Load Data into HBase

Goal

This is a warm-up practice to provide background knowledge of Hadoop MapReduce and HBase Database/Datastore. The following practice will cover:

- Overview for Hadoop and HBase
- Set up the environment for Hadoop and HBase
- Create HBase Database tables for ClueWeb09 [1] data, and load them into HBase
- Test the stored HBase records with the provided java executable

Deliverables

Submit an output file dataTable1.txt containing your results to the Canvas Assignments page by using the provided java class iu.pti.hbaseapp.HBaseTableReader.

Evaluation

None

Introduction

The development of data-intensive problems in recent years has brought new requirements and challenges to storage and computing infrastructures. Researchers are not only doing batch loading and processing of large-scale data, but also demanding the capabilities of incremental updating and interactive analysis. Therefore, extending existing storage systems to handle these new requirements becomes an important research challenge.
In this exercise, we will introduce technologies related to data-intensive computing and storage, namely Hadoop and HBase.

Hadoop MapReduce

The Apache Hadoop MapReduce software library is a framework that allows for the distributed processing of large datasets across clusters of computers using simple programming models. It is designed to scale up from single servers to thousands of machines, each offering local computation and storage. Rather than rely on hardware to deliver high availability, the library itself is designed to detect and handle failures at the application layer, delivering a highly available service on top of a cluster of computers, each of which may be prone to failures [2].
Hadoop has an architecture consisting of a master node with many client workers and uses a global queue for task scheduling, thus achieving natural load balancing among the tasks. The MapReduce

model reduces the data transfer overhead by overlapping data communication with computations when reduce steps are involved. Hadoop performs duplicate executions of slower tasks and handles failures by rerunning the failed tasks using different workers. Data is stored on the Hadoop Distributed File System (HDFS), a distributed parallel file system for data storage, which stores the data across the local disks of the computing nodes while presenting a single file system view through the HDFS API. The architecture is shown in Figure 1.

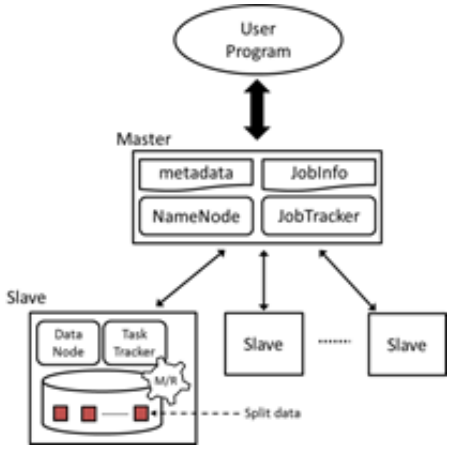

Figure 1. Hadoop MapReduce Architecture

This practice session and upcoming projects mainly use Hadoop MapReduce as our programming model and execution framework.

HBase

HBase is an open source, distributed, column-oriented, and sorted-map datastore modeled after Google's BigTable. Figure 2 shows the data model of HBase. Data is stored in tables; each table contains multiple rows and a fixed number of column families. For every row, there can be a varied amount of qualifiers within a column family, and at the intersections of rows and qualifiers are table cells. Cell contents are both uninterpreted and versioned arrays of bytes. A table can be configured to maintain a certain number of versions for its cell contents. Rows are sorted by row keys, which are implemented as byte arrays [3-4].

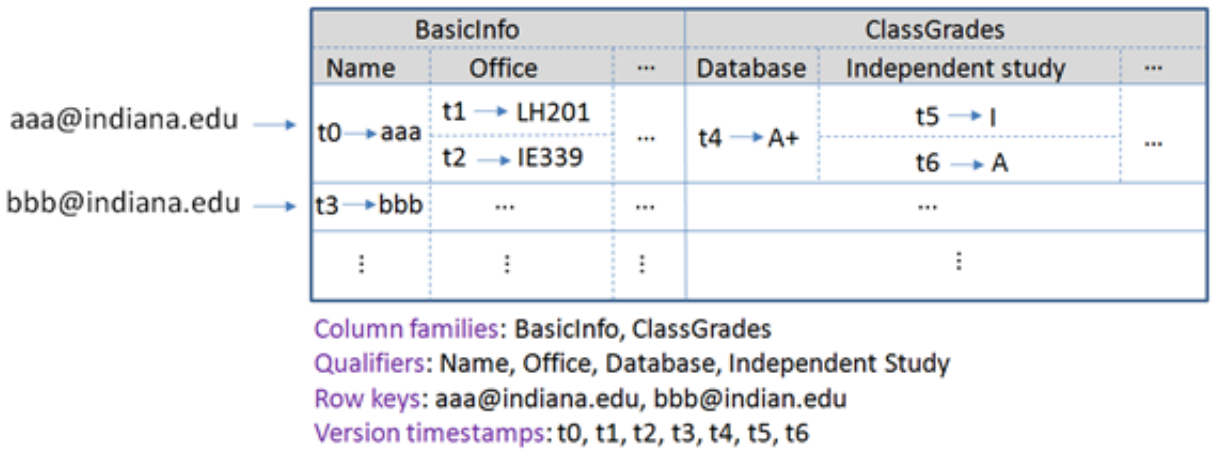

Figure 2. An example of HBase table structure

This practice and upcoming projects use HBase as a datastore which stores the raw input data and the output results, even serving as a database for our search engine.

Exercise: Create Tables and Load Data to HBase

This section provides command-line steps to configure the working environment, start Hadoop and HBase, create HBase tables, load data into HBase, and view the uploaded records. Note that these steps must be executed under the prepared virtual box in your machine.

Start Hadoop and HBase

The following steps direct you to the locations of Hadoop and HBase and give instructions on how to start them. For Hadoop, a provided one-click shell script, MultiNodesOneClickStartUp.sh, is used to automatically start the Hadoop framework. For HBase, we use the default starter.

```
# start hadoop
$ cd /root/software/hadoop-1.1.2/
$ . ./MultiNodesOneClickStartUp.sh /root/software/jdk1.6.0_33/ nodes

# start hbase
$ cd /root/software/hbase-0.94.7/
$ ./bin/start-hbase.sh
```

Configure the working environment

Once Hadoop and HBase have started, we need to copy a configuration file hbase-site.xml to Hadoop's conf directory and update the environment parameter HADOOP_CLASSPATH.

```
# prepare for hadoop and hbase environment

$ cp /root/software/hbase-0.94.7/conf/hbase-site.xml /root/software/hadoop-1.1.2/conf/
$ cd /root/software/hadoop-1.1.2/
$ export HADOOP_CLASSPATH=`/root/software/hbase-0.94.7/bin/hbase classpath`
```

Load data into HBase Tables

After the working environment is ready, two HBase Tables, WordCountTable and clueWeb09DataTable for this homework, are created by the provided java class iu.pti.hbaseapp.clueweb09.TableCreatorClueWeb09. Then we upload the clueWeb09 data to HBase using helper iu.pti.hbaseapp.clueweb09.DataLoaderClueWeb09.

```
export HADOOP_CLASSPATH=`/root/software/hbase-0.94.7/bin/hbase classpath`

# create hbase tables
$ ./bin/hadoop jar lib/cglHBaseMooc.jar iu.pti.hbaseapp.clueweb09.TableCreatorClueWeb09

# create one directory for mapreduce data input
$ mkdir -p /root/MoocHomeworks/HBaseWordCount/data/clueweb09/mrInput

# create input's metadata for HBbase data loader
$ ./bin/hadoop jar lib/cglHBaseMooc.jar iu.pti.hbaseapp.clueweb09.Helpers create-mr-inpu
t /root/MoocHomeworks/HBaseWordCount/data/clueweb09/files/ /root/MoocHomeworks/HBaseWord
Count/data/clueweb09/mrInput/ 1

# copy metadata to Hadoop HDFS
$ ./bin/hadoop dfs -copyFromLocal /root/MoocHomeworks/HBaseWordCount/data/clueweb09/mrIn
put/ /cw09LoadInput
$ ./bin/hadoop dfs -ls /cw09LoadInput

# load data into HBase (takes 10-20 minutes to finish)
$ ./bin/hadoop jar lib/cglHBaseMooc.jar iu.pti.hbaseapp.clueweb09.DataLoaderClueWeb09 /c
w09LoadInput
```

Verify data record from HBase output

Finally, we verify the uploaded records with the iu.pti.hbaseapp.clueweb09.DataLoaderClueWeb09 from the clueWeb09DataTable. The screen redirects the output to dataTable1.txt. This file is used for evaluation and must be uploaded to the Oncourse Assignments page.

```
  $ ./bin/hadoop jar lib/cglHBaseMooc.jar iu.pti.hbaseapp.HBaseTableReader  clueWeb09Dat
aTable details string string string string 1 > dataTable1.txt

  $ vim dataTable1.txt
```

What's next?

You are ready to program your own HBase WordCount.

References

1. Clueweb09 project, http://lemurproject.org/clueweb09/ (http://lemurproject.org/clueweb09/)
2. Hadoop official website, http://hadoop.apache.org/ (http://hadoop.apache.org/)
3. HBase official website, http://hbase.apache.org/ (http://hbase.apache.org/)
4. Xiaoming Gao, Vaibhav Nachankar, and Judy Qiu. 2011. Experimenting lucene index on HBase in