Big Data and Hadoop Development

A C A D G I L D

Page

1

Session

17

:

Overview of

HBase

Assignment

1

Big Data and Hadoop Development

A C A D G I L D

Page

2

Session 17

:

Overview of HBase

Assignment

–

Hbase Introduction

Table of Contents

1.

Introduction

................................

................................

................................

................................

..........

3

2.

Objective

................................

................................

................................

................................

...............

3

3.

Prerequisites

................................

................................

................................

................................

.........

3

4.

Associated Data Files

................................

................................

................................

............................

3

5.

Problem Statement

................................

................................

................................

...............................

3

6.

Expected output

................................

................................

................................

................................

....

4

7.

Approximate Time to Complete Task

................................

................................

................................

...

4

Big Data and Hadoop Development

A C A D G I L D

Page

3

1.

Introduction

In this assignment you need to answer questions on Hbase.

2.

Objective

This assignment will help you to understand Hbase concepts.

3.

Prerequisites

Acadgild’s VM, or Linux operating system with Hadoop installed in it.

4.

Associated Data Files

N/A

5.Problem Statement

Give a brief elaboration of the below questions.

========================================

1.Give a brief difference between HBASE and HDFS.

Ans: HDFS is a distributed file system and has the following properties:  
1. It is optimized for streaming access of large files. You would typically store files that are in the 100s of MB upwards on HDFS and access them through MapReduce to process them in batch mode.  
2. HDFS is optimized for use cases where you write once and read many times like in the case of production logs. You can append to files in some of the recent versions but that is not a feature that is very commonly used. There is no concept of random writes.  
3. HDFS doesn’t do random reads very well.

HBase on the other hand is a distributed column oriented database. The filesystem of choice typically is HDFS owing to the tight integration between HBase and HDFS. Having said that, it doesn’t mean that HBase can’t work on any other filesystem. It’s just not proven in production and at scale to work with anything except HDFS.  
HBase provides you with the following:  
1. It gives you the ability to do random read/writes on your data which HDFS doesnt allow you to.  
2. HBase stores data in the form of key value pairs in a columnar fashion. HBase provides a flexible data model.  
3. Fast scans across tables.  
4. Scale in terms of writes as well as total volume of data.

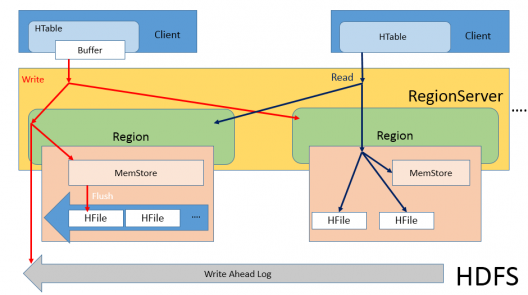
-------------------------------------------------------------------------------

2.List the main components of HBASE.

# Ans:

**HBase Architecture:**

The following figure clearly explains the HBase Architecture.

**[](http://bigdataanalyticsnews.com/wp-content/uploads/2014/01/HBaseARchitecture_PNG2-528x295.png)**

In HBase, there are three main components: Master, Region server and Zoo keeper. The other components are Memstore, HFile and WAL.

As HBase runs on top of HDFS, it utilizes the Master-Slave architecture in which the HMaster will be the master node and the Region Servers are the slave nodes. When the client sends a write request, HMaster gets that request and forward it to the respective Region Server.

---------------------------------------------------------------------------------------------------------------------

3.Does Hbase support sql?

Ans:Yes

----------------------------------------------------------------------------------

4.When should we use HBASE, list some of the scenarios for

the same.

Ans: HBase is a good option only when there are hundreds of millions or billions of rows. HBase can also be used in places when considering to move from an RDBMS to HBase as a complete redesign as opposed to a port.In other words,  HBase is not optimized for classic transactional applications or even relational analytics. It is also not a complete substitute for HDFS when doing large batch MapReduce. Then why should you go for HBase?? If your application has a variable schema where each row is slightly different, then you should look at HBase.

-----------------------------------------------------------------------------------------------------------------------------------------

5.What are the different modes in which Hbase can be run?

Ans: HBase has two run modes: [standalone](http://hbase.apache.org/book.html#standalone) and [distributed](http://hbase.apache.org/book.html#distributed). Out of the box, HBase runs in standalone mode. Whatever your mode, you will need to configure HBase by editing files in the HBase conf directory. At a minimum, you must edit conf/hbase-env.sh to tell HBase which java to use. In this file you set HBase environment variables such as the heapsize and other options for the JVM, the preferred location for log files, etc. Set JAVA\_HOME to point at the root of your java install.

### 5.1. Standalone HBase

This is the default mode. Standalone mode is what is described in the [quickstart](http://hbase.apache.org/book.html#quickstart) section. In standalone mode, HBase does not use HDFS — it uses the local filesystem instead — and it runs all HBase daemons and a local ZooKeeper all up in the same JVM. ZooKeeper binds to a well known port so clients may talk to HBase.

### 5.2. Distributed

Distributed mode can be subdivided into distributed but all daemons run on a single node — a.k.a. pseudo-distributed — and fully-distributed where the daemons are spread across all nodes in the cluster. The pseudo-distributed vs. fully-distributed nomenclature comes from Hadoop.

Pseudo-distributed mode can run against the local filesystem or it can run against an instance of the Hadoop Distributed File System (HDFS). Fully-distributed mode can ONLY run on HDFS. See the Hadoop [documentation](http://hadoop.apache.org/docs/current/) for how to set up HDFS. A good walk-through for setting up HDFS on Hadoop 2 can be found at <http://www.alexjf.net/blog/distributed-systems/hadoop-yarn-installation-definitive-guide>.

#### 5.2.1. Pseudo-distributed

|  |  |
| --- | --- |
|  | Pseudo-Distributed Quickstart  A quickstart has been added to the [quickstart](http://hbase.apache.org/book.html#quickstart) chapter. See [quickstart-pseudo](http://hbase.apache.org/book.html#quickstart_pseudo). Some of the information that was originally in this section has been moved there. |

A pseudo-distributed mode is simply a fully-distributed mode run on a single host. Use this configuration testing and prototyping on HBase. Do not use this configuration for production nor for evaluating HBase performance.

### 5.3. Fully-distributed

By default, HBase runs in standalone mode. Both standalone mode and pseudo-distributed mode are provided for the purposes of small-scale testing. For a production environment, distributed mode is appropriate. In distributed mode, multiple instances of HBase daemons run on multiple servers in the cluster.

Just as in pseudo-distributed mode, a fully distributed configuration requires that you set the hbase-cluster.distributed property to true. Typically, the hbase.rootdir is configured to point to a highly-available HDFS filesystem.

In addition, the cluster is configured so that multiple cluster nodes enlist as RegionServers, ZooKeeper QuorumPeers, and backup HMaster servers. These configuration basics are all demonstrated in [quickstart-fully-distributed](http://hbase.apache.org/book.html#quickstart_fully_distributed).

Distributed RegionServers

Typically, your cluster will contain multiple RegionServers all running on different servers, as well as primary and backup Master and ZooKeeper daemons. The conf/regionservers file on the master server contains a list of hosts whose RegionServers are associated with this cluster. Each host is on a separate line. All hosts listed in this file will have their RegionServer processes started and stopped when the master server starts or stops.

-----------------------------------------------------------------------------

6.Why is zookeeper needed in Hbase?

A distributed Apache HBase installation depends on a running ZooKeeper cluster. All participating nodes and clients need to be able to access the running ZooKeeper ensemble. Apache HBase by default manages a ZooKeeper "cluster" for you. It will start and stop the ZooKeeper ensemble as part of the HBase start/stop process. You can also manage the ZooKeeper ensemble independent of HBase and just point HBase at the cluster it should use. To toggle HBase management of ZooKeeper, use the HBASE\_MANAGES\_ZK variable in conf/hbase-env.sh. This variable, which defaults to true, tells HBase whether to start/stop the ZooKeeper ensemble servers as part of HBase start/stop.

When HBase manages the ZooKeeper ensemble, you can specify ZooKeeper configuration directly in conf/hbase-site.xml. A ZooKeeper configuration option can be set as a property in the HBase hbase-site.xml XML configuration file by prefacing the ZooKeeper option name with hbase.zookeeper.property. For example, the clientPort setting in ZooKeeper can be changed by setting the hbase.zookeeper.property.clientPort property. For all default values used by HBase, including ZooKeeper configuration, see [hbase default configurations](http://hbase.apache.org/book.html#hbase_default_configurations). Look for the hbase.zookeeper.property prefix. For the full list of ZooKeeper configurations, see ZooKeeper’s zoo.cfg. HBase does not ship with a zoo.cfg so you will need to browse the conf directory in an appropriate ZooKeeper download.

You must at least list the ensemble servers in hbase-site.xml using the hbase.zookeeper.quorum property. This property defaults to a single ensemble member at localhost which is not suitable for a fully distributed HBase. (It binds to the local machine only and remote clients will not be able to connect).

©

copyright ACAD

GILD

Zookeeper

•

HBase

uses Zookeeper extensively for region assignment.

•

Zookeeper has simple APIs written in java

•

Directories and files in Zookeeper are called

Znodes

.

•

Clients connect to Zookeeper via a session.

•

Zookeeper is used to ensure only one Master is assigned.

•

HBase

cannot operate without Zookeeper.

------------------------------------------------------------------------------

8 Hbase is a schema less database, what does it mean?

Ans: Moving into a relational database physical schema,

we can adorn the logical diagram with several new attributes that act as primary and foreign keys for relational

database tables.

The many-to-many relationship of questions to positions has an interesting design pattern when translated to the non-relational world. Consider an implementation of this

logical design in another product type, the family

of Bigtable systems (which also includes open source implementations such as Hypertable and HBase). In this setup, we

have more depth than a key/value paradigm, because

the data storage engine does more

with the data in the value itself, providing a more

thorough structure and meta-structure.

Each entity can have "column families" (of which there are a discrete and limited number, established at design time), and with a column family, there can be an unlimited

number of "columns" (which are effectively repeating cells within the column family)

This style of data design is a variant of what is s

ometimes referred to as a

"generic" table, or an "open schema" table.

------------------------------------------------------------------------------

9 What is the minimum number of column family every Hbase table should have?

Ans: Columns in Apache HBase are grouped into column families. All column members of a column family have the same prefix. For example, the columns courses:history and courses:math are both members of the courses column family. The colon character (:) delimits the column family from the column family qualifier. The column family prefix must be composed of printable characters. The qualifying tail, the column family qualifier, can be made of any arbitrary bytes. Column families must be declared up front at schema definition time whereas columns do not need to be defined at schema time but can be conjured on the fly while the table is up and running.

Physically, all column family members are stored together on the filesystem. Because tunings and storage specifications are done at the column family level, it is advised that all column family members have the same general access pattern and size characteristics.

-------------------------------------------------------------------------------

10 What is the benefit of using connection pool in Hbase?