PPARC UK e-Science Postgraduate School '05

A In-Memory Compressed XML Representation of Astronomical Data

O'Neil Delpratt – PhD Student
University of Leicester

Supervisors:
Rajeev Raman, Clive Page, Tony Linde &
Mike Watson – University of Leicester

Astro Grid

University of Leicester

Introduction

- Topics:
  - Outline of Project
  - Role in Project
  - Current Work
  - Conclusion

Outline of Project

AstroGrid Project

Building a data-grid for UK astronomy.

Consist of a wide variety of web/grid services, serving up data to be combined and processed, and ultimately displayed to the user.

- Astronomers Previously used FITS binary.
- International Virtual Observatory Alliance (IVOA) Introduced VOTable xml format.

Concern:

VOTable is designed as a flexible storage and exchange format for tabular data, with emphasis on astronomical tables.

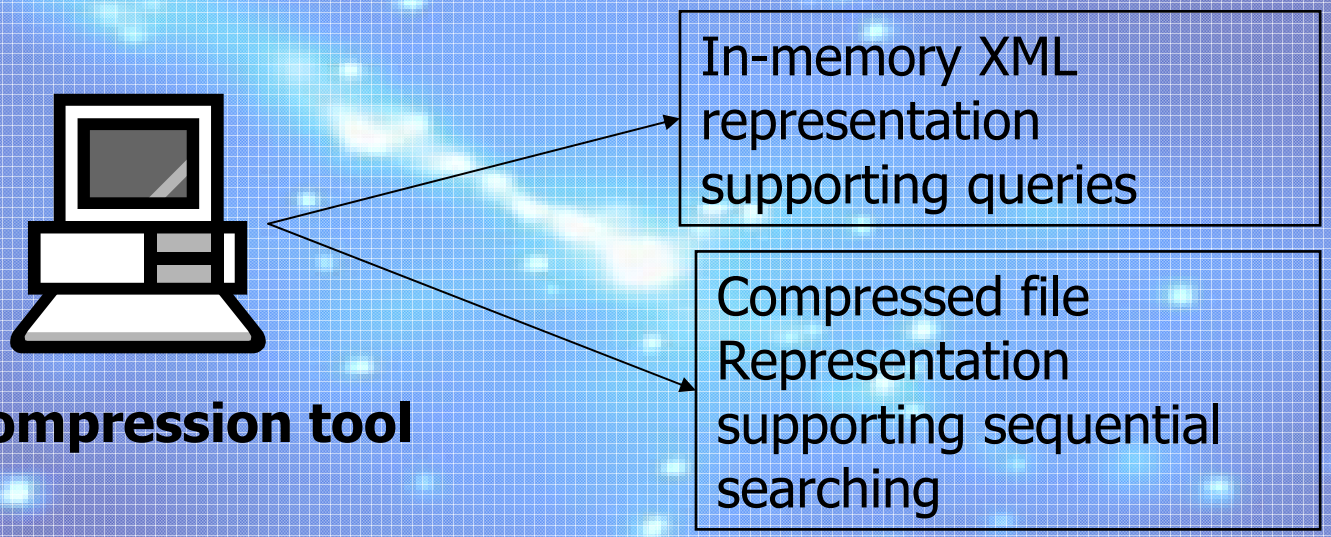
Using the VOTable XML-based astronomical data format we know that it:

- Is Basic (XML representation of a 2d binary table.).
- Has Static Data (features for big-data and grid computing)
- Allows data to be stored separately, with the remote data link

Main Problem

- XML-based files are larger than equivalent formats,
- Network bandwidth is limited.
- Adhoc approach
- Slow searching

Role in Project



- Provide software tool for the Astrogrid Group.
- Remove the adhoc approach of accessing astronomical data in XML documents.
- Remove the limitation of parsing large XML documents, allowing us to make ‘proper’ flexible XML files rather than representation of 2D tables.

An existing XML-specific compression tools are:

- XMILL *[H. Liefke and D. Suciu, 2000]*

  Files are inaccessible until uncompressed.

Therefore we need to develop a software tool that provides both:

- Compression
- Powerful operations on the compressed XML documents.

Requirements

- Produce a software tool for efficient storage and processing of large XML files that arise in the context of IVOA.
- Remove adhoc way of accessing XML data.
- Provide efficient searching facilities of VOTable files.
- Rapid querying operations.
- Reduce bandwidth.
- Allow for merging of XML files.

Current Work

- Topics:
  - Succinct DOM
  - Using PrefixSum Data Structures in DOM
  - Experimental Results

Succinct DOM

- DOM is a specification for representing XML documents in main memory.
- XML file is parsed and represented using our succinct DOM implementation

What is Succinctness?

A *succinct* data structure is one that uses space approaching the information theoretic minimum for the required structure yet ideally supports the desired operations in constant time,

unconstrained by the size of the structure.

Components of Succinct DOM

Parenthesis Data Structure:

```
<Directory>
 <Name>Basic</Name>
 <File type="XML">
  <FName>Basic</FName>
  <Created>
   <Time>
    <Hour>15</Hour>
    <Minute>00</Minute>
   </Time>
   <Date>10/10/05
   </Date>
  </Created>
  <Note>Files details</Note>
 </File>
 <Note>XML files</Note>
</Directory>
```

- Currently using pointers
- Parenthesis we have 2 bits per node

(( ) ( ) (( ) ( )) ( )) ( )) ( ))

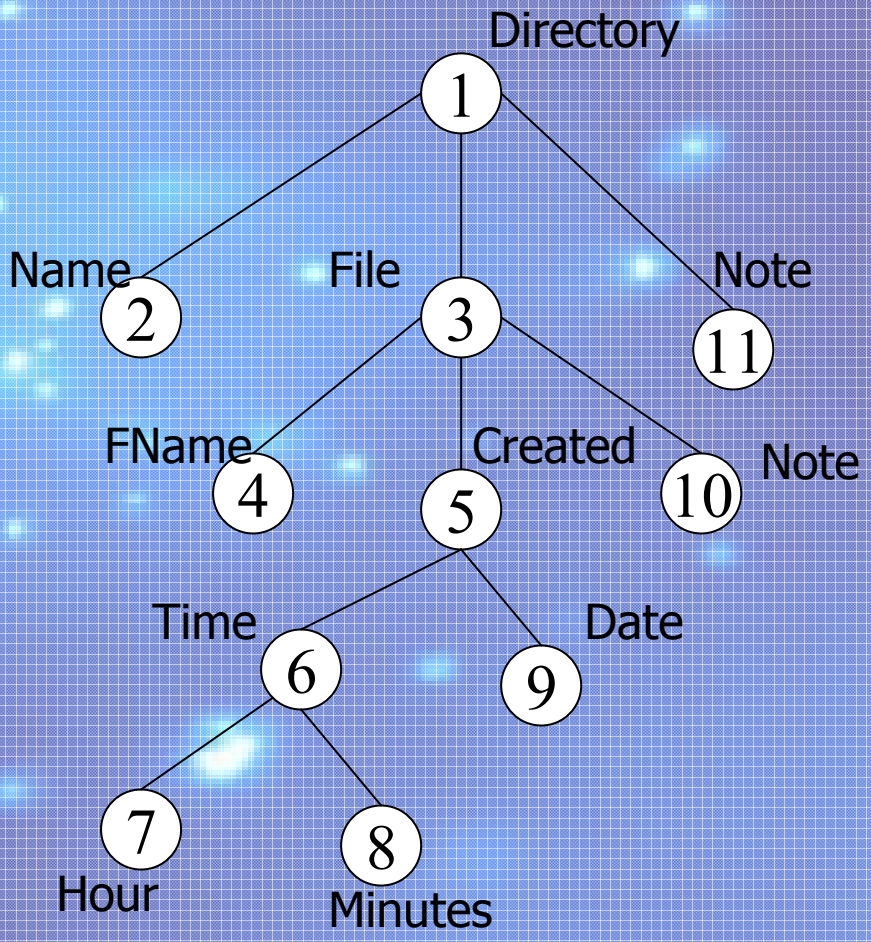
1 2 3 4 5 6 7 8 9 10 11

As a BitVector:

00100100010110111011

[Geary, Rahman, R. Raman, V Raman. 2004]

Tree Representation

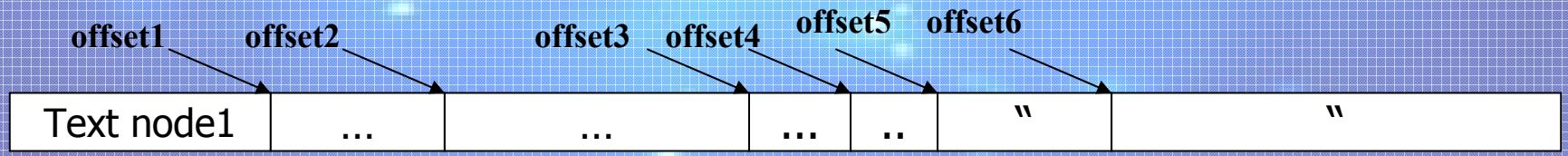


Components of Succinct DOM

PrefixSum Data Structure Motivation:

We can store the text node data from XML files as a concatenated string.

Store the offsets for each text node data in an integer array.



Problem Here:

- The text nodes often take most of the XML tree, sometimes 50%.
- Therefore, the offsets as 32-bit integers in memory would not be space efficient.

Components of Succinct DOM

PrefixSum Data Structure

Given a sequence of k numbers, $n_1,\ldots,n_k$, where $n_i > 0$, for all i, the *prefixsums* for the sequence are $d_1,\ldots,d_k$, where $d_1 = n_1,\ d_2 = n_1 + n_2,\ldots,\ d_k = n_1 + \ldots + n_k$.

The requirements for the PrefixSum Data structure to represent with an Abstract Data Type:

```
ADT PrefixSum is
      ABSTRACT DATA
         a non-negative integer k and k positive integers
         n_1,...,n_k
      OPERATIONS
         Constructor(n_1,...,n_k)
            initialises k and n_1,...,n_k
         Sum(i)
            returns the value n_1+...+n_i, where 1<=i<=k
END
```

PrefixSum Data Structure Continued...

There are existing integer-coding methods that we can use to encode the number 1,…,x compactly.

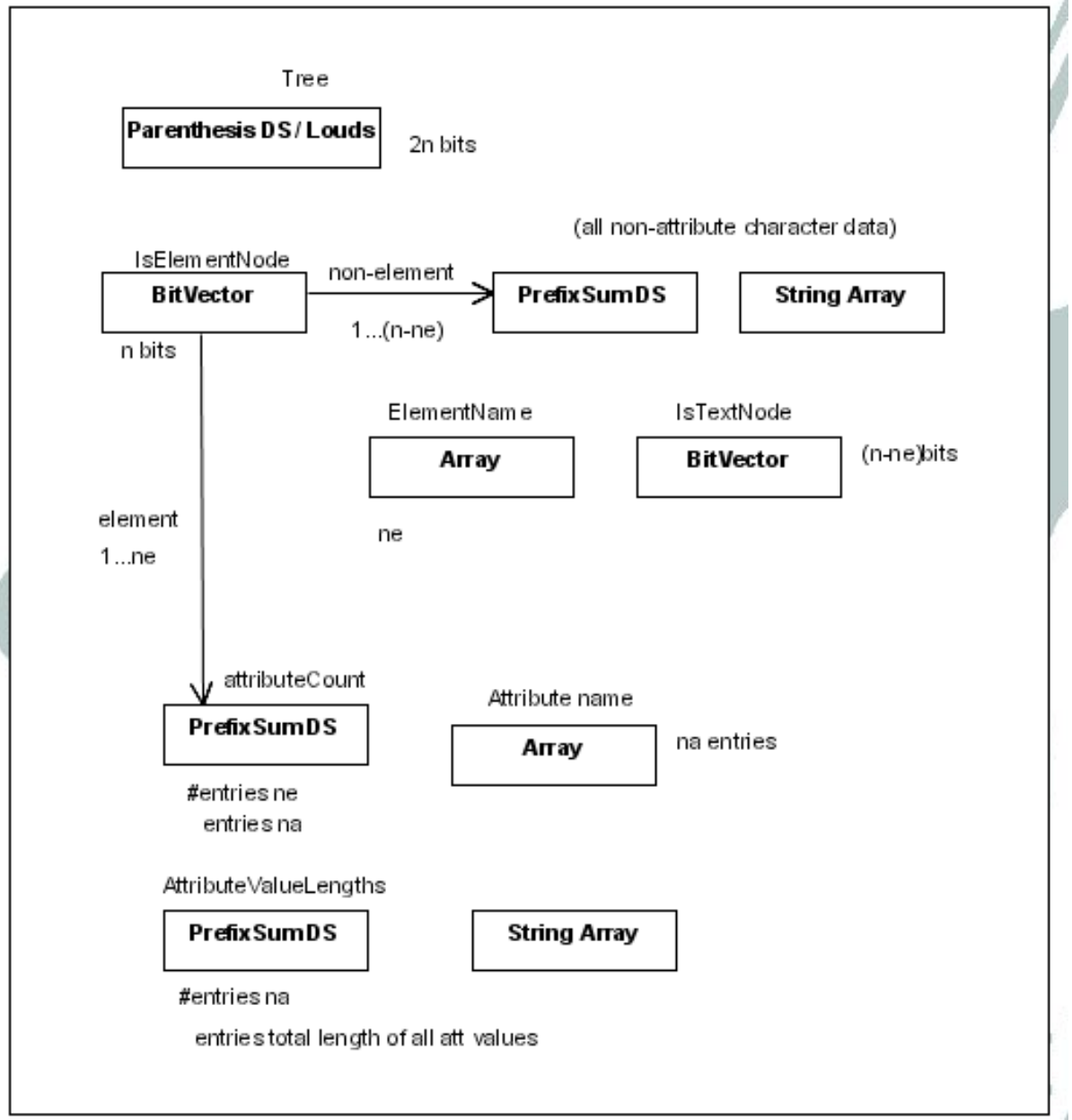
We consider three integer-coding methods that are used in inverted file compression:

- Gamma Code – requires $1+2\lfloor \log x \rfloor$ bits.
- Delta Code –requires $1+2\lfloor \log\log 2x \rfloor+\lfloor \log x \rfloor$ bits.
- Golomb – requires $\lfloor (x-1)/b \rfloor+\log b$ bits.

Components of Succinct DOM

Some Of the DOM methods we will implement:

- childNodes
- parentNode
- Attributes
- nextSibling
- previousSibling
- nodeType
- nodeName
- nodeValue



Experimental Results

Files	Original Size	Bzip2	Succinct DOM
UNSPSC-2.04.XML	1.02MB	107KB	294KB
Mondial-3.0.xml	1.70MB	107KB	324KB
Orders.xml	5.12MB	285KB	1,159KB
xCRL.xml	8.29MB	256KB	1,029KB
Votable2.xml	15.10MB	1,622KB	4,335KB
Nasa.xml	23.80MB	2,688KB	8,279KB
XPath.xml	49.80MB	1,323KB	7,437KB

Challenges that remain

- Improve the results of the succinct DOM and that of the Prefix-sum data structures.
- Challenges to bring space usage closer to bzip2.
- Develop a compressed file and in-memory representation.
- Relate the XML representation data structures to VOTable files for further compression improvements.