Introduction to Hbase

Gkavresis Giorgos - 1470

Agenda

- What is Hbase
- Installation
- About RDBMS
- Overview of Hbase
- Why Hbase instead of RDBMS
- Architecture of Hbase
- Hbase interface
- Summarise

What is Hbase

Hbase is an open source,distributed sorted map modeled after Google's BigTable

Open Source

- Apache 2.0 License
- Commiters and contributors from diverse organizations like Facebook,Trend Micro etc.

Installation

Download link

http://www.apache.org/dyn/closer.cgi/hbase/

Before starting it, you might want to edit

conf/hbase-site.xml and set the directory you want

HBase to write to, hbase.rootdir

Can be standalone or pseudo distributed and distributed

Start Hbase via `$ ./bin/start-hbase.sh`

About Relational DatabaseManagementSystems

- Have a lot of Limitations
- Both read / write throughout not possible(transactional databases)
- Specialized Hardware is quite expensive

Background

- Google releases paper on Bigtable – 2006
- First usable Hbase – 2007
- Hbase becomes Apache top-leven project – 2010
- Hbase 0.26.5 released.

Overview of Hbase

- Hbase is a part of Hadoop
- Apache Hadoop is an open-source system to reliably store and process data across many commodity computers
- Hbase and Hadoop are written in Java
- Hadoop provides:
    - Fault tolerance
    - Scalability

Hadoop advantages

- Data pararell or compute-pararell.For example:
  - Extensive machine learning on <100 GB of image data
  - Simple SQL queries on >100 TB of clickstreaming data

Hadoop's components

- MapReduce(Process)
  - Fault-tolerant distributed processing
- HDFS(store)
  - Self-healing
  - High-bandwidth
  - Clustered storage

Difference Between Hadoop/HDFS and Hbase

HDFS is a distributed file system that is well suited for the storage of large files.HBase, on the other hand, is built on top of HDFS and provides fast record lookups (and updates) for large tables.

HDFS has based on GFS file system.

Hbase is

- Distributed – uses HDFS for storage
- Column – Oriented
- Multi-Dimensional(Versions)
- Storage System

Hbase is NOT

- A sql Database – No Joins, no query engine, no datatypes, no (damn) sql
- No Schema
- No DBA needed

Storage Model

- Column – oriented database (column families)
- Table consists of Rows, each which has a primary key(row key)
- Each Row may have any number of columns
- Table schema only defines Column familes(column family can have any number of columns)
- Each cell value has a timestamp

Static Columns

int	varchar	int	varchar	int
int	varchar	int	varchar	int
int	varchar	int	varchar	int

Something different

- Row1 → ColA = Value
-                     ColB = Value
-                     ColC = Value
- Row2 → ColX = Value
-                      ColY = Value
-                       ColZ = Value

A Big Map
Row Key + Column Key + timestamp => value

Row Key	Column Key	Timestamp	Value
1	Info:name	1273516197868	Sakis
1	Info:age	1273871824184	21
1	Info:sex	1273746281432	Male
2	Info:name	1273863723227	Themis
2	Info:name	1273973134238	Andreas

One more example

Row Key	Data

cutting	Info:{'height':'9ft','state':'CA'}
	Roles:{'ASF':Director','Hadoop':'Founder'}

tlipcon	Info:{'height':5ft7','state':'CA'}
	Roles:{'Hadoop':'Committer'@ts=2010
	'Hadoop':'PMC'@ts=2011
	'Hive':'Contributor'}

Column Families

- Different sets of columns may have different priorities
- CFs stored separately on disk access one without wasting IO on the other.
- Configurable by column family
  - Compression(none,gzip,LZO)
  - Version retention policies
  - Cache priority

Hbase vs RDBMS

	RDBMS	Hbase
Data layout	Row-oriented	Column family oriented
Query language	SQL	Get/put/scan/etc *
Security	Authentication/Authorization	Work in Progress
Max data size	TBs	Hundrends of PBs
Read / write throughput limits	1000s queries/second	Millions of queries per second

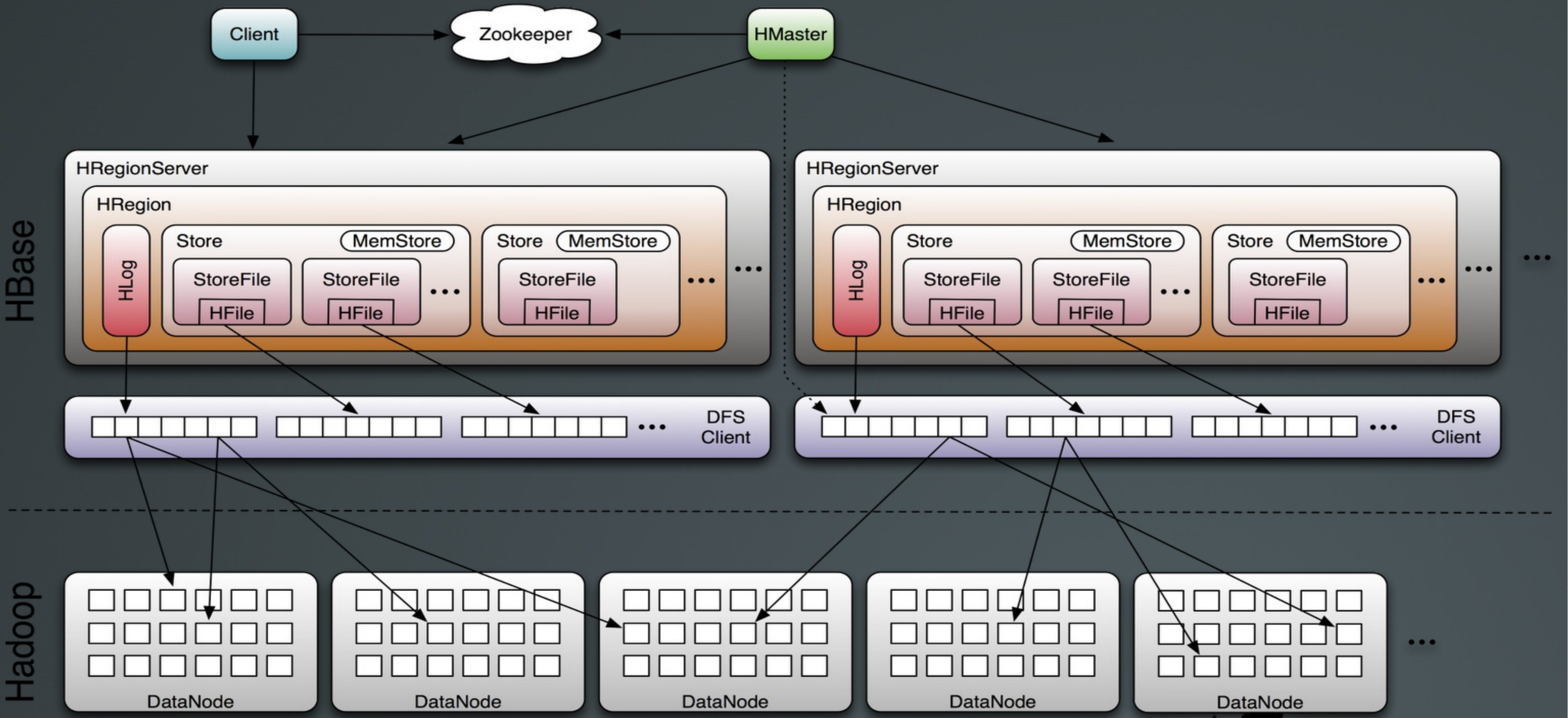
Terms and Daemons

- Region
    - A subset of table's rows,
- RegionServer(slave)
    - Serves data for reads and writes
- Master
    - Responsible for coordinating the slaves
    - Assigns regions, detects failures of Region Servers
    - Control some admin function

Distributed coordination

- To manage master election and server availability we use Zookeeper
- Set up a cluster, provides distributed coordination primitives
- An excellent tool for building cluster management systems

Hbase Architecture



Distributed coordination

- To manage master election and server availability we use Zookeeper
- Set up a cluster, provides distributed coordination primitives
- An excellent tool for building cluster management systems

Hbase Interface

- Java
- Thrift(Ruby,Php,Python,Perl,C++,..)
- Hbase Shell

Hbase API

- get(row)
- put(row,Map<column,value>)
- scan(key range, filter)
- increment(row, columns)
- Check and Put, delete etc.

Hbase shell

```
hbase(main):003:0> create 'test', 'cf'
0 row(s) in 1.2200 seconds
hbase(main):004:0> put 'test', 'row1', 'cf:a', 'value1'
0 row(s) in 0.0560 seconds
hbase(main):005:0> put 'test', 'row2', 'cf:b', 'value2'
0 row(s) in 0.0370 seconds
hbase(main):006:0> put 'test', 'row3', 'cf:c', 'value3'
0 row(s) in 0.0450 seconds
```

Hbase shell cont.

```
hbase(main):007:0> scan 'test'
ROW        COLUMN+CELL
row1       column=cf:a, timestamp=1288380727188, value=value1
row2       column=cf:b, timestamp=1288380738440, value=value2
row3       column=cf:c, timestamp=1288380747365, value=value3
3 row(s) in 0.0590 seconds
```

Hbase in java

```
HBaseConfiguration conf = new HBaseConfiguration();
 conf.addResource(new Path("/opt/hbase-0.19.3/conf/hbase-site.xml"));


 HTable table = new HTable(conf, "test_table");
 BatchUpdate batchUpdate = new BatchUpdate("test_row1");
 batchUpdate.put("columnfamily:column1", Bytes.toBytes("some value")
 );
 batchUpdate.delete("column1");
 table.commit(batchUpdate);
```

Get Data

Read one column value from a row

```
Cell cell = table.get("test_row1", "columnfamily1:column1");
```

To read one row with given columns, use HTable#getRow() method.

```
RowResult singleRow = table.getRow(Bytes.toBytes("test_row1") );
```

A ”tough” facebook application

- Realtime counters of URLs shared, links ”liked”, impressions generated
- 20 billion events/day (200K events/sec)
- ~30 sec latency from click to count
- Heavy use of incrementColumnValue API
- Tried MySQL,Cassandra, settled on Hbase

Use Hbase if

- You need random wrire,random read or both (but not neither)
- You need to do many thousands of operations per sec on multiple TB of data
- Your access patterns are simple

Thank you \../