Improving Pattern Matching Performance in XSLT

John Lumley  Michael Kay

jωL Research  Saxonica

Saxonica
Synopsis

Some XSLT frameworks use lots of generic pattern templates

\[\text{*[predicate]}\]

with high pattern-matching costs

Improving performance for these:

- Investigating the pattern matching
- Common pattern preconditions
- Other 'oracle' possibilities
- Configuring such tuning
introductory apologies

• I have assumed you have some familiarity with XSLT

• We discuss specific XSLT stylesheets (*DITA-OT*) operating on a particular XSLT engine (*Saxon*)

*If not, then this talk might still amuse you with lots of graphs & pictures*

*As the Americans caution: your mileage may vary*
XSLT push operation

<xs1:apply-templates
    mode="mode"
    select="expr"/>

<xs1:template
    mode="mode"
    match="pattern">
    instructions....
</xs1:template>
XSLT 'push' templates

- exists(@match) and @mode=#current
- eval(@match,$context-item) = true()
- highest import precedence
- highest pattern priority
- selected template set

empty
one
() execute template body
two+ error or last
What Saxon does

- attribute
  - class
- element
  - alpha
  - bravo
- *...
- @*...

Rank order
Differing vocabulary/framework architectures - *DocBook*

```xml
<d:itemizedlist>
  <d:listitem>
    <d:para>Suspending rule ambiguity checking. </d:para>
  </d:listitem>
</d:itemizedlist>

<xsl:template
  match="d:itemizedlist/d:listitem">
  ...
</xsl:template>
```
Differing vocabulary/framework architectures - DITA

<ul class="- topic/ul ">
  <li class="- topic/li ">
    Regeneration parts</li>

<codeph class="+ topic/ph pr-d/codeph ">

<xsl:template match=""
  *[contains(@class, ' topic/ul ')]/
  *[contains(@class, ' topic/li ')]">

  ... ...
A sample transformation

<dita>

DITA-OT
transform.topic2fo.main

XSLT1.0/2.0

• 2.66 MB
• XML tree:
  • 13,066 elements
  • 46,831 attributes
  • 6,093 text

• 58 source files
• 70 modes
• Templates:
  • 418 pattern (258 #default)
  • 155 named

• 19,441 elements
• 91,048 attributes
• 6,140 text

80 pages
262 tables
4,8673 cells
## Significant Modes

<table>
<thead>
<tr>
<th>Mode</th>
<th>Purpose</th>
<th>invocations</th>
<th>time</th>
<th>#template patterns in mode</th>
<th>#templates matched</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td>#</td>
<td>%</td>
<td>/ ms</td>
<td>%</td>
</tr>
<tr>
<td>#default</td>
<td>General</td>
<td>13,095</td>
<td>17.2</td>
<td>4,330</td>
<td>97.8</td>
</tr>
<tr>
<td>toc</td>
<td>Table of Contents</td>
<td>22,088</td>
<td>29.1</td>
<td>51</td>
<td>1.1</td>
</tr>
<tr>
<td>bookmark</td>
<td>Bookmarks</td>
<td>37,752</td>
<td>49.7</td>
<td>33</td>
<td>0.8</td>
</tr>
<tr>
<td>all templates</td>
<td></td>
<td>75,950</td>
<td></td>
<td></td>
<td></td>
</tr>
</tbody>
</table>

<table>
<thead>
<tr>
<th>Mode</th>
<th>element(*)</th>
<th>element(named)</th>
<th>attribute(named)</th>
<th>#templates matched</th>
</tr>
</thead>
<tbody>
<tr>
<td>#default</td>
<td>240</td>
<td>19</td>
<td>8</td>
<td>39</td>
</tr>
<tr>
<td>toc</td>
<td>2</td>
<td>4</td>
<td>0</td>
<td>3</td>
</tr>
<tr>
<td>bookmark</td>
<td>2</td>
<td>5</td>
<td>0</td>
<td>3</td>
</tr>
</tbody>
</table>
Template 'Rank'

this is the most important slide in this presentation

Template rank distribution
mode: #default

KEEP CALM AND PAY ATTENTION
Templates used

Templates used with mode: #default
Most frequent templates
## Frequent patterns, mode `#default`

<table>
<thead>
<tr>
<th>Order</th>
<th>Rank</th>
<th>%calls</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>52</td>
<td>26</td>
<td>28.5</td>
<td>*[contains(@class, 'pr-d/codeph ')]</td>
</tr>
<tr>
<td>204</td>
<td>5</td>
<td>25.0</td>
<td>*[contains(@class, 'topic/tbody ')]/ *[contains(@class, 'topic/row ')]/ *[contains(@class, 'topic/entry ')]</td>
</tr>
<tr>
<td>151</td>
<td>9</td>
<td>8.5</td>
<td>*[contains(@class, 'topic/p ')]</td>
</tr>
<tr>
<td>199</td>
<td>5</td>
<td>7.5</td>
<td>*[contains(@class, 'topic/strow ')]/ *[contains(@class, 'topic/stentry ')]</td>
</tr>
<tr>
<td>206</td>
<td>5</td>
<td>5.3</td>
<td>*[contains(@class, 'topic/tbody ')]/ *[contains(@class, 'topic/row ')]</td>
</tr>
<tr>
<td>205</td>
<td>5</td>
<td>5.1</td>
<td>*[contains(@class, 'topic/thead ')]/ *[contains(@class, 'topic/row ')]/ *[contains(@class, 'topic/entry ')]</td>
</tr>
</tbody>
</table>
Detailed time measurement

Template matching times
mode: #default
Most time-expensive patterns

<table>
<thead>
<tr>
<th>order:rank</th>
<th>% time</th>
<th>Pattern</th>
</tr>
</thead>
<tbody>
<tr>
<td>204:5</td>
<td>31.2</td>
<td>@C{ topic/tbody }/@C{ topic/row }/@C{ topic/entry }</td>
</tr>
<tr>
<td>52:26</td>
<td>10.6</td>
<td>@C{ pr-d/codeph }</td>
</tr>
<tr>
<td>151:9</td>
<td>9.9</td>
<td>@C{ topic/p }</td>
</tr>
<tr>
<td>199:5</td>
<td>9.9</td>
<td>@C{ topic/strow }/@C{ topic/stentry }</td>
</tr>
</tbody>
</table>

Longest processed templates:
mode:#default
Costly templates
Costly templates ii

@class has been searched ~200 times already for this node

I search @class

so do I

so do I

so do I

so do I

so do I

so do I

so do I

so do I

so do I

so do I

so do I

so do I

so do I

so do I
Can we improve?

• Rule preconditions — partitioning large rule sets by common (boolean) conditions

• Using *oracle guarantees*, shortcuts not applicable to all stylesheets:
  – Exploiting template mutual exclusivity
  – Pre-processing significant data
  – Pattern rewrites

• Configuring stylesheet execution
Common preconditions

- `chapter/title[condition1], chapter/title[condition2], chapter/para, chapter/section ...`

- `exists(parent::chapter) → chapter/title[condition1], chapter/title[condition2], chapter/para, chapter/section ...`

- `pre: exists(parent::chapter) → title[condition1], title[condition2], para, section ...`
Preconditions for DITA-OT

exists(@class) \times \text{ they all have one}
contains(@class, string_i) \times \text{ very little commonality}

\textbf{GOAL:} \quad p \text{ preconditions each shared by } \sim m \text{ patterns}
\text{ 'minimum work': } p \approx m \approx \sqrt{N}

precondition-for (contains(@class, string_i)) \rightarrow
contains(@class, any-substring-of(string_i))

<table>
<thead>
<tr>
<th>Initial Substring size</th>
<th>1</th>
<th>2</th>
<th>3-5</th>
<th>6</th>
<th>7</th>
<th>8</th>
</tr>
</thead>
<tbody>
<tr>
<td># preconditions</td>
<td>1</td>
<td>12</td>
<td>14</td>
<td>16</td>
<td>46</td>
<td>75</td>
</tr>
<tr>
<td>Largest set</td>
<td>250</td>
<td>146</td>
<td>121</td>
<td>121</td>
<td>121</td>
<td>17</td>
</tr>
</tbody>
</table>

contains(@class,'abcdef') \land\land
pre:contains(@class,'abc') \times \rightarrow contains(@class,'def')
Substring precondition distribution

Precondition distribution
mode: #default
2-character initial substring
Implementing preconditions

\[ *[\text{contains(@class,} \text{string}_i)] \rightarrow *[\text{contains(@class,} \text{substring}(\text{string}_i,1,2))]* \]
Substring preconditions
Consulting the *oracle*

- Reassurances as practical truths, not applicable to all stylesheets:
  - Mutual exclusivity of templates:
    - Suspending rule ambiguity checks
    - Reordering templates & imports
  - Pre-tokenizing significant data
Mutual exclusivity: 'Un-disambiguating' rules

selected template set

empty

one

execute template body

two+

error or last

Match this...

... no need to check these
'Mutually exclusive': promoting stylesheets
'Mutually exclusive': promoting stylesheets
Pre-tokenizing @class data

R1: *[contains(@class,' topic/entry ')]
R2: *[contains(@class,' topic/row ')]
R3: *[contains(@class,' topic/row ')]/
   *[contains(@class,' topic/entry ')]

R1: *[tokenize(@class,'s+')='topic/entry']
R2: *[tokenize(@class,'s+')='topic/row']
R3: *[tokenize(@class,'s+')='topic/row']/
   *[tokenize(@class,'s+')='topic/entry'

\$tokens.self.class := tokenize(self::*/@class,'s+')
\$tokens.parent.class := tokenize(parent::*/@class,'s+')

\$precondition_M := \$tokens.self.class = 'topic/entry'
\$precondition_N := \$tokens.self.class = 'topic/row'
\$precondition_P := \$tokens.parent.class = 'topic/row'

R1: \$precondition_M && *
R2: \$precondition_N && *
R3: \$precondition_P && \$precondition_M && *
Configuring the tuning

Define preconditions via patterns (cf. Snelson):

\[
\text{contains}\left(@\text{class},\ \ s[\text{starts-with}(.,' ') \ \text{and} \ \text{ends-with}(.,' ')\right) \rightarrow \text{contains}\left(@\text{class},\ \ \text{substring}(s,1,2)\right)
\]
Unifying for preconditions

* [contains(@class, 'ui-d/screen')] unifies with?

contains()
attribute::attribute(Q{}/class)
p.withPredicate
nodeTest
cvUntyped 'ui-d/screen'
data slash
treat attribute::attribute(Q{}/class)
dot

contains() filter
Ss and
starts-with() ends-with()
dot dot

$s := 'ui-d/screen'

binds variable
qualifies value
grounded eval
contains()
attribute::attribute(Q{}/class)
substrinct()
'ui-d/screen' 1 2
contains(@class,'u')

XMLLondon 2015 - John Lumley
27 May, 2015
Conclusions

• Large sets of *[$predicate] XSLT patterns can be very expensive (DITA is paying a lot for @class extensibility)

• Preconditions are practical: but which ones?

• Other oracle measures can help
  – 'This document is mostly tables'

• 'Tuning' can be configured via patterns
  – Watch Saxonica