ONYX: Towards an Internet-Scale XML Dissemination Service

Yanlei Diao, Shariq Rizvi, Michael J. Franklin (VLDB’04)

Presented by Mengmeng Liu

Several slides are courtesy of Yanlei Diao

Background: XQuery

- Xpath: i.e., //section/*title and returns a node set
- A query is a FLWOR expression enclosed by a constant tag
- XML filtering and transformation

```xml
<sections>
  { for $s in $doc//section
    where $s/title = "XML"
    and $s/figure/title = "XML processing"
    return <sections>
      { $s//section/title }
      { $s/figure }
    </sections>
  }
</sections>
```

YFilter: High-Performance XML filtering

- A large set of linear path expressions:
  - Q1=/a/b
  - Q2=/a/c
  - Q3=/a/b/c
  - Q4=/a//b/c
  - Q5=/a/*/c
  - Q6=/a//c
  - Q7=/a/*/*/c
  - Q8=/a/b/c

A single NFA sharing all the common prefixes:

Path sharing is the key to efficiency and scalability!

CBDD (Content-Based Data Dissemination)

- Data sources: Continuously publish XML data items.
- The service: Delivers to each user the XML data items that match her data interests; the delivered results are presented in a customized format.

Applications of XML Dissemination

- News feeds via RSS (Really Simple Syndication)
  - My Yahoo!: updated headlines from BBC, CNN, NPR.
- Mobile services
  - Mobile operators: connect content providers with millions of clients running a multitude of operating systems.
- Stock tickers
  - QuoteMedia: fast access to real-time and historical stock data.
- Online auctions
  - Freebidingtools.com: create your own feed for your favorite eBay search.
- Network monitoring
  - Ganglia: a distributed monitoring system for clusters and grids.

Why distributed processing?

- Privacy
  - Regulations: e.g., CA Senate Bill No. 1386.
  - Policies: e.g., customers’ data stay behind the firewall.
- Locality of data interests
  - Disseminate regional data directly to local subscribers.
- Scalability
  - Data volume: number of messages per second up to thousands, message size from 1 KB to 20 KB.
  - Query population: up to millions.
  - Frequency of query updates: from a daily basis to every few minutes.
  - Result volume: can amplify the input data volume by a large factor.
Here comes ONYX…

Design space: Expressiveness & Scalability

**Expressiveness:** data model + query language a service supports

<table>
<thead>
<tr>
<th>Distribution</th>
<th>Yes</th>
<th>No</th>
</tr>
</thead>
<tbody>
<tr>
<td>Data Source</td>
<td>ONX</td>
<td>YFilter</td>
</tr>
<tr>
<td>Broker</td>
<td>M1</td>
<td>squticate</td>
</tr>
<tr>
<td>Expressions</td>
<td>No</td>
<td>Expressions</td>
</tr>
<tr>
<td>Xref Ties</td>
<td>Yes</td>
<td>Xref Ties</td>
</tr>
<tr>
<td>Indexing</td>
<td>No</td>
<td>Indexing</td>
</tr>
</tbody>
</table>

**Benefits:**
- Full structure matching.
- A small maintenance cost for query updates.
- Extensibility for supporting new operators.

Using YFilter as a message broker

Q1:
```
/nslf [head/pubdata[@edition.area="SF"]]
[subject/subject[@subject.type="Stock"]]
[subject/subject[@subject.name="Tide Forecasts"]]
```

Q2:
```
/nslf [head/pubdata[@edition.area="SF"]]
[subject/subject[@subject.name="Tide Forecasts"]]
```

Benefits:
- Full structure matching.
- A small maintenance cost for query updates.
- Extensibility for supporting new operators.

ONYX core techniques

**Processing planes:** Query Plane and Data Plane

<table>
<thead>
<tr>
<th>System Tasks</th>
<th>Query Plane</th>
<th>Data Plane</th>
</tr>
</thead>
<tbody>
<tr>
<td>Content-Driven Routing</td>
<td>Building Routing Tables</td>
<td>Lookup in routing tables</td>
</tr>
<tr>
<td>Incremental transformation</td>
<td>Building transformation plans</td>
<td>Executing transformation plans</td>
</tr>
<tr>
<td>Final query processing</td>
<td>Building query plans</td>
<td>Executing query plans</td>
</tr>
</tbody>
</table>

Routing Table Design

- **Routing query**
- The data interests of queries down from an output link.
- Data interests of a query: XPath expression (equivalent of the for and where clauses of FLOWR expressions).
- Multiple routing queries can be connected by or.
- **Routing table representation**
  - Merge routing queries into a single combined operator network.
  - Map from output links to routing queries.
- Construction algorithm
  - Map(): a user query → a routing query in the canonical form.
  - Collect(): routing queries sent from child brokers → a routing table.
  - Aggregate(): all the routing queries (at a node) → a new routing query.

An example

```
Q1 = ns1[head/pubdata[@edition.area="SF"][subject/subject[@subject.type="Stock"][subject/subject[@subject.name="Tide Forecasts"]]]]
Q2 = ns1[head/pubdata[@edition.area="SF"][subject/subject[@subject.name="Tide Forecasts"]]]
```
**Routing Table Construction**

![Routing Table Diagram]

**Sharing and Short-cut evaluation**

- A problem with sharing
  - Separate routing query representations: short-cut evaluation.
  - Combined one: sharing may sacrifice the short-cut evaluation strategy.
- Solution: dynamic pruning of the operator network at runtime.
  - Each broker A state has a static set of broker ids that it can reach.
  - System keeps a dynamic set of broker ids that have been reached.
  - YF execution is extended to prune the operator network using these sets.

**Other Routing Considerations**

- Filtering Power of Routing
  - Fraction of messages filtered by routing.
    - Selectivity of the union of the user queries at the node.
    - Loss in precision in the routing queries representing this node.
  - If inherently low, partition the query population to improve it.
    - An Exclusiveness Pattern: e.g., "/a/b[@id=?]
    - Identify a set of such patterns, and partition queries using them.
- Routing Efficiency
  - Number of messages routed per second
  - Conflicting goals to filtering power of routing: tradeoff

**Incremental Message Transformation**

- Goal: reduce message size and avoid redundancy
  - Early projection
    - Remove unnecessary data.
  - Early restructuring
    - Transcode messages based on user profiles such as wireless users.
    - Be agreed upon all users downstream of a link.
  - Method: attaching transformation queries to the output links of brokers on the path of routing

**Relations to other techniques**

- Multicast
  - Augmenting content-based routing features is proposed.
  - Doesn't support fine-grained customized user profile.
- Publish/Subscribe system
  - Doesn't necessarily support XML messages.
- XML-based overlay network
  - Such as mesh-based overlay network we discussed.
  - Doesn't talk about XML query processing issues.
  - Doesn't support customized transformations.
Conclusion

- ONYX enables internet-scale XML CBDD.
- YFilter’s NFA-based operator network can do routing.
- Locality of data interests is key to filtering power.

- Fixed brokers and Fault-Tolerance
- Rethinking routing query representation
- Optimizations
  - Incremental message transformation
  - Routing table design