A performance comparison of Hive and Pig

Julius Bladh, Andreas Pålsson
DD2471

May 27, 2015

Abstract

Information extraction has recently received significant attention due to the rapid growth of unstructured text data. However, this is computationally intensive and MapReduce and parallel database management systems have been used to analyze large amounts of data. In the paper *A performance comparison of parallel DBMSs and MapReduce on large-scale text analytics* the performance of a Hadoop implementation of MapReduce has been compared to one of the more popular parallel DBMSs. However, the authors only compared the performance when using one specific high level language over Hadoop.

The aim of this project is to compare the performance of the Hadoop/Pig implementation of MapReduce with Hadoop/Hive. We will use some of the benchmark methods mentioned in the paper to do this comparison.

1 Introduction

The amount of text data grows every day on the Internet, for example on social media, news articles or webpages. However, this data is largely unstructured and its usefulness is limited. Therefore, information extraction (IE) is introduced in order to increase the usefulness of unstructured text. However, performing IE tasks is computationally intensive and MapReduce and parallel database management systems have been used to analyze large amounts of data. A common way to process large sets of data is using Apache Hadoop. Hadoop is a Java-implemented framework that allows for the distributed processing of large data sets across clusters of computers. Since writing MapReduce jobs in Java can be difficult, Hive and Pig has been developed and works as platforms on top of Hadoop. Hive and Pig allows users easy access to data compared to implementing their own MapReduce in Hadoop.

1.1 Purpose

In this project, we will measure the response performance of Hive and Pig in order to understand which platform is a better choice for large IE tasks. Pig and Hive were chosen because of their widespread use and due to the fact that they are the two most popular implementations of MapReduce on Hadoop[1].

1.2 Related Work

A performance comparison on IE between Hadoop/Pig and a parallel DBMSs called Vertica has recently been done [1]. The study used popular methods of performing IE from unstructured data. The methods included regular expression-based IE and sequential modeling-based IE used in conjunction with conditional random fields.

The study focused on response time between the two platforms and concluded that Vertica outperformed Pig by 5-9 times. However, the authors wants to extend their future work by also comparing with Hive.

2 Background

2.1 Hive & Pig

Both Hive and Pig are platforms optimized for analyzing large data sets and are built on top of Hadoop. Hive is a platform that provides a declarative SQL-like language whereas Pig requires users to write a procedural language called PigLatin. Users that are used to more traditional databases might therefore lean away from using Pig due to having to learn new tools, but also due to the need of a changed workflow.

2.2 Information extraction

Information extraction (IE) is the task of automatically extracting structured information from unstructured data. This can be done in several ways, but two methods are introduced in the article "A Performance Comparison of Parallel DBMSs and MapReduce on Large-Scale Text Analytics"[1].

The first method mentioned is using a sequence model in conjunction with conditional random fields (CRF). This means using statistical models and machine learning to perform certain tasks such as named entity extraction. The other method the authors present is using hand-written regular expressions to extract data from unstructured text.

3 Method

A data collection of significant size was needed to compare the performance of Hive and Pig. The data collection used in the following comparisons is a dump of Wikipedia articles, which resulted in a 2 gigabyte XML-file.

In order to conduct various IE tasks, the Wikipedia articles were preprocessed in the following way. Firstly, a sentence splitter written by the Cognitive Computation Group at University of Illinois at Urbana-Champaign[2] was used. This tool parsed the data in the articles and formatted the output so that one sentence is written per line. In addition to this, we wrote a C++-program to add an article ID and a sentence ID to the beginning of each line in the dump. This CSV-formatted (comma separated values) files were easy to import to both Hive

and Pig.

From the aforementioned CSV-formatted file, we imported the data into the table sentences in Hive and Pig in the following schema:

Attr.name	Attr.type
article_id	int
sentence_id	int
sentence	string

Table 1: Sentences table.

In addition to this, we were interested in extracting named entities from the articles. To accomplish this task, we tokenized each word in the articles and used a part of speech-tagger (PoS-tagger) developed by the The Stanford Natural Language Processing Group at Stanford University[4]. A PoS-tagger assigns part of speech to each word, such as noun, verb, adjective etc. For example, "Mike is cool" would be "Mike_NNP is_VB cool_JJ", where NNP is proper noun, VB is verb and JJ is adjective.

This allowed us to create the following schema in Pig and Hive, where article_id is used to identify which article the token comes from. The sentence_id is used to identify the sentence from which the token came and pos is what PoS-tag the token has:

Attr.name	Attr.type
article_id	int
sentence_id	int
token	string
pos	string

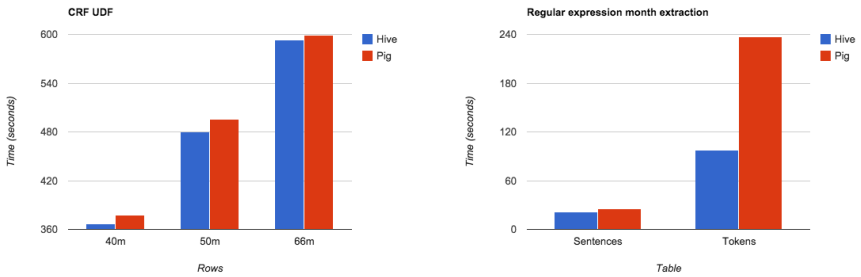
Table 2: Tokens table.

The first test we performed was to see how well Hive and Pig performed regarding extracting data based on regular expressions. We extracted the names of months from both table Sentences and Tokens.

The second test was conducted in order to more thoroughly compare the performance of Pig and Hive when the tasks are more computationally intensive. This test examined the speed at which the platforms could extract all the nouns in the Tokens table and classify each proper noun as a location, organisation, people or other. To accomplish this, we wrote our own user defined functions which took advantage of Stanford's Named Entity Recognizer[4]. The software provides a general implementation of linear chain Conditional Random Field (CRF) sequence models. That is, by training your own models, you can actually use this code to build sequence models for any task. However, the library comes pre-trained with models trained with the CoNLL 2003[3] English training data and we found these sufficient for our purpose. The test was conducted on three different amounts of data: 40 million rows, 50 million rows and 66 million rows.

4 Result

The result for Regular expression-based IE, is shown in figure (b). The Sentences table contained 1.2 million rows, while the Tokens table contained 66 million rows. The results indicate that Hive performs better than Pig when using regular expressions. Figure (a) is the result for the named entity extraction with the CRF. The test was executed on three different amount of data, containing 40 million to 66 million rows. The results also indicate that Hive has faster respond time than Pig, when using user defined functions with CRF.



(a) User defined function IE runtimes (b) Regular expression IE runtimes

Figure 1

5 Conclusion

In this report a comparison between Hive and Pig has been carried out, to see how well each platform perform when conducting IE tasks. Tests show that Hive outperformed Pig for both regular expression-based and CRF-based IE. However, this study was performed on small data sets and Pig and Hive might not perform as well on such small data. For future work, it would be fundamental to compare Hive and Pig on bigger data collections.

References

[1] Fei Chen and Meichun Hsu. A performance comparison of parallel dbmss and mapreduce on large-scale text analytics. In *Proceedings of the 16th International Conference on Extending Database Technology*, EDBT '13, pages 613–624, New York, NY, USA, 2013. ACM.

[2] University of Illinois at Urbana-Champaign Cognitive Computation Group. Sentence segmentation toolr.

[3] Erik Tjong Kim Sang et al. Language-independent named entity recognition.

[4] Jenny Finkel et al. Stanford named entity recognizer.