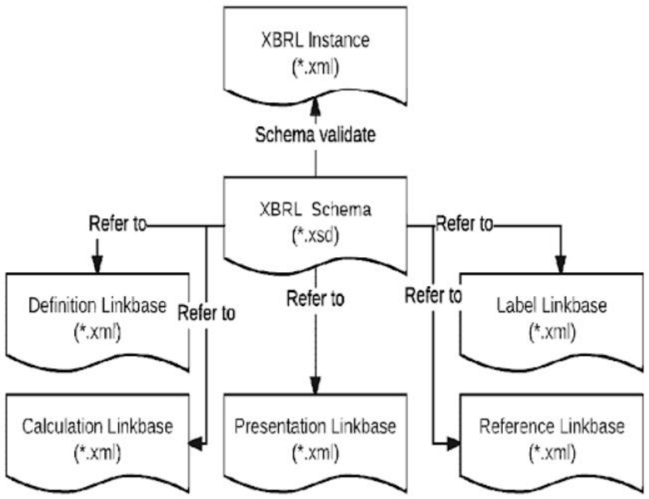
XBRL(extensible business report language) Report

--By QiYao

Introduction

What's XBRL?

XBRL[1] (eXtensible Business Reporting Language) is a freely available and global standard for exchanging business information. XBRL allows the expression of semantic meaning commonly required in business reporting. The language is XML-based and uses the XML syntax and related XML technologies such as XML Schema, XLink, XPath, and Namespaces. One use of XBRL is to define and exchange financial information, such as a financial statement. The XBRL Specification is developed and published by XBRL International, Inc. (XII).
XBRL[2] is a standards-based way to communicate and exchange business information between business systems. These communications are defined by metadata set out in taxonomies, which capture the definition of individual reporting concepts as well as the relationships between concepts and other semantic meaning. Information being communicated or exchanged is provided within an XBRL instance.
In general XBRL consists of an **XBRL instance**, containing primarily the business facts being reported, and **a collection of taxonomies** (called a Discoverable Taxonomy Set (DTS)), which define metadata about these facts, such as what the facts mean and how they relate to one another. It includes a schema file defining the facts' meanings as well as linkbase files. And Linkbases are made up with 5 different categories:Definition linkbase, label linkbase, reference linkbase, calculation linkbase and presentation linkbase. And the structure shown as the below figure[3]:



XBRL is currently wided used by many companies which needs to provide their information to regulators or need to accurately move information around within a complex group, governments that are simplifying the process of business reporting to government or any analysts and investors who want to understand the investment risk and need to compare the companies' potential in terms of the finance performance based on the current investment XBRL files provides. So XBRL is actually a collection of rules and instances which play important role nowadays.

Why?

What problem exist?

Since the widely use of XBRL in the international companies, every company's quarterly, half-year,annual reports all use XBRL as the standard of reports. So there are quite massive XBRL financial reports produced each season every year. So the traditional databases and methods system can't hold on such size of data and analyze this amount of data. And worse more, with the complexity of the reference relationship and hierarchies among the data, it make the data much more difficult to be used and analyzed. And every company, government as well as investor all want to make use of huge amount of data. So we have to use a effective method to make use of the kind of distributed system or make use of parallel methods to deal with current problems above.

What?

What methods has been proposed?

T Feng, ZY Zhang etc proposed a kind of Frequent pattern mining for massive XBRL Data in parallel in a Internet Information System.[4]

They proposed a method based on Hadoop Map/Reduce ideas. They map the XBRL data into key-value pair and use Map functions to other key-value pairs. And output the results merged by the Reduce functions. They use three different steps:First, try to upload the XBRL instances to HDFS. All information in the massive XBRL instances can be completely saved in the HDFS for further analyzed. Second, they use XQuery language to get the constructured bool matrix of XBRL transaction data. Finally, partition boolean matrix of XBRL transaction data into small pieces and the frequent pattern by Map/Reduce fucntions and get the parallel data mining for XBRL frequent pattern. And thus they extract the frequent pattern about these XBRL data.

Y.F Pan, Y Zhang, Ken Chiu process the xml making use of multicore CPUs.[5]

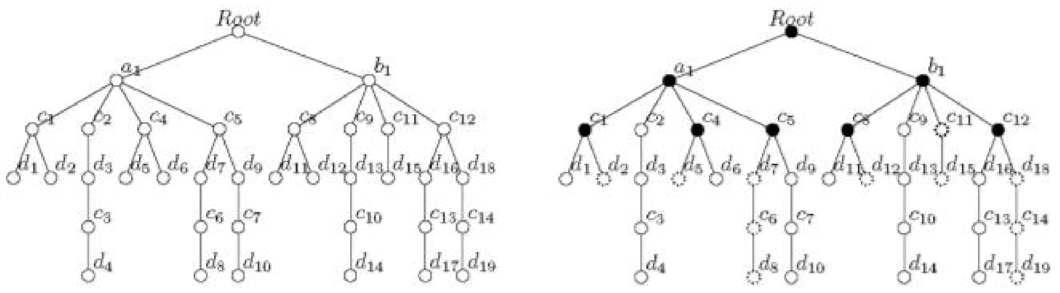
Achieving data parallelism by dividing the XML document into chunks and then independently processing all chunks in parallel is difficult, however, because the state of an XML parser at the first character of a chunk depends potentially on the characters in all preceding chunks. The preparsing is sequential, however, and thus limits speedup. In this work, we parallelize the preparsing pass itself by using a simultaneous finite transducer (SFT), which implicitly maintains multiple preparser results. Each result corresponds to starting the preparser in a different state at the beginning of the chunk. This addresses the challenge of determining the correct initial state at beginning of a chunk by simply considering all possible initial states simultaneously.

How?

How to deal with the XBRL in parallel with Spark?

There are some methods based on Hadoop but no methods based on Spark yet. So I think we can use similar methods. The problems are mainly about the two parts. **First, we can process multiple xbrl instances in parallel. Then within each instance we can process multiple branches in parallel**.

So in general, it has **two parts**:
First, for multiple XBRL instances, we take each instance as a <K,V> pair, where K is the company name and report time while V is the concrete content of the xbrl instances. So first we can have a set of data like *D(K,V):{(CompanyA_time1, Statement),(CompanyA_time2, Statement),(CompanyB_time1, Statement)...}*. And we can read them into JavaPairRDD. Further more, we can repartition it and can use Parse multiple XBRL instances in parallel.
Second part: Parse each XBRL instance in parallel. For each instance, within the Parse function, we can achieve parallel too. XBRL is still a XML file, which has quite clear hierarchic structure. So we use tree structure to represent the XBRL and we can try to parse it into several subtrees. And therefore we can process these subtrees in parallel. The basic idea is that in XML file, within each element,it includes several other elements or value. So we can recognize the different element names and divide each element into a partition.


Root
a_1
b_1
c_1 c_2 c_4 c_5 c_8 c_9 c_{11} c_{12}
d_1 d_2 d_3 d_5 d_6 d_7 d_9 d_{11} d_{12} d_{13} d_{15} d_{16} d_{18}
c_3 c_6 c_7 c_{10} c_{13} c_{14}
d_4 d_8 d_{10} d_{14} d_{17} d_{19}

As the figure shows, from the /root we have several branches at each level. And we can separate the data with same base directory into different machines. And we can regard any root-leaf path as an path instance. For example, in the figure above, if we have two machines, the black nodes are the data will be stored in the all the machines. And the according to different data dotted circle nodes is stored in one machine, while normal circle nodes is stored in the other one. And these can lead to process the data in parallel.
In pseudo code:

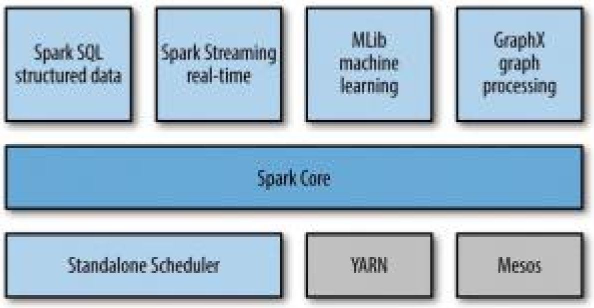
```
//First parallel processing multiple XBRL instances
Have N XBRL instances:D={(K1,V1),(K2,V2)...(KN,VN)} Take as RDD And K machines:
Take D as PairRDD and take into K partitions.
For machine k= 1 to K (Each machine has N/K sets)do:
    For(i=1 to N/K) do:
        Result=ParseXBRL(Vi);
    End For
End For
Return Result.

ParseXBRL(String V):
InterList=Root(V) //Get all Elements, Using Queue to store the element in order
DepthLimit=k. //Any nodes deeper than k will be regarded as IN nodes
IN=InnerNodes(InterList,DepthLimit)
$DN \leftarrow \bigcup_{E \in IN} Ancessor(E)DN$     // DN is the root nodes,ancestor nodes of IN
Duplicate each node in DN to all the machines
```

```
For each element    E ϵ IN
    Do group E according to getParent(E)
    //group together all the IN nodes having same Parent nodes
i=0
End For
For each group E do
    Distribute Subtree(E) to Machine_i
    i++
End For
For each i=1 to K do:
    Result_i=getContent(Subtree(E))    //Understand each subtree in parallel

Return SumAll(Result_i)
```

How is the spark framework implementing this?
In general,Spark has a various software stack including Spark Streaming, Shark SQL,MLib etc. Spark streaming offers the most basic construction of SparkConf to handle the real-time data streaming. SQL gives spark the ability to operate the SQL while ML makes run ML algorithms. Here we should use MLib and Straming.



And all these ensure a solid environment for various applications.
Spark also uses a **host/workers architecture** which means the host has a scheduler to schedule the job and assign all the tasks to the worker node where executors run.And Partition is the minimum operation part in Spark. How many partitions there are, how many tasks there will be.
And the job driver program is running on the master node and create the SparkContext. And then submit the job to all the worker nodes and get scheduled by the cluster manager process.
And as for this XBRL problem, we first focus on processing multiple XBRL instances in parallel where Spark PairRDD provide the opportunity to do so. And within each instance, we can process each instance with the corresponding tree structure and take the nodes with the same parent node as the the same subtree and we can process all subtrees in parallel too to get the tree structure into subtree partition which will help a lot for processing the XBRL instance.

Reference:

[1] https://en.wikipedia.org/wiki/XBRL
[2] https://www.xbrl.org/the-standard/what/an-introduction-to-xbrl/

[3] https://books.google.com/books?id=sF7FCwAAQBAJ&pg=PA138&lpg=PA138&dq=XBRL+parallel&source=bl&ots=v-Az6siyOD&sig=oocKPzwpG1CSpS2AbFH98uqy3w4&hl=zh-CN&sa=X&ved=2ahUKEwjTgqnBxd7aAhVCwVkKHZJpBL04ChDoATABegQIABA0#v=onepage&q&f=true
[4] Feng T, Zeng Z Y. Frequent Pattern Mining for Massive XBRL Data in Internet Information Disclosure System[C]//Software Engineering and Information Technology: Proceedings of the 2015 International Conference on Software Engineering and Information Technology (SEIT2015). 2016: 138-142.
[5] Pan Y, Zhang Y, Chiu K. Simultaneous transducers for data-parallel XML parsing[C]//Parallel and Distributed Processing, 2008. IPDPS 2008. IEEE International Symposium on. IEEE, 2008: 1-12.