Week 13
HBase: Column-Oriented DBMS On Top Of Hadoop

Mastering Cloud Computing

Coleman Kane
(based on material by Paul Talaga)

UNIVERSITY OF Cincinnati

Column-Oriented Database

Store data in a manner similar to a spreadsheet

Common layout for most “SQL”-type databases

Data records typically consist of one or more “keys” and reference a much larger set of “data” cells

UNIVERSITY OF Cincinnati

History: Google BigTable

Originally developed in 2004, published paper in 2006.
Recently offered as consumer service in 2015.

Simplified design, though inspired by RDBMS:

- Multi-dimensional sorted map
- Un-typed data
- Non-relational
- Sparse storage model
- Primary Key is always row-id (key in the map)
- Value is associative array of columns
- Secondary key could be considered “has-a” column relationships

http://research.google.com/archive/bigtable.html
https://cloud.google.com/bigtable/

UNIVERSITY OF Cincinnati

Intro: Apache HBase

Modeled on Google BigTable, following paper

Built to use HDFS as data store

Tables are distributed across node by dividing their contents up into “regions” (groups of rows - BigTable calls these “tablets”)

Integrates with Hadoop MapReduce

https://hbase.apache.org/

UNIVERSITY OF Cincinnati

Apache HBase Architecture

There are three core components to HBase:

HMaster: Manages how data is distributed around the cluster, and provides gateway to database - similar to YARN ResourceManager. Can have “backup” master(s)

HRegion: Manager of one “region” of the database, providing storage for one or more “regions” from tables

ZooKeeper: Coordinates the distributed processes running across the cluster - provides centralized configuration and state management to facilitate high-availability operation

Apache HBase Data Model

The HBase Data Model is summarized as follows:

- Each row has a “row id”, acts as the primary key
- Each row can have one or more “columns”, which really just acts as a sparse associative array
- Columns organized into column groups, which are added or removed infrequently
- Columns can be added/removed within a colgroup at any time with minimal penalty
- Data in cells (anywhere not the “row id”) is untyped and of arbitrary length
- Common to store whole images and other “big” data in cells

UNIVERSITY OF Cincinnati

Sparse Storage Optimization

The “sparse” storage optimization instantiates columns within the record, only if they contain data. If they’re empty, then they do not occupy space.

It works very much like a large table/spreadsheet, from a programming model perspective. There’s at least one column that always contains a value in each row, known as the “primary key”.

PK	Name	Variant	Count	Owner	User
Game01	Risk	---	7	James	Edward
Food01	Banana	Yellow	4	Edward	---
Food02	Sandwich	Bacon	3	Melissa	---

UNIVERSITY OF Cincinnati

Programming Model

Generally speaking, the database layout could be implemented using simple data structure constructs:

```
big_table = {} # A dictionary of dictionaries
```

Or, if you prefer C++:

```
std::map<std::key_type,std::map> big_table;
```

Or, in Java:

```
Map<KeyType,Map> big_table = null;
```

UNIVERSITY OF Cincinnati

Programming Model

```
#!/usr/bin/python3
import sys
big_table = {}

# Start with a fixed column named row-id:
print("Enter row-id, or just hit ENTER to stop adding data: ")
while True:
    input_data = sys.stdin.readline().strip()
    if len(input_data) == 0:
        print("Nothing entered, ending data entry loop")
        break

    # Create a new row object, that's a dictionary, with one entry for
    # mandatory row-id
    row = {'row-id': input_data}

    print("Enter column name (just press ENTER to end row data entry): ")

    while True:
        # Get another input record
        colname = sys.stdin.readline().strip()

        if len(colname) == 0:
            print("Completed row '{0}'".format(row['row-id']))
            big_table[row['row-id']] = row
            break

        print("Enter column value (just press ENTER for no value): ")
        input_data = sys.stdin.readline().strip()
        if len(input_data) > 0:
            row[colname] = input_data
        print("Enter column name (just press ENTER to end row data entry): ")

    print("Enter row-id, or just hit ENTER to stop adding data: ")

print(big_table)
```

https://github.com/ckane/CS6065-Cloud-BigTableDemo

UNIVERSITY OF Cincinnati

Programming Model

```
Enter row-id, or just hit ENTER to stop adding data:
Test1
Enter column name (just press ENTER to end row data entry):
cf:a
Enter column value (just press ENTER for no value):
12
Enter column name (just press ENTER to end row data entry):
cf:b
Enter column value (just press ENTER for no value):
77
Enter column name (just press ENTER to end row data entry):

Completed row 'Test1'
Enter row-id, or just hit ENTER to stop adding data:
Test2
Enter column name (just press ENTER to end row data entry):
cf:b
Enter column value (just press ENTER for no value):
25
Enter column name (just press ENTER to end row data entry):
cf:c
Enter column value (just press ENTER for no value):
77
Enter column name (just press ENTER to end row data entry):
cf:d
Enter column value (just press ENTER for no value):
32
Enter column name (just press ENTER to end row data entry):
cr:1
Enter column value (just press ENTER for no value):
88
Enter column name (just press ENTER to end row data entry):

Completed row 'Test2'
Enter row-id, or just hit ENTER to stop adding data:

Nothing entered, ending data entry loop
{'Test1': {'cf:a': '12', 'cf:b': '77', 'row-id': 'Test1'},
 'Test2': {'cf:c': '77', 'cf:b': '12', 'row-id': 'Test2', 'cf:d': '324132'}}
```

Table View

row-id	cf:a	cf:b	cf:c	cf:d	cr:1
Test1	12	77			
Test2		25	77	32	88

UNIVERSITY OF Cincinnati

API - Python - HappyBase

Using the “Thrift” interface from HBase:
`hbase-root/bin/hbase thrift start`

There’s a Python framework named “HappyBase”:

- https://happybase.readthedocs.io

```
#!/usr/bin/env python
import happybase

db_conn = happybase.Connection('laboratory')
table = db_conn.table('test2')

for key, data in table.scan():
    print(key)
    print(data)
```

UNIVERSITY OF Cincinnati

API - Java - HBase

There's a Java framework as well, with HBase

- http://hbase.apache.org/apidocs/