Name: Sihang Wang

Course: DSCI 551

Assignment: Week 5 Summary

Date: February 14, 2025

Week 5 Summary

This week's topics are focusing on the reaming contents of HDFS and LXML/XML. In first lecture, we talked about some very important operations in HDFS which is getBlockLocations(), create(), append(), addBlock(), readBlock(), WriteBlock(), copyBlock(), replaceBlock(). Here are some breakdowns for the important notes:

Phase 1: Client Interacts with the NameNode

This phase involves metadata operations where the client communicates with the NameNode to get information about data blocks.

Reading Data (Fetching Block Locations)

1. The client wants to read a file.
2. It contacts the NameNode by calling getBlockLocations() and provides:
    - File name
    - Offset (starting position of the data)
    - Length (amount of data to read)
3. The NameNode responds with:
    - A list of block locations (which DataNodes store the file's blocks).
    - The replication information (e.g., 3 copies of each block).

Writing Data (Allocating Blocks)

1. The client first creates or appends to a file by calling create() or append().
2. It requests new block allocation by calling addBlock().
    - The NameNode assigns a new block.
    - It also provides the list of DataNodes where the block should be stored (to maintain replication).

Phase 2: Client Interacts with DataNodes

In this phase, the actual data transfer occurs between the client and DataNodes.

Reading Data (Retrieving Blocks)

1. The client connects to the nearest DataNode from the block locations provided by the NameNode.
2. It requests the block data using readBlock().
3. If the selected DataNode is slow or unavailable, the client automatically switches to another DataNode storing a replica.

Writing Data (Storing Blocks)

1. The client connects to the first DataNode (as provided by the NameNode).
2. It writes data to the first DataNode using writeBlock().
3. The first DataNode forwards the block to the second DataNode, which then sends it to the third DataNode (if replication factor = 3).
4. Once all DataNodes acknowledge successful writing, the process is complete.

For LXML and XML content

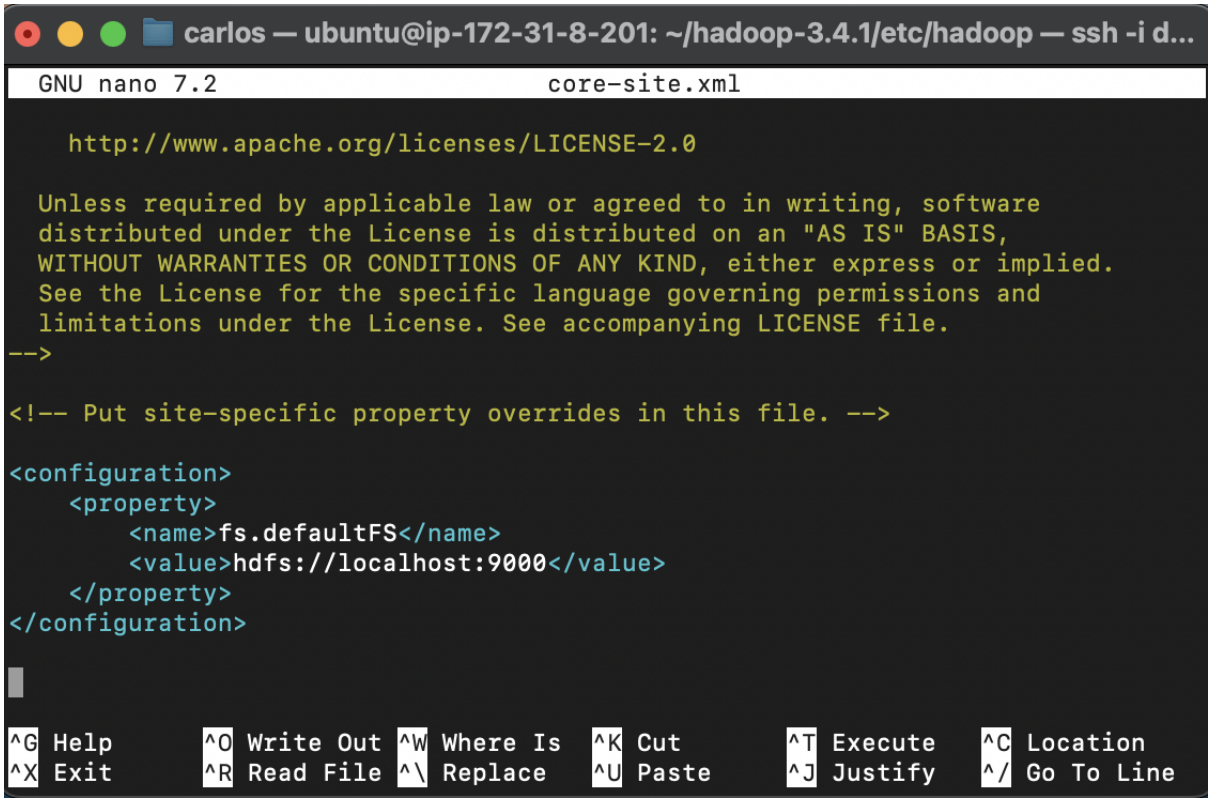
XML is a syntax (serialization format) for data. Important noticed that XML document has a single root element, and elements names are case-sensitive. XML structure is an ordered tree, and it is self-describing and semi-structured data. The index of XML starts at 1.

For Xpath, the lecture introduces lots of functions. We need to know functions such substring, starts-with, substring-before, and substring-after

Function	Description	Example	Result
substring(string, start, length)	Extracts a substring from string, starting at start (1-based index) for length characters.	substring('XPath Functions', 7, 4)	"Func"
substring(string, start)	Extracts a substring from string, starting at start (1-based index) to the end.	substring('XPath Functions', 7)	"Functions"
starts-with(string, prefix)	Checks if string starts with prefix, returns true() or false().	starts-with('XPath Functions', 'XPath')	true()

contains(string, substring)	Checks if string contains substring, returns true() or false().	contains('XPath Functions', 'Func')	true()
substring-before(string, delimiter)	Returns the substring before delimiter.	substring-before('XPath Functions', ' ')	"XPath"
substring-after(string, delimiter)	Returns the substring after delimiter.	substring-after('XPath Functions', ' ')	"Functions"
string-length(string)	Returns the length of string.	string-length('XPath')	5

Technical showcases



```
>>> printf(tree.xpath('/bib/book/author[starts-with(., "J")]'))
<author>Jeffrey D. Ullman</author>

>>> printf(tree.xpath('/bib/book/author[substring-before(., "J")]'))
>>> printf(tree.xpath('/bib/book/author[substring-before(., "e")]'))
<author>Serge Abiteboul</author>

<author>Jeffrey D. Ullman</author>

>>> printf(tree.xpath('/bib/book/author[substring-before(., "e") = "S"]'))
<author>Serge Abiteboul</author>

>>> printf(tree.xpath('/bib/book/author[substring-after(., "ma")]'))
<author>Jeffrey D. Ullman</author>

>>> printf(tree.xpath('/bib/book/author[substring-after(., "ma") = "n"]'))
<author>Jeffrey D. Ullman</author>

>>>
```

```
>>> printf(tree.xpath('/bib/book/author[substring-before(., "J")]'))
>>> printf(tree.xpath('/bib/book/author[substring-before(., "e")]'))
<author>Serge Abiteboul</author>

<author>Jeffrey D. Ullman</author>

>>>
```

carlos — ubuntu@ip-172-31-8-201: ~/class — ssh -i dsci551.pem ubuntu...

```
<book price="55">
        <publisher>Freeman</publisher>
        <author>Jeffrey D. Ullman</author>
        <title>Principles of Database and Knowledge Base Systems</title>
        <year>1998</year>
</book>

<book>
        <title>xyz</title>
        <author age="25"/>
</book>

>>> printf(tree.xpath('/bib/book/author/age'))
>>> printf(tree.xpath('/bib/book/author/@age'))
20
25
>>> printf(tree.xpath('/bib/book[contains(author, "ull")]/title'))
>>> printf(tree.xpath('/bib/book/author[contains(.,"ull"]/title'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "src/lxml/etree.pyx", line 2342, in lxml.etree._ElementTree.xpath
  File "src/lxml/xpath.pxi", line 342, in lxml.etree.XPathDocumentEvaluator.__ca
ll__
  File "src/lxml/xpath.pxi", line 210, in lxml.etree._XPathEvaluatorBase._handle
_result
lxml.etree.XPathEvalError: Invalid expression
>>> printf(tree.xpath('/bib/book/author[contains(.,"ull"]'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "src/lxml/etree.pyx", line 2342, in lxml.etree._ElementTree.xpath
  File "src/lxml/xpath.pxi", line 342, in lxml.etree.XPathDocumentEvaluator.__ca
ll__
  File "src/lxml/xpath.pxi", line 210, in lxml.etree._XPathEvaluatorBase._handle
_result
lxml.etree.XPathEvalError: Invalid expression
>>> printf(tree.xpath('/bib/book/author[contains(., "ull")]'))
<author><first-name>Rick</first-name><last-name>Hull</last-name></author>

>>> printf(tree.xpath('/bib/book/author[contains(., "ull")]'))
<author><first-name>Rick</first-name><last-name>Hull</last-name></author>

>>> printf(tree.xpath('/bib/book/author[contains(., "Ull")]'))
<author>Jeffrey D. Ullman</author>

>>> printf(tree.xpath('/bib/book/author[substring(., 1, 1) = "J"]))
  File "<stdin>", line 1
    printf(tree.xpath('/bib/book/author[substring(., 1, 1) = "J"]))
                      ^
SyntaxError: unterminated string literal (detected at line 1)
>>> printf(tree.xpath('/bib/book/author[substring(., 1, 1) = "J"]'))
<author>Jeffrey D. Ullman</author>

>>> printf(tree.xpath('/bib/book/author[starts-with(., 1, 1) = "J"]'))
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
  File "src/lxml/etree.pyx", line 2342, in lxml.etree._ElementTree.xpath
  File "src/lxml/xpath.pxi", line 342, in lxml.etree.XPathDocumentEvaluator.__ca
ll__
  File "src/lxml/xpath.pxi", line 210, in lxml.etree._XPathEvaluatorBase._handle
_result
lxml.etree.XPathEvalError: Invalid number of arguments
>>> printf(tree.xpath('/bib/book/author[starts-with(., "J")]'))
<author>Jeffrey D. Ullman</author>

>>>
```

carlos — ubuntu@ip-172-31-8-201: ~/class — ssh -i dsci551.pem ubuntu...

```
>>> printf(tree.xpath('/bib/book[1]/@price'))
35
>>> printf(tree.xpath('/bib/book[year=1995]/title'))
<title>Foundations of Databases</title>

>>> printf(tree.xpath('/bib/book[year >=1995 and year <=2000]/title'))
<title>Foundations of Databases</title>

<title>Principles of Database and Knowledge Base Systems</title>

>>> printf(tree.xpath('/bib/book[year >=1995 or year <=2000]/title'))
<title>Foundations of Databases</title>

<title>Principles of Database and Knowledge Base Systems</title>

>>> printf(tree.xpath('/bib/book[year >=1995 or not(year <=2000)]/title'))
<title>Foundations of Databases</title>

<title>Principles of Database and Knowledge Base Systems</title>

<title>xyz</title>

>>> printf(tree.xpath('/bib/book[price]/title'))
<title>Foundations of Databases</title>

>>> printf(tree.xpath('/bib/book[price]/title'))
<title>Foundations of Databases</title>

>>> printf(tree.xpath('/bib/book[author/first-name = "Rick"]/title'))
<title>Foundations of Databases</title>

>>> printf(tree.xpath('/bib/book[author/@age]/title'))
<title>Foundations of Databases</title>

<title>xyz</title>

>>> print(tree.xpath('/lib/book'))
[]
>>> printf(tree.xpath('/lib/book'))
>>> printf(tree.xpath('/bib/book'))
<book price="35">
        <publisher>Addison-Wesley</publisher>
        <author>Serge Abiteboul</author>
        <author><first-name>Rick</first-name><last-name>Hull</last-name></author
>
        <author age="20">Victor Vianu</author>
        <title>Foundations of Databases</title>
        <year>1995</year>
        <price>38.8</price>
</book>

<book price="55">
        <publisher>Freeman</publisher>
        <author>Jeffrey D. Ullman</author>
        <title>Principles of Database and Knowledge Base Systems</title>
        <year>1998</year>
</book>

<book>
        <title>xyz</title>
        <author age="25"/>
</book>
```

```
>>> printf(tree.xpath('/bib/book[1]/author/first-name/text()'))
Rick
>>> printf(tree.xpath('/bib/book[1]/price'))
<price>38.8</price>

>>> printf(tree.xpath('/bib/book[1]/@price'))
35
>>> printf(tree.xpath('/bib/book[year=1995]/title'))
<title>Foundations of Databases</title>

>>> printf(tree.xpath('/bib/book[year >=1995 and year <=2000]/title'))
<title>Foundations of Databases</title>

<title>Principles of Database and Knowledge Base Systems</title>

>>> printf(tree.xpath('/bib/book[year >=1995 or year <=2000]/title'))
<title>Foundations of Databases</title>

<title>Principles of Database and Knowledge Base Systems</title>

>>> printf(tree.xpath('/bib/book[year >=1995 or not(year <=2000)]/title'))
<title>Foundations of Databases</title>

<title>Principles of Database and Knowledge Base Systems</title>

<title>xyz</title>
```

```
>>> printf(tree.xpath('/bib/book[1]/author/first-name/text()'))
Rick
```

```
>>> printf(tree.xpath('/bib/book[1]/publisher'))
<publisher>Addison-Wesley</publisher>

>>> printf(tree.xpath('/bib/book[1]/publisher/text()'))
Addison-Wesley
>>> printf(tree.xpath('/bib/book[1]/author'))
<author>Serge Abiteboul</author>

<author><first-name>Rick</first-name><last-name>Hull</last-name></author>

<author age="20">Victor Vianu</author>
```

```
>>> printf(tree.xpath('/bib/book[1]'))
<book price="35">
        <publisher>Addison-Wesley</publisher>
        <author>Serge Abiteboul</author>
        <author><first-name>Rick</first-name><last-name>Hull</last-name></author
>
        <author age="20">Victor Vianu</author>
        <title>Foundations of Databases</title>
        <year>1995</year>
        <price>38.8</price>
</book>
```

```
>>> printf(tree)
<bib>
<cd>abc</cd>
<book price="35">
        <publisher>Addison-Wesley</publisher>
        <author>Serge Abiteboul</author>
        <author><first-name>Rick</first-name><last-name>Hull</last-name></author
>
        <author age="20">Victor Vianu</author>
        <title>Foundations of Databases</title>
        <year>1995</year>
        <price>38.8</price>
</book>
<book price="55">
        <publisher>Freeman</publisher>
        <author>Jeffrey D. Ullman</author>
        <title>Principles of Database and Knowledge Base Systems</title>
        <year>1998</year>
</book>
<book>
        <title>xyz</title>
        <author age="25"/>
</book>
</bib>
>>> printf(tree.xpath('/bib/book'))
<book price="35">
        <publisher>Addison-Wesley</publisher>
        <author>Serge Abiteboul</author>
        <author><first-name>Rick</first-name><last-name>Hull</last-name></author
>
        <author age="20">Victor Vianu</author>
        <title>Foundations of Databases</title>
        <year>1995</year>
        <price>38.8</price>
</book>

<book price="55">
        <publisher>Freeman</publisher>
        <author>Jeffrey D. Ullman</author>
        <title>Principles of Database and Knowledge Base Systems</title>
        <year>1998</year>
</book>

<book>
        <title>xyz</title>
        <author age="25"/>
</book>
```

```
ubuntu@ip-172-31-8-201:~/class$ ls
bibs.xml  fsimage564.xml  hello  helper.py  note
ubuntu@ip-172-31-8-201:~/class$ python3
Python 3.12.3 (main, Jan 17 2025, 18:03:48) [GCC 13.3.0] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>> from lxml import etree
>>> from helper import printf
>>> tree = etree.parse('bibs.xml')
>>> tree
<lxml.etree._ElementTree object at 0x70bf6d53f4c0>
>>> printf(tree)
<bib>
<cd>abc</cd>
<book price="35">
        <publisher>Addison-Wesley</publisher>
        <author>Serge Abiteboul</author>
        <author><first-name>Rick</first-name><last-name>Hull</last-name></author
>
        <author age="20">Victor Vianu</author>
        <title>Foundations of Databases</title>
        <year>1995</year>
        <price>38.8</price>
</book>
<book price="55">
        <publisher>Freeman</publisher>
        <author>Jeffrey D. Ullman</author>
        <title>Principles of Database and Knowledge Base Systems</title>
        <year>1998</year>
</book>
<book>
        <title>xyz</title>
        <author age="25"/>
</book>
</bib>
```

```
ubuntu@ip-172-31-8-201:~$ ls class/
bibs.xml  fsimage564.xml  hello  note
ubuntu@ip-172-31-8-201:~$
```

```
sftp> lcd "/Users/carlos/Downloads"
sftp> put fsim
stat fsim: No such file or directory
sftp> put fsimage564.xml
Uploading fsimage564.xml to /home/ubuntu/fsimage564.xml
fsimage564.xml                              100%   16KB 242.6KB/s   00:00
sftp> put bibs.xml
Uploading bibs.xml to /home/ubuntu/bibs.xml
bibs.xml                                    100%  659    10.3KB/s   00:00
```