HDFS KV-Store

Large Scale Data Management

2018/2019

The main goal of this guide is to introduce the HDFS environment and to develop a Key-Value Store over the HDFS file system.

Documentation is available at:

- HDFS
- Hadoop The Definitive Guide - Book
- FileSystem API
- SequenceFile API
- Writable Data Types

Deployment

Requirements: Java 8 Oracle or above.

1. Download **Hadoop 2.9.2** binary.

```
wget
http://mirrors.up.pt/pub/apache/hadoop/common/hadoop-2.9.2/hadoop-2.9.2.tar
.gz
```

2. Update the ***etc/hadoop/*-site.xml*** files according the templates.
3. Define the java version on ***hadoop-env.sh*** (*export JAVA_HOME= …)*
4. Add the Hadoop variables (templates) to ***~/.bashrc***, and ***source*** it

```
source ~/.bashrc
```

5. Generate a *ssh-key* for HDFS elements communication

```
ssh-keygen -t rsa -P "" -f hdfs_key
cat ~/.ssh/hdfs_key.pub >> ~/.ssh/authorized_keys
```

6. Format NameNode

```
hdfs namenode -format
```

7. Start HDFS

```
cd hadoop-2.9.2/sbin/
./start-dfs.sh
```

8. Validate which Hadoop daemons are running

```
jps

6949 DataNode
6810 NameNode
7132 SecondaryNameNode
7282 Jps
```

9. Check the HDFS WebUI

```
http://localhost:50070/
```

10. To stop HDFS

```
./stop-dfs.sh
```

KVStore: Part 1

The main goal of this part is to create a simple key-value store using the HDFS-client. The KV-Store should implement the ***KVinterface*** (provided with the guide's codebase). All key-value pairs must be written/read to/from ***Hadoop SequenceFiles*** and must be represented with the *Writable data types* provided by *hadoop.io* API***.***

KVInterface:

```
boolean put (int key, String value);
String get (int key);
Object scan ();
```

Creating a Writer:

```
SequenceFile.Writer writer = SequenceFile.createWriter (
      configuration,
      SequenceFile.Writer.file(path),
      SequenceFile.Writer.keyClass(key class),
      SequenceFile.Writer.valueClass(value class),
      SequenceFile.Writer.appendIfExists(true));
```

Creating a Reader:

```
SequenceFile.Reader reader = new SequenceFile.Reader (
      configuration,
      SequenceFile.Reader.file(path),
```

```
SequenceFile.Reader.bufferSize(buffer size));
```

KVStore: Part 2

The goal of this part is to extend the previous KV-Store solution by supporting the insertion of two values for each entry of the KV-Store.

```
[Key]: [Value1, Value2]
```

The KV-Store should now follow the ***KVInterface2*** and all pairs must be written/read to/from ***Hadoop SequenceFiles***. Additionally, each value should now comprehend a **timestamp** marking its creation or modification, and a **status** element to mark its state.

KVInterface2:

```
boolean put (int key, COLS columns, Object content);
String get (int key, COLS columns);
boolean update (int key, COLS columns, Object content);
boolean delete (int key, COLS column);
Object scan (COLS column);
```

KVStore: Part 3

1. Instead of writing all key-value entries to a single ***SequenceFile***, how could one optimize the key-value entries distribution?
2. Considering the previous key-value entries distribution, how could one optimize the *scan* method?
3. Implement a read and a write caches that serve read- and write-based requests, respectively. All values should now be written to the *WriteCache,* which in turn will then write a ***SequenceFile*** to the HDFS file system.

Learning outcomes

Experiment the HDFS file system interface. Understand the challenges and design choices involved on developing HDFS applications.