INFO-H413 Heuristic Optimization
Implemenation Exercise 1

Dany S Efila ULB 000349567

Universite Libre de Bruxelles

Avril 2017

INFO-H415 Adanvanced Databases Documents store and cloudant

Dany S EFILA
Michel NOUTCHA

Universite Libre de Bruxelles

Decembre 2017

0.1 Objectives

The aim of this project is to understand how the document oriented databases works and how the Cloudant technology can be exploited based on that kind of database model. For this purpose we first have to introduce the NoSQL databases, the state of the art and their link with the document store in the first part. The second part will be the cloudant technology in itself and the last part will focused on the details of the implementation.

0.2 Introduction

With the increase in internet bandwidth, as well as the reduction of IT hardware costs, new opportunities have emerged in the field of distributed computing. The transition to the 21st century by the WEB 2.0 revolution saw the data volume of some companies increase considerably. These data come mainly from social networks, medical databases, economic indicators, etc. The increasing computerization of all kinds of processing has resulted in an exponential increase in this volume of data, which is now in petabytes, the Anglo-Saxon called it Big Data. Managing these volumes of data has become a problem that relational databases have not been able to manage. The NoSQL brings together many databases that no longer rely on a relational representation logic. However, it is not easy to explicitly define what a NoSql database is, since no standard has yet been established.

0.3 State of the art

0.3.1 NoSQL Databases

The basic need that NoSQL answers is performance. In order to solve "Big Data" problems, developers from companies such as Google and Amazon have made compromises on the RDBMS ACID properties. These compromises on the relational notion have allowed the RDBMS to free themselves from their brakes on horizontal scalability. Another important aspect of NoSql is that it responds to the CAP theorem that is more suitable for distributed systems. The other benefit of NoSQL is financial. Serving to handle large volumes of data, it is therefore intended for large companies. By 2010, NoSql began to expand to smaller companies. Mostly start-ups can not afford to buy Oracle licenses that have developed their own DBMS by mimicking Google and Amazon products.Four major categories stand out among these.

Key Value databases

The key-value database is considered the most basic. Its principle is very simple, each stored value is associated with a unique key. Only with this key will it be possible to execute queries on the value. The structure of the stored object is free and therefore the responsibility of the developer of the application. A considerable advantage of this type of database is that

it is very easily extensible, so we speak of horizontal scalability. Indeed, in the case where the volume of data increases, the load is easily distributable between the different servers by simply redefining the key intervals between each server. As far as the need for vertical scalability is concerned, this is very much reduced by the simplicity of the queries which are PUT, GET, DELETE, UPDATE. This type of database thus displays very good read / write performance and tends to reduce the number of requests made, their choice being restricted. However, queries can only be executed on keys because of the simplicity of the principle. Allowing data storage without schema, the client is forced to retrieve all the contents of the blob and can not obtain a result of thinner queries such as results from SQL queries could allow. This type of key-value-oriented database therefore involves very specific use cases, because of their simplicity of model and the wide range of choice of proposed queries. These systems are therefore mainly used as a repository provided that the types of requests required are very simple. They are found as a cache storage system or distributed sessions, especially where the integrity of the data is insignificant. Today,The best known solutions that have adopted the key-value pair system are Voldemort (LinkedIn), Redis and Riak.

Column Oriented Databases

Column-oriented databases have been designed by the web giants to cope with the management and processing of large volumes of data growing rapidly from day to day. They often integrate a system of minimalist queries close to SQL. Although widely used, there is still no official method or defined rules for a column-oriented database to be qualified as quality or not. The principle of a column database consists in their storage by column and not by row. That is, in a row-oriented database (conventional DBMS) the data is stored in such a way as to favor the rows by grouping all the columns of the same row together. Column-oriented databases will store data so that all data in the same column is stored together. These bases can evolve over time, either in number of rows or in number of columns. In other words, and contrary to a relational database where the columns are static and present for each row, those columns oriented columns are called dynamic and therefore present only in case of need. In addition, the storage of a "null" is 0. Take the example of the registration of a customer requiring his name, first name and address. In a relational database, you must first create the schema (the table) that will store this information. If a client does not have an address, the rigidity of the relational model requires a null marker, signifying an absence of value which will however cost space in memory. In a column-oriented database, the address column will simply not exist. Column databases have been designed to store millions of columns and are therefore perfect for managing one-to-many storage.

document Oriented Databases

Document databases are an evolution of key-value databases. Here the keys are no longer associated with values in the form of a binary block but with a document whose format is not imposed. It can be of several different types such as JSON or XML, provided that the database is able to manipulate the format chosen to allow processing on documents.

Otherwise, it would be equivalent to a key-value database. Although the documents are structured, this basic type is called "schemaless". It is therefore not necessary to define the fields of a document.

The advantage of document databases is that they can retrieve a set of hierarchically structured information from a key. A similar operation in the relational world would be equivalent to multiple table joins. Document databases are therefore very much in demand by Web applications that broadcast entire pages obtained by a set of joins. These applications can also cache information in an intelligible form. For the majority of them, the base can be directly put on the RAM memory allowing a considerable gain of performance. A negative point about these databases is the duplication of information. Because data is grouped by document, integrity issues may arise, with some data necessarily duplicated across multiple documents. In addition, by the very great flexibility of their schema (a priori an asset), it is very easy to make an error when inserting data. Relational databases provide greater data consistency, because in a case such as this, the server will refuse to execute the query if the database schema is not followed.

Graph Oriented Databases

Although key-value, column, or document databases derive their main benefit from the performance of data processing, graph-oriented databases can solve very complex problems that a relational database would be unable to solve. to do. Social networks (Facebook, Twitter, etc.), where millions of users are connected in different ways, are a good example: friends, fans, family etc. The challenge here is not the number of elements to manage, but the number of relationships that can exist between all these elements. Indeed, there are potentially n relations to store for n elements. Even if there are solutions such as joins, relational databases very quickly face performance problems as well as complexity problems in the development of queries. The graphical approach is therefore inevitable for social networks such as Facebook or Twitter.

Graph databases are also used for managing large IT structures. They also allow the development of links between the various interests that a user could have, in order to be able to offer him products likely to interest him. Thus, advertisements appearing on Facebook are very often in relation to the searches made on Google, and the proposals for purchases of online shopping sites such as eBay and Amazon are related to purchases already made (for example : People who bought this product also bought ...). As its name suggests, these databases are based on graph theory, with three elements to remember: An object (in the context of Facebook we will say that it is a user) will be called a Node. Two objects can be linked together (like a friendship relationship). Each object can have a number of attributes (social status, first name, name etc.) The data is stored on each node, itself organized by relations. From there, it will be much easier to perform operations that would have been very complex and cumbersome in a relational universe.

0.3.2 Document store

A document store is characterized by the fact that it has a schema free organization of data, which means that records do not need to have the same structure; column can have arrays as value and record can have nested structure.Document store can be used with two formats JSON or XML. JSON is the mostly used because can be processed directly by the applications, it is the main category of NoSQL database.

```
{"widget": {
    "debug": "on",
    "window": {
        "title": "Sample Konfabulator Widget",
        "name": "main_window",
        "width": 500,
        "height": 500
    },
    "image": {
        "src": "Images/Sun.png",
        "name": "sun1",
        "hOffset": 250,
        "vOffset": 250,
        "alignment": "center"
    },
    "text": {
        "data": "Click Here",
        "size": 36,
        "style": "bold",
        "name": "text1",
        "hOffset": 250,
        "vOffset": 100,
        "alignment": "center",
        "onMouseUp": "sun1.opacity = (sun1.opacity / 100) * 90;"
    }
}}
```

Figure 1: Example of JSON document

The principle behind is that document databases store all informations for a given object in a single instance in the database, this make mapping object into the database a simple task in contrast to Relational databases which ones generally store data in separate tables that are defined by the programmer, and a single object may be spread across several tables.

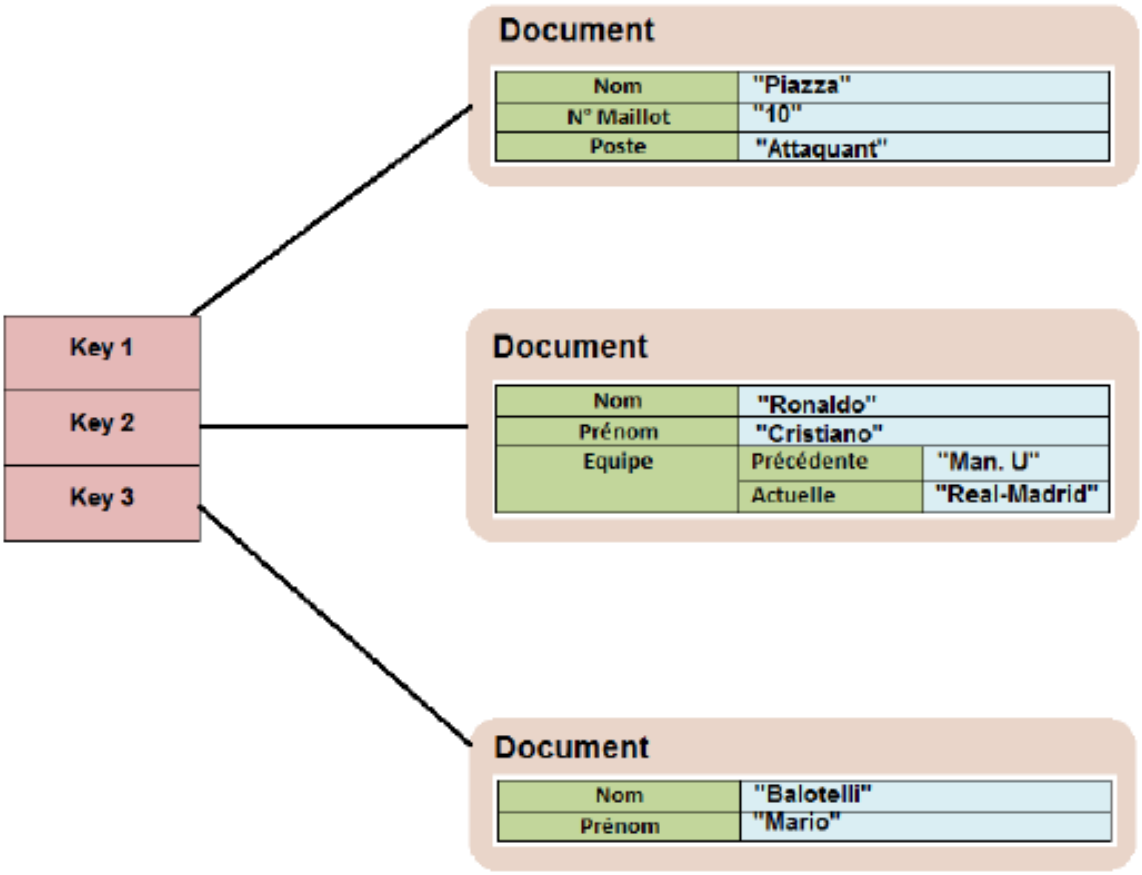

Figure 2: Example of key value document

0.4 How does it work

MapReduce paradigm

Distributed data processing raises some questions, how do you distribute the work between the servers? How to synchronize the different results? How to manage a failure of a processing unit? MapReduce addresses these issues. It is not a database element, but a programming model inspired by functional languages and more specifically Lisp language. It allows to process a large amount of data in parallel, distributing them on various nodes of a cluster. This mechanism was put forward by Google in 2004 and has been very successful with companies using DataCenters such as Facebook or Amazon.

Principle of MapReduce

The principle of MapReduce is simple: it involves cutting a task handling a large volume of data into several tasks each processing a subset of these data. MapReduce is seen in two stages. The first step, called "map" is to dispatcher a task, as well as the data that deals in several sub-tasks each processing a subset of data. The second step is called "Reduce". In this part it is a question of recovering the results of the various servers (namely the sub-tasks) and of consolidating them. We will come back in more detail on the subject below. Although the principle is relatively simple, it is mainly thanks to the contribution of the model MapReduce simplifying to the maximum the complexities of distributed computing. It allows developers to focus on the actual processing. The fact that MapReduce is developed in Java allows you to get away from the hardware architecture so that a MapReduce framework can run on a set of heterogeneous machines.

Functioning of MapReduce

As said before, the MapReduce model consists of two steps, both represented by functions. In the map function we take as input a set of "key-values", the node does an analysis of the problem and it will separate it into sub-tasks, to be able to redistribute them on the other nodes of the cluster. In the necessary case, the nodes receiving the subtasks repeat the same process recursively. The subtasks of the different nodes are processed each one in their respective map function and will return an intermediate result. The second step, called "Reduce", consists of putting all the results back to their respective parent nodes. The results thus go up from the lowest node to the root. With the Reduce function, each node will calculate a partial result in associating all the values corresponding to the same key into a single pair (key - value). Once this pair (key-value) is obtained, the result is sent back to the parent node, which will repeat the same process recursively to the root node. When a node finishes processing a task, a new block of data is assigned to a Map task. Here is a diagram illustrating all that.

Here is an example of use. The problem here will be to count the number of occurrences of each word. As input data we will have text files. The purpose of the Map function is to decompose the text into a couple of key-value words. The Reduce function will take couples with the same key and count all occurrences to make only one key-value pair per word. [(m, [1,1,1, ...]) m2, [1,1,1, ..], ...]. The illustration shows only part of the process and is incomplete.

Advantages and disadvantages

Advantages Here is a list of some benefits that MapReduce brings: Ease of use: the user can focus on his own code and neglect the aspect of distributed processing, treated in a completely transparent way. Versatility: It is adaptable to many types of data processing, such as data mining, size calculations of several thousand documents. Its ability to break down a process into multiple distributable tasks on a multitude of nodes.

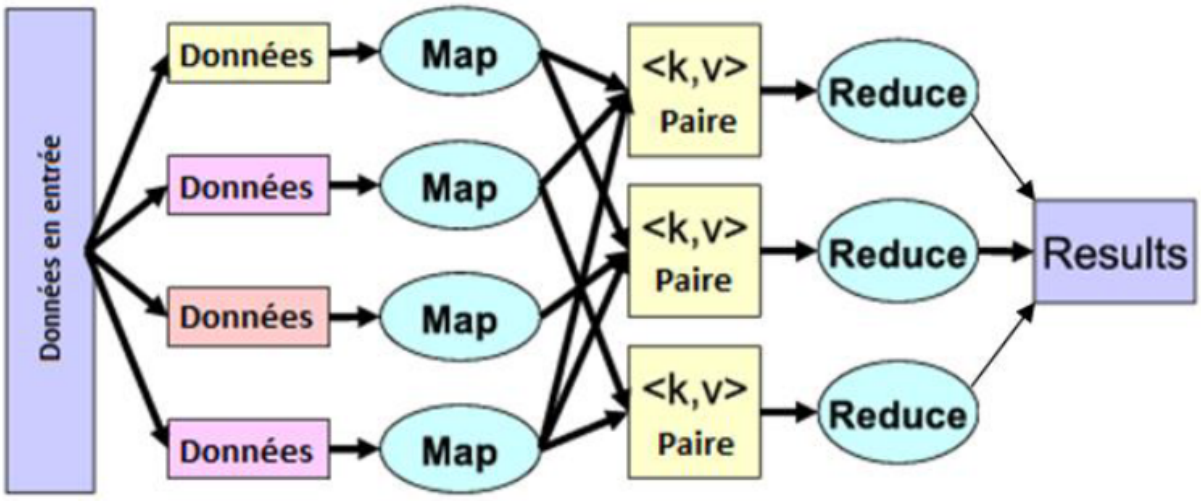

Figure 3: Example of MapReduce functioning

Disadvantages Here is a list of some disadvantages of MapReduce: There is only one entry for the data. As a result, algorithms that require multiple input elements are poorly supported, as MapReduce is designed to read a single input element and produce an output element. Fault tolerance and horizontal scalability mean that MapReduce operations are not always optimized for input / output. In addition, the two-step data stream makes it very rigid, because to go to the next step, the current step must be completed. All of these problems reduce performance. Does not support high-level languages such as SQL.

0.5 Cloudant

Cloudant is a document-oriented DataBase as a Service (DBaaS). It stores data as documents in JSON format. It's built with scalability, high availability, and durability in mind. It comes with a wide variety of indexing options including map-reduce, IBM Cloudant NoSQL DB for IBM Cloud Query, full-text indexing, and geospatial indexing. The replication capabilities make it easy to keep data in sync between database clusters, desktop PCs, and mobile devices.

The storage

Cloudant NoSQL DB stores documents using JSON (JavaScript Object Notation) encoding, so anything encoded into JSON can be stored as a document. Files containing media, such as images, videos, and audio, are called BLOBs (Binary Large OBjects), and can be stored as attachments associated with documents.

0.6 Cloudant query

SELECT * FROM Cloudant

Anyone working in database administration or software development in the last two decades probably used SQL (Structured Query Language). The applications work with the data, so that an administrator / developer would learn a quantity of MySQL, MSSQL, DB2, or even Microsoft Access RDBMS - enough to persist their data and write queries to retrieve and parse them. This means that most administrators / developers are used to creating expressions like this: For new users, it often helps to draw some rough comparisons between the SQL they already know and the querying and indexing systems present in Cloudant.

Drop the JOINs

With Cloudant, records are organized into JSON documents, not broken into tables made of rows and columns. This approach eliminates the concept of a FROM clause altogether. And because you dont have tables, you cant JOIN them together, either. Instead, you unnormalized relational data when moving it into Cloudant, so more of it is stored together in the first place.This means you can send JSON requests to DB with queries similar to the following: In SQL terms, you can treat the selector JSON as being like the SQL WHERE clause. Similarly, the fields is like SELECT, sort is like ORDER BY and the limit and skip fields are roughly equivalent to LIMIT and OFFSET. All that said, selector is the only required portion. However, unlike in SQL where you can simply write and execute a valid query and it will work, even if it isnt performant, you cant do this with Cloudant Query.

0.7 Implementation

For this part w have used a trial version of IBM Cloudant by creating a Bluemix account. Then we have follow the instruction until the end of the configuration. We have created 15000 document to be able to estimate the necessary processing time.

Cloudant is a NoSQL document database meaning that a database contains a collection of JSON formatted documents. Now you can use the Cloudant Dashboard to create a database, and then to add data to your database, you add a new document to the database.

0.8 Conclusion

This document is an overview and a real world test of Cloudant.The conclusion of this is that a document store is a lightweight fast and reliable database optimized for the web and high load application with its lock free multi access. The database installation and maintenance is really straightforward and simple and the database request are plain HTTP request that require a few lines of code. But even with all those advantages, CloudantDB is not always the best document store.MongoDB is recommended in application that dont need

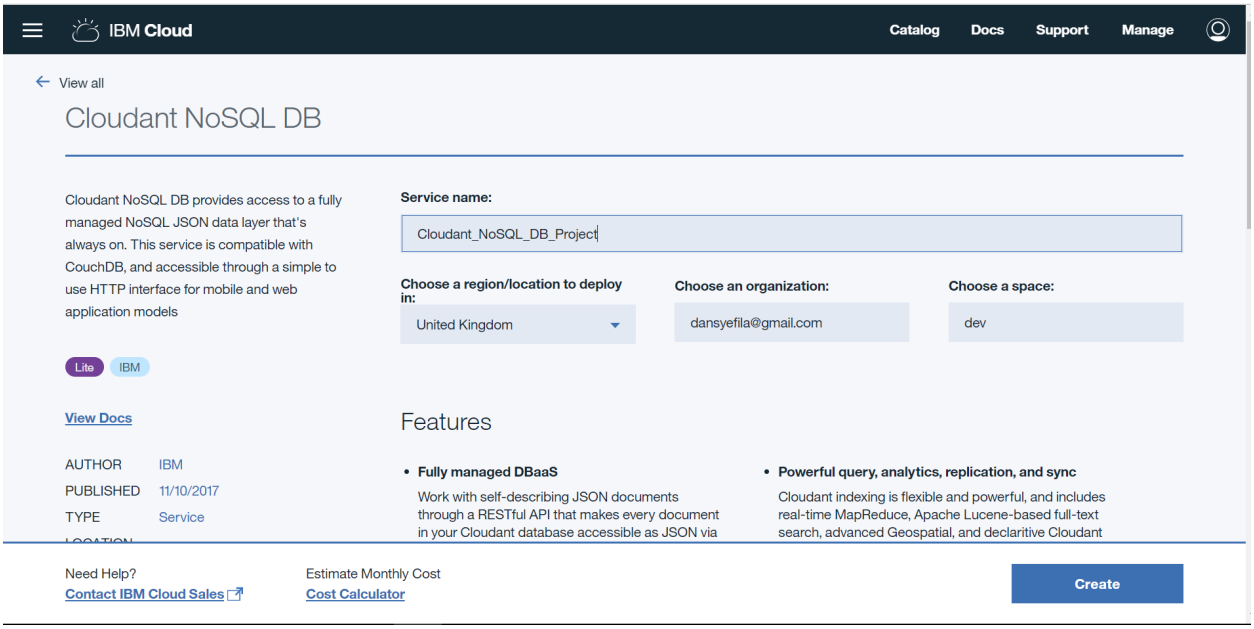

Figure 4: Creation of a cloudant DB

the consistency and the history feature of Cloudant but benefice more of the speed advantage of MongoDB. This shows that even when a No-SQL database is better, CloudantDB could not be the recommended one to use When the application needs complex request like join or relation, SQL is still more appropriate. Document store also doesnt guaranty that an application will always access the last version of the data. Because of those, its not destined to replace SQL but rather to complement it.