SIXML Version 1.2

Succinct Indexable In-memory Representations for XML Documents

Succinct Indexable XML (SIXML) version 1.2 provides an efficient in-memory representation of large static XML documents, with stable and predictable memory usage, which can be used as a plug-in to create a variety of XML processing APIs. SIXML is based on succinct data structures, which use an information-theoretically minimum amount of space to represent a given data type (see Wikipedia page: http://ira.le.ac.uk/handle/2381/3363).

Representing XML Document

- In main-memory

Case Study: Xerces-C 2.8

In Xerces-C the DOM tree is represented using parents, see Fig. 1 which shows the 4 pointers required as minimum per node in the DOM tree. Typically node types representations such as DOM Element and Node Text require 413 and 2716 bits per node, respectively. The DOM implementation gives a robust API for the DOM tree, at the memory usage cost due the use of parents.

Memory Usage

<table>
<thead>
<tr>
<th>File</th>
<th>Size</th>
<th>SIXDOM</th>
<th>Xerces-C</th>
<th>JAXON</th>
<th>SIXDOM- XML</th>
</tr>
</thead>
<tbody>
<tr>
<td>Chair.c</td>
<td>1PB</td>
<td>32%</td>
<td>453%</td>
<td>153%</td>
<td>12%</td>
</tr>
<tr>
<td>Linenom.xml</td>
<td>12MB</td>
<td>28%</td>
<td>399%</td>
<td>164%</td>
<td>15%</td>
</tr>
<tr>
<td>XCDOM.xmi</td>
<td>607MB</td>
<td>50%</td>
<td>491%</td>
<td>100%</td>
<td>14%</td>
</tr>
</tbody>
</table>

Memory usage typically less than 50% of the size. SIXDOM MX compresses the text, space usage is even better.

Running times

In the paper [EBTT 08], we provide a comprehensive performance test:

- Test: Full navigation of document, retrieving all text nodes
- Results: SIXDOM 1.1 was ~1.8 times slower than Xerces-C DOM

SIXDOM benefits:
- Very low memory footprint
- Good for mobile devices
- Fast processing
- Query-friendly

SIXML 1.2

Fast Parsing,

Plug-in to XML Processing APIs,

Highly space efficient,

Fast navigation and data access.

Pointerless Data Structure

Is there a succinct ("pointerless") in-memory representation of XML documents, which can give the full XML processing functionality?

We can represent the tree in Figure 1.2 as a parentheses string:

- Document order numbering
- Very fast navigation using 2.6 bits/node
- [BRF06, WE06, EDBT 08]

Using parentheses string and other "success" building blocks we are able to represent 1000s in < 4 bits per node, rather than the minimum 2.9 bits in Xerces-C

SIXDOM 1.1

Provides a highly efficient in-memory representation of XML documents in main memory. Features:

- DOM API (Level 2 and partial 3)
- A rich library interface
- C++

We refer to SIXML with a DOM API. This we call SIXDOM 1.1

Forthcoming SIXML 1.3

Released Summer 2013

Features:

- Fast memory efficient parsing. Using the Expat parser (creation of James Clark). Very low memory footprint
- Cross platform support to languages such as Java and C# Net
- Release XML processing APIs based on Succinct data structures

To maximise the potential of SIXML we are interested in Industrial support and partnerships

O'Neil Delpratt
onl1@mcs.le.ac.uk

(Joint work with Rajeev Raman and Naila Rahman)

Website under construction: http://www.cs.le.ac.uk/SIXML

University of Leicester

xmlprague