HBase web application
Big Data Processing

November 27, 2015

This assignment will cover HBase. You will learn how to:

- Design a schema.
- Use HBase with the command line.
- Use the Java API to access HBase.

Warming up

The lecture briefly covers some of the most important aspects of schema design in HBase. In order to get a more thorough understanding of how HBase differs from RDBMS and what drives schema design in HBase, you should read at least these three things:

- HBase reference guide - Data Model [1]
- HBase reference guice - Schema Design [2]
- Google Big Table - Schema Design [3]

For more in dept information, we also advise you to look at the Google Big Table paper.[4] It is easy to read and provides insight into how HBase works.

After reading the above three articles, you should be able to answer the following questions. Make sure you always include a clear line of reasoning in your answer. Just stating facts without explanation will not gain you points.

Exercise 1

The purpose of a row key is, of course, to uniquely identify a row. But it has another very important goal! What is that goal?

Exercise 2

Imagine you wish to store a large collection of web pages. You want to identify each site by it's domain name. Explain why it is better to store each site with it's reversed domain name as row key.

I.e.: Explain why it is better to use **org.apache.hbase** instead of **hbase.apache.org**.

Hint: use your answer from exercise 1

Exercise 3

[1] `https://hbase.apache.org/book.html#datamodel`
[2] `https://hbase.apache.org/book.html#schema`
[3] `https://cloud.google.com/bigtable/docs/schema-design`
[4] `https://static.googleusercontent.com/media/research.google.com/en//archive/bigtable-osdi06.pdf`

In traditional RDMBS, it is very common to use an auto-incrementing value as primary key. Using an auto-incrementing row key in HBase however, could cause performance problems. Explain why.

Exercise 4

In traditional database systems, it is considered bad practice to denormalize data (That is, store the same data multiple times to avoid joins). Is this also the case with HBase? Explain why or why not.

Exercise 5

Scalability is nice, but there are some severe drawbacks when comparing HBase to traditional databases. One of which is that HBase is not ACID compliant. Think of **two other** disadvantages.

Introducing ImPress

By now you hopefully got a taste of how HBase can and cannot be used. Based on your amazing skill in reading the reference guide, you are hired at ImPress, a start-up building an (equivalently named) image sharing service for photographers and artists.

The web-service allows people to post high-resolution images to their profile page, and view other peoples profiles. In it's current version though, there is a problem. When signing up for an account, or uploading an image, this data is kept only in the applications working memory (RAM). The application never persists any data to disk! This means that all data is lost when the company's server shuts down. Worse: the application can only handle as much data as fits in memory!

It is your task to replace this in-memory storage with persistent storage. You believe HBase is the way to go and start working.

Installing the web-application

To develop ImPress, you need a Cloudera VM inside Virtualbox or VMWare. Any code you write will be inside this VM because HBase will not work outside Cloudera.

1. Launch your Cloudera VM and open a terminal.
2. Type the following **exact** commands

```
cd ~/workspace
git clone https://github.com/txsmith/ImPress.git
```

3. Launch Eclipse.
4. Go to `File` → `Import...`
5. Choose `Maven` → `Existing Maven Projects`
6. Browse to the `ImPress` folder and click `OK`, `Finish`.

This could take a while! If all goes well you should now have the complete web application imported in Eclipse. To run the application, simply select `start-webapp` from the run or debug menu. You will see a lot of messages in the console. The output should eventually show these lines:

```
...
2015-11-02 04:22:35.350:INFO:oejs.Server:main: Started @32003ms
[INFO] Started Jetty Server
```

You can now start a browser and go to `localhost:8080` to test the application.

Designing the Schema

In Eclipse, open the package `nl.impress.app.data`. This package has two objects that have to be persisted: **User** and **Upload**. Each field of these objects has to be stored and retrieved with HBase. To see what exactly you have to store, look at the fields that these classes contain.

Exercise 6

When storing images, what would you choose as the row key? Why?

Each image is stored in three different sizes: thumbnail, medium and full-resolution. Thumbnails are shown on the home page of the website, medium images are shown on the profile page, and the full-resolution images are only accessed when the 'download' button is clicked.

Exercise 7

Knowing this, what column family or families would you create for storing the image data? Motivate your choice.

Exercise 8

Describe briefly how you would store user data (username, password, e-mail address, age, etc).

Exercise 9

Imagine: ImPress launches successfully and grows to 100 million users. Now the marketing department wants to analyse user profiles based on their age. With your schema in mind, how would you be able to query all users between 20 and 30 years old? Would this cause any problems, and if so how would you solve them?

Creating HBase tables via the command line

Enough theory, let get our hands dirty! Open a terminal in Cloudera and type the following to launch the HBase shell:

```
hbase shell
```

It may take a while for the HBase shell to start. Don't mind the deprecation warnings, they are normal. When the shell has started, type:

```
help
```

for a list of available commands. If you have trouble with the long output of the `help` command, it may be more convenient to refer to the HBase wiki[5].

Exercise 10

Take your results from exercise 7, 8 and 9 and create the schema! Provide a list of all commands you used to do so.

Test first

Now let's step it up a bit, open the ImPress project in Eclipse on Cloudera and look in the `src/test/java` folder. It contains a test hierarchy for the `Storage` classes.

Run the `UserStorageTest` class and note that the tests pass on the `InMemoryUserStorage` but fail for `HBaseUserStorage`.

Exercise 11

Open the class: `nl.tudelft.bigdata.data.`**`HBaseUserStorage`** and implement the class. Make sure the tests all pass!

Important note: you have to implement the **`cleanup`** method in `HBaseUserStorageTest` in order for the tests to work properly. If you don't, your database will be polluted with test-data.

Useful resources:

- HBase Java API reference[6]
- HBase Reference Guide - Examples [7]

Switching to `HBaseUserStorage`

When your tests pass, the `InMemoryUserStorage` can be replaced by `HBaseUserStorage` in the real application code. To do this, we need to change one line of code in the **MainModule** class.

```
1 public void configureServlets() {
2    // ...
3    bind(UserStorage.class).to(InMemoryUserStorage.class);
4    // ...
5 }
```

This method in the `MainModule` class tells the application what classes should be used in order make things work. Don't mind the other lines; we only care about the line highlighted above. It configures the application to always use `InMemoryUserStorage` as concrete class whenever a `UserStorage` is needed.

Changing this to:

```
1 bind(UserStorage.class).to(HBaseUserStorage.class);
```

makes the switch to our HBase implementation. Run the application to see if it works! User data should now persist even if you restart the application.

[5] `https://wiki.apache.org/hadoop/Hbase/Shell`
[6] `https://hbase.apache.org/apidocs/`
[7] `https://hbase.apache.org/book.html#_examples`

Implementing `HBaseImageStorage`

Image data however, is still stored using the `InMemoryImageStorage` class. Let's change this. First run the `ImageStorageTest` class and note that the tests pass on the `InMemoryImageStorage` but fail for `HBaseImageStorage`.

Exercise 12

Open the class: `nl.tudelft.bigdata.data.`**`HBaseImageStorage`** and implement the class. Make sure the tests all pass!

Important note: you have to implement the **`cleanup`** method in `HBaseImageStorageTest` in order for the tests to work properly.

Switching to `HBaseImageStorage`

When your tests pass, the `InMemoryImageStorage` can be replaced by `HBaseImageStorage` in the real application code. To do this, we again need to change one line of code in the **MainModule** class:

```
1 public void configureServlets() {
2    // ...
3    bind(ImageStorage.class).to(InMemoryImageStorage.class);
4    // ...
5 }
```

Change this to:

```
1 bind(ImageStorage.class).to(HBaseImageStorage.class);
```

Run the application and check if everything works as expected.

Devlivery instructions

1. Put a **`PDF file`** with answers to each exercise inside the project's **`assignment`** folder.
2. Make a **`ZIP file`** of the entire project using the following commands in the terminal. Substitute `<student-number>` with your own student number.

```
cd ~/workspace/ImPress
mvn clean
zip -r <student-number>.zip . -x *.git* *.settings* .classpath .project
```

3. Deliver the resulting zip file.