CSC: Supporting Queries on Compressed Cached XML

Paper by Stefan Böttcher and Rita Hartel
University of Paderborn, Germany

Presentation by Sebastian Maneth
( NICTA / UNSW, Australia )
Problem Description

Client sends many queries to a (huge) XML document
wants to obtain XML result subtrees to the queries

How can we **minimize the data exchange** between Client and Server?
Possible Solutions

Given a client query, the Server sends…

a) only answer values ("query shipping")

b) all data values "relevant" to the query

c) all data values needed to → evaluate the query ("read set")
  → display answers (output set)

d) complete (huge) XML document
Possible Solutions

Given a client query, the Server sends...

a) only answer values (“query shipping”)

b) all data values “relevant” to the query

c) all data values needed to → evaluate the query (“read set”)
    → display answers (output set)

d) complete (huge) XML document
Possible Solutions

Given a client query, the Server sends…

a) only answer values ("query shipping")

b) all data values “relevant” to the query

c) all data values needed to → evaluate the query ("read set")
   → display answers (output set)

d) complete (huge) XML document

→ difficult to determine a small read set
→ Approach use same evaluation algorithm for Client and Server;
   Server knows Client’s state & determines which data is
   missing for evaluation.
XML Particulars

An XML document can be divided into

→ tree structure  (induced by begin/end tags)
→ data values   (text and attribute values)

Tree structure is
• smaller than data part
• usually highly compressible  (<1% of size of XML document)
• used repeatedly (e.g. to display answer subtrees)

Approach  always send the full tree structure of the queried XML document to the Client. (in compressed form!)

→ Data Values are shipped on demand
Example

```
<authorList>
  <author>
    <name>J. Ziwago</name>
    <pubs>50</pubs>
    <age>30</age>
  </author>
  <author>
    <name>B. Pitt</name>
    <pubs>41</pubs>
    <age>50</age>
  </author>
  <coauthor id="328">
    <author>
      <name>J. Pizzicato</name>
      <pubs>88</pubs>
      <age>91</age>
    </author>
  </coauthor>
</authorList>
```

CSC: Supporting Queries on Compressed Cached XML / 7
Example

Tree structure (known to Client)

```
Example bibs
  authorList
    author
      pubs age name
        J. Ziwago 50 30
        B. Pitt 41 50
    author
      pubs age name
        J. Pizzicato 88 91
  titleList
```

CSC: Supporting Queries on Compressed Cached XML / 8
Example

Tree structure (known to Client)

Data values (unknown to Client)

CSC: Supporting Queries on Compressed Cached XML / 9
Example

Tree structure (known to Client)

Q1 = //author[@age>50]

Data values (unknown to Client)
Example

Q1 = //author[@age>50]

Potential match nodes of Q1

Data values (unknown to Client)
Example

Data values (unknown to Client)

Q1 = //author[@age>50]

Server sends

50_30_41_50_
88_91_J. Pizzicato

Potential match nodes of Q1

CSC: Supporting Queries on Compressed Cached XML / 12
Example

Q1 = //author[@age>50]

Potential match nodes of Q1

Server sends 50_30_41_50_

88_91_J. Pizzicato

Answer to Q1

Needed by pre-order evaluator.
Q1 = //author[@age>50]

Next query:
Q2 = //author/@pubs

Data values
Now known to Client!

50 30 41 50
J. Ziwago

58 91
J. Pizzicato

41 50
B. Pitt
Q1 = //author[@age>50]

Next query:

Q2 = //author/@pubs

Server sends (empty sequence)

Answers to Q2
Example

Q1 = //author[@age>50]

Next query:

Q2 = //author/@pubs

Server sends (empty sequence)

In fact, for Q2 no data is exchanged with the server.

→ Client realizes that it has all required data values!
Example

Q1 = //author[@age>50]

Q2 = //author/@pubs

Next query:

Q3 = //author/name

Server sends J. Ziwago_B. Pitt
Performance Evaluation

Dataset: XMark [Schmidt at al. 2002]
(sizes 34K to 11M)

Compared three different models.

1) CSC (Compressed Structure Caching) as explained, send complete (compressed) tree structure at begin, then data values on demand

2) Compression only ship query answers, but in compressed form

3) Direct only ship query answers

<table>
<thead>
<tr>
<th>Dataset</th>
<th>XML</th>
<th>tree</th>
<th>compr. tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>XMark</td>
<td>XML</td>
<td>tree</td>
<td>compr. tree</td>
</tr>
<tr>
<td>.0001</td>
<td>34K</td>
<td>8K</td>
<td>3.4K</td>
</tr>
<tr>
<td>.001</td>
<td>116K</td>
<td>37K</td>
<td>8K</td>
</tr>
<tr>
<td>.01</td>
<td>1.1M</td>
<td>374K</td>
<td>48K</td>
</tr>
<tr>
<td>.1</td>
<td>11M</td>
<td>3.6M</td>
<td>364K</td>
</tr>
</tbody>
</table>
Performance Evaluation

XMark 0.0001

<table>
<thead>
<tr>
<th>XMark</th>
<th>XML</th>
<th>tree</th>
<th>compr. tree</th>
</tr>
</thead>
<tbody>
<tr>
<td>.0001</td>
<td>34K</td>
<td>8K</td>
<td>3.4K</td>
</tr>
<tr>
<td>.001</td>
<td>116K</td>
<td>37K</td>
<td>8K</td>
</tr>
<tr>
<td>.01</td>
<td>1.1M</td>
<td>374K</td>
<td>48K</td>
</tr>
<tr>
<td>.1</td>
<td>11M</td>
<td>3.6M</td>
<td>364K</td>
</tr>
</tbody>
</table>

Transferred data volume

send/receive for 10 consecutive queries
Performance Evaluation

Transferred data
Total of send/receive for 10 queries

Scaling document sizes

CSC: Supporting Queries on Compressed Cached XML / 20
Summary

Less transferred data than (compressed) query shipping through

→ only sending **difference** in data values, needed by Client for evaluating the given query.

→ The **difference** is determined by Server through running/simulating the given query

Data values are sent compressed,

   and in the order as needed by evaluator.

→ Performance gain depends on the choice of (a) **query evaluator** and (b) **compressor**
Conclusions

• Combination of Compression and Caching (CSC)

• Less transferred data than (compressed) query shipping

• Modular system (compression engine & query evaluator can be replaced)

• Read Set can be reduced by more clever query evaluator (not strict pre-order)