Optimizing the Xopus XSL pipeline

November 22nd 2008

Laurens van den Oever

Introducing

- Laurens van den Oever
  CEO of Xopus BV

- Xopus BV
  Friendly XML editor
  Since 2007, 10 people

- Q42 Internet BV
  Friendly internet technology
  Since 2000, 25 people

The M&M Game

Rules:

- Every slide with M&M's contains a question
- The first **correct** answer is rewarded with a baglet of M&M's!

Xopus Overview

- Browser based XML editor
- Non-technical target audience
- MVC
  - XML
  - XSL
  - XSD
- 100% Javascript & XSL

Demo

Recipe Demo

Question

xopus

Why do we write our software in Javascript?

Answer

xopus

Javascript does not require a client side install.

XSL Explained

- XSL is a transformation from one XML document into another
- Xopus: customer domain specific XML => XHTML
- The V in MVC

XSL Example: XML Input


```
<recipe xml:lang="en-US">
  <title>Traditional Christmas Ham</title>
  <author>Joanna</author>
  <ingredients>
    <ingredient> ... </ingredient>
    <ingredient> ... </ingredient>
  ...
</recipe>
```

XSL Example: XSL Transform

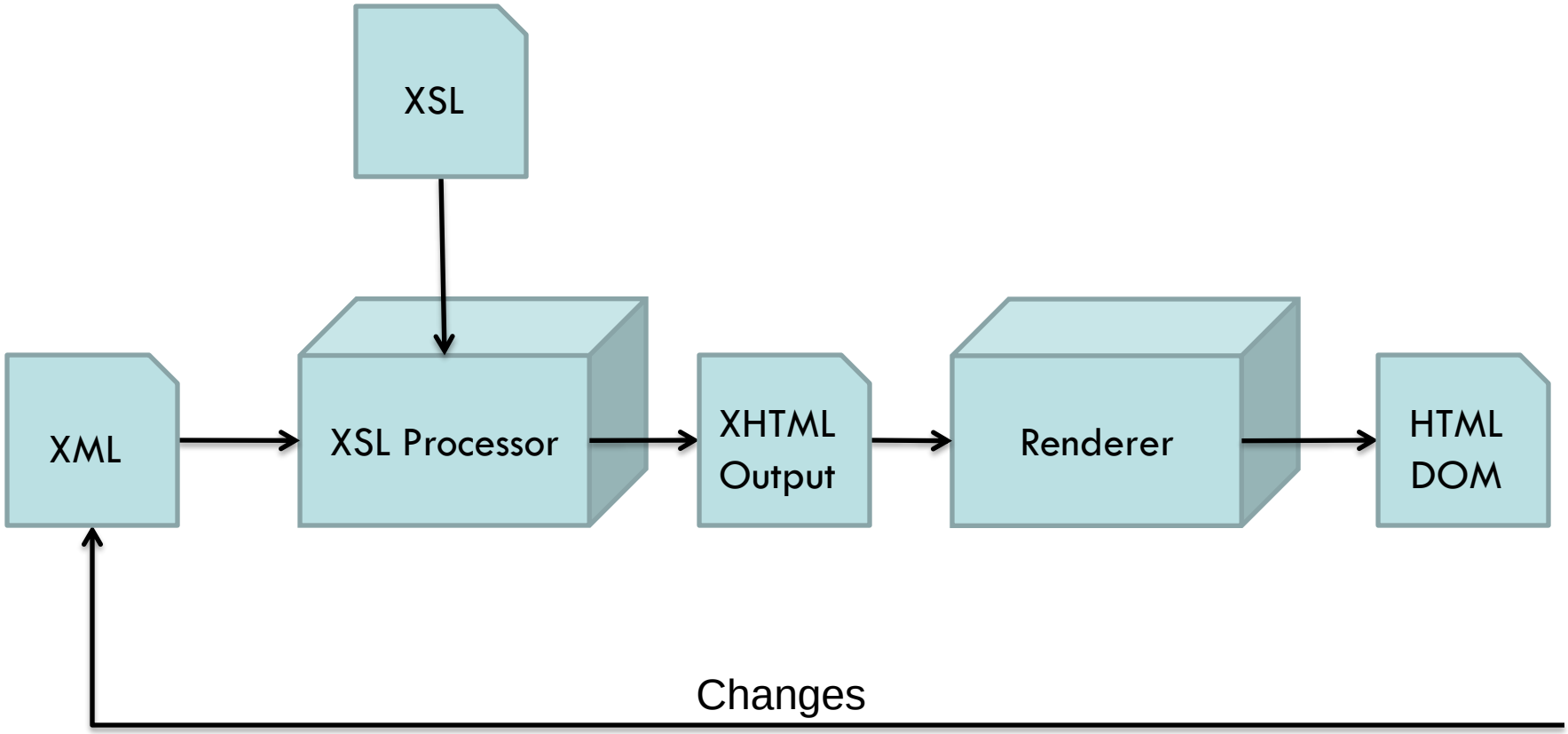

```
<xsl:template match="author">
  <div class="author">
    <span class="by">by </span>
    <span class="author-name">
      <xsl:value-of select="." />
    </span>
  </div>
</xsl:template>
```

XSL Example: HTML Output


```
<div class="author">
  <span class="by">by </span>
  <span class="author-name">Joanna</span>
</div>
```

Xopus XSL Pipeline

xopus



Xopus XSL Pipeline

- Executed after every change
- Performance is critical:

User experience:	Great	Unacceptable
Startup	<1s	>10 s
Enter	<150 ms	>1000 ms
Typing	<15 ms	>50 ms

- Need to support large XML documents

⇒ Optimizing maximum document size

⇒ Target: 10MB (99,9% of XML documents)

Version 1: Full XSL

- Entire XML is transformed
- XHTML output replaces entire HTML DOM
- Full support of XSL standard
- Performance proportional to document size

Version 1: Full XSL

xopus

- Performance for 100KB document:

  (P4 3GHz, 3GB, WinXPSP3, IE7, MSXML6)

	XSL (ms)	Renderer (ms)	Total (ms)
t = 0	400	700	1100
t > 0	300	700	1000

Question

xopus

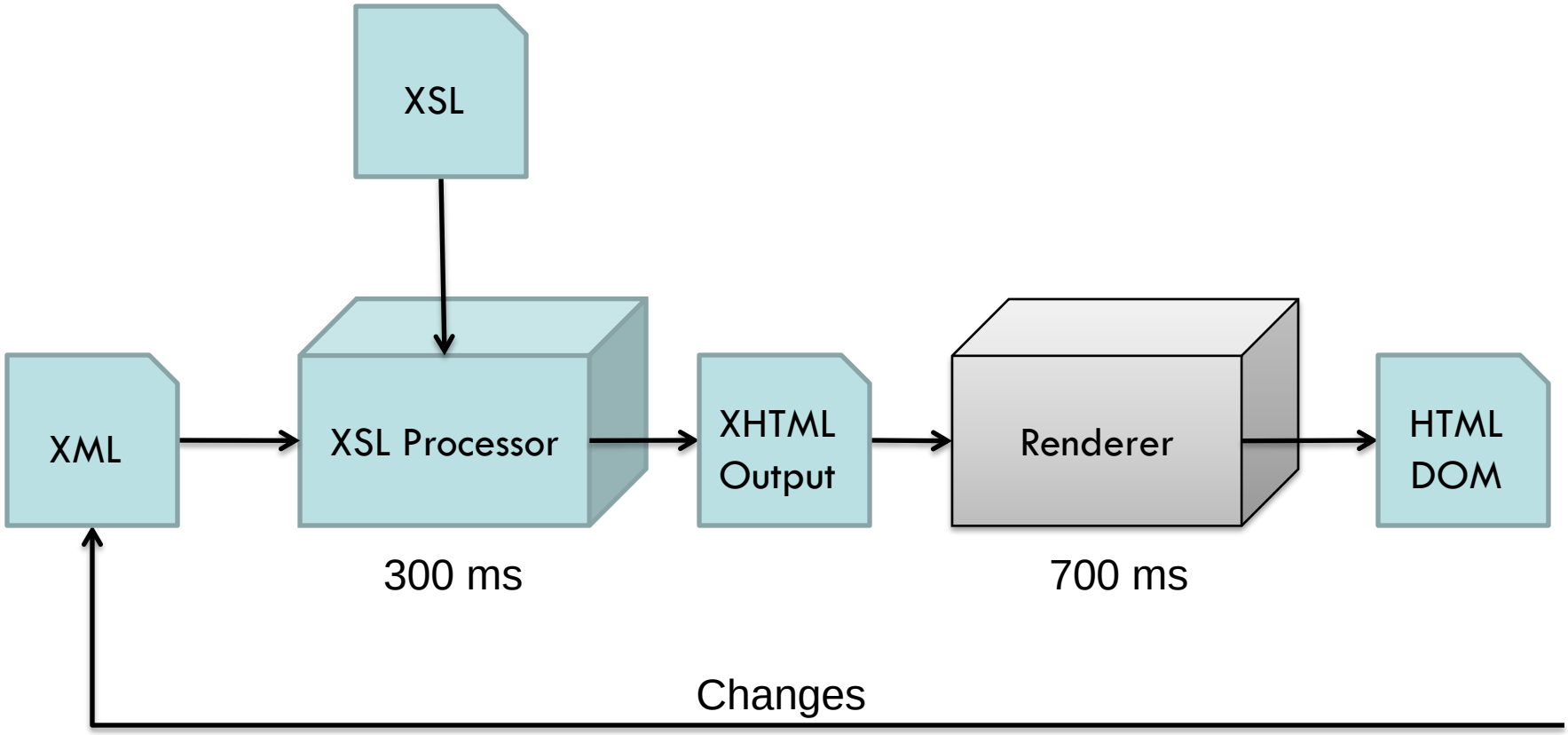
What is the best way to minimize the runtime of an algorithm?

Answer

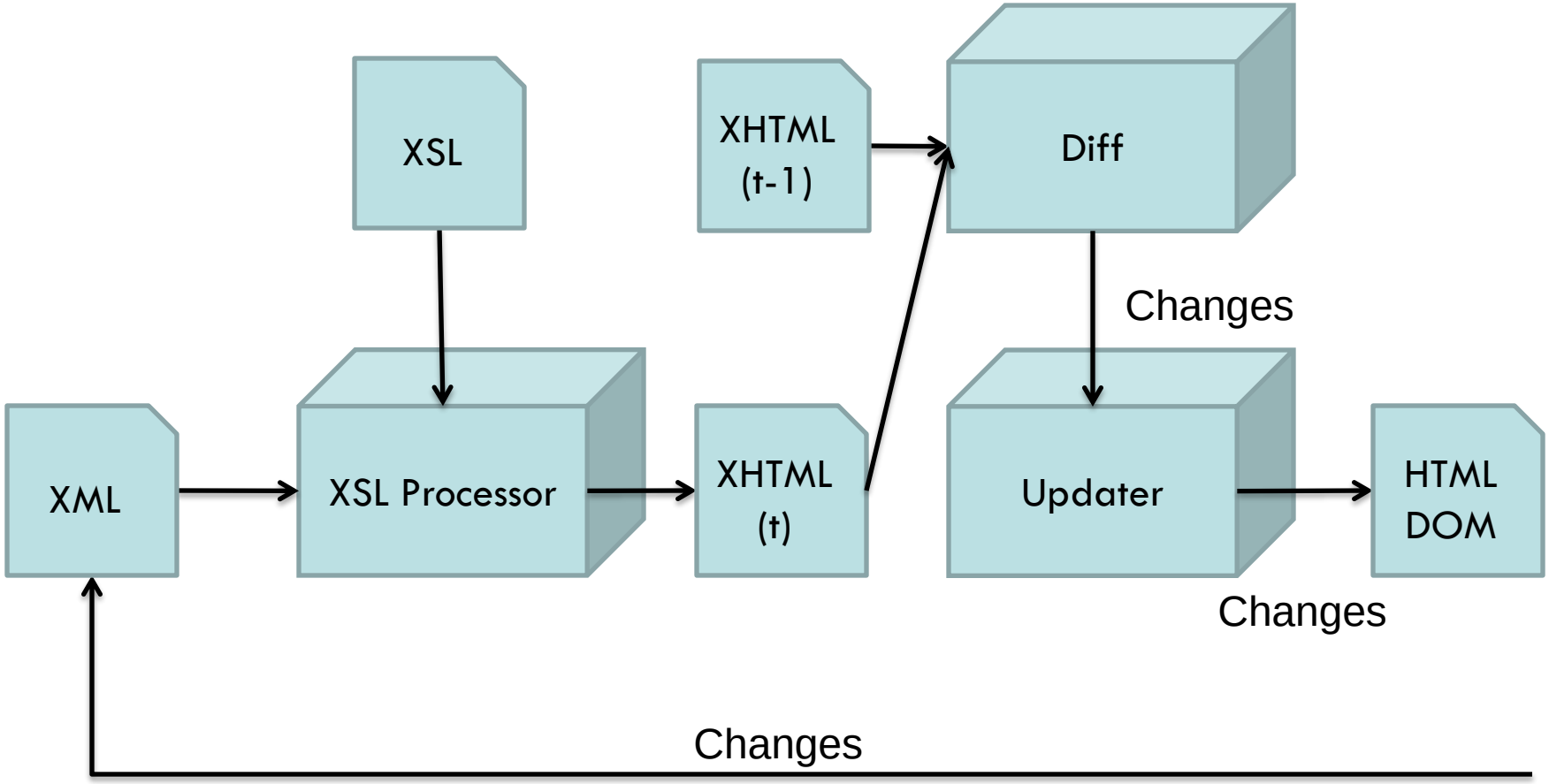
Don’t run it.

Xopus XSL Pipeline

xopus



Version 2: Differential Rendering



Question

Which problem is not solved by differential rendering?

Answer

xopus

The first rendering (t = 0) is still slow.

Because differential rendering can only render changes.

Version 2: Differential Rendering

- Entire XML is transformed
- XHTML output is compared with previous XHTML output
- Changes are applied to HTML DOM
- Full support of XSL standard
- Performance still proportional to document size

Version 2: Differential Rendering

Performance for 100KB document:

- Full XSL:

(ms)	XSL	Renderer	Total
t > 0	300	700	1000

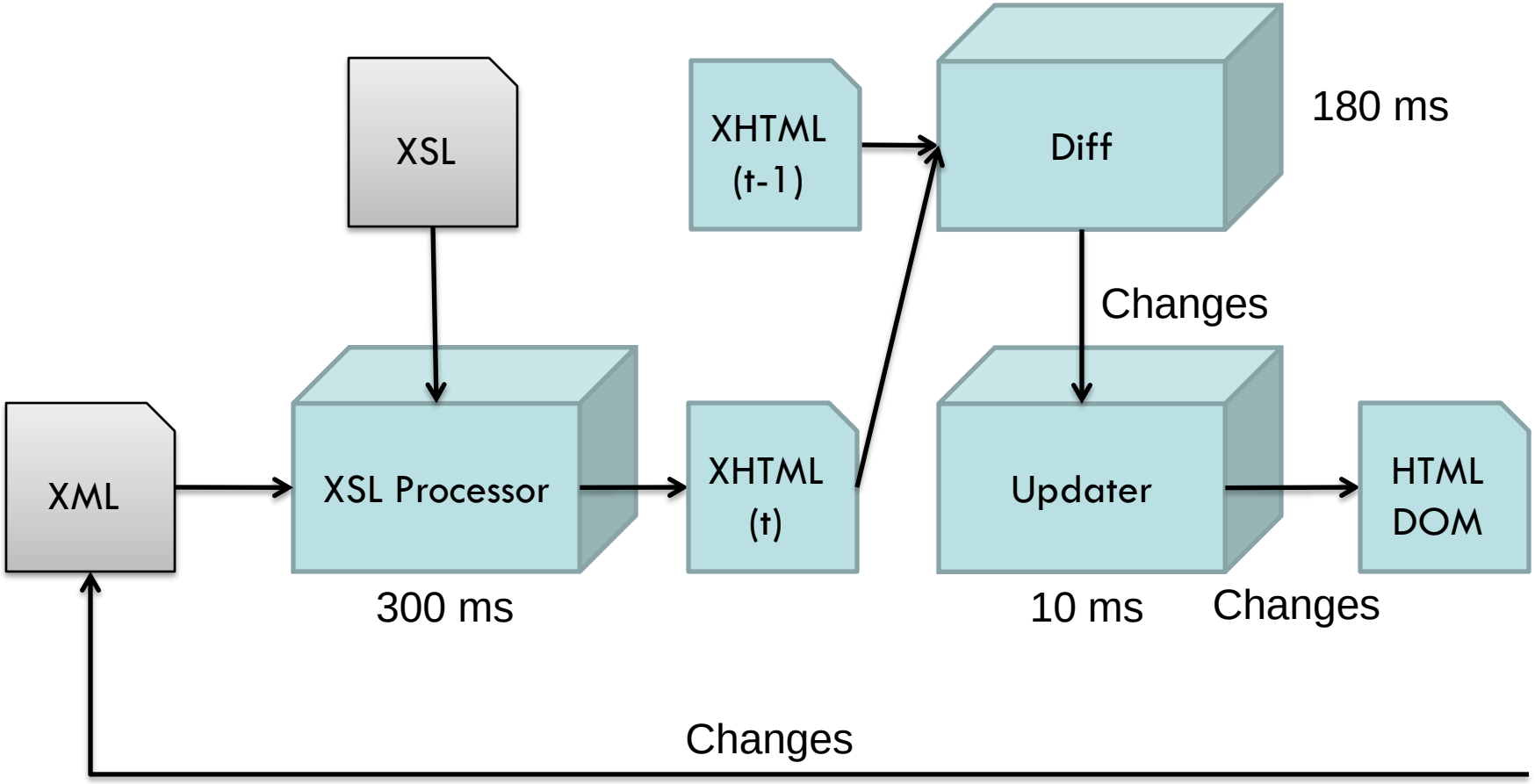
- Differential Rendering:

(ms)	XSL	Diff	Updater	Total
t > 0	300	180	10	490

$\Rightarrow$ 2x improvement!

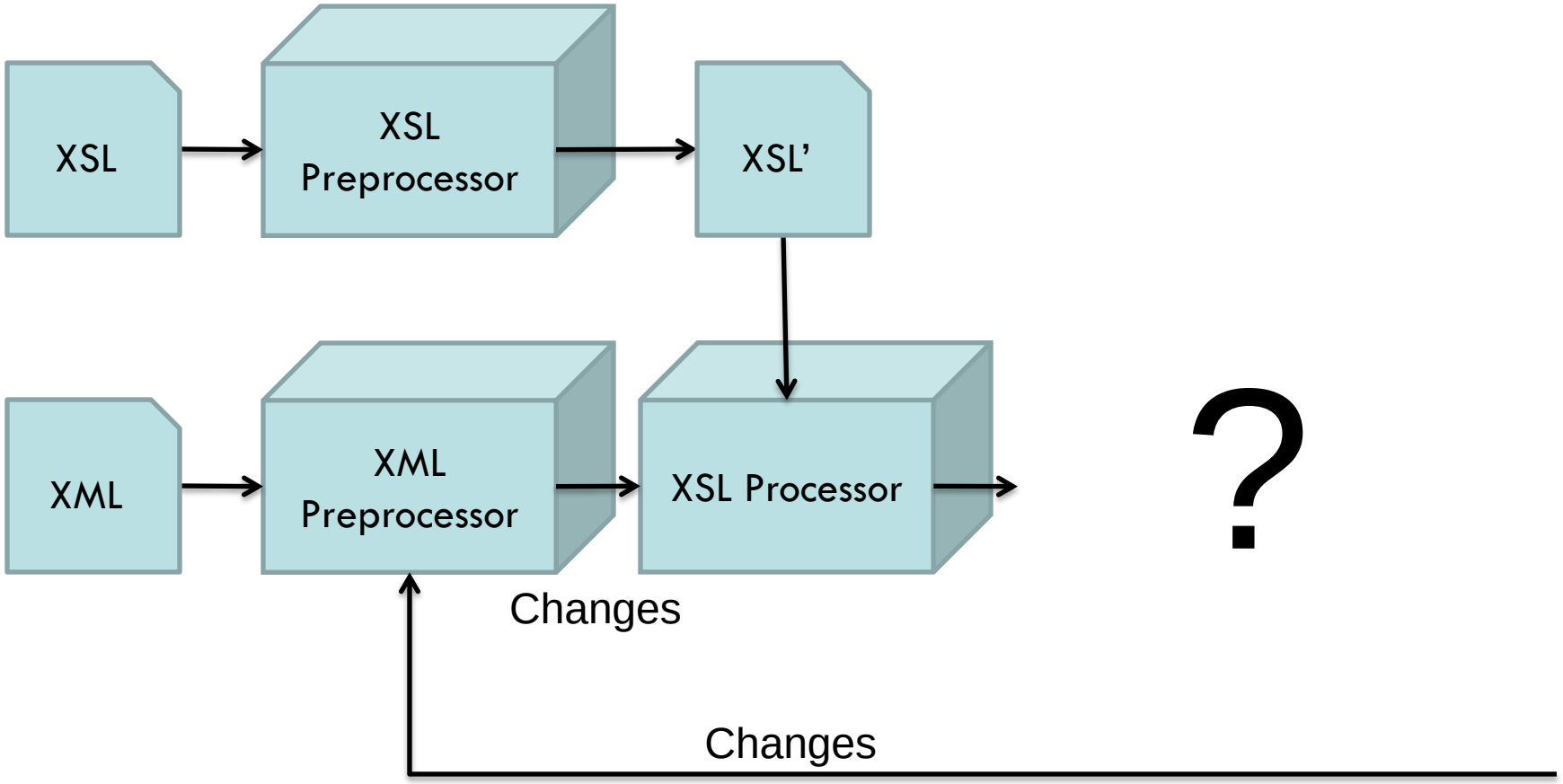
Version 2: Differential Rendering

xopus



Version 3: Partial XSL

xopus



Question

xopus

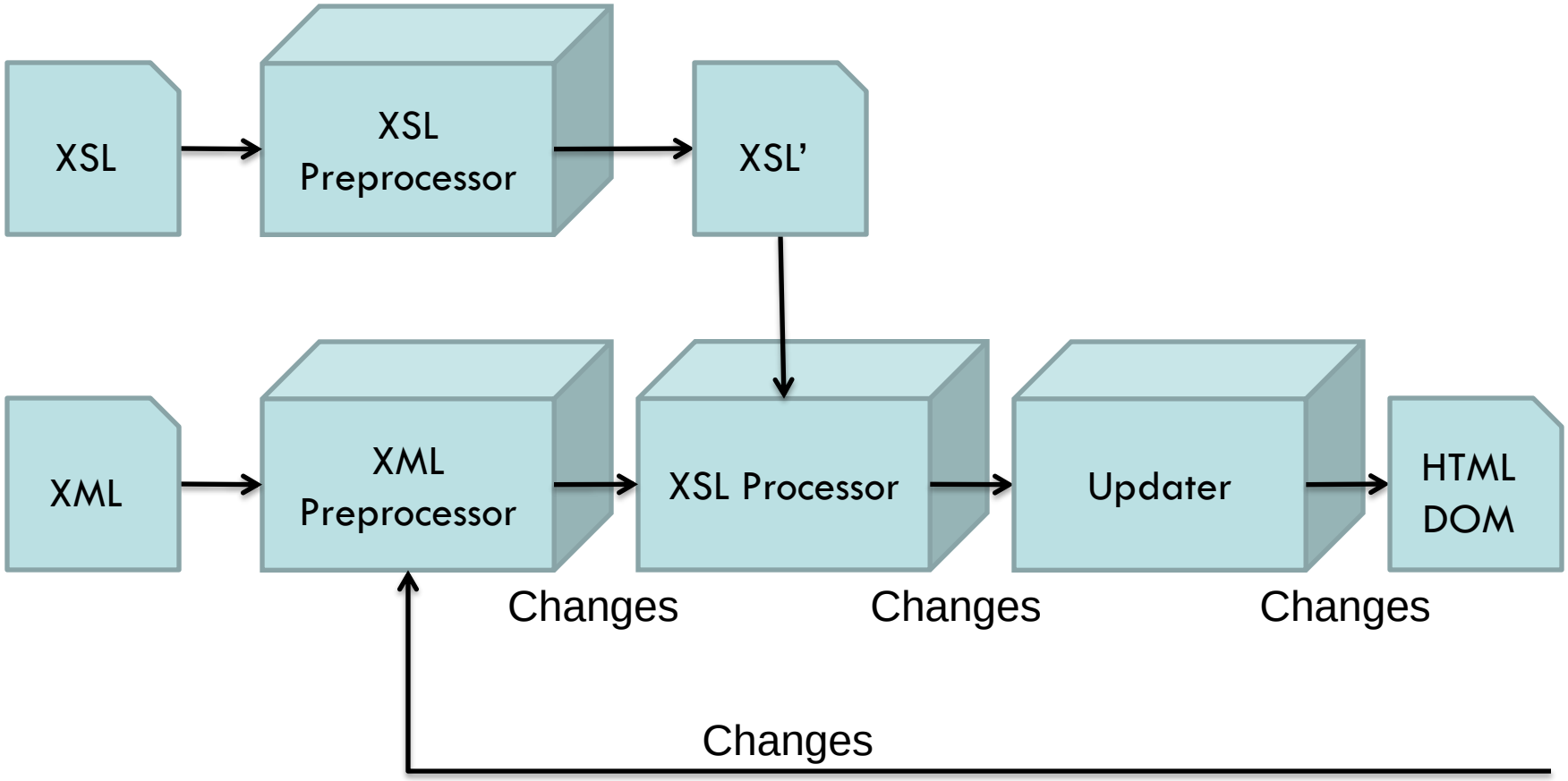
Which bonus do we get from this architecture?

Answer

We no longer need to diff since we now only transform changes.

Version 3: Partial XSL

xopus



Version 3: Partial XSL

- Changes are tagged in XML
- XSL only transforms changes
- Changes are applied to HTML DOM
- Limited support of XSL standard
- Performance proportional to changed fragment size

Version 3: Partial XSL

xopus

Performance for 100KB document:

- Differential Rendering:

(ms)	XSL	Diff	Updater	Total
t > 0	300	180	10	490

- Partial XSL:

(ms)	Pre	XSL	Updater	Total
t > 0	0	10	10	20

$\Rightarrow$ 25x improvement!

Demo

xopus

```
<xsl:template match="paragraph">
  <xsl:variable name="l"
                select="string-length(.) mod 25"/>
  ...
</xsl:template>
```

Demo

xopus

```
<xsl:template match="paragraph">
  <xsl:variable name="l" .../>
  <xsl:variable name="color">
    rgb(<xsl:value-of select="10 * $l"/>,
        <xsl:value-of select="255 - (5 * $l)"/>,
        <xsl:value-of select="255 - (10 * $l)"/>)
  </xsl:variable>
  ...
</xsl:template>
```

Demo

xopus

```
<xsl:template match="paragraph">
  <xsl:variable name="l" .../>
  <xsl:variable name="color">
    ...
  </xsl:variable>
  <p style="background: {$color}">
    <xsl:apply-templates select="node()"/>
    [<xsl:value-of select="string-length(.)"/>]
  </p>
</xsl:template>
```

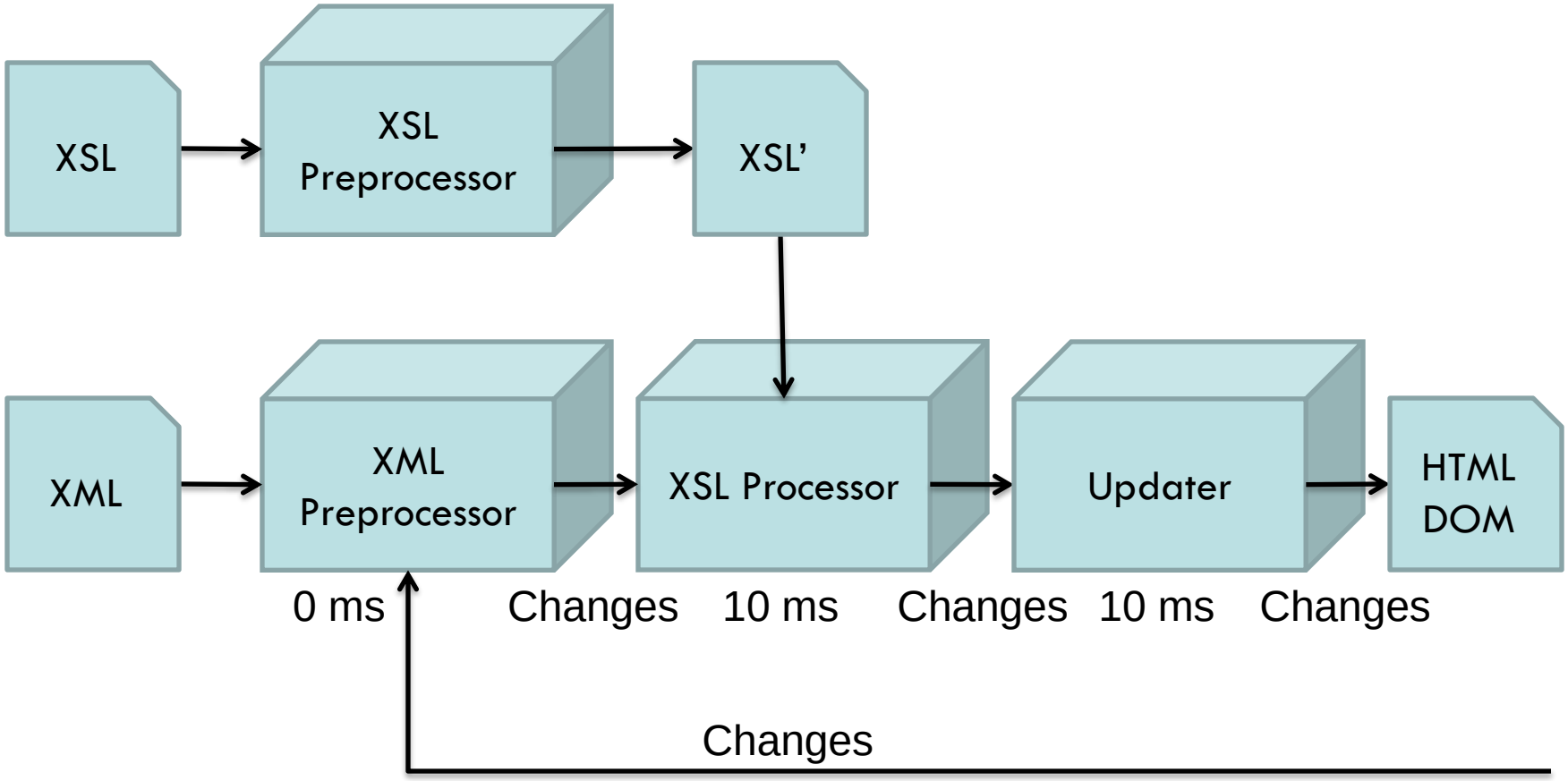
Demo

Partial XSL Demo

- 100KB XML document
- XSL executed for every keystroke
- Ugly

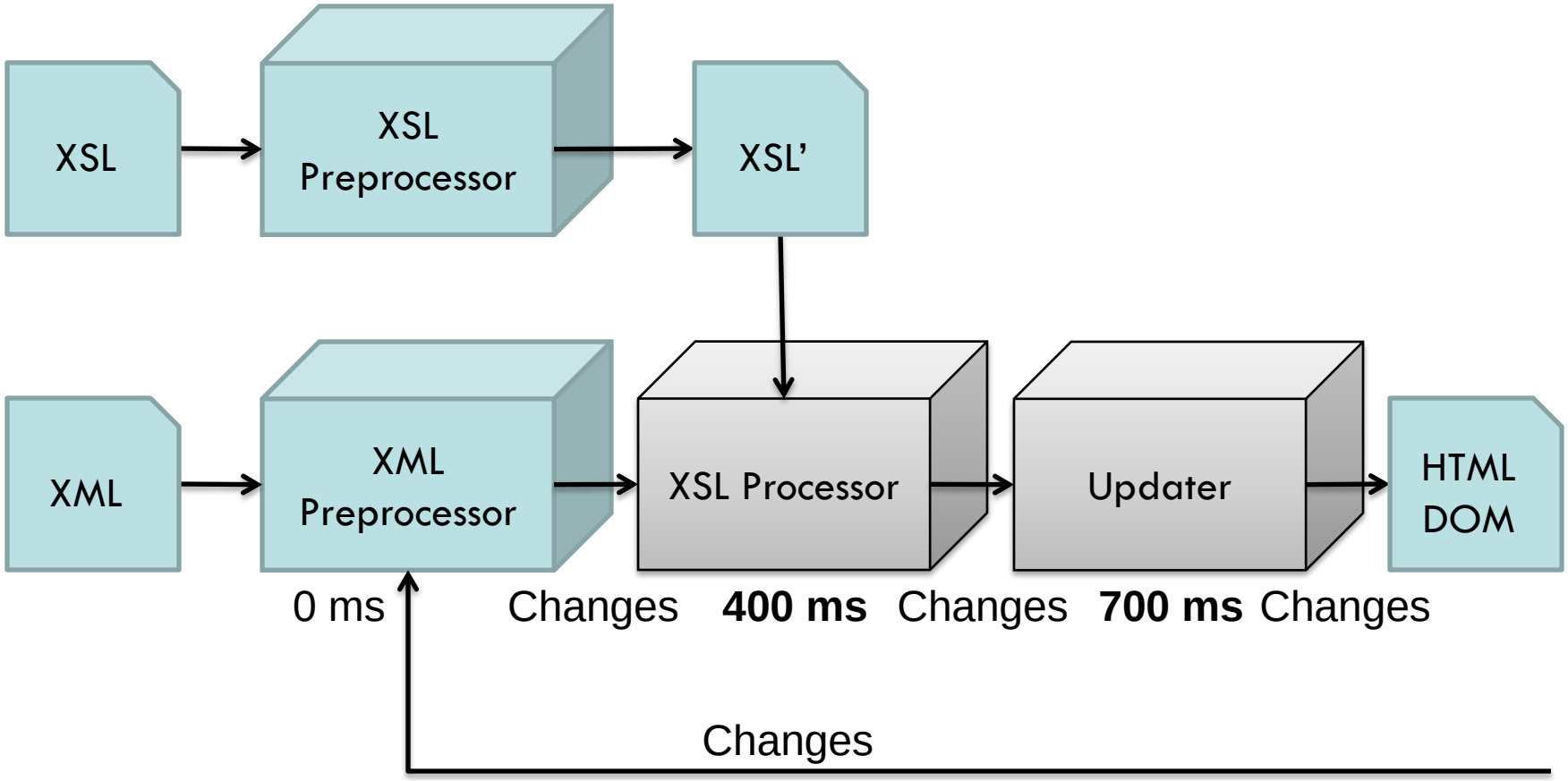
Version 3: Partial XSL (t > 0)

xopus



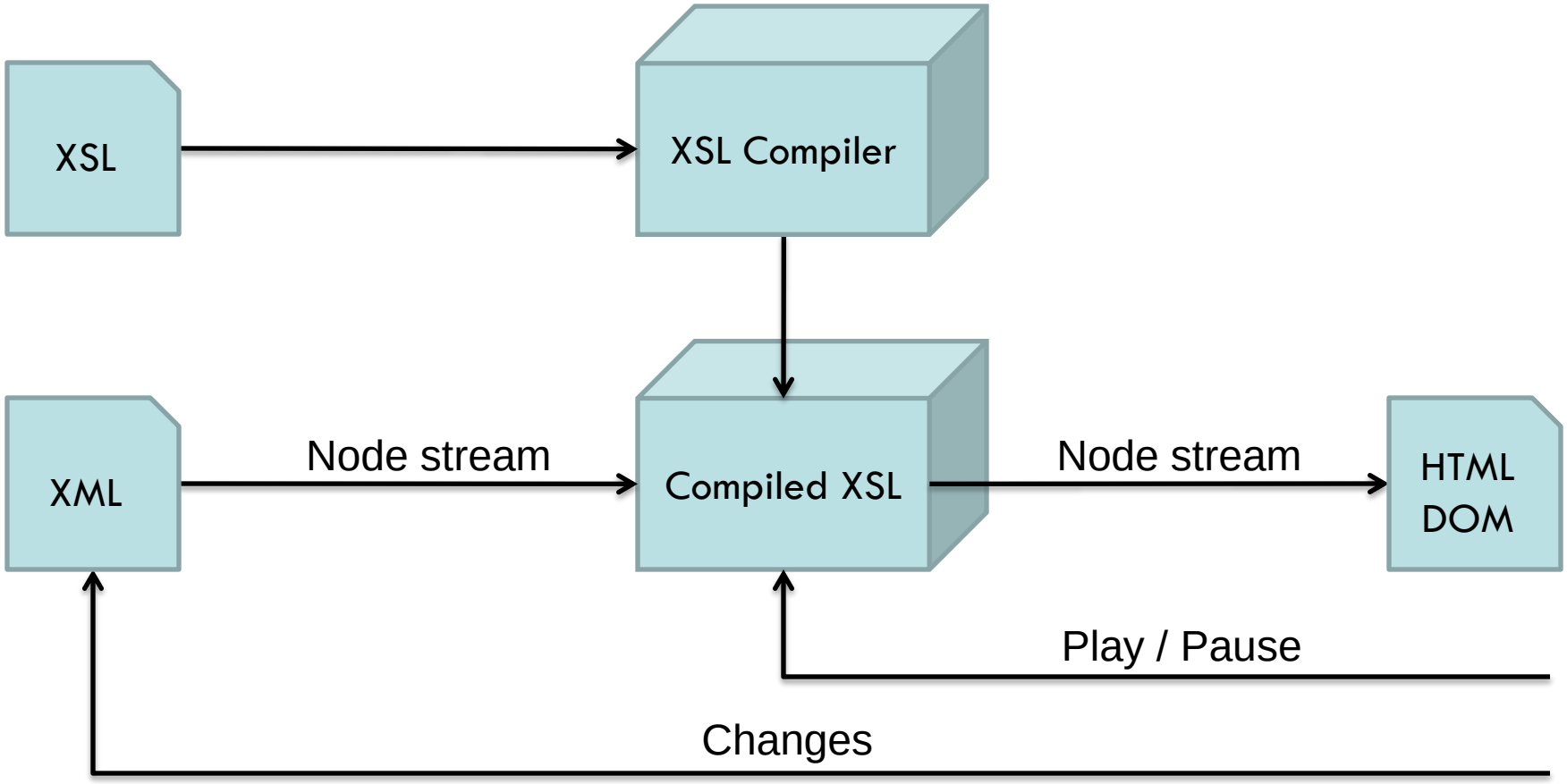
Version 3: Partial XSL (t = 0)

xopus



Version 4: Incremental XSL

xopus



Version 4: Incremental XSL

xopus

- XSL is compiled into DOM updating Javascript functions
- Processing is paused when screen is full
- Scrolling and editing continues processing
- Entire XML is downloaded and parsed
- Currently limited support of XSL standard
  (full support is feasible, possibly better than 3rd party)

Version 4: Incremental XSL

Performance for 100KB document:

- Partial XSL:

(ms)	Pre	XSL	Updater	Total
t = 0	0	400	700	1100
t > 0	0	10	10	20

- Incremental XSL:

(ms)	Total
t = 0	220
t > 0	10

⇒ 5x startup improvement!

Version 4: Incremental XSL

Performance for 10MB document:

- 100KB:

(ms)	Total
t = 0	220
t > 0	10

- 10MB:

(ms)	Total
t = 0	1500
t > 0	10

⇒ Sub linear scaling!

Demo

Incremental XSL Demo

- Proof of Concept
- 10MB XML document
- No editing yet

Conclusions

- Maximal document size increased from 8KB to 10MB
- Network speed is now main bottleneck

$\Rightarrow$ With incremental XSL performance is no longer an issue

Questions?

More information:

http://xopus.com

Laurens van den Oever

laurens@xopus.com

BTW: We're looking for new developers!