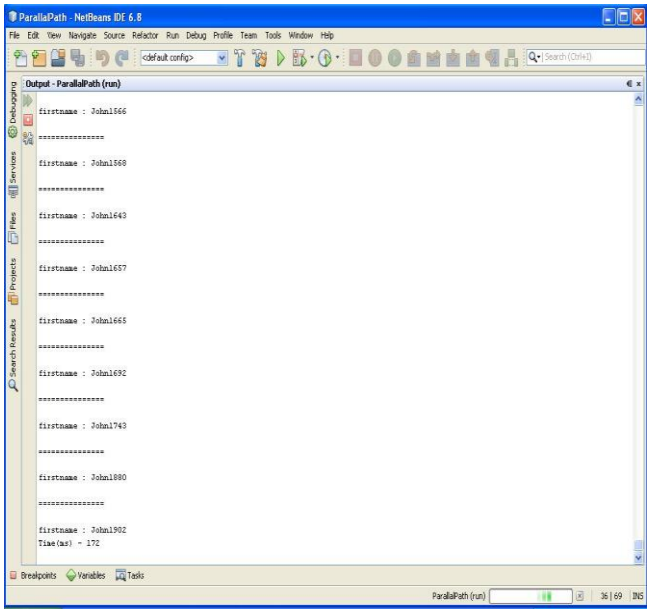

Figure 11: Showing the result of P-path and its execution time

The above figure shows the result of P-path query and its execution time in completing the process. P-path takes the input from the XPath query and processes it to give the output.

5. CONCLUSION

In this paper we described a XML-based fuzzy system which is used for both crisp and fuzzy XML data. A prior application can be used permanent with this layer of fuzzy system, while extra functionality of fuzzy data storage and fuzzy queries are given by our system. Here we have used XPath query for processing the query. XPath can accept any type of fuzzy statement in any situation in the given query part. Performance is the major issues in current time. For improving the performance of X-PATH, we have used Parallel Path Stack algorithm. Parallel Path Stack algorithm propagates the speed of XML Query processing performance significantly. The results shows that the process running parallel in the given system. Thus, algorithm increases the performance of the XPath Query by lowering the execution time of the system.

6. REFERENCES

[1] Dey D. and Sumit S., "A Probabilistic Relational Model and Algebra," ACM TODS, Vol.21, pp.339-369, Sep.1996.

[2] Buckles B.P. and Petry F.E., "A Fuzzy Representation of Data for Relational Databases," Fuzzy Sets and Systems, 7, pp.213 -226, 1982.

[3] Fernandez M., Tan W. -C., and Suciu D., "Silk Route: Trading between Relations and XML". Proceedings of WWW, Amsterdam , May 2000.

[4] Fong J., Pang F. and Bloor C., "Converting Relational Database into XML Document", Proceedings of the International Workshop on Electronic Business Hubs , September, pp.61 -65, 2001.

[5] Lee, D., Mani, M., Chi u, F., and Chu W. W., "Schema Conversion Methods between XML and Relational Models," Knowledge Transformation for the Semantic Web, 2003.

[6] Lee, D., Mani, M., Chiu, F., and Chu W. W., "NeT&CoT: translating relational schemas to XML schemas using semantic constraints," Proceedings of ACM CIKM, 2002.

[7] Lee, J., Fanjiang, Y., Kuo, J., and Lin, Y., "Modeling Imprecise Requirements with XML," Fuzzy Systems, 2, pp.861-866, May 2002.

[8] TurowskiK. andWeng U., "Representing and processing fuzzy information –an XML -based approach ,"Knowledge-Based Systems , Vol.15, pp.67 -75, 2002.

[9] G. Koloniari, E. Pitoura, Peer-to-peer management of XML data, issues and research challenges, SIGMOD Record 34 (2), 2005.

[10] R. Nayak, M. Zaki (Eds.), Knowledge discovery from XML documents, PAKDD 2006 workshop proceedings, Lecture Notes in Computer Science, vol. 3915, Springer-Verlag, Heidelberg, 2006.

[11] A. Boukottaya, C. Vanoirbeek, Schema matching for transforming structured documents, The 2005 ACM Symposium on Document engineering, Bristol, United Kingdom, 2005.

[12] R. Nayak, R. Witt, A. Tonev, Data mining and XML documents, The 2002 International Workshop on the Web and Database (WebDB 2002), June 24–27, 2002.

[13] Oracle Berkeley DB XML, http://www.oracle.com/database/berkeley-db/xml/index.html.

[14] J.M. Medina, J. Galindo, F. Berzal, J.M. Serrano, \Using Object Relational features to build a Fuzzy Database Server", VIII Intl. Conf. Of information processing and management of uncertainty in knowledge-based systems (IPMU 2002), pp 307-314. July 1-5, 2002.

[15] J.C. Cubero, N. Mar¶³n, J.M. Medina, O.Pons, M.A. Vila, \Fuzzy object Management in an Object-Relational Framework", X Intl.Conf. of information processing and management of uncertainty in knowledge-based systems, pp.1767-1774. July 4-9 2004.

[16] H. Prade, C. Testemale, \Generalizing Database Relational Algebra for the Treatmentof Incomplete or Uncertain Information andVague Queries", Information Sciences Vol.34, 1984

[17] M. Zemankova-Leech, A. Kandel, "Fuzzy Relational Databases a Key to Expert Systems", KÄ oln, Germany, TÄUV Rheinland, 1984.

[18] S. Fukami, M. Umano, M. Muzimoto, H. Tanaka, "Fuzzy Database Retrieval and Manipulation Language", IEICE Technical Reports, Vol. 78, N. 233, pp. 65{72, AL-78-85 (Automata and Language) 1979.

[19] M. Umano, \Freedom-O: A Fuzzy Database System", Fuzzy Information and Decision Processes. Gupta-Sanchez edit. NorthHoland Pub. Comp. 1982.

[20] J. Galindo, J.M. Medina, O.Pons, J.C. Cubero, "A Server for Fuzzy SQL Queries", Flexible Query Answering Systems, eds.T. Andreasen, H. Christiansen and H.L.Larsen, Lecture Notes in Arti¯cial Intelligence (LNAI) 1495, pp. 164{174. Ed. Springer, 1998.

[21] JianLiu·Z.M. Ma·XueFeng, "Storing and querying fuzzy XML data in relational databases", Springer Science+Business Media New York, 2013.

[22] R.D. Rodrigues, A.J.O. Cruz, R.T. Cavalcante, "Aliança: A proposal for a fuzzy database architecture incorporating XML", Elsevier B.V., 2008.

[23] Ying Jin, SangeethaVeerappan,"A Fuzzy XML Database System: Data Storage and Query Processing", IEEE, 2010.

[24] Ting Yang, Jinmao Wei, Baoquan Fan, Xu Wang, Haiwei Zhang, "Structural Similarity Computation Based On Extended Edge Matching Method", International Conference on Fuzzy Systems and Knowledge Discovery, 2012.

[25] Ying Jin, Hemal J Mehta, "Composite Event Processing in an Active Rule-Based Fuzzy XML Database System",IEEE IRI, 2011.

[26] Jian Liu, Z. M. Ma, Li Yan, "FTwig: Efficient Algorithm for Processing Fuzzy XML Twig Pattern Matching", IEEE, 2010.