I. INTRODUCTION

XML has become the de facto standard for data representation and exchange over the Internet. Nevertheless, along with the increasing size of XML documents and complexities to evaluate XML queries, existing query processing performance in a single-centralized environment will deteriorate. Parallelism is, thus, a viable solution. One possible approach of exploiting parallelism is to adopt a PC cluster system, in which tens of commodity PCs are interconnected with a high-speed network, due to its recent popularization and commercialization.

The core operation of XML query processing is to find query patterns in XML data. One of the techniques to find query patterns is the holistic twig joins algorithms, which are important family algorithms to enable us to process queries consisting branches holistically. To the best of our knowledge, parallel query processing based on holistic twig joins has not been studied very intensively.

At this stage of the project we focus on finding methods of partitioning XML data and distributing them to cluster PCs. The first method, Grid Metadata for XML (GMX), aims at partitioning XML data obtained from heterogeneous XML documents for inter query parallelism. The second method, streams-based partitioning, aims at partitioning XML data on the fly for intra query parallelism.

II. GRID METADATA FOR XML (GMX)

A. Parallel Architecture

A shared-nothing PC cluster is utilized, one node is selected as the coordinator and others are processing nodes. All cluster PCs hold the entire XML metadata. Each processing node maintains its own respective XML data as the results of the partitioning method. The coordinator is responsible for accepting queries from users, delivering them to appropriate processing nodes, merging the results, and dispatching the results to users. Meanwhile, processing nodes are responsible for computing the holistic twig joins in parallel without communication among cluster nodes and delivering the results to the coordinator. In this case, the partitioning method must ensure that partitioned streams for a specific processing node do not contribute data dependency to other processing nodes.

B. Conceptual Model

Grid Metadata for XML (GMX) [1] is a model for XML metadata that is maintained in a two-dimensional structure for describing a relationship between XML documents represented as streams of XML nodes and twig queries. The main objective of this model is to provide abstraction for describing streams and partitioning them, by taking into account the coherency between query twigs and XML documents.

GMX has a document dimension and a root-to-leaf path (query path) dimension. The relationship between the two dimensions determines a cost of processing a query path for an XML document. Since a query path is a part of a query, some query path dimension values belonged to the same query can be aggregated for a query dimension. Thus, the relationship between the query dimension and the document dimension determines a query processing cost for an XML document.

C. Cost Model

A cost is the workload measurement of a partition. We introduce a cost model that considers retrieval costs of disk I/O access, computation costs, and communication costs. The cost model is expressed as follows:

\[ cf(d, q) = \frac{1}{R_{I/O}} + \frac{1 + \beta + \gamma}{R_{comp}} + \frac{\gamma}{R_{comm}} \sum_{t \in q \text{ nodes}} |S_t| \]

where \( \beta \) is a fraction of the InputSize to estimate root-to-leaf path solutions, \( \gamma \) is a fraction of the InputSize to estimate the final solutions, and \( |S_t| \) is a stream size of node \( t \). (ii) A query is decomposed to its root-to-leaf paths and each path is processed by one or more processing nodes. The root-to-leaf path query cost is expressed as follows:

\[ cf(d, p) = \alpha \left( \frac{1}{R_{I/O}} + \frac{1 + 2\beta}{R_{comp}} + \frac{\gamma}{R_{comm}} \right) \sum_{t \in p \text{ nodes}} |S_t| \]

D. Partitioning Method

The main objective of the partitioning method is to partition streams of XML nodes by means of GMX and to provide ways of refining partitions for balancing workloads in the distribution.

There are three steps of the partitioning method:
1) Clustering XML Documents: Heterogeneous XML documents are grouped according to their similar elements. A hierarchical clustering technique is used to give users more flexible analysis to decide the best clustering results. Values of the feature vector are normalized and proximities are measured by Euclidian distance. Thus, the clustering results reflect vertical partitions on the GMX.

2) Clustering Queries: Each cluster resulted from clustering XML documents is further grouped according to similar query elements by using the same clustering technique. In this step, values of the feature vector are not normalized due to 1 or 0 values, and thus, Manhattan distance is more appropriate to measure the proximities. The clustering results reflect horizontal partitions on the GMX. So far, the GMX is partitioned in vertical-horizontal wise to produce fine partitions.

3) Partition Refinement: Before conducting this step, the results of step 1 and 2 are distributed to cluster PCs by using Round Robin allocation. If the result of distribution leads to load imbalance, the highest workload of a partition is further refined and redistribute to the lowest workload of a cluster PC. The computation is repeatedly performed until workloads of all cluster PCs are about equal balance. There are three cases of partition refinement: (i) if a partition is composed from one or more queries associated with many XML documents, the partition is split on document wise, (ii) if a partition is composed from many queries associated with a single XML document, the partition is split on query wise, and (iii) if a partition is composed from a query associated with a single XML document, the partition is split according to root-to-leaf paths of the query.

III. STREAMS-BASED PARTITION

A. Parallel Architecture

We used the same idea of coordinator and processing nodes in the parallel architecture. However, in this case the entire XML data is kept by the coordinator. Whenever a query request occurs, the coordinator computes the partition and distributes them to processing nodes on the fly. Each processing node computes the holistic twig joins independently in parallel and delivers the results to the coordinator. The streams-based partition must guarantee that partitioned streams for a specific processing node do not contribute data dependency to other processing nodes.

B. Basic Notions

Streams-based partition [2] exploits positional properties of XML nodes. Streams are the underlying data structure of Holistic Twig Joins. They consist of a sequence of XML nodes having the same name and represented in 3-tuple structure $(DocId, LeftPos:RightPos, Level)$. Briefly the positional properties are defined as follows:


**Left Containment** An occurrence of an ancestor node $a$ and a descendant node $d$ satisfies: $a.DocId = d.DocId$ and $a.LeftPos < d.LeftPos$.


C. Partition Plan

Given a query, each query node is associated with a stream. Streams are partitioned according to the relationships between query nodes. The largest stream is selected as the initial stream to be partitioned. The size of a partition is determined by a window size value. Each partition of the selected stream is a substream. Based on the initial partitions, the partition is propagated to the root node of a query; it is called Bottom-Up Propagation. Then, the partition is propagated from the root node to the rest of query nodes; it is called Top-Down Propagation.

In both partitioning propagations, basically we try to find stream nodes that satisfy the containment property in relation with each partition in the base stream. However, finding every node occurrence satisfying containment properties is time consuming. Instead, we use the idea of most left position $mostL$ and most right position $mostR$ of each partition in the base stream. Also, we skip finding the first node lying within the range of $mostL$ and $mostR$ and just find the last node lying within the range of $mostL$ and $mostR$ for each partition in the base stream. Therefore, for each partition we efficiently perform one time search only.

D. Distribution Plan

The distribution plan is to distribute partitions of streams to cluster PCs by using Round Robin allocation. As the time is very crucial for on-the-fly partition, in the case of workload imbalance as the results of distribution we do not make further attempts to rebalance the workloads.

IV. SUMMARY AND FUTURE WORKS

In summary, we have conducted two types of XML data partition for both inter and intra query parallelism and devised parallel algorithms for holistic twig joins. While other partitioning methods partition XML data by considering the XML tree structure carefully, our proposed partitioning methods consider mainly the relationships between queries, XML elements, and streams of nodes.

As for the future works, in order to improve the parallel query system performance, we intend to integrate both partitioning methods and utilize multi-cores CPU optimally.

REFERENCES
