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

```xml
<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...

XML use is thus increasing rapidly. Analyst Ron Schmelzer with market-research firm ZapThink predicted XML will rise from 3 percent of global network traffic in 2003 to 24 percent by 2006, as Figure 1 shows, and to at least 40 percent by 2008.

However, XML's growing implementation raises a key concern: Because it provides considerable metadata about each element of a document's content, XML files can include a great deal of data. They can thus be inefficient to process and can burden a company's network, processor, and storage infrastructures, explained IBM Distinguished Engineer Jerry Cuomo.
A tree interpretation...

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

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 \( \Pi \) 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 whole decompression

XML-queriable compressors (like XPress, XGrind, XQzip,...)
- poor compression and scan of the whole (compressed) file

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
We proposed the XBW-transform that mimics on trees the nice structural properties of the Burrows-and-Wheeler Transform 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**

**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

Step 2.
Stably sort according to $S_\pi$

Key fact
Nodes correspond to items in $(S_{\text{last}}, S_\alpha)$
rows corresponding to last children

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

XBW takes optimal $t \log |\Sigma| + 2t$ bits
XBzip – a simple XML compressor

XBzip = XBW + PPMd

[Ferragina et al, WWW '06]
XBW is navigational

Two useful properties:
- Children are contiguous and delimited by 1s
- Children reflect the order of their parents

XBW is searchable

XBW indexing [reduction to string indexing]:
- Store succinct and efficient \text{Rank} and \text{Select} data structures over these three arrays

Paolo Ferragina, Università di Pisa
XBzipIndex: XBW + FM-index

[Upto 36% improvement in compression ratio]

Query (counting) time $\approx 8$ ms, Navigation time $\approx 3$ ms

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

The overall picture on Compressed Indexing...

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)