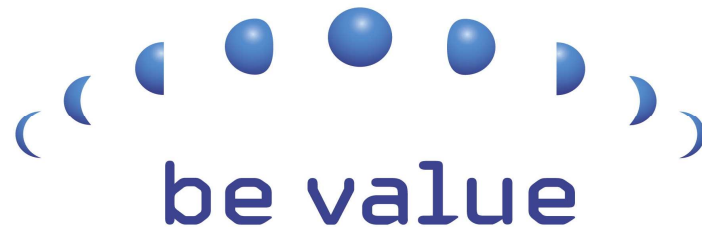


# Cocoon and Large Documents

Performance,  
Scalability,  
Optimization

Nico Verwer, Be Value



The work which lead to this presentation was done on a project for a large publishing company in the Netherlands. One of the tasks in this project for which Cocoon is used involves assembling large numbers of documents into a publication. In order to do this effectively, some issues concerning performance and scalability had to be addressed.

# What is a large document?

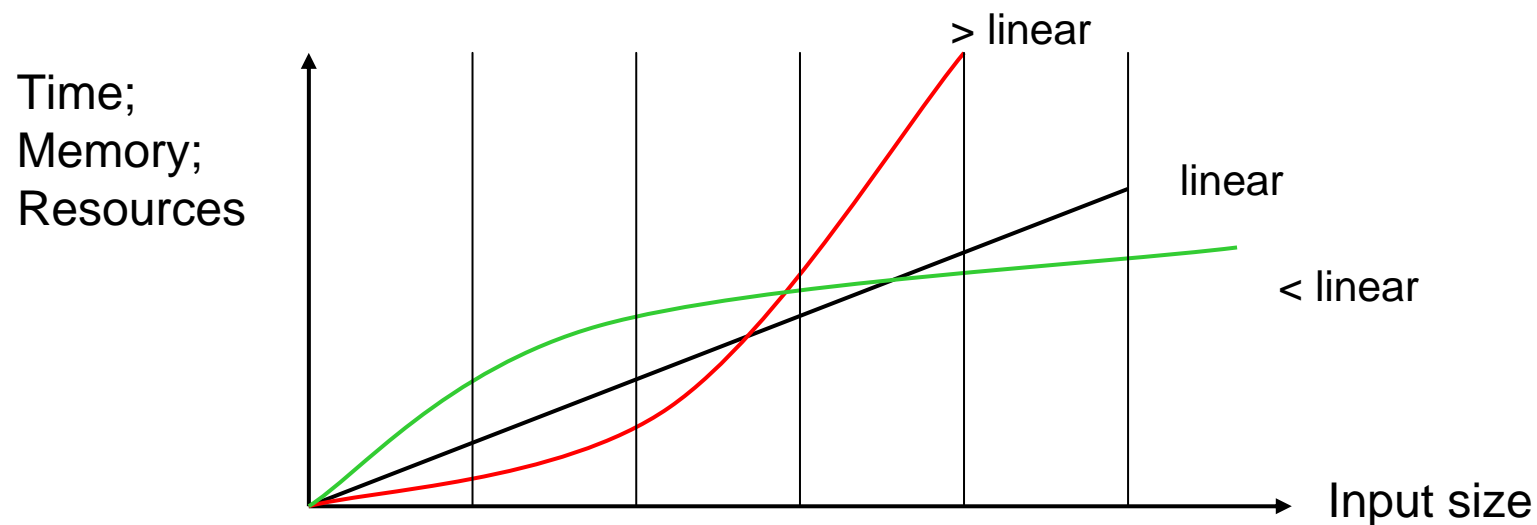
- Hundreds of megabytes – serialized.
- Hundreds of thousands of XML elements.
- Too big for the Cocoon profiler
  - `org.apache.cocoon.ProcessingException: Failed to execute pipeline.: org.xml.sax.SAXException: Index too large`
  - The profiler keeps a `XMLByteStreamCompiler` representation of all SAX events.
- Out of memory errors. Performance tips don't help. XSLT has been optimized. Memory keeps running out.
  - <http://cocoon.apache.org/2.1/performance/tips.html>
  - <http://www.dpawson.co.uk/xsl/sect4/N9883.html>
- `java -Xms32M -Xmx3072M ...`  
is the maximum (without 64-bit JVM extensions and lots of RAM).
- It is the size, not the number of documents.
  - <http://wiki.apache.org/cocoon/CocoonPerformanceResults>

# An alternative profiler

- This will at least show how bad the problem is.
- 'Log-file Output Writing Profiler' (LOWProfiler).
  - <http://wiki.apache.org/cocoon/ProfilingPipelinesWithBigSaxEventStreams>
  - ```
<map:match pattern="test">  
  <map:generate src="bigfile.xml"/>  
  ...do some transformations etc...  
  <map:transform type="lowprofiler"/>  
  <map:serialize type="xml"/>  
</map:match>
```
  - Generates profilerinfo (total time, no detailed times) plus heap-usage.

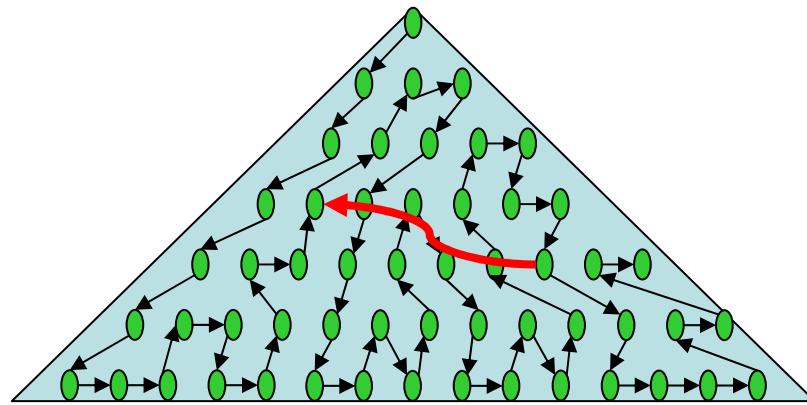
# Using Saxon instead of Xalan

- 1,5 – 10 times faster.
- Uses more memory. ☹!
- Does not solve greater-than-linear behaviour, which is a scalability problem.



# XSLT is a problem

- XSLT transformers must *always* have a complete representation of the input document.
  - Even SAX transformers!
  - Needed for expressions like  
`<xsl:copy-of select="//somewhere[@location='far away']"/>`



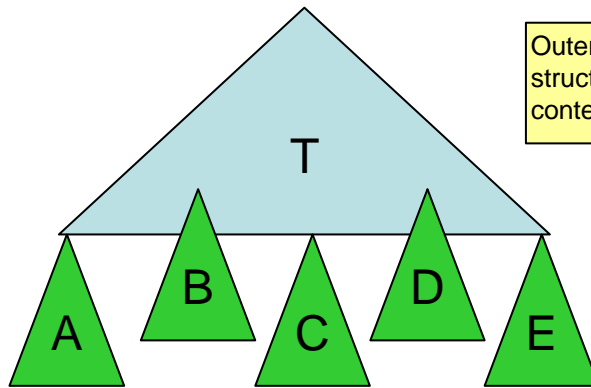
- Optimize XSLT transformations, based on algorithmic pattern.

# Two algorithmic patterns

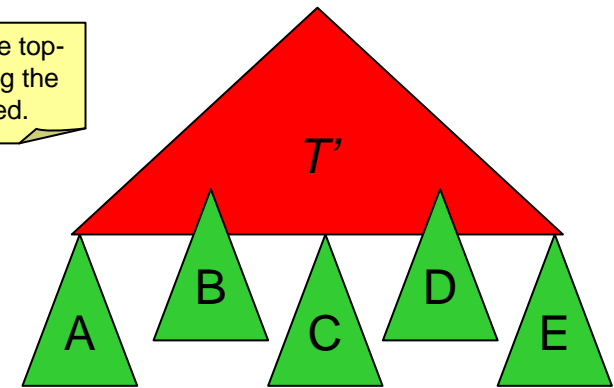
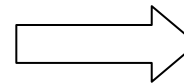
- Many transformations are a combination of (or can be turned into a combination of)

The border between outer and inner can be chosen arbitrarily. Usually it is marked by a specific element (the root-element of the fragments).  
The names reduce and map come from list-operations with a similar pattern.

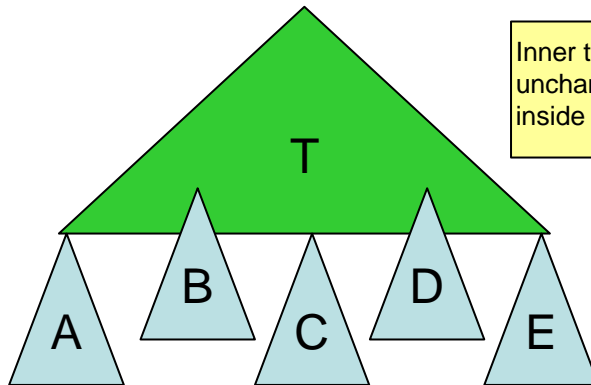
Outer Transformation (reduce)



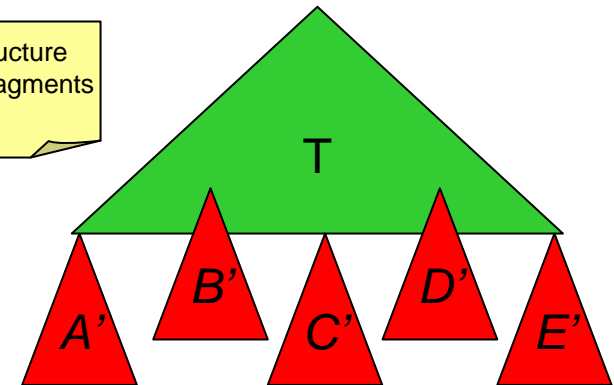
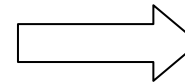
Outer transformations only change the top-structure of an XML document, leaving the content inside this structure unchanged.



Inner Transformation (map)

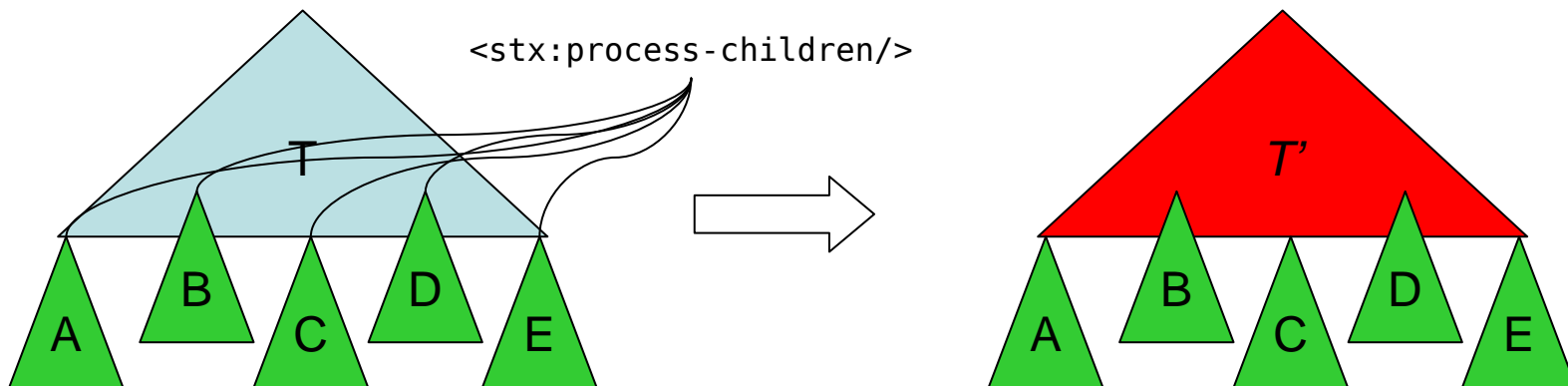


Inner transformations leave the top-structure unchanged, and change the content-fragments inside this structure in the same way.



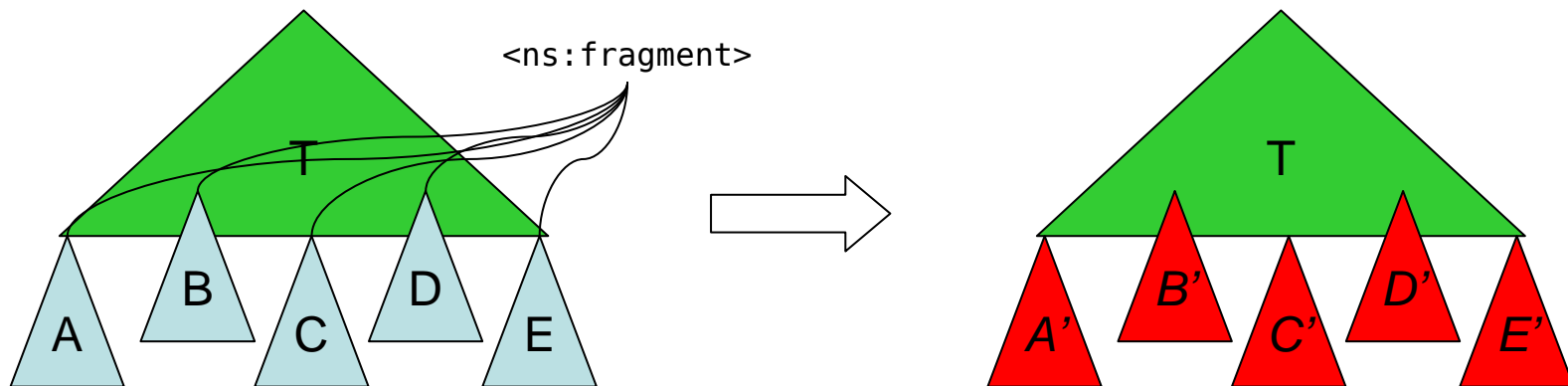
# Streaming Transformations (STX)

- Optimizes outer transformations, but is not expressive enough for all transformations.
- Streaming Transformations for XML (STX)
  - Open Source standard (<http://stx.sourceforge.net/>).
  - High-speed, low-memory.
  - One-pass transformation, therefore:
    - Limited access to the document (STXPath).
    - Syntax resembles XSLT, but is sometimes very different.
    - The Joost STX transformer is available as a Cocoon block.
    - Is actively being developed (Oliver Becker, Humboldt-Universität zu Berlin)



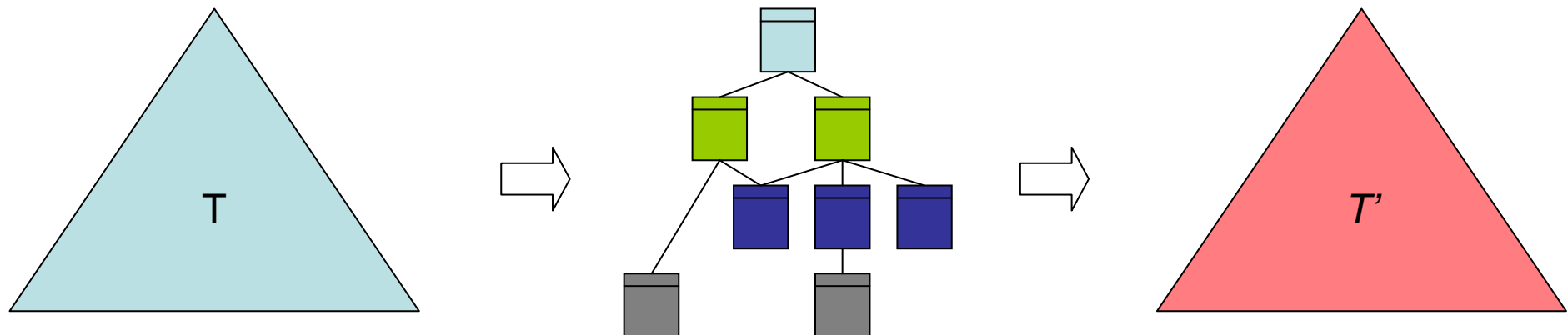
# MultiFragmentTraxTransformer

- Optimizes inner transformations, even if they behave > linear.
  - $n * (s/n)^p < s^p$
- Based on standard XSLT transformer (Saxon or Xalan).
  - Same XSLT stylesheet applied to every fragment.
  - Copies everything outside (above) fragments.
  - No exchange of information between fragments.
  - <http://wiki.apache.org/cocoon/MultiFragmentTraxTransformer>
- ```
<map:transform type="multifragment-xslt" src="stylesheet.xsl">  
  <map:parameter name="fragment.namespace" value="http://namespace"/>  
  <map:parameter name="fragment.element" value="fragment"/>  
  <!-- Parameters for the stylesheet. -->  
</map:transform>
```



# Rewriting transformers in Java

- Optimizes other cases.
- Fastest option, but has disadvantages.
  - More developer effort.
  - Lower level of abstraction.
  - More difficult to maintain.
- Keep it small, simple, configurable and single-purpose!
  - Extend AbstractTransformer or AbstractSAXTransformer.
  - Use analyze / calculate / synthesize approach, with an intermediate data structure.



# Results

		XSLT		Java + XSLT		Java + XSLT + MultiFragment	
XML elements	File size (MB)	Time (s)	Mem. (MB)	Time (s)	Mem. (MB)	Time (s)	Mem. (MB)
700	0.37	10	~24	6	19	1	89
63000	36	~3375 (56'15)	~3600	1295 (21'35)	899	123 (2'03)	174
97000	52	#	#	2632 (43'52)	1510	264 (4'24)	209
447000	242	#	#	#	#	949 (15'49)	313

# Results

