Plan

Micro-benchmark principles
Choosing XPath/XQuery micro-benchmarks
Micro-benchmark results on:

- CDuce/CQL
- eXist
- Galax
- MonetDB
- QizX
- Saxon

Lessons learned:

- XQuery performance
- Methodology

Conclusion
Micro-benchmarking principles

Study (measure) one feature at a time

• Avoid noise, allow understanding
• Not easy: real life applications are not this way.
• But: feature-based results are better predictors.

Vary one parameter at a time

• Typical «query execution time» depends on
  – 5 hardware parameters
  – 10 software parameters
  – 5 query parameters, 5 data parameters

(General soundness principle) Document thoroughly all parameters.
Micro-benchmarking XQuery

Plenty of language features to test!

XPath navigation:

- 13 axes, combinations thereof
- value and positional predicates
- XPath 2.0: joins
- Aggregation functions

XQuery:

- Result construction, let clauses, nested blocks, functions and function calls… Updates coming…

+ General aspects

- Document loading time, query compilation time, robustness to lexical variations, cache impact, index / view impact etc.
Micro-benchmarking XQuery: our take

XPath features tested:
• Navigation along: child, descendant, attribute axes
• Positional predicates
• Existential branches

XQuery features tested:
• For-return expressions
• Return varying data volumes
• Returned data is retrieved at varying depth in the input
• Some returned trees should be copied, some should not
• Nested for-return expressions
XQuery systems

Freely available XQuery processing systems with a large user community and/or published descriptions

• CDuce/CQL v0.4.0. Implemented in OCaml. No node identity.
• eXist v1.0 beta 1. Implemented in Java. Updates!
• Galax v0.5.0. Implemented in OCaml. No persistence used here.
• MonetDB v0.10. Implemented in C++. Main-memory.
• QizX v1.0. Implemented in Java.
• Saxon v8.6.1. Implemented in Java. Type support commercial.

The list is not exhaustive. Limitation: manpower.

Measures as of April 21st 2006. Some systems have changed 😊

Other targets: BerkeleyDB/XML, Timber…
XML documents (1)

Some tested systems work totally in memory. We settled for moderate-sized documents of 11 MB.

- Big enough to see something
- Small enough to (almost !) avoid memory issues

Document exponential2.xml:
Some tested systems work totally in memory. We settled for moderate-sized documents of 11 MB.

- Big enough to see something
- Small enough to (almost !) avoid memory issues

Document layered.xml:
XML documents (3)

Some tested systems work totally in memory. We settled for moderate-sized documents of 11 MB.

- Big enough to see something
- Small enough to (almost !) avoid memory issues

Document mixed.xml:
XML documents (4)

Why these tree shapes?

Control fanout variation with depth

Two extremes: vs.

Control tag distribution with level

Two extremes , vs.

Every element has a numeric @id attribute, unique integer values

No text nodes. No schema.
Test setting

All systems were used with «standard» options
  • Clearly, other results are possible.

We measured 1 cold and 3-5 hot executions. We report average hot execution times.

We sometimes also report partial times (interesting insights)

Tests are not performed on the same machine 😞  Directly comparable only:
  • QizX and eXist
  • Saxon and MonetDB
  • Galax
  • CDuce

Disclosure !
XPath queries and results
XPath queries

Q11(n): /t1/t2/t3/…/tn
Q12(n): /t1/t2/t3/…/tn/@id
Q13(n): (/t1/t2/t3/…/tn)[1]/@id
Q14(n): (/t1/t2/t3/…/tn)[position()=last()]/@id
Q15(n): /t1[t2/t3/…/tn]/@id
Q16(n): /t1[t2/t3/…/tn]/t2/…/tn/@id
Q17(n): //tn

N varies from 1 to 19. XPath queries tested on:
/t1/t2/.../tn absolute times (different machines)
All systems are clearly not equal; different scaleup behavior. 

Closer look?
eXist detailed times, /t1/t2/.../tn
eXist detailed times, /t1/t2/.../tn
QizX detailed times, /t1/t2/…/tn
/t1/t2/\ldots/tn/@id scaleup with n

Execution time/time for n=1

n

- Exist
- QizX
- CDuce
- MonetDB
- Saxon
- Galax
\((/t_1/t_2/.../t_n)[1]/@id\) scaleup with \(n\)

<table>
<thead>
<tr>
<th>Type</th>
<th>Symbol</th>
</tr>
</thead>
<tbody>
<tr>
<td>Exist</td>
<td>⚝</td>
</tr>
<tr>
<td>QizX</td>
<td>⚫</td>
</tr>
<tr>
<td>CQL</td>
<td>⬤</td>
</tr>
<tr>
<td>MonetDB</td>
<td>⬤</td>
</tr>
<tr>
<td>Saxon</td>
<td>⬤</td>
</tr>
<tr>
<td>Galax</td>
<td>⬤</td>
</tr>
</tbody>
</table>
\((/t_1/t_2/\ldots/t_n)[\text{position()}=\text{last()}]/@id\) scale up with \(n\)
/t1[t2/.../tn]/@id scaleup with n

Execution time / time for n=1

- Exist
- QizX
- Cduce
- MonetDB
- Saxon
- Galax

n
/t1[t2/.../tn]/t2/.../tn/@id

scaleup with n
On exponential2.xml, /t1/t2/…/tn is the same as //tn.

/t1/t2/…/tn scaleup with n
XQuery queries and results
XQuery queries (1)

Q21(n): for $x$ in /t1/t2/
    return <res>{ for $y$ in $x/*/*/…/*
    return fn:data($x/@id) }
    </res>

Q22(n): for $x$ in /t/t2
    return <res>{ $y/*/*/…/* } 
    </res>
Q21(n) scaleup with n

Execution time / time for n=1

n

Exist
QizX
Cduce
MonetDB
Saxon
Galax
Q22(n) scaleup with n
XQuery queries (2)

Q23(n): for $x$ in /t1/t2/
    return <res>{$x//t3, $x//t4, …, $x//tn}</res>

Q24(n): for $x$ in /t/t2
    return <res>{$x/*[position()<=n]}</res>

Q25(n): for $x$ in /t1/t2
    return $x/*[position()<=n]
Q23(n) scaleup with n

Execution time / time for n=1

n

Exist
QizX
Cduce
MonetDB
Saxon
Galax
Q24(n) scaleup with n

Execution time / time for n=1

n

Exist
QizX
Cduce
MonetDB
Saxon
Galax
Q25(n) scaleup with n

Execution time / time for n=1

n

0 1 2 3 4 5 6 7 8 9

Exist
QizX
Cduce
MonetDB
Saxon
Galax
XQuery queries (3)

Q26(0): for $x$ in /t1/t2
    return <res>
    </res>

Q26(1): for $x$ in /t1/t2
    return <res>{for $x1 in $x/* return
        <res>{for $x2 in $x1/* return
            <res>{for $x3 in $x2/* return
                <res>{$x3//t13}</res>
            </res>
        </res>
    </res>}
</res>

Q26(2): for $x$ in /t1/t2
    return <res>{for $x1 in $x/* return
        <res>{for $x2 in $x1/* return
            <res>{for $x3 in $x2/* return
                <res>{$x3//t13}</res>
            </res>
        </res>}
    </res>}
</res>
Q26(n) scaleup with n

Execution time / time for n=1

n

Exist
QizX
Cduce
MonetDB
Saxon
Galax
Lessons learned
Performance lessons learned

Navigation: plain vs. tag (or other) indices

Optimizing positional predicates
  • Simple algorithms for [1]
  • At least 4 alternatives for [position()=last()] (cf. Michael Kay)

Time to locate nodes compared with the time to return them !...

XQueries: optional navigation in the where clause relatively well supported across the board
  • Complex, overlapping results do crash some systems
  • Underlying platform quirks (mainly Caml, also programming issues)
Methodology lessons learned

Micro-benchmarking pays

- Interesting insights even for undocumented systems
- Corner cases and bugs signalled to implementors
- Varying document, parameters etc. outlines impact of each exact parameter

Collecting thousands of points 😞 😞 😞. Automated tools?

Intermediary times extremely interesting. Are they well reported?