IEEE Focs '05
+
WWW '06

XML
Compression and Indexing

Paolo Ferragina
Dipartimento di Informatica, Università di Pisa

[Joint with F. Luccio, G. Manzini, S. Muthukrishnan]

Five years ago... [now, J. ACM 05]

FOCS 2000

The 41st Annual Symposium on Foundations of Computer Science

Opportunistic Data Structures with Applications

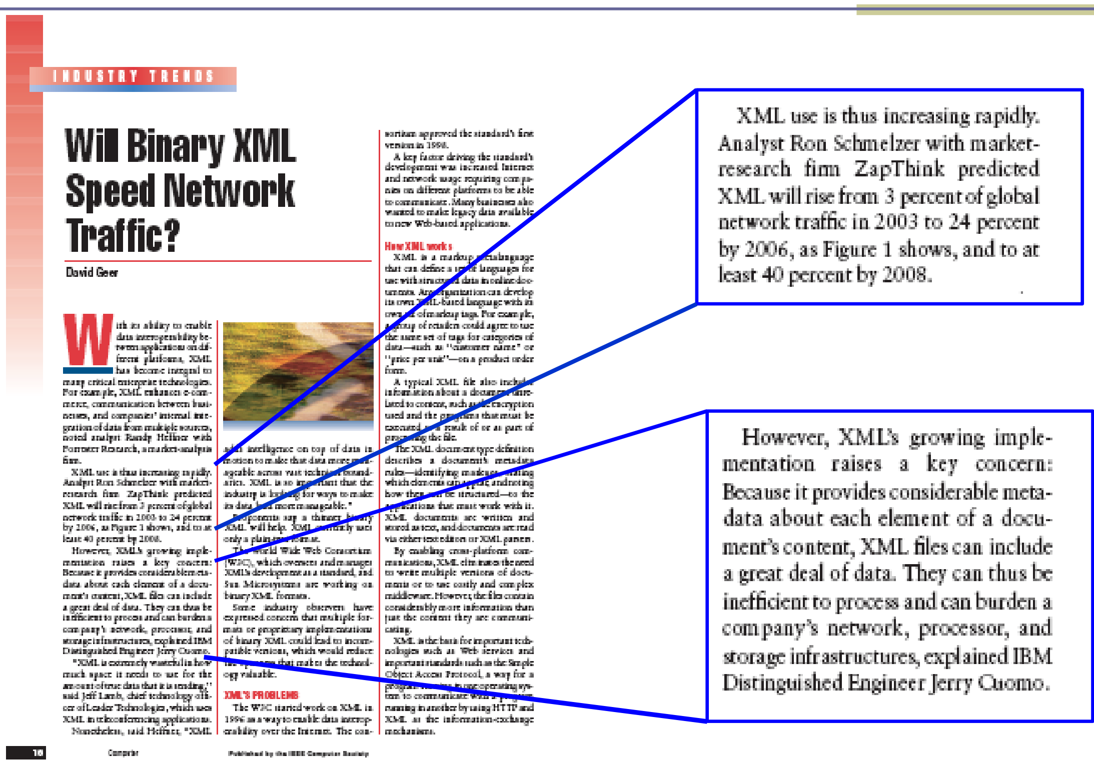
P. Ferragina, G. Manzini

Survey by Navarro-Makinen cites more than 50 papers on the subject !!

An XML excerpt

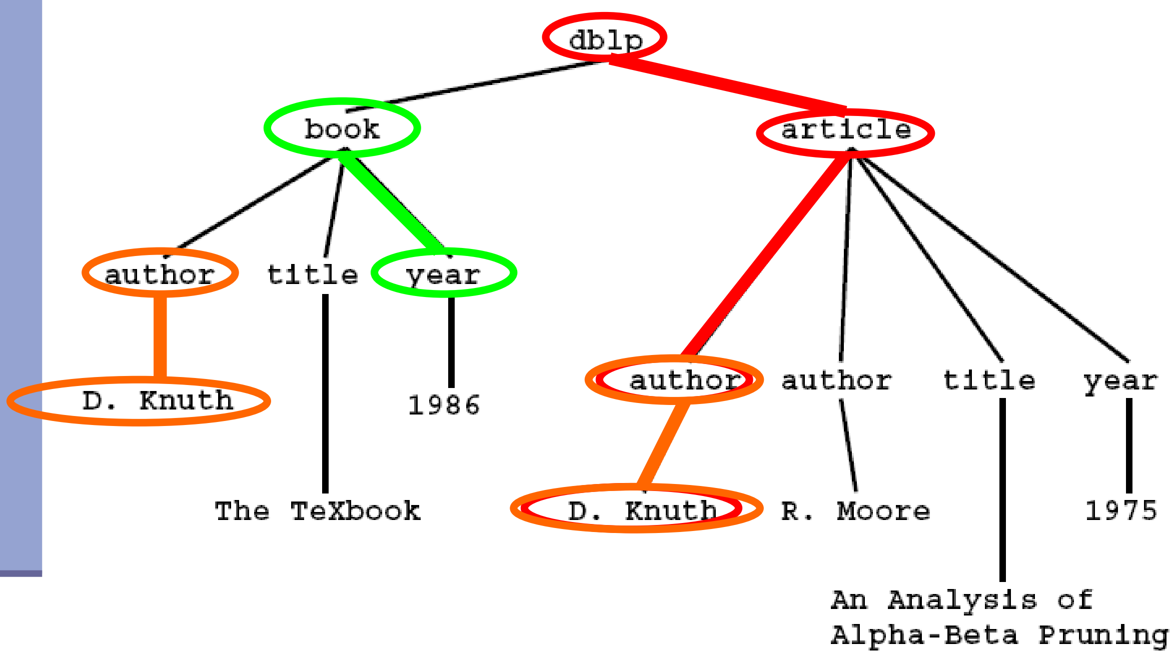
```
<dblp>
    <book>
        <author> Donald E. Knuth </author>
        <title> The TeXbook </title>
        <publisher> Addison-Wesley </publisher>
        <year> 1986 </year>
    </book>
    <article>
        <author> Donald E. Knuth </author>
        <author> Ronald W. Moore </author>
        <title> An Analysis of Alpha-Beta Pruning </title>
        <pages> 293-326 </pages>
        <year> 1975 </year>
        <volume> 6 </volume>
        <journal> Artificial Intelligence </journal>
    </article>
...
</dblp>
```


A key concern: Verbosity...



IEEE Computer, April 2005

A tree interpretation...



- ☑ XML document exploration ≡ Tree navigation
- ☑ XML document search ≡ Labeled subpath searches

Subset of XPath [W3C]

The Problem

We wish to devise a compressed representation for a labeled tree T that efficiently supports some operations:

- ✓ Navigational operations: parent(u), child(u, i), child(u, i, c)
- ✓ Subpath searches: given a sequence Π of k labels
- ✓ Content searches: subpath + substring search
- ✓ Visualization operation: given a node, visualize its descending subtree

- ☑ XML-aware compressors (like XMill, XmlPpm, ScmPpm,...)
  - ➢ need the w
- ☑ XML-queriable
  - ➢ **poor comp**

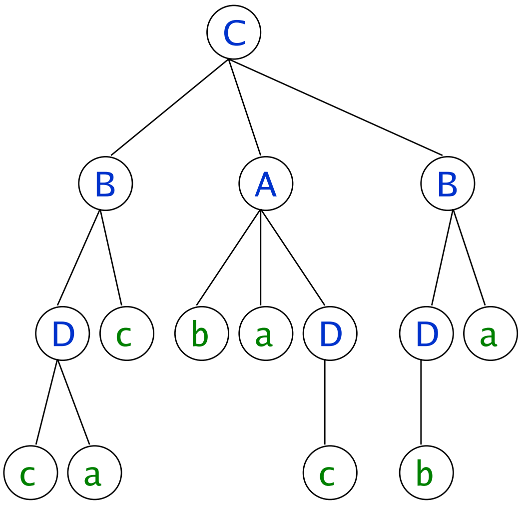
XML-native search engines
need this tool as a core block for
query optimization and (compressed) storage

- ☑ Summary indexes (like Dataguide, 1-index or 2-index)
  - ➢ **large space** and **do not** support "content" searches
- ☑ Theoretically do exist many solutions, starting from [Jacobson, IEEE Focs '89]
  - ➢ **no subpath/content** searches, and **poor performance** on labeled trees

A transform for "labeled trees" [Ferragina *et al*, *IEEE* Focs '05]

- ☑ We proposed the XBW-transform that mimics on trees the nice structural properties of the Burrows-and-Wheeler Trasform on strings (*do you know bzip !?*).
- ☑ The XBW **linearizes** the tree T in **2 arrays** s.t.:
  - ☑ the compression of T ***reduces to*** use any *k*-th order entropy compressor *(gzip, bzip,...)* over these two arrays
  - ☑ the indexing of T ***reduces to*** implement simple rank/select query operations over these two arrays

The XBW-Transform



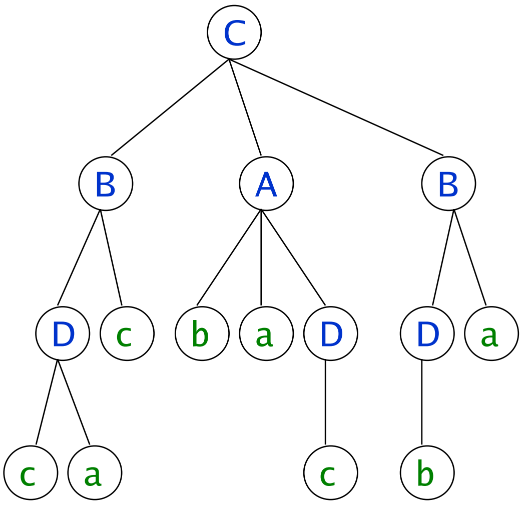
S	S_π
αC	ε
B	C
D	B C
c	D B C
a	D B C
c	B C
A	C
b	A C
a	A C
D	A C
c	D A C
B	C
D	B C
b	D B C
a	B C

Permutation of tree nodes

upward labeled paths

Step 1.
Visit the tree in pre-order.
For each node, write down its label and the labels on its upward path

The XBW-Transform



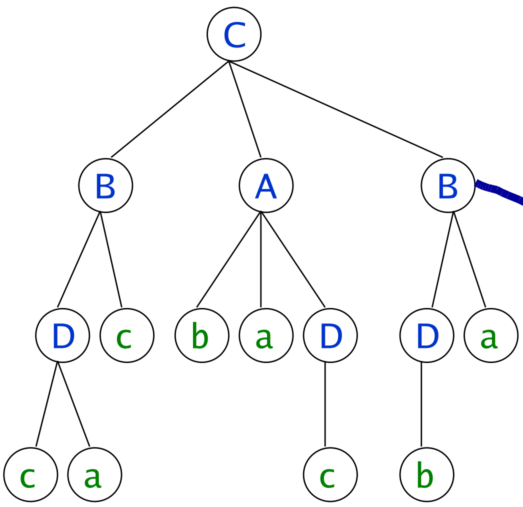
S	S_π
αC	ε
b	A C
a	A C
D	A C
D	B C
c	B C
D	B C
a	B C
B	C
A	C
B	C
c	D A C
c	D B C
a	D B C
b	D B C

upward labeled paths

Step 2.

Stably sort according to S_π

The XBW-Transform



S_{last}	S	S_π
1	αC	ε
0	b	A C
0	a	A C
1	D	A C
0	D	B C
1	c	B C
0	D	B C
1	a	B C
0	B	C
0	A	C
1	B	C
1	c	D A C
0	c	D B C
1	a	D B C
1	b	D B C

XBW

XBW can be built and inverted in *optimal* O(t) time

Step 3.

Add a binary array S_{last} marking the rows corresponding to *last* children

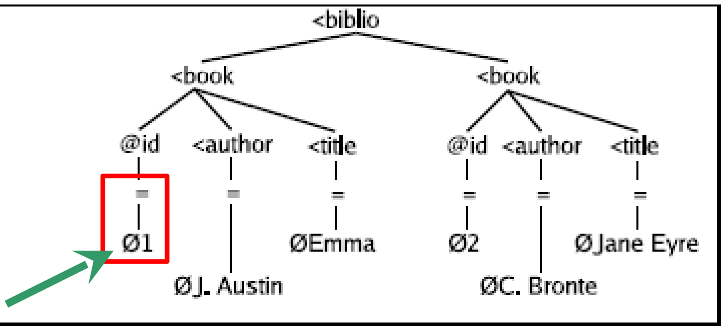
Key fact

Nodes correspond to items in <S_{last},S_α>

XBW takes *optimal* t log |Σ| + 2t bits

XBzip – a simple XML compressor

```
<biblio>
  <book id=1>
    <author>J. Austin</author>
    <title>Emma</title>
  </book>
  <book id=2>
    <author>C. Bronte</author>
    <title>Jane Eyre</title>
  </article>
</biblio>
```



Rk	S_{last}	S_α	S_π
1	1	<biblio	*empty string*
2	1	-	<author<book<biblio
3	1	-	<author<book<biblio
4	0	<book	<biblio
5	1	<book	<biblio
6	0	@id	<book<biblio
7	0	<author	<book<biblio
8	1	<title	<book<biblio
9	0	@id	<book<biblio
10	0	<author	<book<biblio
11	1	<title	<book<biblio
12	1	-	<title<book<biblio
13	1	-	<title<book<biblio
14	1	-	@id<book<biblio
15	1	-	@id<book<biblio
16	[redacted]	ØJ. Austin	-<author<book<biblio
17	[redacted]	ØC. Bronte	-<author<book<biblio
18	[redacted]	ØEmma	-<title<book<biblio
19	[redacted]	ØJane Eyre	-<title<book<biblio
20	[redacted]	Ø1	-@id<book<biblio
21	[redacted]	Ø2	-@id<book<biblio

Tags, Attributes and symbol =

Pcdata

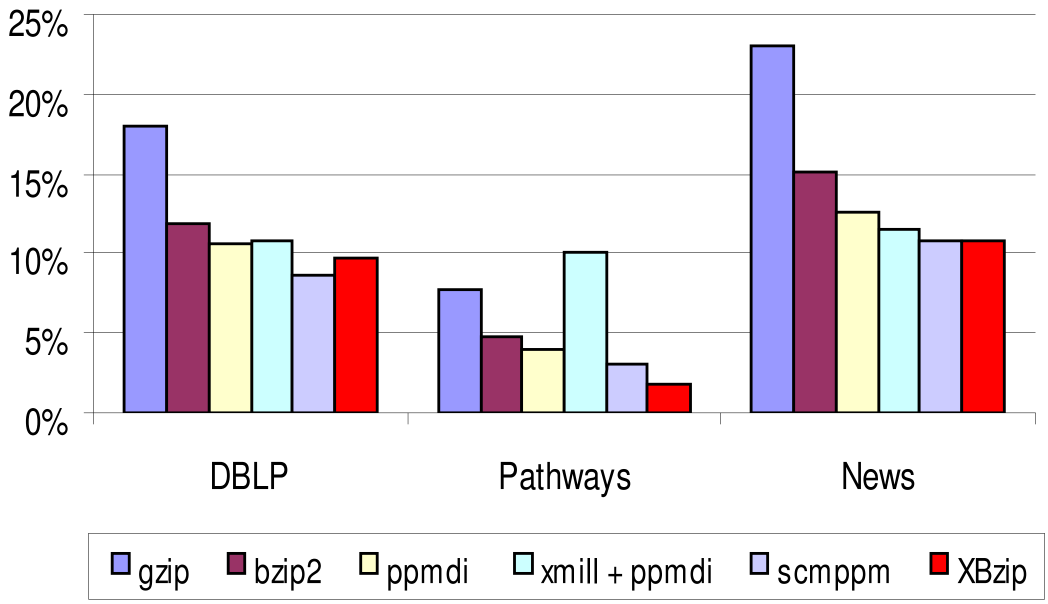
XBW is compressible:

❶ S_α and S_{pcdata} are locally homogeneous

❷ S_{last} has some structure

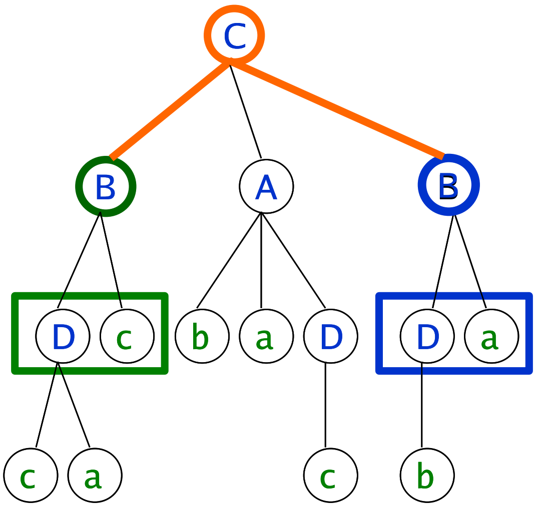
XBzip = XBW + PPMd

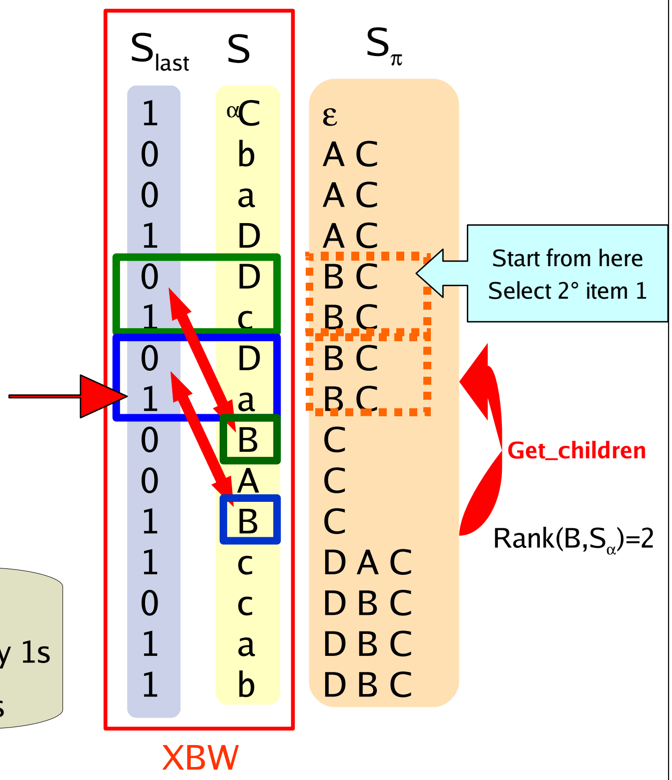
[Ferragina *et al*, WWW '06]



String compressors are not so bad: within 5%

XBW is navigational



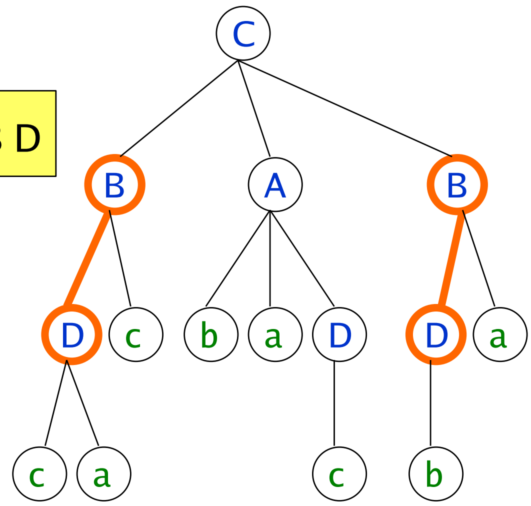


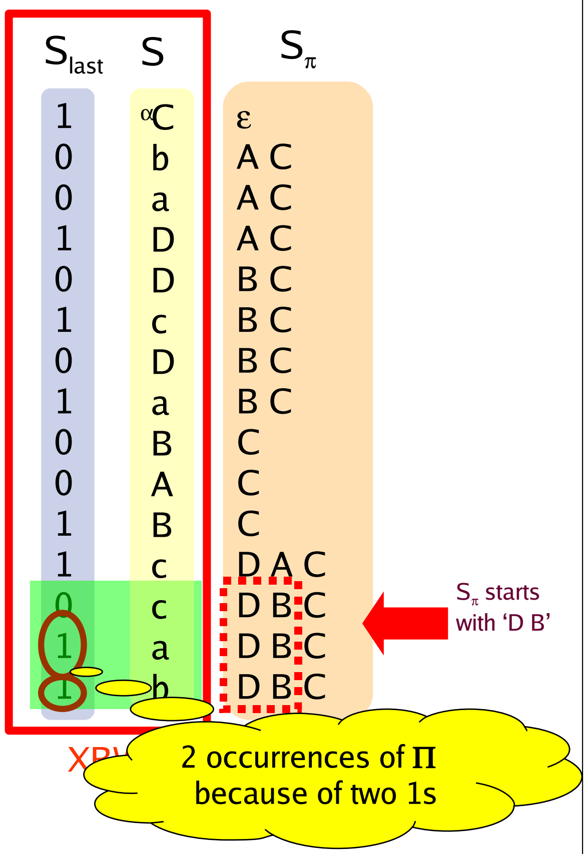
Two useful properties:

- Children are contiguous and delimited by 1s
- Children reflect the order of their parents

XBW is searchable

Π = B D

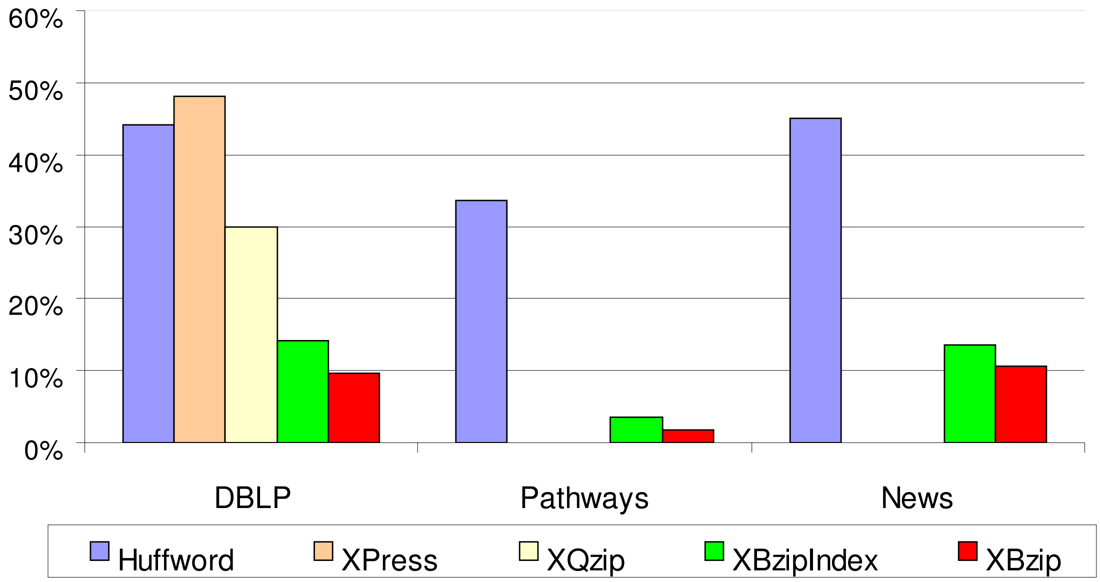




XBW indexing [reduction to string indexing]:

- Store succinct and efficient **Rank** and **Select** data structures over these three arrays

XBzipIndex: XBW + FM-index [Ferragina *et al*, WWW '06]



DBLP: 1.75 bytes/node, Pathways: 0.31 bytes/node, News: 3.91 bytes/node

Upto 36% improvement in compression ratio
Query (counting) time ≅ 8 ms, Navigation time ≅ 3 ms

The overall picture on Compressed Indexing...

Data type

Text

Labeled Tree

This is a powerful paradigm to design compressed indexes:

2. Transform the input in few arrays (via BWT or XBW)
3. Index (+ Compress) the arrays to support rank/select ops

Theory: Soda '06 (2), Cpm '06 (2), Icalp '06 (2), DCC '06 (1)
Experimental: Wea '06 (2)