L11_L12

Introduction to Cassandra

Agenda

- Apache Cassandra – An Introduction
- Features of Cassandra
  - Peer-to-Peer Network
  - Writes in Cassandra
  - Hinted Handoffs
  - Tunable Consistency: Read Consistency and Write Consistency
- CQL Data Types
- CQLSH
- CRUD : Insert, Update, Delete and Select
- Collections : Set, List and Map
- Time To Live (TTL)
- Import and Export

Apache Cassandra – An Introduction

- Apache Cassandra was born at Facebook. After Facebook open sourced the code in 2008, Cassandra became an Apache Incubator project in 2009 and subsequently became a top-level Apache project in 2010.

- It is a column-oriented database designed to support peer-to-peer symmetric nodes instead of the master–slave architecture.

- It is built on Amazon's dynamo and Google's BigTable.

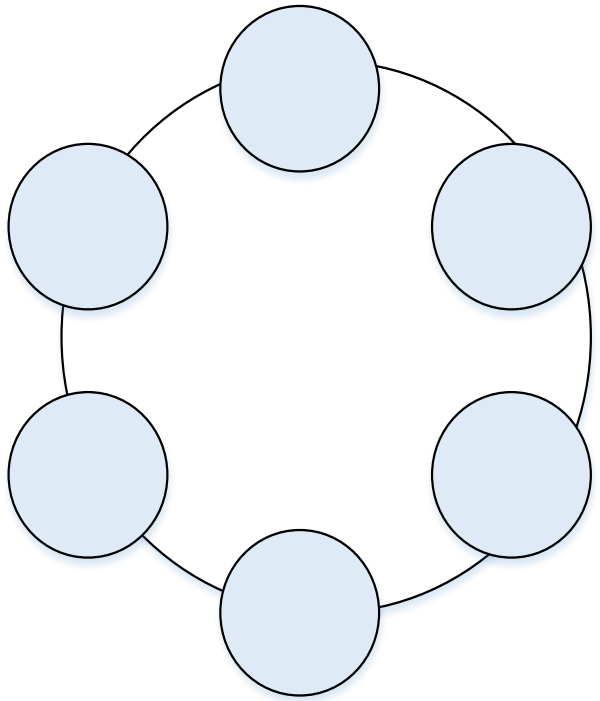
Features of Cassandra

Features of Cassandra

- Open Source
- Distributed
- Decentralized (Server Symmetry)
- No single point of failure
- Column-oriented
- Peer to Peer
- Elastic Scalability

Peer to Peer Network

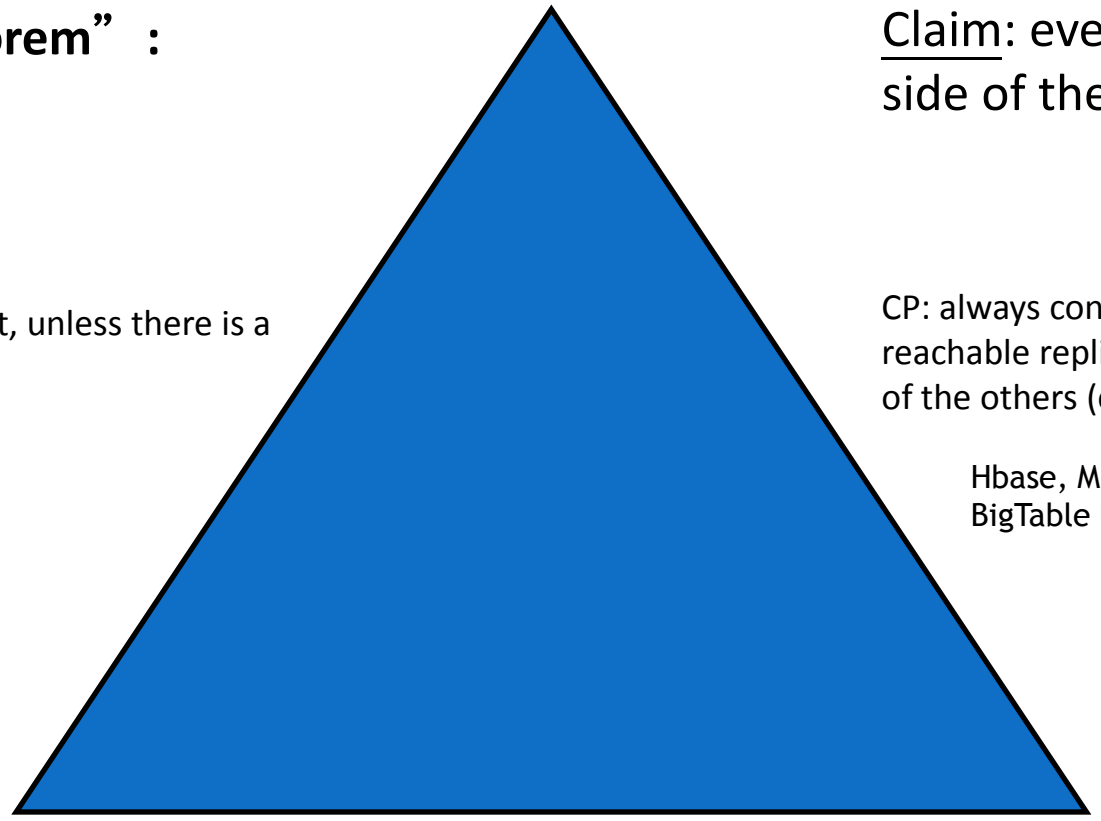
Sample Cassandra Cluster



Fox&Brewer "CAP Theorem": C-A-P: choose two.

consistency

C



A

Availability

P

Partition-resilience

Claim: every distributed system is on one side of the triangle.

CA: available, and consistent, unless there is a partition.

Traditional RDBMS
PostgreSQL, MYSQL
Etc.

CP: always consistent, even in a partition, but a reachable replica may deny service without agreement of the others (e.g., quorum).

Hbase, MongoDB, Redis,
BigTable like Systems

AP: a reachable replica provides service even in a partition, but may be inconsistent if there is a failure.

Riak, Cassandra, CouchDB,
Dynamo like systems

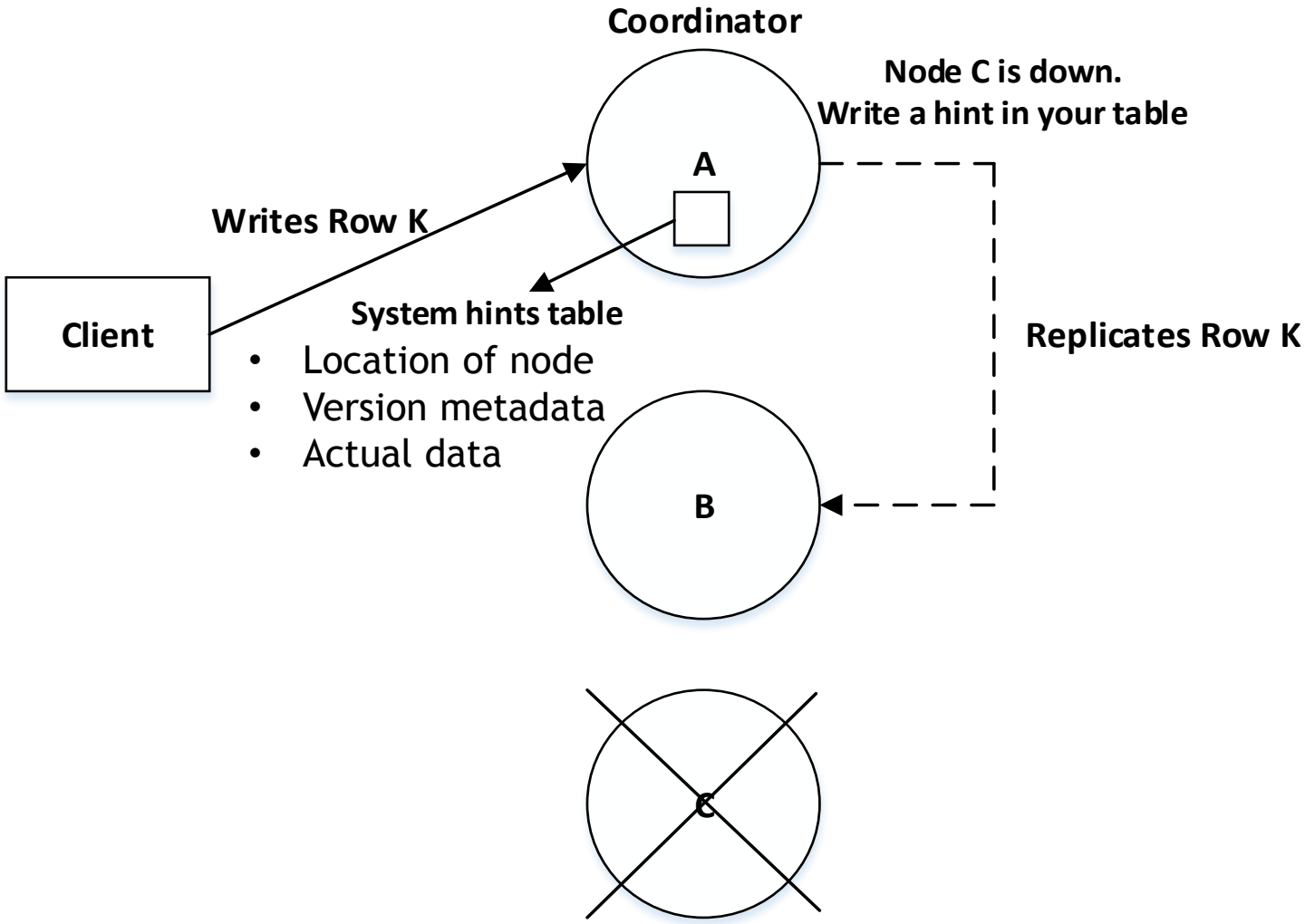
Writes in Cassandra

Writes in Cassandra

- A client that initiates a write request.
- It is first written to the commit log. A write is taken as successful only if it is written to the commit log.
- The next step is to push the write to a memory resident data structure called Memtable. A threshold value is defined in the Memtable.
- When the number of objects stored in the Memtable reaches a threshold, the contents of Memtable are flushed to the disk in a file called SSTable (Stored string Table). Flushing is a non-blocking operation.
- It is possible to have multiple Memtables for a single column family. One out of them is current and the rest are waiting to be flushed.

Hinted Handoffs

Hinted Handoffs



Tunable Consistency

Read Consistency

ONE	Returns a response from the closest node (replica) holding the data.
QUORUM	Returns a result from a quorum of servers with the most recent timestamp for the data.
LOCAL_QUORUM	Returns a result from a quorum of servers with the most recent timestamp for the data in the same data center as the coordinator node.
EACH_QUORUM	Returns a result from a quorum of servers with the most recent timestamp in all data centers.
ALL	This provides the highest level of consistency of all levels and the lowest level of availability of all levels. It responds to a read request from a client after all the replica nodes have responded.

Write Consistency

ALL	**This is the highest level of consistency of all levels as it necessitates that a write must be written to the commit log and Memtable on all replica nodes in the cluster.**
EACH_QUORUM	A write must be written to the commit log and Memtable on a quorum of replica nodes in all data centers.
QUORUM	A write must be written to the commit log and Memtable on a quorum of replica nodes.
LOCAL_QUORUM	A write must be written to the commit log and Memtable on a quorum of replica nodes in the same data center as the coordinator node. This is to avoid latency of inter-data center communication.
ONE	A write must be written to the commit log and Memtable of at least one replica node.
TWO	A write must be written to the commit log and Memtable of at least two replica nodes.
THREE	A write must be written to the commit log and Memtable of at least three replica nodes.
LOCAL_ONE	A write must be sent to, and successfully acknowledged by, at least one replica node in the local data center.

CQL Data types

CQL Data types

Int	32 bit signed integer
Bigint	64 bit signed long
Double	64-bit IEEE-754 floating point
Float	32-bit IEEE-754 floating point
Boolean	True or false
Blob	Arbitrary bytes, expressed in hexadecimal
Counter	Distributed counter value
Decimal	Variable – precision integer
List	A collection of one or more ordered elements
Map	A JSON style array of elements
Set	A collection of one or more elements
Timestamp	Date plus time
Varchar	UTF 8 encoded string
Varint	Arbitrary-precision integers
Text	UTF 8 encoded string

CQLSH

CRUD - Keyspace

To create a keyspace by the name “Students”

```
CREATE KEYSPACE Students WITH REPLICATION = {
        'class':'SimpleStrategy',
        'replication_factor':1
 };
```

To describe all existing keyspaces

```
DESCRIBE KEYSPACES;
```

CRUD – Create Table

To create a column family or table by the name “student_info”.

```
CREATE TABLE Student_Info (
  RollNo int PRIMARY KEY,
  StudName text,
  DateofJoining timestamp,
  LastExamPercent double
  );
```

To lookup names of all tables

Describe Tables;

To describe table Student_info

Describe table Student_info;

CRUD - Insert

To insert data into the column family “student_info”.

```
BEGIN BATCH
INSERT INTO student_info
(RollNo,StudName,DateofJoining,LastExamPercent)
VALUES (1,'Michael Storm','2012-03-29', 69.6);
INSERT INTO student_info
(RollNo,StudName,DateofJoining,LastExamPercent)
VALUES (2,'Stephen Fox','2013-02-27', 72.5);
APPLY BATCH;
```

CRUD - Select

To view the data from the table “student_info”.

```
SELECT *  FROM student_info;

SELECT *  FROM student_info limit 2;

SELECT rollno, studname as Name  FROM student_info;


SELECT *  FROM student_info where rollno in (1,2,3);


SELECT *  FROM student_info where studname = ‘Stephen Fox’;
```

CRUD – Create Index

To create an index on the “studname” column of the “student_info” column family use the following statement

CREATE INDEX ON student_info(studname);

SELECT rollno FROM student_info where studname = ‘Stephen Fox’;

CRUD – Update

To update the value held in the “StudName” column of the “student_info” column family to “David Sheen” for the record where the RollNo column has value = 2.

Note: An update updates one or more column values for a given row to the Cassandra table. It does not return anything.

UPDATE student_info SET StudName = 'David Sheen' WHERE RollNo = 2;

CRUD - Delete

To delete the column “LastExamPercent” from the “student_info” table for the record where the RollNo = 2.

Note: Delete statement removes one or more columns from one or more rows of a Cassandra table or removes entire rows if no columns are specified.

DELETE LastExamPercent FROM student_info WHERE RollNo=2;

To delete the record from the “student_info” table where the RollNo = 2.

DELETE FROM student_info WHERE RollNo=2;

Collections

Collections – Set, List, Map

When to use collection?

Use collection when it is required to store or denormalize a small amount of data.

What is the limit on the values of items in a collection?

The values of items in a collection are limited to 64K.

Where to use collections?

Collections can be used when you need to store the following:

1. Phone numbers of users.
2. Email ids of users.

Collections - Set

To alter the schema for the table “student_info” to add a column “hobbies”.

ALTER TABLE student_info ADD hobbies set<text>;

Select * from student_info;

Collections - Set

To update the table “student_info” to provide the values for “hobbies” for the student with Rollno =1.

```
UPDATE student_info
    SET hobbies = hobbies + {'Chess’,’ Table Tennis'}  WHERE RollNo=1;
Select  rollno, hobbies from student_info;
```

To remove an element from the set

```
UPDATE student_info
    SET hobbies = hobbies -{'Chess’}  WHERE RollNo=1;
```

To remove all elements of hobbies set for the student with Rollno =1.

```
UPDATE student_info
    SET hobbies =  { } WHERE rollNo=1;
Select * from student_info where rollno = 1;
```

Collections - List

To alter the schema of the table “student_info” to add a list column “language”.

ALTER TABLE student_info ADD language list<text>;

Select * from student_info;

Collections - List

To update values in the list column, “language” of the table “student_info”.

```
UPDATE student_info
        SET language = language + ['Hindi’, ‘English']
            WHERE RollNo=1;

Select rollno, language from student_info where rollno = 1;
```

To prepend / append an element to the list

```
UPDATE student_info
        SET language = [‘Kannada’] + language WHERE RollNo=1;

Select rollno, langugage from student_info;
```

Collections - List

To update values in the list column, “language” at a particular index

```
UPDATE student_info
        SET language[0] = ‘Kannada’
            WHERE RollNo=1;

Select rollno, language from student_info;
```

To delete an element to the list

```
UPDATE student_info
        SET language = language - [‘Kannada’] WHERE RollNo=1;
Select rollno, langugage from student_info;
Delete language[0] from student_info where rollno=1;
```

Collections - Map

To alter the “users” table to add a map column “todo”.

```
ALTER TABLE student_info
    ADD todo map<timestamp, text>;
```

Collections - Map

To update the todo map:

UPDATE student_info SET todo = { '2014-9-24': 'Cassandra Session','2014-10-2 12:00' : 'MongoDB Session' } WHERE rollno =1;

Select rollno, todo from student_info;

To modify / delete map:

UPDATE student_info set todo['2014-9-24'] = 'Hive Session' where rollno=1;

Delete todo['2014-9-24'] from student_info where rollno=1;

Time To Live

Time To Live

Data in a column, other than a counter column, can have an optional expiration period called TTL (time to live). The client request may specify a TTL value for the data. The TTL is specified in seconds.

```
CREATE TABLE  userlogin(
userid int primary key, password text
);
```

```
INSERT INTO userlogin (userid, password)  VALUES (1,'infy') USING TTL 30;
```

```
SELECT TTL (password)
        FROM userlogin
                WHERE userid=1;
```

Export to CSV

Export data to a CSV file

Export the contents of the table/column family “elearninglists” present in the “students” database to a CSV file (d:\elearninglists.csv).

COPY elearninglists (id, course_order, course_id, courseowner, title) TO 'd:\elearninglists.csv';

Import from CSV

Import data from a CSV file

To import data from “D:\elearninglists.csv” into the table “elearninglists” present in the “students” database.

COPY elearninglists (id, course_order, course_id, courseowner, title)
FROM 'd:\elearninglists.csv';

Answer a few quick questions …

Answer Me

- What is Cassandra?
- Comment on Cassandra writes.
- What is your understanding of tunable consistency?
- What are collections in CQLSH? Where are they used?

References ...

Further Readings

- http://www.datastax.com/documentation/cassandra/2.0/cassandra/gettingStartedCassandraIntro.html
- http://www.datastax.com/documentation/cql/3.1/pdf/cql31.pdf
- http://www.datastax.com/documentation/cassandra/2.0/cassandra/dml/dml_config_consistency_c.html

Thank you