LOGBASE

Getting Started Guide

Release 0.1.b1

Database System Research Group

School of Computing

National University of Singapore

May 19, 2013

CONTENTS

CHAPTER

ONE

INTRODUCTION

LogBase [1] is an open-source, scalable log-structured database system in the Cloud that adopts log-only storage structure for removing the write bottleneck observed in write-heavy environments, e.g., continuous stream processing.

LogBase leverages the Hadoop Distributed File System (HDFS) [2] to maintain log files, which constitute the only data repository in the system. LogBase is implemented in Java, inherits basic infrastructures from the open-source HBase [3], and adds new features for log-structured storages including access to log files and in-memory indexes.

In this release, LogBase provides basic API for schema definition and data operations. In the next releases, advanced features that are in the pipeline include secondary indexes, query processing engine, and system recovery. Further information of LogBase project can be found at [1].

1. http://www.comp.nus.edu.sg/~logbase/
2. http://hadoop.apache.org
3. http://hbase.apache.org/

The remaining chapters of this document are organized as follows. Chapter 2 introduces the data model of LogBase, and the provided API that handles data managed in LogBase. Example codes on how to use this API are also presented in this chapter. Chapter 3 elaborates on the configuration steps and running LogBase.

Please feel free to contact us at the following email addresses should you have any query.
huanghao@comp.nus.edu.sg
wangsh@comp.nus.edu.sg
voht@comp.nus.edu.sg

CHAPTER

TWO

DATA MODEL AND API

2. 1 Data Model

While most scalable cloud storage systems such as Cassandra and HBase employ key-value model, the data model of LogBase is based on the widely-accepted relational data model where data are stored as tuples in relations, i.e., tables, and a tuple comprises of multiple attributes' values. This design choice ensures the suitability of LogBase to provide scalable storage services for database-centric applications in the Cloud.

2. 2 API

The API for handling data in LogBase is defined in the following package in the source code folder: *./src/main/java/sg/edu/nus/LogBaseAPI*.

LogBase's API is divided into two groups, namely *LogAdmin* for managing tables' schema, and *LogTable* for accessing data within a table.

(1) API for *LogAdmin*

(1.1) **public void createTable(String tableName, String[] columnName)**

Description:
This function creates a table with a specified table name and column names in LogBase.

Parameters:

tableName - The String of the specified table name.

columnName - The String array of which each element is one of the specified column names.

(1.2) **public LogTable getExistingTable(String tableName)**

Description:
This function checks whether there is a table with a specified table name in LogBase.

Parameters:
tableName - The String of the specified table name.
columnName - The String array of which each element is one of the specified column names.

Returns:
If the table exists in LogBase, this function returns a LogTable object handler for users to access the data within the table; otherwise, the function returns null.

(2) API in *LogTable*

(2.1) **public void put(byte[] row, byte[] column, byte[] value)**

Description:
This function puts a new value at specified row and column of a LogTable which invokes the function.

Parameters:
row - The name of the specified row in format of byte[].
column - The name of the specified column in format of byte[].
value - The value to be record with format of byte[].

(2.2) **public void put(byte[] row, byte[][] columns, byte[][] values)**

Description:
This function puts new values at specified row and columns of a LogTable which invokes the function.

Parameters:
row - The name of the specified row in format of byte[].
columns - The names of the specified columns in format of byte[] array of which each element refers to a column name.

values - The values to be record with format of byte[] array of which each element is a value to be recorded at the corresponding column.

(2.3) **public Result get(byte[] row)**

Description:
This function retrieves the newest version of the record at a specified row of a LogTable which invokes the function.

Parameters:
row - The name of the specified row in format of byte[].

(2.4) **public Result get(byte[] row, byte[] column)**

Description:
This function retrieves the newest record at specified row and column of a LogTable which invokes the function.

Parameters:
row - The name of the specified row in format of byte[].
column - The name of the specified column in format of byte[].

(2.5) **public Result get(byte[] row, byte[][] columns)**

Description:
This function retrieves the newest record at specified row and columns of a LogTable which invokes the function.

Parameters:
row - The name of the specified row in format of byte[].
columns - The names of the specified columns in format of byte[] array of which each element refers to a column name.

2. 3 Example Codes Using API

We provide sample codes that use both LogAdmin and LogTable API in the following class sg.edu.nus.test.Test:

```
package sg.edu.nus.test;
import java.io.IOException;
import org.apache.hadoop.hbase.HBaseConfiguration;
import org.apache.hadoop.hbase.client.Result;
import org.apache.hadoop.hbase.util.Bytes;
import sg.edu.nus.LogBaseAPI.LogAdmin;
import sg.edu.nus.LogBaseAPI.LogTable;
```

```

public class Test {
    public static void main (String[] argv) throws IOException{
        LogAdmin admin = new LogAdmin(HBaseConfiguration.create());
        //Step 1. Create a table with a specified table name and a specified column name.
        admin.createTable("testLogBase", new String[]{"c1", "c2"});
        final LogTable table = admin.getExistingTable("testLogBase");
        //Step 2 Insert a record in the table
        byte[] row = Bytes.toBytes("r1"); //row name
        byte[][] cols = new byte[][]{Bytes.toBytes("c1"), Bytes.toBytes("c2")}; //columns' names
        byte[][] value = new byte[][]{Bytes.toBytes("v1"), Bytes.toBytes("v2")}; //values
        table.put(row, cols, value);
        //Step 3. Get a record with row name as the key
        Result ret = table.get(row);
        for(int i=0; i<ret.size(); i++){
          System.out.println(" get = " + (new String (ret.raw()[i].getValue())));
        }
        //Step 4. Get a record with row name and column name as keys
        ret = table.get(row, Bytes.toBytes("c1"));
        for(int i=0; i<ret.size(); i++){
          System.out.println(" get = " + new String (ret.raw()[i].getValue()));
        }
    }
}
```

CHAPTER

THREE

GETTING STARTED

In this chapter, we elaborate how to set up the configurations and get started with LogBase in cluster environments.

3. 1 Preliminary Setup

3.1.1 Configure Loopback IP Address

Open */etc/hosts* and modify loopback IP address as 127.0.0.1 (the default setting is 127.0.1.1), then there should looks like this:

```
127.0.0.1 localhost
127.0.0.1 ubuntu.ubuntu-domain ubuntu
```

3.1.2 Download and Unpack the Latest Stable Release of LogBase

The latest stable release of LogBase can be download from the project web page: http://www.comp.nus.edu.sg/~logbase/.

Then, an environment variable should be added into the environment file as follows:
Open */etc/environment*, and insert the following string

```
HBASE_HOME="/home/logbase-0.1.b1"
```

where "/home/logbase-0.1.b1" is an example of the path where the logbase files are unpacked. Please change it if you unpack the logbase files in another file folder. Note that the server has to be rebooted for the environment variable to take effect.

3. 2 Configuration for Hadoop

The distributed environment configurations include the following steps.

(1) ssh
ssh must be installed and sshd must be running to use Hadoop's scripts to manage remote Hadoop and HBase daemons. You must be able to ssh to all nodes, including your local node, using passwordless login (Google “ssh passwordless login” for reference).

(2) NTP
Inconsistent system time in a cluster may results in querying problem and “weird” cluster operations. Thus, please run NTP on your cluster before running LogBase. Detail information of NTP can be found at
http://en.wikipedia.org/wiki/Network_Time_Protocol.

(3) ulimit and nproc
Massive files are used in LogBase at a same time. Hence, you should adjust the upper bound on the number of file descriptors, since the default user file limit is only 1024 on most Linux systems (Type unimit -n in the Terminal, you can see the default setting). We suggest increasing the user file limit to 10K.

In addition, to avoid OutOfMemoryError, you should also increase the users’ nproc setting. We also suggest setting it as 10K.

(4) Hadoop configurations
Since LogBase leverage HDFS to maintain log files. Therefore, make sure that you have installed Hadoop version 0.20.2, and then follow the examples below to configure the *core-site.xml*, *hdfs-site.xml*, *masters*, and *slaves* as required by Hadoop.

(4.1) Configure *core-site.xml* like this:

```
<configuration>
  <property>
    <name>fs.default.name</name>
    <value>hdfs://awan-1-23-0:12345</value>
    <description>Name node.</description>
  </property>
  <property>
    <name> hadoop.tmp.dir </name>
    <value>/data1/epic-data/hadoop/tmp</value>
  </property>
  <property>
    <name>fs.inmemory.size.mb</name>
    <value>200</value>
    <description>Larger amount of memory allocated for the in-memory file-system used to
      merge map-outputs at the
    reduces.</description>
  </property>
  <property>
    <name>io.sort.factor</name>
    <value>100</value>
    <description>The number of streams to merge at once while sorting files. This determines
    the number of open file handles.</description>
  </property>
  <property>
    <name>io.sort.mb</name>
    <value>200</value>
    <description>The total amount of buffer memory to use while sorting files, in megabytes.
    By default, gives each merge stream 1MB, which should minimize seeks.</description>
  </property>
  <property>
    <name>io.file.buffer.size</name>
    <value>131072</value>
    <description>The size of buffer for use in sequence files. The size of this buffer should
      probably be a multiple of hardware page size (4096 on Intel x86), and it determines how
      much data is buffered during read and write operations.</description>
  </property>
</configuration>
```

Please modify the values of "fs.default.name" and "hadoop.tmp.dir" according to your own situations.

(4.2) Configure *hdfs-site.xml* like this:

```
<configuration>
  <property>
    <name>dfs.replication</name>
    <value>3</value>
  </property>
  <property>
    <name>dfs.block.size</name>
    <value>134217728</value>
    <description>The default block size for new files.</description>
  </property>
  <property>
    <name>dfs.http.address</name>
    <value>0.0.0.0:50070</value>
    <description>
    The address and the base port where the dfs namenode web ui will listen on.
    If the port is 0 then the server will start on a free port. </description>
  </property>
  <property>
```

```
        <name>dfs.secondary.http.address</name>
        <value>0.0.0.0:5004</value>
        <description>
          The secondary namenode http server address and port. If the port is 0 then the server
          will start on a free port. </description>
    </property>
    <property>
        <name>dfs.datanode.address</name>
        <value>0.0.0.0:5003</value>
        <description>
        The address where the datanode server will listen to. If the port is 0 then the server will
        start on a free port. </description>
    </property>
    <property>
        <name>dfs.datanode.ipc.address</name>
        <value>0.0.0.0:5002</value>
        <description>
          The datanode ipc server address and port. If the port is 0 then the server will start on a
          free port. </description>
    </property>
    <property>
        <name>dfs.datanode.http.address</name>
        <value>0.0.0.0:5001</value>
        <description>
        The datanode http server address and port. If the port is 0 then the server will start on a
        free port. </description>
    </property>
</configuration>
```

The above “address” values can be reset according to your own situations.

3. 3   Configure LogBase

(1) conf/hbase-site.xml

Open the file *./conf/hbase-site.xml*, and edit it like this:

```
<configuration>
  <property>
        <name>hbase.zookeeper.quorum</name>
        <value>awan-0-11-0</value>
        <description>The directory shared by RegionServers.
        </description>
      </property>
<property>
        <name>hbase.zookeeper.property.dataDir</name>
        <value>/data1/epic-data/logbase/data</value>
        <description>Property from ZooKeeper's config zoo.cfg.
        The directory where the snapshot is stored. </description>
      </property>
      <property>
        <name>hbase.rootdir</name>
        <value>hdfs://awan-1-23-0:12345/logbase</value>
        <description>The directory shared by RegionServers.
        </description>
      </property>
      <property>
        <name>hbase.cluster.distributed</name>
        <value>true</value>
      </property>
      <property>
        <name>hbase.master.maxclockskew</name>
        <value>180000</value>
      </property>
</configuration>
```

Note that the node information ("awan-1-23-0") and the port ("12345") in the value of "hbase.rootdir" should be consistent with those in the value of "fs.default.name" in Hadoop settings.

(2) conf/regionservers

Open the file *./conf/regionservers*, and list the nodes which work as regionservers:

```
awan-1-00-0
awan-1-01-0
awan-1-02-0
```

We suggest setting the nodes of regionservers being consistent with the data nodes in Hadoop.

3.4 Work with LogBase

Make sure that Hadoop is running, then open the command terminal, and change the directory to the LogBase home folder. Then, type the following command to start the LogBase system up.

```
./bin/start-logbase.sh
```

When LogBase has been started, its log tables can be accessed and manipulated via the provided API as described in Chapter 2.

Finally, to stop Logbase, please type the following command in Terminal.

```
/bin/stop-logbase.sh
```