HBase vs Neo4j

Technical overview

Name: Vladan Jovičić
CR09 Advanced Scalable Data
(Fall, 2017)

École Normale Superiuere de Lyon

12th October 2017
## Contents

1. Introduction ........................................... 3
2. Overview of HBase and Neo4j models ................. 4
3. Features of HBase and Neo4j ......................... 5
4. Use cases of the databases .......................... 6
5. Conclusion ............................................ 8
1 Introduction

Nowadays, big data is everywhere around us. The data is usually disorganized and without any form. Thus, there is a need for storing it in a some way so that it can be easily analyzed. The problem of storing and analyzing data efficiently exists since 50s. However, after foundation of some big companies, the amount of data has exploded. So, new techniques to handle this data were needed. This was the main reason for developing a non-traditional, non-relational databases which were used in the past. It is important to say that relational databases can also handle big data although a much more time is required for processing. Two databases which are the main topic of this report are founded after the “data explosion”.

This report gives an overview of HBase and Neo4j database models and presents their features, situations in which these database systems are applicable. The report also provide a comparison between the mentioned database systems. This report will not focus on the difference between general databases systems (e.g. relational vs non-relational). We assume that a reader is familiar with standard notions used when one discusses about databases. However, in the section where we describe features, we define almost all notions (unless its meaning is obvious). This is done in order to recall and highlight the importance of these features. This report does not provide how to use HBase or Neo4j, nor it discusses the architecture of databases.

The report is organized as follows. We firstly give an overview of both databases. We discuss an underlying models and how data is organized in a storage using these models. Then we start discussion about features. For each feature, we say if it is supported by the studied databases or not. At the end, we give an use-cases for the databases, i.e., when they should be used and when they should not.

The main reference for HBase is [3]. For Neo4j, information relies on the official Neo4j documentation [1] and developer guide [2].
2 Overview of HBase and Neo4j models

In this section we give a short overview of HBase and Neo4j models. HBase is built on the top of Hadoop file system (shortly HDFS). The underlying model of HBase is a hash table and as such, it provides a quick random access to large amount of structured data. HBase database consists of one or more tables. A table in database contains multiple rows. Table also contains column families and every column family contains multiple (usually different) columns. Table 1 provides a graphical overview of what we have described.

<table>
<thead>
<tr>
<th>row / col family</th>
<th>A</th>
<th>B</th>
<th>C</th>
</tr>
</thead>
<tbody>
<tr>
<td>col name</td>
<td>col1</td>
<td>col2</td>
<td>col3</td>
</tr>
<tr>
<td>1</td>
<td>a</td>
<td>b</td>
<td>c</td>
</tr>
</tbody>
</table>

Table 1: Organization of data in HBase database

Recall that we said that the underlying model of HBase is a hash table. So, one might be wondering how to represent the above structure as a hash table. A correct answer is to use multidimensional hash tables. Every row has a key and its value is again a hash table. For the sake of ease of description, we will call these second level hash tables. The keys of second level hash tables are called column families. Again, value of each key is a hash table (third level hash tables). Finally, each key of these hash tables is called column. Following the syntax of JavaScript used for hash tables, the structure represented in Table 1 can be represented as follows:

```javascript
{
  "1" : {
    "A" : { "col1" : "a", "col2" : "b", "col3" : "c" },
    "B" : { "col4" : "d", "col5" : "e" },
    "C" : { "col6" : "f" }
  }
}
```

Neo4j is an example of so called graph database and as such, the underlying structure of a Neo4j database is a graph. A common thing to all graph databases is that one can model different relations between entities almost perfectly. This is due to nature of its underlying model. The organization of data in Neo4j database is much simpler than in HBase database. All data is
Figure 1: An example of Neo4j database model

represented as a set of nodes (entities or objects). There is an edge between two different nodes if there is a relationship between them. Both nodes and edges can contain labels (or attributes). Node labels describe what the node represents (e.g. person, book, movie). Edge labels describe a relationship between different nodes. For example, if there is a node with value Alice and label PERSON and a node with name Introduction to databases and label BOOK one might add an edge between the two nodes and set label BELONGS in order to describe that a book Introduction to databases belongs to Alice. Figure 1 describes this.

3 Features of HBase and Neo4j

In this section we present a different features of HBase and Neo4j. Some of them can be deduced directly from the description of data organization in previous section.

SCALABILITY: The notion of scalability is used to refer to capabilities of the database system to handle growth of amount of data. Both HBase and Neo4j are horizontally scalable, meaning that we scale it by adding more machines into our pool of resources. Note that in literature researchers often refer to horizontal scalability as linear scalability.
Fault Tolerance: We say that a system is a fault tolerant if it can continue working properly when one of its components fails. HBase is fault tolerant by default since it is built on the top of HDFS. Neo4j is fault tolerant as well. Both systems handle this problem by replicating data. The only difference is that HBase allows to select replication factor while Neo4j follows Raft protocol.

Parallel processing and MapReduce: When it comes to big data, parallel processing of data should be one of key features. Both HBase and Neo4j allow parallel processing. However, Neo4j does not offer a support for MapReduce while HBase does.

Foreign keys: Foreign keys allow to uniquely identify a field of one table with a row or column of another table. It is easy to notice that the notion of foreign keys is more suitable for relational databases. However, due to nature of Neo4j, it is possible to create foreign keys between different entities. HBase does not support this.

Load balancing: The notion of load balancing is used to refer to possibilities of a system to distribute requests (or queries) evenly among working units. Both databases have their own, modifiable load balancers.

Optimization of queries: HBase supports block cache to improve read performance. Neo4j has two types of caches: file buffer cache and object cache. Their techniques improve both write and read performance.

Other features: Both databases support server-side scripts. Neo4j supports a few more programming languages. The data in HBase database is accessed using either Java API or RESTful HTTP API. Neo4j offers the same and in addition a few more. A Neo4j specific access method is Cypher Query Language. One advantage of Neo4j compared to HBase is that it supports ACID transaction concept (support to ensure data integrity after non-atomic manipulations of data).

4 Use cases of the databases

The previous sections give us enough information to derive when one should use HBase or Neo4j. Although both databases offer a lot of functionalities, we can not say that they are applicable in any situation and for any need. Usually, we have to consider different parameters in order to decide. Some
of them are: type of the application, data volume, data importance, data type, etc. Here, we will state in which cases HBase and Neo4j are the best solutions. We leave for a reader to investigate if one of these databases is a good solution in the cases when application requirements (parameters) are different.

The following properties are indicators that HBase is a good solution for an application.

- **A huge data volume**: an application deals with a huge amount of data and the data is accessed frequently.

- **Non-transactional application**: HBase is not suitable for transactional applications.

- **Non-relational data**: If the data that application accesses is not highly relational but rather has a variable schema with slightly different rows, then HBase should be surely considered.

- **A good hardware support**: HBase uses distributed file system. Having a HBase database on only one machine does not exploit performance of HBase system.

From the other hand, one should consider Neo4j database in the following situations:

- **Strongly related data**: When different units of data have a different relationships with other units, Neo4j is definitely worth considering.

- **Queries consists of joins**: Neo4j has a good support if the application access data from different tables and combines it.

- **Need for local and global data information**: Sometimes, the applications needs a global information about data. For example, assume that the application needs to answer how many restaurants each person liked on average. In such cases, Neo4j is one of the best candidates.

At the end of this section, we present a few big companies which uses HBase and Neo4j.

- **HBase**: Facebook, Twitter, Yahoo, and Adobe use HBase

- **Neo4j**: a service of eBay, a service of Telenor, Walmart, etc.
5 Conclusion

We gave a short overview of HBase and Neo4j models, as well as the key features which these two databases offer. Also, we saw some cases when one should consider using these databases. Implicitly, we also defined when one should not use them. By presenting the models and features, we can say that use cases of databases are very different. Thus, it is not possible to say that one database is strictly better than other; they are incomparable. However, we can say that HBase is one of the best databases systems when we speak about NoSQL databases, and that Neo4j is the best in the case of graph databases.

References