Technical Aspects of XML Format – Case Study
Differences between Saving Data into Element and Attribute

Ondřej Bureš
Faculty of Informatics and Management, University of Hradec Králové, Rokitanského 62, Hradec Králové, Czech Republic

Keywords: XML, Data, Java, PHP, Visual Basic.

Abstract: XML technology is used for data transmission on daily basis. While processing huge files, every millisecond per node can play its part in the process. In total time it can lead to extending the whole procedure by minutes or even tens of minutes which can make it very ineffective in matter of time and cost. Goal of this study is to put saving data into elements in contrast with saving into attributes of XML files and compare final results. In order to receive the best overview, three applications in different selected programming languages were tested and results were compared.

1 INTRODUCTION

XML data transmission is based on many steps. It starts with creating data, which is mostly printed directly from relational database Speed of this step is limited only by actual database settings. Next step means sending or downloading data file from exporter to importer. Time spent on this step is determined by connection speed therefore network settings. In final destination file is processed by a XML parser and then again sent into database or directly to frontend of some web portal.

Let's focus on data parsing. Assuming that XML is valid (Grijzenhout and Marx, 2013) it can contain basically unlimited count of nodes, a single millisecond can cause a lot of time lapse during the file processing. In case of 60 thousand nodes in file, one millisecond per each can lead up to one minute delay. If we had couple of this sized files, simple math can tell us how big delay would we get during parsing.

This consideration can make us wonder. What if there is a difference in time required for parsing elements and attributes which would cause delay while reading big XML file containing hundreds thousands nodes or even millions of them. Finding out that there is difference in data saving approach could save many resources for companies in which XML data transmission is one of the key processes.

2 TESTING CONDITIONS

In order to be able to test our assumption, we need to set up an environment with equal conditions. That way we are able to get objective results. Nowadays applications are written and created in many different programming languages. It would be the best to test them all, but in this phase of study we will settle with sample of the most popular and the most used languages.

2.1 Programming Languages

TIOBE programming community is doing monthly statistics and creating list of most popular programming languages (TIOBE Software BV, 2014). These statistics are based on number of skilled engineers world-wide, courses and third party vendors. Results are calculated using popular search engines such as Google, Bing Yahoo!, Wikipedia and many others.

It's not a chart of either the best programming languages or those in which most lines of code have been written. Main purpose of this chart is overview which should serve for programmers to check if their skill are still up to date. In spite of the fact, we will use this index to determine which languages are the most commonly used for creating applications.

Bureš O..
Technical Aspects of XML Format – Case Study - Differences between Saving Data into Element and Attribute.
DOI: 10.5220/0005154503540359
In *Proceedings of the International Conference on Knowledge Management and Information Sharing* (KMIS-2014), pages 354-359
ISBN: 978-989-758-050-5

Table 1: TIOBE index for June 2014 (http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html).

#	Language	Rating
1	C	16.191%
2	Java	16.113%
3	Objective-C	10.934%
4	C++	6.425%
5	C#	3.944%
6	(Visual) Basic	3.736%
7	PHP	2.848%
8	Python	2.710%
9	Javascript	2.000%
10	Visual Basic .NET	1.914%

Out of programming languages mentioned above, this study includes testing in applications written in three of them, which are Java, Visual Basic and PHP. Unfortunately we don't have C or any of its modifications on the list, but for needs of case study it is not so important.

Each programming language has different conditions to be able to run it locally on machine using Windows operating system.

In order to be able to run Java program, it is important to have Java Development Kit (also known as JDK) installed. This package is provided directly on Oracle website.

Visual Basic requires Internet Informational Services server (also known as IIS) which is usually part of Windows OS default installation. All what user needs is to enable this service on control panel as a feature.

PHP programming language requires Apache server to be installed. There are many solutions that avoid whole process of installing and setting up whole server, one of them is EasyPHP package that was used during tests.

2.2 Testing XML File

It is important to give both elements and attributes the same starting conditions in order to achieve comparable values. As a starter, files for testing elements and for testing attributes must be same sized. Only that way we'd avoid time difference caused by loading file.

Next important feature is string length. We need to be sure to avoid time differences while loading strings of different lengths.

We will achieve both objectives by using only one file in which we will have both elements and attributes while in one row we have same values for both element and attribute.

```
<item param="32781.00">32781.00
</item>
```

Values are generated randomly so we would avoid any caching issues. XML also needs to be fully valid, so we will include proper header at the beginning of the file:

```
<?xml version="1.0" encoding="utf-8" ?>
```

Next we will try to figure out if file size matters during the whole process. That's why three different files were created. The first one contains one hundred thousand records, second contains five hundreds thousands and the third one contains one million records.

Table 2: File sizes.

Records count	File size (bytes)
100 000	3 898 783
500 000	19 206 579
1 000 000	38 495 351

2.3 Testing Environment

As last parameter we have to mention, that all three languages were tested with the same computing power. Only that way we are able to compare every application with each other. Configuration of the used computer is 2 GHz dual core Intel processor and 2 GB memory operated by OS Windows 7 Professional edition.

3 TESTING APPLICATIONS

All of programming languages selected for testing differences between parsing elements and attributes have its own XML parser in default so there is no need in installing any additional libraries.

3.1 Application in PHP

PHP in its default settings uses memory of only 128 megabytes, which is not enough for processing file containing one million records. Therefore we need to enlarge this parameter to 1024 megabytes using designated command right inside the application:

```
ini_set("memory_limit","1024M")
```

PHP itself has XML parser called SimpleXML and the source code for processing elements in the file is following:

```
  $source = simplexml_load_file
('export_1m.xml');
  $items = $source->xpath
("/root/item");
  foreach ($items as $item) {
  $array[$i] = (string)$item;
  $i++;
  }
```

We are saving every line into array just to be sure that this line is processed. Script for parsing data from `param` attribute looks very similar:

```
  $source = simplexml_load_file
('export_1m.xml');
  $items = $source->xpath
("/root/item");
  foreach ($items as $item) {
  $array[$i] = (string)$item['param'];
  $i++;
  }
```

Start time and end time of application is tracked with function `microtime()`, which is called before loading XML file and after loading the last node in XML file.

3.2 Application in Java

According to TIOBE index, this programming language was the most popular world-wide until year 2012. Java uses 256 megabytes of memory in default, which is sufficient for only around 150 thousands records. Therefore it is also required to enlarge memory limit up to 1024 megabytes to be limitless in our testing using Xmx parameter.

Java has its default XML parser called XPath API which is part of basic Java package since version 5, but thanks to its popularity, wide variety of libraries extending and improving work with XML files is available all over community forums.

Source code of application processing elements is following:

```
  XPath xpath =
XPathFactory.newInstance().newXPath();
  NodeList nodes = (NodeList)
xpath.evaluate("/root/item/text()";,
new InputSource('export_1m.xml'),
XPathConstants.NODESET);
  int size = nodes.getLength();
  String[] valueArr = new String[size];
  for (int i = 0; i < size; i++) {
  valueArr[i] =
nodes.item(i).getNodeValue();
  }
```

And with slight modification we get source code of application which parses attributes of given XML file:

```
  XPath xpath =
XPathFactory.newInstance().newXPath();
  NodeList nodes = (NodeList)
xpath.evaluate("/root/item/@param", new
InputSource('export_1m.xml'),
XPathConstants.NODESET);
  int size = nodes.getLength();
  String[] valueArr = new String[size];
  for (int i = 0; i < size; i++) {
  valueArr[i] =
nodes.item(i).getNodeValue();
  }
```

Duration of running application is in this case monitored using function nanoTime() which is again called twice, once before file is loaded and nodes parsed and once after whole process is finished.

3.3 Application in Visual Basic

The youngest of all used languages is Visual Basic developed by Microsoft Company. According to the TIOBE index this language is losing its popularity since year 2010.

Unlike the two already mentioned languages, this one does not have any memory limitation in default, so there is no need for initial settings modification. Application used for parsing elements has following source code:

```
  xml.LoadXmlFile('export_1m.xml')
  Dim item = xml.FirstChild()
  While Not (item Is Nothing)
  Dim value As String = item.Content
  item = item.NextSibling()
  End While
```

While again with slight modifications we get an application which parses attributes from XML file.

```
  xml.LoadXmlFile('export_1m.xml')
  Dim item = xml.FirstChild()
  While Not (item Is Nothing)
  Dim value As String =
item.GetAttrValue("param")
  item = item.NextSibling()
  End While
```

Time required for code execution is in case of Visual Basic tracked with function `now().Ticks`.

4 RESULTS

While running every application in testing mode, each returned different time results for every try of processing XML file. Because of this observation every application ran 10 times for both cases meaning parsing elements and attributes.

4.1 Results of PHP Application

For the first language we got quite interesting results. Values are in milliseconds:

Table 3: Values collected from PHP application.

	Element	Attribute
100 000	397.7	468.3
500 000	1900.4	2278.5
1 000 000	3800.0	4633.6

From initial view we can see that element processing hundred thousands of elements is 70 milliseconds faster than processing same number of attributes. For a better overview data collected from all three counts of nodes in XML file were counted to hundred thousand and put into a graph.

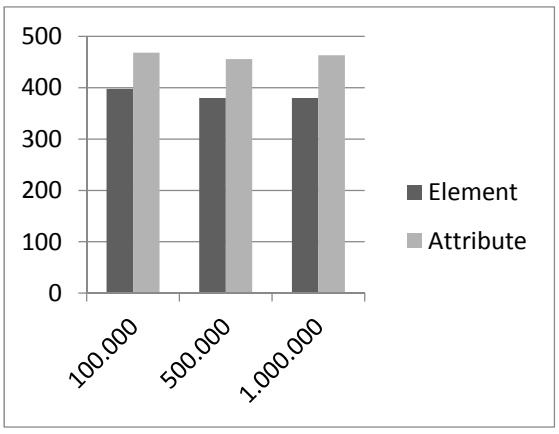

Figure 1: Graph of values collected from PHP application. On y axis duration of processing 100 000 elements, on x axis number of nodes in tested file.

We can see that in PHP results are approximately the same regardless number of nodes in the file, but it is faster to process element that it is to process attribute. While reading results from the table, it is obvious that while parsing a million of nodes, difference is almost one second.

4.2 Results of Java Application

While testing in Java, we got surprising results. First we take a look on the table with averaged values.

Table 4: Values collected from Java application.

	Element	Attribute
100 000	2211.3	2346.8
500 000	30044.2	30362
1 000 000	114047.3	111374.8

Right now we can see that time required for processing million records is extreme. It takes almost two minutes and even processing of 100 thousands nodes takes much longer than it does in PHP (almost 6x in numbers).

This is caused by constant calling garbage collector even when memory limit is set higher than default. If we tried only about two thousands nodes, we would get almost the same time per node as for one hundred thousand, but with higher greater number of nodes time increases exponentially.

As mentioned during introducing Java application, thanks to popularity of this language there are alternatives for XML parsing. One of the most suggested is VDT-XML library. In order to be completely honest with all programming languages and to get the best and the most reliable results, we tried testing also using this library while getting following results:

Table 5: Values collected from Java application using VDT-XML library.

	Element	Attribute
100 000	460.9	455.7
500 000	1377.2	1395.9
1 000 000	2795.2	2555.1

We can see that time required for processing improved a lot. Again we count collected values for 100 thousands nodes and put them into graph for better overview.

Interesting fact is that the more nodes we have, the faster processing time per node is. It means that application written in Java takes significant time just for opening and loading the file. This time is constant and the more nodes file has, the lower average value of this time is.

Unlike PHP application, in case of Java there is no such big difference in time required for processing element and for attribute. At count of 500 thousands the time is almost the same for both (275.44 milliseconds per hundred thousand elements

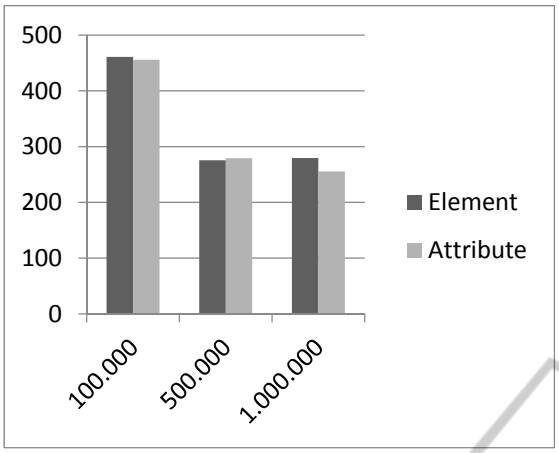

Figure 2: Graph of values collected from Java application using VDT-XML library. On y axis duration of processing 100 000 elements, on x axis number of nodes in tested file.

and 279.18 milliseconds per hundred thousand attributes).

4.3 Results of Visual Basic Application

Using this programming language got us notable results as well.

Table 6: Values collected from Visual Basic application.

	Element	Attribute
100 000	1161.7	1684.2
500 000	5820.0	8216.0
1 000 000	10398.6	15570.0

Just before using graphical data representation we can see that processing XML in Visual Basic is the slowest out of all tested languages. Now let's take a look on a graph.

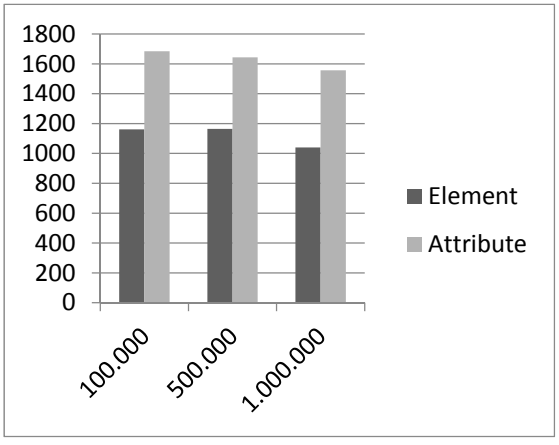

Figure 3: Graph of values collected from Visual Basic application. On y axis duration of processing 100 000 elements, on x axis number of nodes in tested file.

In this case it is obvious, that processing element is much faster than processing attribute regardless number of nodes inside the file. As well as in Java application, average values are decreasing per number of nodes inside XML file, which again means that application written in Visual Basic also needs some time for opening and loading whole file, but it is not as remarkable as it was while using Java application.

5 CONCLUSION

As we found out, there is almost no difference using Java language and only a minor difference using PHP in behalf of elements. While using these two languages we don't need to care that much about whether to save data into elements or attributes.

On the other hand time difference while using Visual Basic language is obvious. Processing attribute takes much more time than processing element, it is around 1/3 of total time regardless count, which can lead up to big delay while processing big file or more smaller files.

If we were exporters and were completely sure about language used by data importer, we could take different values per programming languages from this study in consideration, but when it is very likely that our data importers may vary in using technology, it is best to provide them with most of data saved into elements, because it can lead to faster data processing, therefore better cooperation between companies, at least in the area of data transmission.

At this point it might be very interesting to do such a research for other languages as well, at least for the most popular modifications of C language to find out whether results in Visual Basic are just some anomaly or not.

ACKNOWLEDGEMENT

This work was supported by the project No. CZ.1.07/2.2.00/28.0327 Innovation and support of doctoral study program (INDOP), financed from EU and Czech Republic funds.

REFERENCES

Bureš, O. (2014) 'Comparing suggested approaches for XML design with current situation in Czech Republic

(case study).‘ *Proceedings of the 23rd International Business Information Management Association 2014*, 481-487

Department of Interior of Czech Republic (2009) *The methodology for creating XML schemas in information systems of public administration.* Available at: http://www.mvcr.cz/clanek/metodika-tvorby-xml-schemat-v-oblasti-informacnich-systemu-verejne-spravy.aspx

Grijzenhout, S. and Marx, M. (2013) ‘The quality of the XML Web.‘ *Journal of web semantics*, 19, 59-68

Ogbuji, Uche (2004) ‘*Principles of XML design: When to use elements versus attributes.*’ Available at: http://www.ibm.com/developerworks/xml/library/x-eleatt/index.html

TIOBE Software BV (2014) *TIOBE Index for June 2014*. Available at: http://www.tiobe.com/index.php/content/paperinfo/tpci/index.html (Accessed: 20 June 2014)

Walmsley, Priscilla (2012) *Definitive XML Schema*. Prentice Hall PTR