Optimization techniques for XML query processing

Ioana Manolescu-Goujot

Gemo/IASI group
INRIA Saclay–Île-de-France and LRI, Université de Paris Sud-11

October 30, 2009
Research activity since my PhD (2001)

Optimization techniques for XML query processing

- Materialized views for XQuery
- Compressed XML databases

Optimization for ActiveXML

XML data management in structured P2P networks
Research activity since my PhD (2001)

Optimization techniques for XML query processing

- Materialized views for XQuery
- Compressed XML databases

Optimization for ActiveXML

XML data management in structured P2P networks

Also:

Web application modeling and specification WebML
XQuery performance and benchmarking XIME-P, MemBeR
Data-intensive, interactive workflows
Performance evaluation ExpDB, SIGMOD R&W
Outline of this talk

Part I
Materialized views for XQuery

Part II
XML data management on DHTs

Closing
Related works
Context of our work
Perspectives
Part I

Materialized views for XQuery
Materialized views for XQuery: outline

1. Motivation

2. The XAM language
   - XPath \{./, //, *, []\}
   - Nesting
   - Optionality
   - IDs
   - Semantics

3. From XQuery to XAMs

4. XAM query rewriting
Motivation

Historical context

1998 XML standard is out, gaining traction
1998 W3C holds XML QL workshop
2000 Intelligent XML node identifiers for persistent storage
2001 W3C work on XQuery under way...
then come XPath, XSchema, XSL
2001 Shredding-based implementations in RDBMSs. Query translation dependent on shredding
2002 Main-memory tree-based implementations (Galax, Qizx, Saxon)
2003 SQL/XML: built-in type xml, XML import and outport, some XPath
2004 XQuery translated to relational algebra for column-based in-memory store (MonetDB/XQuery), general RDBMSs (PathFinder)

No generic framework for access path selection
1998  XML standard is out, gaining traction
Motivation

Historical context

1998 XML standard is out, gaining traction
1998 W3C holds XML QL workshop
Motivation

Historical context

1998  XML standard is out, gaining traction
1998  W3C holds XML QL workshop
2000  Intelligent XML node identifiers for persistent storage
**Historical context**

1998  XML standard is out, gaining traction
1998  W3C holds XML QL workshop
2000  Intelligent XML node identifiers for persistent storage
2001  W3C work on XQuery under way...
### Historical context

1998  XML standard is out, gaining traction
1998  W3C holds XML QL workshop
2000  Intelligent XML node identifiers for persistent storage
2001  W3C work on XQuery under way... then come XPath, XSchema, XSL
Historical context

1998  XML standard is out, gaining traction
1998  W3C holds XML QL workshop
2000  Intelligent XML node identifiers for persistent storage
2001  W3C work on XQuery under way... then come XPath, XSchema, XSL
2001  Shredding-based implementations in RDBMSs. Query translation dependent on shredding
Historical context

1998  XML standard is out, gaining traction
1998  W3C holds XML QL workshop
2000  Intelligent XML node identifiers for persistent storage
2001  W3C work on XQuery under way... then come XPath, XSchema, XSL
2001  Shredding-based implementations in RDBMSs. Query translation dependent on shredding
2002  Main-memory tree-based implementations (Galax, Qizx, Saxon)
Historical context

1998 XML standard is out, gaining traction
1998 W3C holds XML QL workshop
2000 Intelligent XML node identifiers for persistent storage
2001 W3C work on XQuery under way... then come XPath, XSchema, XSL
2001 Shredding-based implementations in RDBMSs. Query translation dependent on shredding
2002 Main-memory tree-based implementations (Galax, Qizx, Saxon)
2003 SQL/XML: built-in type `xml`, XML import and outport, some XPath
Historical context

1998  XML standard is out, gaining traction
1998  W3C holds XML QL workshop
2000  Intelligent XML node identifiers for persistent storage
2001  W3C work on XQuery under way... then come XPath, XSchema, XSL
2001  Shredding-based implementations in RDBMSs. Query translation dependent on shredding
2002  Main-memory tree-based implementations (Galax, Qizx, Saxon)
2003  SQL/XML: built-in type $xml$, XML import and outport, some XPath
2004  XQuery translated to relational algebra for column-based in-memory store (MonetDB/XQuery), general RDBMSs (PathFinder)
Historical context

1998  XML standard is out, gaining traction
1998  W3C holds XML QL workshop
2000  Intelligent XML node identifiers for persistent storage
2001  W3C work on XQuery under way... then come XPath, XSchema, XSL
2001  Shredding-based implementations in RDBMSs. Query translation dependent on shredding
2002  Main-memory tree-based implementations (Galax, Qizx, Saxon)
2003  SQL/XML: built-in type $xml$, XML import and outport, some XPath
2004  XQuery translated to relational algebra for column-based in-memory store (MonetDB/XQuery), general RDBMSs (PathFinder)

No generic framework for access path selection
Contributions

1. Language for XML materialized views: XAMs [ABM05]
   - Describe storage structures, materialized views, indices
   - Features from XQuery and modern stores
   - Access path selection relies on view-based query rewriting

2. Access path selection for an XQuery subset
   - Algebraic decomposition [ABM+06]
     - XQuery = query XAMs + join/restructuring
   - Algorithms for rewriting XAM queries using XAM views
     - Under Dataguide constraints [ABMP07]
     - In the general case [MZ09]

3. Prototype implementations in ULoad [ABMV05] and ViP2P [MZ09]
Motivation

Access path selection for XQuery

Rewriting

Rewriting

Rewriting

XMLize

Query engine

Cost-based optimizer

Execution plan

XML result

XQuery to XAMs
(Algebraic translation)

XQuery query

Rewriting

Rewriting

Rewriting

Rewriting

Rewriting

Rewriting

Rewriting

Rewriting
XAMs: materialized views for XQuery

Motivation

Rewriting

Query engine

Execution plan

Cost-based optimizer

XML result

Rewrite (q1)

Rewrite (q3)

Rewrite (q2)

Rewrite (q3)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)

Rewrite (q1)
XML Access Modules (XAMs)
XML Access Modules (XAMs)
XML Access Modules (XAMs)

The XAM language

XPath { , , , [ ] }

I. Manolescu (Gemo/IASI)

Optimizations techniques for XML

October 30, 2009 11 / 64
XML Access Modules (XAMs)

The XAM language

XPath { /*, [ ] }
XML Access Modules (XAMs)

for $x$ in /a, $y$ in $x$/b, $z$ in $x$/f
return $y$, $z$
XML Access Modules (XAMs)

```
for $x$ in /a, $y$ in $x$/b, $z$ in $x$//f
return $y$, $z$
```

<table>
<thead>
<tr>
<th>$b_{cont}$</th>
<th>$f_{cont}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨b⟩⟨c⟩⟨d⟩/⟨e⟩some⟨/e⟩⟨/c⟩⟨/b⟩</td>
<td>⟨f/⟩</td>
</tr>
</tbody>
</table>
XML Access Modules (XAMs)

for $x$ in /a, $y$ in $x$/b, $z$ in $x$/f
return $y$, $z$

<table>
<thead>
<tr>
<th>$b_{cont}$</th>
<th>$f_{cont}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨b⟩ ⟨c⟩ ⟨d/⟩ ⟨e⟩ some ⟨/e⟩ ⟨/c⟩ ⟨/b⟩</td>
<td>⟨f/⟩</td>
</tr>
<tr>
<td>⟨b⟩ ⟨b⟩ ⟨d⟩ text ⟨/d⟩ ⟨/b⟩ ⟨d/⟩ ⟨/b⟩</td>
<td>⟨f/⟩</td>
</tr>
</tbody>
</table>
XML Access Modules (XAMs)

The XAM language

XPath \( \{/\, //\, *, [\]] \}\)

for $x$ in /a, $y$ in $x$/b, $z$ in $x$/f

return $y, \, z$

<table>
<thead>
<tr>
<th>$b_{cont}$</th>
<th>$f_{cont}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨b⟩⟨c⟩⟨d/&gt;⟨e⟩some⟨/e⟩⟨/c⟩⟨/b⟩</td>
<td>⟨f/&gt;</td>
</tr>
<tr>
<td>⟨b⟩⟨b⟩⟨d⟩text⟨/d⟩⟨/b⟩⟨d/&gt;⟨/b⟩</td>
<td>⟨f/&gt;</td>
</tr>
</tbody>
</table>
The XAM language

XPath\{//,//*[@]\}

XML Access Modules (XAMs)

for $x$ in /a, $y$ in $x/b$, $z$ in $x/f$
return $y/text()$, $z$

<table>
<thead>
<tr>
<th>$b_{val}$</th>
<th>$f_{cont}$</th>
</tr>
</thead>
<tbody>
<tr>
<td>some</td>
<td>⟨f/⟩</td>
</tr>
<tr>
<td>text</td>
<td>⟨f/⟩</td>
</tr>
</tbody>
</table>
XML Access Modules (XAMs)

XPath $\{/\mathbin{//},*,[]\}$

I. Manolescu (Gemo/IASI)  Optimizations techniques for XML  October 30, 2009  15 / 64
XML Access Modules (XAMs)

The XAM language

Nesting

I. Manolescu (Gemo/IASI)  
Optimizations techniques for XML  
October 30, 2009  
16 / 64
XML Access Modules (XAMs)

The XAM language

Optionality

XML Access Modules (XAMs)
XML Access Modules (XAMs)

The XAM language

IDs

XML Access Modules (XAMs)

I. Manolescu (Gemo/IASI)  Optimizations techniques for XML  October 30, 2009  18 / 64
Formal XAM semantics

Nested relational tables including $\bot$
Two equivalent specifications:

**Natural semantics**
Based on tree embeddings
Useful for reasoning about containment

**Algebraic semantics**
Based on a canonical database and structural (outer-,) (nested ) joins
Useful for pattern extraction from XQuery
Access path selection for XQuery

XQuery to XAMs (Algebraic translation)

Rewriting

Rewriting

Rewriting

Rewriting

XQuery query

Rewriting

Rewriting

Rewriting

Rewriting

XMLize

XML result

Execution engine

Query plan

Cost-based optimizer

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4

v1 v2 v3 v4
From XQuery to XAMs

Algebraic approach based on nested relational model [MP05, ABM⁺06]

1. XQuery dialect: nested for-where-return blocks
2. Syntax-driven translation to a nested relational algebra expression
3. Algebraic operation reordering leads to identifying sub-expressions corresponding to XAMs
Access path selection for XQuery
Let $q$ be a query XAM and $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$ a set of views XAMs. A rewriting of $q$ using $\mathcal{V}$ is an algebraic expression 

$$e(v_1, v_2, \ldots, v_k)$$

such that $e(d) = q(d)$ for any document $d$
Algebraic rewriting & operators

Let $q$ be a query XAM and $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$ a set of views XAMs. A rewriting of $q$ using $\mathcal{V}$ is an algebraic expression

$$e(v_1, v_2, \ldots, v_k)$$

such that $e(d) = q(d)$ for any document $d$.

Algebra operators

- $\text{scan}(v)$
- $\pi_{\text{cols}}(op)$
- $\pi^0(op)$
- $\text{sort}_{\text{cols}}(op)$
- $\sigma_{\text{cond}}(op)$
- $\text{nav}_{i, np}(op)$ evaluates $np$ over the $\text{cont}$ attribute $op.i$
The \textit{nav} operator

\[
\text{nav}_{b, \text{cont}, //c, \text{cont}}
\]

\[
\begin{align*}
\langle b \rangle \langle c \rangle \langle d \rangle \langle e \rangle \text{some} \langle /e \rangle \langle /c \rangle \langle /b \rangle \\
\langle b \rangle \langle b \rangle \langle d \rangle \text{text} \langle /d \rangle \langle /b \rangle \langle d \rangle \langle /b \rangle
\end{align*}
\]
Rewriting example

```
q
  a
    b
      c_id, cont
      e_val
    d[ val=5 ]
  v_1
    a_id
    b_id, cont
  v_2
    c_id, cont
  v_3
    a_id
    d_val
```
Rewriting example

```
q
  a
  b
    c_{id, cont}
      v_1
    d_{val=5}
    e_{val}
      v_2
      v_3
  d_{val}
    a_{id}
  a_{id}
      b_{id, cont}
      c_{id, cont}
```

```
n_{val}^{b.cont,e.val}
  v_1
```
Rewriting example

\[
q
\]

\[
\sigma_{b.id < c.id}
\]

\[
\nabla_{a.id < b.id}
\]

\[
\text{nav}_{b.\text{cont}, e.\text{val}}
\]

\[
v_1 \quad v_2 \quad v_3
\]

\[
\begin{align*}
&d_{val} \\
&c_{id, \text{cont}} \\
&b_{id, \text{cont}}
\end{align*}
\]

\[
\begin{align*}
&a_{id} \\
&c_{id, \text{cont}}
\end{align*}
\]
Rewriting example

\[
q
\]

\[
\begin{array}{c}
\text{a} \\
\text{b} \\
\text{c}_{id,cont} \\
\text{e}_{val} \\
\text{d}_{val} \\
\text{a}_{id} \\
\text{b}_{id,cont} \\
\text{c}_{id,cont} \\
\end{array}
\]

\[
\begin{array}{c}
\text{v}_1 \\
\text{v}_2 \\
\text{v}_3 \\
\end{array}
\]

\[
\begin{array}{c}
\boxdot_{a.id=a.id} \\
\sigma_{b.id \prec c.id} \\
\sigma_{val=5} \\
\boxdot_{a.id \prec b.id} \\
\sigma_{val=5} \\
\sigma_{val=5} \\
\text{nav}_{b.cont, e.val} \\
\end{array}
\]

\[
\begin{array}{c}
\text{v}_3 \\
\text{v}_2 \\
\text{v}_1 \\
\end{array}
\]
Rewriting example
Rewriting algorithms

Based on subset enumeration
Test if a rewriting can be built out of a view subset
Recalls bucket algorithm or [TYÖ+08]
Test minimality at the end
Exponential complexity, polynomial problem
Rewriting algorithms

Based on subset enumeration
Test if a rewriting can be built out of a view subset
Recalls bucket algorithm or [TYÖ⁺08]
Test minimality at the end
Exponential complexity, polynomial problem

Bottom-up algorithms
Use smaller partial rewritings to build bigger ones
- Dynamic Programming Rewriting
- Greedy based on the biggest query coverage
Reuse of earlier information ⇒ more efficient
**Architecture(s)**

**Store**
- Plain relational (no nesting) in Postgres
- Our own built on BerkeleyDB
- Relational + SQL/XML on Oracle 12

**Outside the store**
- XQuery analysis and XAM extraction
- XAM rewriting
  - ULoad: full XAMs, under Dataguide constraints [ABMP07]
  - ViP2P: conjunctive XAMs, no constraints [MZ09]
- Optimizer
- Execution engine
Part II

XML data management on DHTs
Motivation

KadoP: XML indexing on DHTs
- Indexing and query processing
- Scaling up

ViP2P: materialized views on DHTs
- View materialization
- View indexing
Distributed data management

Old goal (1970)
Distributed data management

Old goal (1970)
- distributed versions of industrial-strength DBMSs
- massively parallel with map/reduce
Old goal (1970)
- distributed versions of industrial-strength DBMSs
- massively parallel with map/reduce

Still missing: the flexible federation
- high independence of the sites: when to be in, what to store
- data distribution transparency
- . . . with the usual performance requirements
Motivation: distributed warehouses of Web content

Web content
structured documents, schemas, annotations, concepts, mappings, Web services, inter-document links

Distributed Web content warehouse operations
- publish resources
- connect (annotate, map, link...) existing resources
- update resources
- enhance resources by combining them

In the style of the RNTL WebContent project (2005-2009)
Distributed hash tables
Distributed hash tables

\[ \text{put}(k_1, v_1) \]
Distributed hash tables

\texttt{put}(k_1, v_1) \to (k_1, v_1)
Distributed hash tables

\[ \text{put}\left(k_1, v_1\right) \]
Distributed hash tables

\((k_1, \{v_1, v_2\})\)

\(\text{put}(k_1, v_1)\)

\(\text{put}(k_1, v_2)\)
Distributed hash tables

Motivation

I. Manolescu (Gemo/IASI) Optimizations techniques for XML October 30, 2009 33 / 64
Motivation

Distributed hash tables

\begin{equation}
(k_1, \{v_1, v_2\})
\end{equation}

put\((k_1, v_1)\)

get\((k_1)\)

put\((k_1, v_2)\)
Motivation

Distributed hash tables

\[
\text{put}(k_1, v_1)
\]

\[
\text{put}(k_1, v_2)
\]

\[
(k_1, \{v_1, v_2\})
\]

\[
\text{get}(k_1)
\]

\[
\{v_1, v_2\}
\]
From DHTs to distributed data management

DHTs provide:
- logical network maintenance
- efficient message routing
- shared (key, value) repository
From DHTs to distributed data management

DHTs provide:
- logical network maintenance
- efficient message routing
- shared (key, value) repository

Still need:
- data indexing algorithms
From DHTs to distributed data management

DHTs provide:
- logical network maintenance
- efficient message routing
- shared (key, value) repository

Still need:
- data indexing algorithms
- storage for application data and even DHT index data
From DHTs to distributed data management

DHTs provide:
- logical network maintenance
- efficient message routing
- shared (key, value) repository

Still need:
- data indexing algorithms
- storage for application data and even DHT index data
- local query processing
From DHTs to distributed data management

DHTs provide:
- logical network maintenance
- efficient message routing
- shared (key, value) repository

Still need:
- data indexing algorithms
- storage for application data and even DHT index data
- local query processing
- distributed query processing: operators, including data transfers, optimization
DHT index queries

The part of a user query that can be answered directly by consulting the DHT content index
Typically less precise than the user query

- Find the **IDs of documents matching** the query
- Find the **IDs of documents which may match** the query
DHT index queries

The part of a user query that can be answered directly by consulting the DHT content index
Typically less precise than the user query

- Find the IDs of documents matching the query
- Find the IDs of documents which may match the query

Many trade-offs
Trade-offs in DHT indexing and query processing

Level of detail of the indexing algorithm:
- index query precision $\uparrow \Rightarrow$ execution time $\downarrow$
- data publication time $\uparrow$, possibly execution time $\uparrow$

Data re-placement or clustering:
- fewer peers contacted for a query (message no. $\downarrow$, execution time $?$)
- data transfers in the absence of queries (message no. $\uparrow$, total message size $\uparrow$)
Building XML stores on DHTs

- Peers retain control over the data they store/publish
  - no global schema
  - documents published independently
    - annotations, triples, links can freely connect content
- peers collaborate for storing the index
- load balancing
Building XML stores on DHTs

- Peers retain control over the data they store/publish
  - no global schema
  - documents published independently
  - annotations, triples, links can freely connect content
- peers collaborate for storing the index
- load balancing

Systems

- XML indexing: KadoP [AMP05, AMP⁺08]
Building XML stores on DHTs

- Peers retain control over the data they store/publish
  - no global schema
  - documents published independently
  - annotations, triples, links can freely connect content
- peers collaborate for storing the index
- load balancing

Systems

- XML indexing: KadoP [AMP05, AMP^{+}08]
- XML materialized views in P2P networks: ViP2P [MZ09]
KadoP: DHT-based XML indexing
KadoP: DHT-based XML indexing

- doc1.xml
- ⟨article⟩
- ⟨title⟩XML⟨/title⟩
- ⟨/article⟩

Diagram:

- Nodes labeled p1, p2, p3, p4, p5, p6, p7, p8
- Arrows connecting nodes to represent relationships
- XML structure depicted with ⟨article⟩, ⟨title⟩, and ⟨/article⟩ tags
KadoP: DHT-based XML indexing

<table>
<thead>
<tr>
<th>doc₁.xml</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨article⟩</td>
</tr>
<tr>
<td>⟨title⟩XML⟨/title⟩</td>
</tr>
<tr>
<td>⟨/article⟩</td>
</tr>
</tbody>
</table>
KadoP: DHT-based XML indexing

(doc1.xml)

⟨article⟩
⟨title⟩XML⟨/title⟩
⟨/article⟩

(article,(doc1,1,3))

(title,(doc1,2,2))

('XML',(doc1,3,1))

I. Manolescu (Gemo/IASI)
KadoP: DHT-based XML indexing

doc₂.xml
<article>
  ⟨title⟩Web⟨/title⟩
  ⟨/article⟩

(doc₂,xml)
(article,(doc₁,1,3))
(title,(doc₁,2,2))
(’XML’,(doc₁,3,1))

doc₁.xml
<article>
  ⟨title⟩XML⟨/title⟩
  ⟨/article⟩

(doc₁,xml)
(article,(doc₁,1,3))
(title,(doc₁,2,2))
(’XML’,(doc₁,3,1))
KadoP: DHT-based XML indexing

**doc₂.xml**
- `<article>`
- `<title>`Web`</title>`
- `</article>`

**doc₁.xml**
- `<article>`
- `<title>`XML`</title>`
- `</article>`

Diagram:
- `p₁`
- `p₂`
- `p₃`
- `p₄`
- `p₅`
- `p₆`
- `p₇`
- `p₈`

Connections:
- `(article,(doc₁,1,3))` from `p₃` to `p₂`
- `(article,(doc₂,1,3))` from `p₃` to `p₆`
- `(title,(doc₁,2,2))` from `p₃` to `p₄`
- `(title,(doc₂,2,2))` from `p₃` to `p₄`
- `(title,(doc₁,2,2))` from `p₄` to `p₂`
- `(title,(doc₂,2,2))` from `p₄` to `p₂`
- `(’Web’,(doc₂,3,1))` from `p₇` to `p₆`
- `(’XML’,(doc₁,3,1))` from `p₈` to `p₁`
KadoP: DHT-based XML indexing

<table>
<thead>
<tr>
<th>doc₂.xml</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨article⟩</td>
</tr>
<tr>
<td>⟨title⟩ Web ⟨/title⟩</td>
</tr>
<tr>
<td>⟨/article⟩</td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>doc₁.xml</th>
</tr>
</thead>
<tbody>
<tr>
<td>⟨article⟩</td>
</tr>
<tr>
<td>⟨title⟩ XML ⟨/title⟩</td>
</tr>
<tr>
<td>⟨/article⟩</td>
</tr>
</tbody>
</table>

(article,(doc₁,1,3))
(article,(doc₂,1,3))
(title,(doc₁,2,2))
(title,(doc₂,2,2))
('Web',(doc₂,3,1))
('XML',(doc₁,3,1))
KadoP: DHT-based XML indexing

\[
\begin{align*}
\text{doc}_1.xml &\quad \text{article} \quad \text{title} \quad \text{XML} \quad \text{/article} \\
\text{doc}_2.xml &\quad \text{article} \quad \text{title} \quad \text{Web} \quad \text{/title} \quad \text{/article}
\end{align*}
\]

//article[cont(.,’XML’)]//title

Optimizations techniques for XML
KadoP: DHT-based XML indexing

\[
\text{doc}_2.\text{xml} \\
\langle \text{article} \rangle \\
\langle \text{title} \rangle \text{Web} \langle /\text{title} \rangle \\
\langle /\text{article} \rangle
\]

\[
\text{doc}_1.\text{xml} \\
\langle \text{article} \rangle \\
\langle \text{title} \rangle \text{XML} \langle /\text{title} \rangle \\
\langle /\text{article} \rangle
\]

//article[cont(.,'XML')]/title

I. Manolescu (Gemo/IASI)
KadoP: DHT-based XML indexing

\[
\begin{array}{|c|}
\hline
\text{doc}_2.\text{xml} \\
\langle \text{article} \rangle \\
\langle \text{title} \rangle \text{Web} \langle /\text{title} \rangle \\
\langle /\text{article} \rangle \\
\hline
\end{array}
\]

\[
\langle /\text{article} \rangle [\text{cont}(., \text{'XML'})] \langle /\text{title} \rangle \ 
\begin{array}{|c|}
\hline
\text{doc}_1.\text{xml} \\
\langle \text{article} \rangle \\
\langle \text{title} \rangle \text{XML} \langle /\text{title} \rangle \\
\langle /\text{article} \rangle \\
\hline
\end{array}
\]

\(\text{//article[cont(.,'XML')]/title}\) 
\((\text{doc}_1,2,2)\)

\((\text{article, (doc}_1,1,3))\)
\((\text{article, (doc}_2,1,3))\)
\((\text{title, (doc}_1,2,2))\)
\((\text{title, (doc}_2,2,2))\)

\(('\text{Web}',(\text{doc}_2,3,1))\)
\(('\text{XML}',(\text{doc}_1,3,1))\)
KadoP: DHT-based XML indexing

//article[cont(.,’XML’)]//title (doc₁,2,2)

I. Manolescu (Gemo/IASI) Optimizations techniques for XML October 30, 2009 39 / 64
KadoP: DHT-based XML indexing

I. Manolescu (Gemo/IASI)  
Optimizations techniques for XML  
October 30, 2009  39 / 64
Scaling up KadoP

Engineering issues:
1. DHT values were too large for efficient storage ⇒ new store
2. Blocking get operation ⇒ pipelined get

Scalability: longest posting list in a query
- long posting list = frequent term [LHSH04]
- distributed B-tree organization ⇒ parallelized posting list transfers

Also Bloom filters to reduce transfers
KadoP indexing experiments on Grid5K

![Graph showing total publishing time for different configurations of publishers and peers.]

- □ - 1 publisher, 200 peers
- ○ - 1 publisher, 500 peers
- ● - 1 publisher, 500 peers (with DPP)
- ♦ - 25 publishers, 500 peers
- × - 50 publishers, 500 peers

Total publishing time (minutes)
KadoP querying experiments on Grid5K

The graph shows the total time for query processing with and without Dynamic Path Preprocessing (DPP). The solid line represents the total time with DPP, while the dashed line represents the total time without DPP. As the load increases, the difference in processing time becomes more pronounced.
Lessons learned with KadoP

- Performant message routing (redundant fingers)
Lessons learned with KadoP

- Performant message routing (redundant fingers)
- Simulation $\neq$ deployment
Lessons learned with KadoP

- Performant message routing (redundant *fingers*)
- Simulation ≠ deployment
- (Some) DHTs were not built for intensive, detailed indexing. This somehow improved with time.
Lessons learned with KadoP

- Performant message routing (redundant *fingers*)
- Simulation $\neq$ deployment
- (Some) DHTs were not built for intensive, detailed indexing. This somehow improved with time.
- Indexing takes time (orders of magnitude *wrt* first try)
Lessons learned with KadoP

- Performant message routing (redundant *fingers*)
- Simulation $\neq$ deployment
- (Some) DHTs were not built for intensive, detailed indexing. This somehow improved with time.
- Indexing takes time (orders of magnitude *wrt* first try)
- Parallelism a big plus
5 Motivation

6 KadoP: XML indexing on DHTs
   - Indexing and query processing
   - Scaling up

7 ViP2P: materialized views on DHTs
   - View materialization
   - View indexing
ViP2P: views in peer-to-peer
ViP2P: views in peer-to-peer

The peers may store:
- documents
The peers may store:
- documents
- views
ViP2P: materialized views on DHTs

ViP2P: views in peer-to-peer

When $q$ arrives:

- View definition
- Lookup
- Rewriting
- Execution of physical plan
When $q$ arrives:
- view definition lookup
When $q$ arrives:
- view definition
- lookup
- rewriting
ViP2P: views in peer-to-peer

When $q$ arrives:
- view definition
- lookup
- rewriting
- execution of physical plan
When $d$ arrives:

- Search view definitions for which $v(d) \neq \emptyset$.
- Compute $v(d)$.
- Send results.
When $d$ arrives:

- search view definitions for which $v_i(d) \neq \emptyset$
ViP2P: views in peer-to-peer

When $d$ arrives:
- search view definitions for which $v_i(d) \neq \emptyset$
- compute $v_i(d)$
When $d$ arrives:

- search view definitions for which $v_i(d) \neq \emptyset$
- compute $v_i(d)$
- send results
View materialization experiment

1000 peers, 250 machines, 2000 documents, 500 views (70 views contribute to all the documents)
View indexing and lookup for query rewriting

view to index

query to look up
**View indexing and lookup for query rewriting**

![Diagram showing view indexing and query lookup]

<table>
<thead>
<tr>
<th>LI index keys</th>
<th>LI &amp; RLI lookup keys</th>
<th>RLI index keys</th>
<th>LPI index keys</th>
<th>LPI &amp; RPI lookup keys</th>
<th>RPI index keys</th>
</tr>
</thead>
<tbody>
<tr>
<td>$a, b$</td>
<td>$a, b$</td>
<td>$b, c$</td>
<td>$a/b/c, a/b/d, a/e$</td>
<td>$a/b, a/c$</td>
<td>$a/b$</td>
</tr>
<tr>
<td>$c, d$</td>
<td>$c, d$</td>
<td></td>
<td>$b/c, b/d$</td>
<td>$a/d, a/e$</td>
<td>$a/b/c$</td>
</tr>
<tr>
<td>$e$</td>
<td>$e$</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
View look up performance

We used 1440 views related to but different from query $q$
View look up performance

We used 1440 views related to but different from query \( q \)
Performance of rewriting algorithms

![Graph showing performance of rewriting algorithms]

- DFR Total
- DPR Total
- DFR First
- DPR First

Rewriting time (ms) vs. Number of views
Query execution: sample plan

(9) SimpleProject
@195.83.212.160:7000,7001

(5) STD($1 anc $1)
@195.83.212.160:7000,7001

(0) Scan(4nodesView)
@195.83.212.160:7000,7001

(4) Receive
@195.83.212.160:7000,7001

(1) Scan(2nodesView)
@172.20.1.2:7000,7001
Query execution

(0) Scan(4nodesView)
@195.83.212.160:7000,7001

(1) Scan(2nodesView)
@172.20.1.2:7000,7001

(4) Receive
@195.83.212.160:7000,7001

(5) STD($1 anc $1)
@195.83.212.160:7000,7001

(9) SimpleProject
@195.83.212.160:7000,7001

Execution time (ms)

Total document size (MB)
Part III

Closing
Closing

8. Related works

9. Context of this work

10. Perspectives
Related works

Distributed data management [ÖV99, Kos00]

XPath query rewriting [BOB+04, XO05, CDO08, TYÖ+08]
- XPath: wildcard *, union
- Rewritings: intersection, navigations, joins

DHT-based relational data management [LHSH04, HRVM08, APV07]

DHT-based XML indexing [GWJD03, BC06, SHA05, AMP+08]

DHT-based shared XML caches [LP08]

Layered architecture for Web content warehousing [AAC+08]

RDF querying and reasoning on DHT [KMK08, LIK06]
Main research grants

MDP2P (2003-2006) Massive Data Management in Peer-to-Peer (P. Valduriez, INRIA)


CODEX (2009-2012) Efficiency, Dynamicity and Composition for XML (I. Manolescu)

DataRing (2009-2012) Peer-to-peer Data Mgmt. (P. Valduriez)

WebDam (2009-2013) Web Data Management (S. Abiteboul)
PhD students (co-) advised


Spyros Zoupananos (2006-2009)  Efficient Peer-to-Peer Data Management (with S. Abiteboul, INRIA)
PhD students (co-) advised


Spyros Zoupanos (2006-2009)  Efficient Peer-to-Peer Data Management (with S. Abiteboul, INRIA)

Also visiting PhD students:

Melanie Weiss (2005)  Declarative XML data cleaning
**Ongoing PhDs**


Konstantinos Karanasos (2009-)  Semantic Web and XML Data Management (with F. Goasdoué, IASI, U. Paris Sud-11)

Asterios Katsifodimos (2009-)  Peer-to-peer Optimization
Distributed data management

We have only seen the beginning

Why this time will be better

Unparalleled opportunities

- hardware? clouds
- storage and data placement? key-value stores, cheap disks
- connectivity? P2P infrastructures with reliability
- syntactic interoperability? XML
- semantic interoperability? Semantic Web
- tables and joins on the Web? Google and Search Computing ERC

Is the Web the database?
Distributed data management

We have only seen the beginning

Why this time will be better

Unparalleled opportunities

- hardware? clouds
- storage and data placement? key-value stores, cheap disks
- connectivity? P2P infrastructures with reliability
- syntactic interoperability? XML
- semantic interoperability? Semantic Web
- tables and joins on the Web? Google and Search Computing ERC

Is the Web the database? No, just part of it
Ways to go

Data access transparency for new distributed architectures
- Automatic strategies for establishing which data to store where
- Efficient ways of exploiting any data from anywhere (RDF, XML, annotations...)
- Query rewriting, execution, optimization

High-level models for expressive data management applications
- The right compromise between WfMC and Java programming
- Natural and expressive manipulation paradigms (also liquid queries...)
- *If you build it, they will come*
New INRIA/University team: Leo
Thank you!


Structured materialized views for XML queries.

ULoad: Choosing the right storage for your XML application.

Constructing and querying peer-to-peer warehouses of XML resources.
In *ICDE ’05: Demo Session*, 2005.

XML processing in DHT networks.

Data currency in replicated DHTs.

[BC06] Angela Bonifati and Alfredo Cuzzocrea.
Storing and retrieving XPath fragments in structured P2P networks.
*Data Knowl. Eng.*, 59(2), 2006.

A framework for using materialized XPath views in XML query processing.

XPath rewriting using multiple views: Achieving completeness and efficiency.

Locating data sources in large distributed systems.


[KMK08] Zoi Kaoudi, Iris Miliaraki, and Manolis Koubarakis. RDFS reasoning and query answering on top of DHTs. 


[LHSH04] Boon Thau Loo, Ryan Huebsch, Ion Stoica, and Joseph M. Hellerstein. The case for a hybrid P2P search infrastructure.


<table>
<thead>
<tr>
<th>Reference</th>
<th>Authors</th>
<th>Title</th>
</tr>
</thead>
</table>
Minimal canonical rewritings

Let $q$ be a query XAM and $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$ a set of views XAMs. A rewriting of $q$ using $\mathcal{V}$ is an algebraic expression

$$e(v_1, v_2, \ldots, v_k)$$

such that $e(d) = q(d)$ for any document $d$
Minimal canonical rewritings

Let $q$ be a query XAM and $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$ a set of views XAMs. A rewriting of $q$ using $\mathcal{V}$ is an algebraic expression $e(v_1, v_2, \ldots, v_k)$ such that $e(d) = q(d)$ for any document $d$.

**Minimal rewritings**

Do not use a view if we could do without it.
Minimal canonical rewritings

Let $q$ be a query XAM and $\mathcal{V} = \{v_1, v_2, \ldots, v_k\}$ a set of views XAMs. A rewriting of $q$ using $\mathcal{V}$ is an algebraic expression

$$e(v_1, v_2, \ldots, v_k)$$

such that $e(d) = q(d)$ for any document $d$.

Minimal rewritings

Do not use a view if we could do without it.

Canonical rewritings

Operators are organized in a certain way (avoids equivalent plan explosion).
Minimal canonical rewriting

\[
\begin{align*}
q & \quad a \\
b & \quad d_{\text{val}=5} \\
c_{\text{id},\cont} & \quad e_{\val} \\
v_1 & \quad v_2 & \quad v_3
\end{align*}
\]

\[
\begin{align*}
b_{\text{id},\cont} & \quad a_{\text{id}} \\
c_{\text{id},\cont} & \quad d_{\val}
\end{align*}
\]

\[
\begin{align*}
\pi_{c.\cont,e.\val} & \quad \pi_{a.\id=a.\id} \\
\sigma_{b.\id<c.\id} & \quad \sigma_{\val=5} \\
\times_{a.\id<b.\id} & \quad v_3
\end{align*}
\]

\[
\begin{align*}
\text{nav}_{b.\cont,e.\val} & \quad v_2 \\
v_1 & \quad v_2
\end{align*}
\]

I. Manolescu (Gemo/IASI)  
Optimizations techniques for XML  
October 30, 2009  
64 / 64
Minimal canonical rewriting

\[
q \quad \pi_{c.cont,e.val} \\
\sigma_{a.id=a.id \land b.id < c.id \land a.id < b.id \land d.val = 5} \\
\times \\
\times \\
\times \\
\times
\]

I. Manolescu (Gemo/IASI) Optimizations techniques for XML October 30, 2009 64 / 64