Apache Cassandra

STL Java Users Group

Cliff Gilmore

DataStax Solutions Architect / Engineer

Aug 14, 2014

Agenda

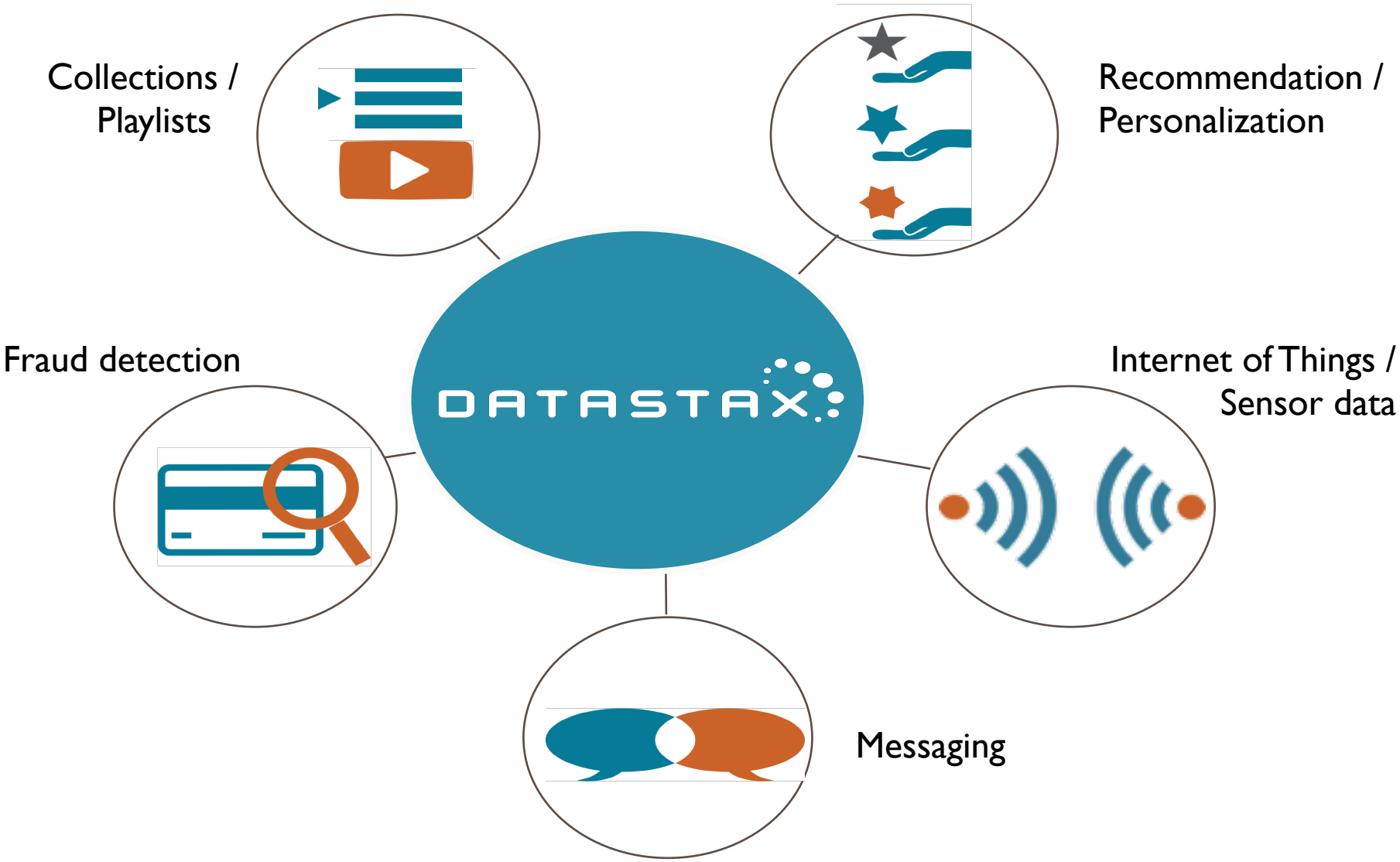
DATASTAX

- Cassandra Overview
- Cassandra Architecture
- Cassandra Query Language
- **Interacting with Cassandra using Java**
- About DataStax

DATASTAX

CASSANDRA OVERVIEW

Who is using DataStax?

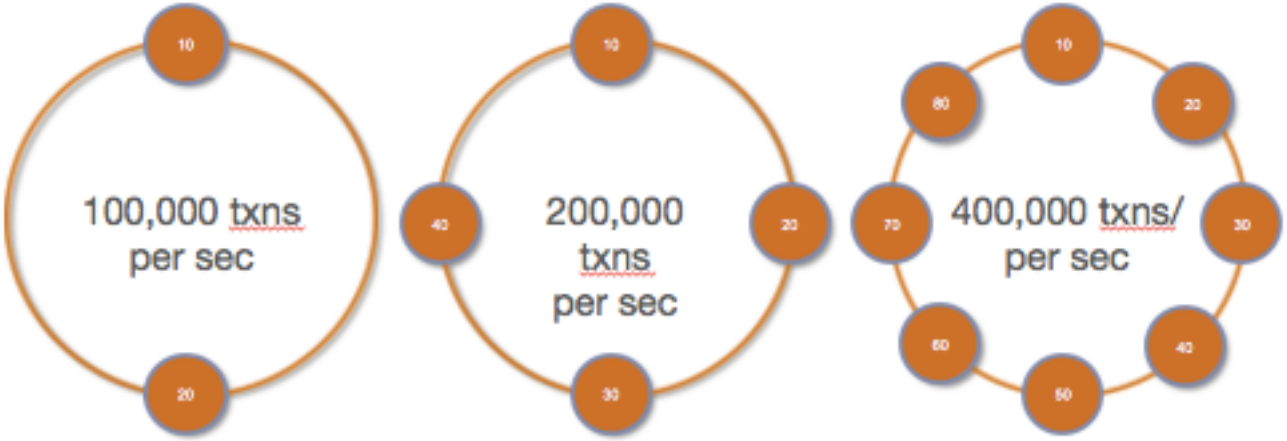


What is Apache Cassandra?

DATASTAX

Apache Cassandra™ is a massively scalable NoSQL database.

- Continuous availability
- High performing writes and reads
- Linear scalability
- Multi-data center support



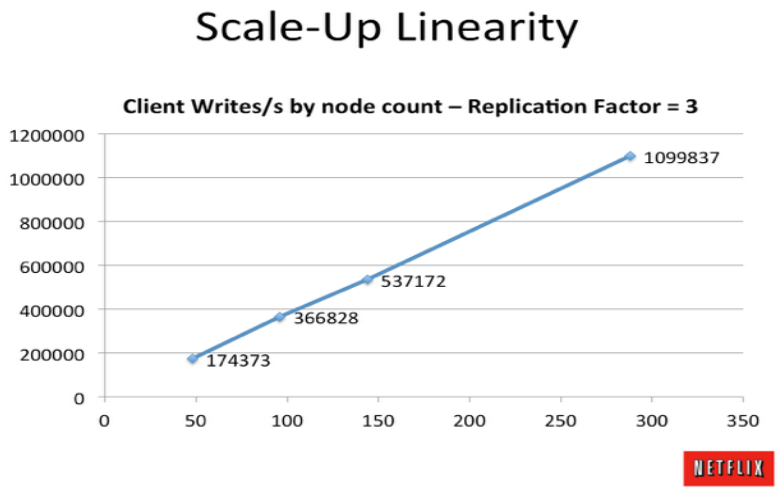
The NoSQL Performance Leader

DATASTAX

"In terms of scalability, there is a clear winner throughout our experiments. Cassandra achieves the highest throughput for the maximum number of nodes in all experiments with a linear increasing throughput."

Source: Solving Big Data Challenges for Enterprise Application Performance Management benchmark paper presented at the Very Large Database Conference, 2013.

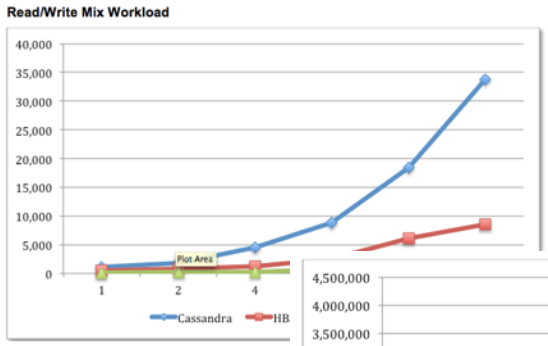
Netflix Cloud Benchmark…



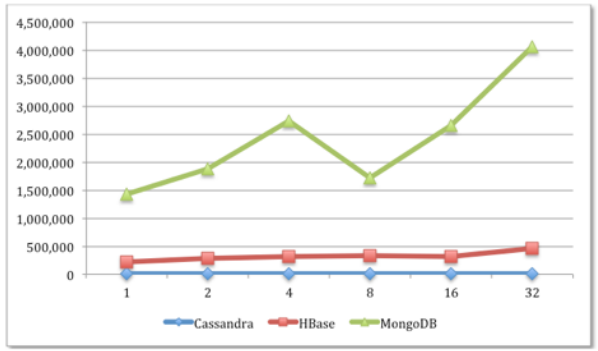
Source: Netflix Tech Blog

End Point Independent NoSQL Benchmark

Highest in throughput…

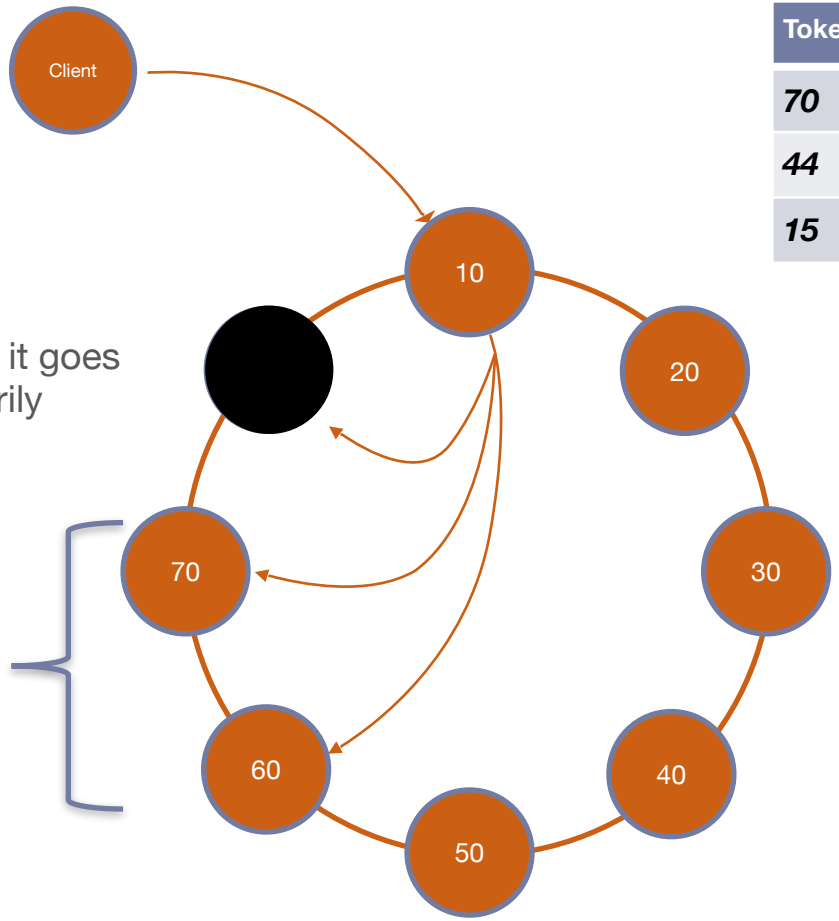


Lowest in latency…



Cassandra is Fault Tolerant

DATASTAX



Token	Order_id	Qty	Sale
70	*1001*	*10*	*100*
44	*1002*	*5*	*50*
15	*1003*	*30*	*200*

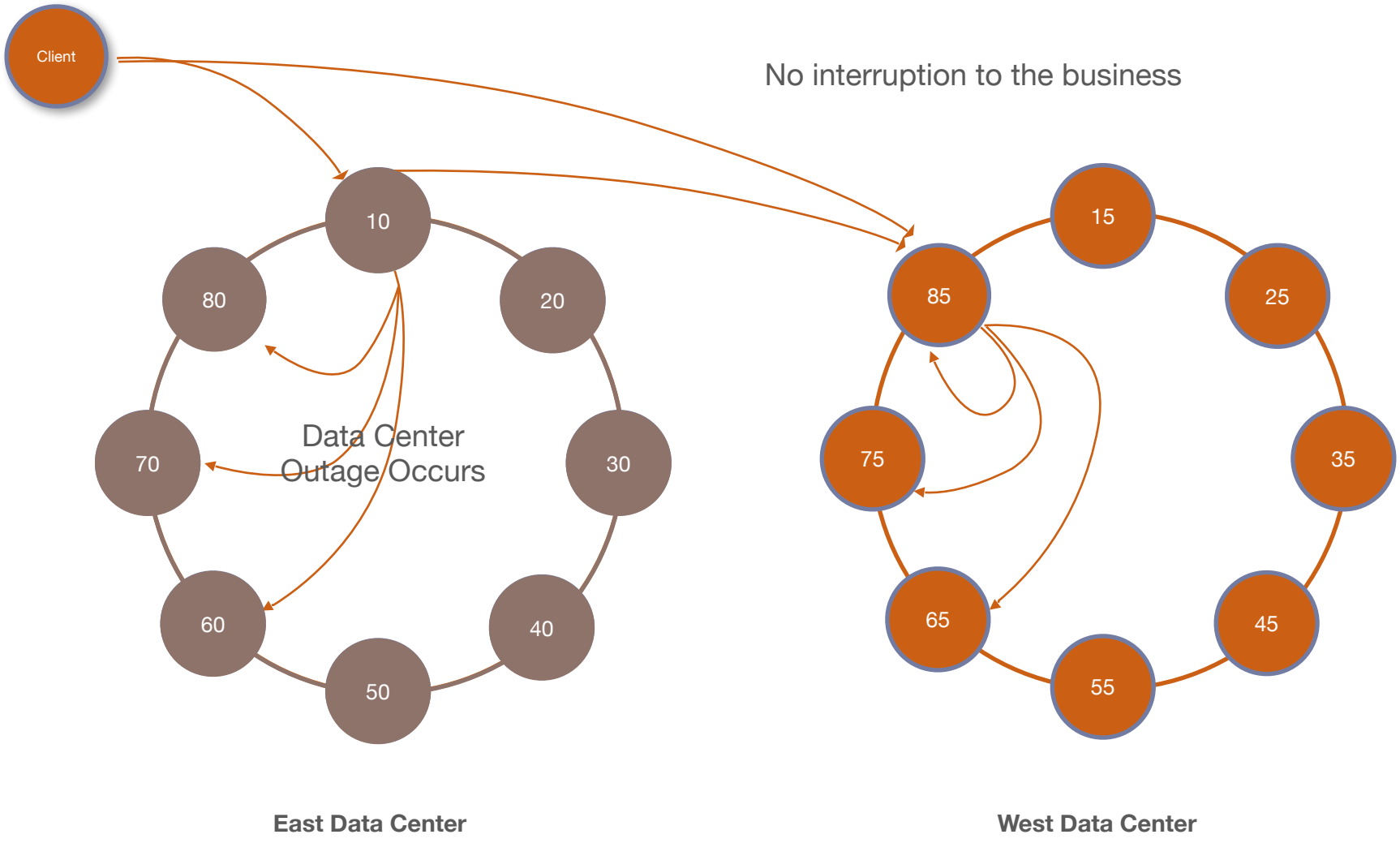
Node failure or it goes down temporarily

We could still retrieve the data from the other 2 nodes

Replication Factor = 3

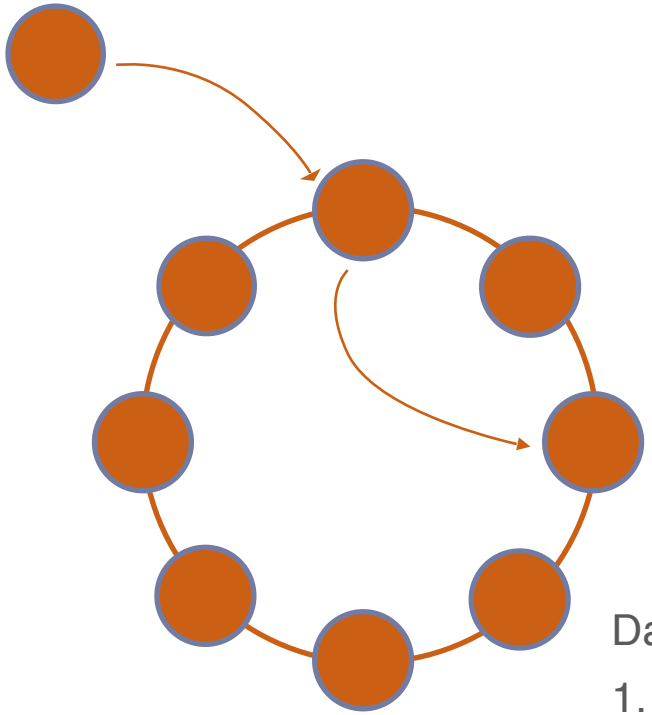
Multi Data Center Support

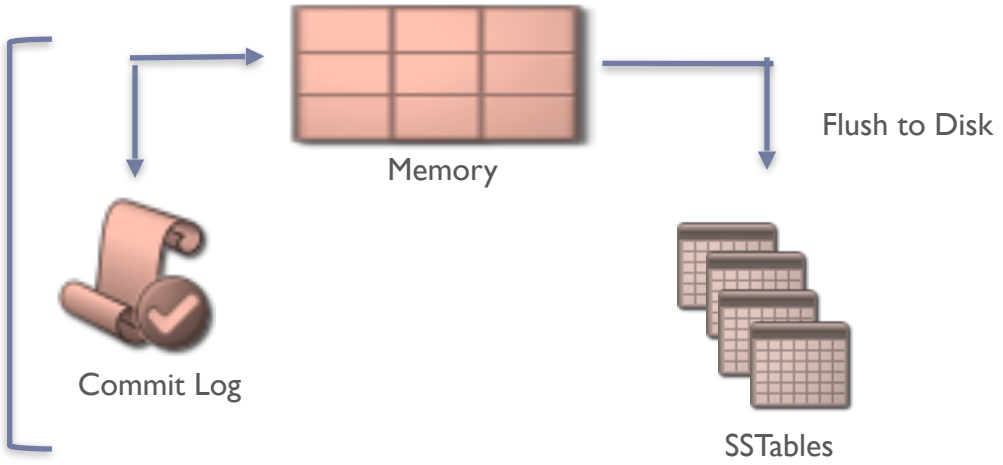
DATASTAX



Writes in Cassandra

DATASTAX





Data is organized into Partitions

1. Data is written to a Commit Log for a node (durability)
2. Data is written to MemTable (in memory)
3. MemTables are flushed to disk in an SSTable based on size.

SSTables are immutable

Tunable Data Consistency

DATASTAX

Writes

- Any
- One
- Quorum
- Local_Quorum
- Each_Quorum
- All

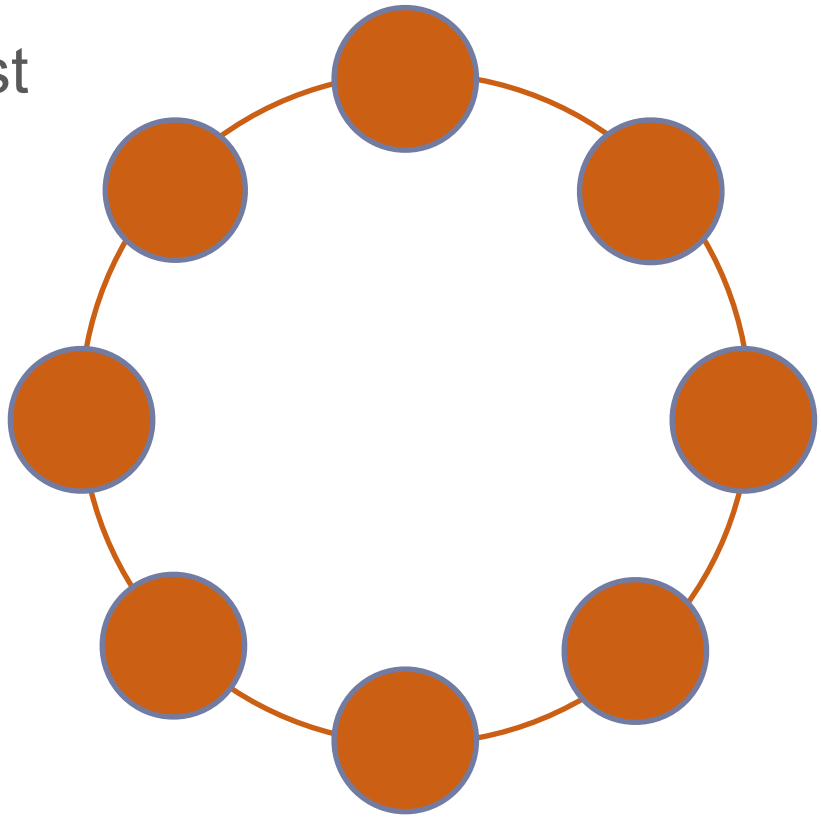
Reads

- One
- Quorum
- Local_Quorum
- Each_Quorum
- All

Built for Modern Online Applications

DATASTAX

- Architected for today's needs
- Linear scalability at lowest cost
- 100% uptime
- Operationally simple



DATASTAX

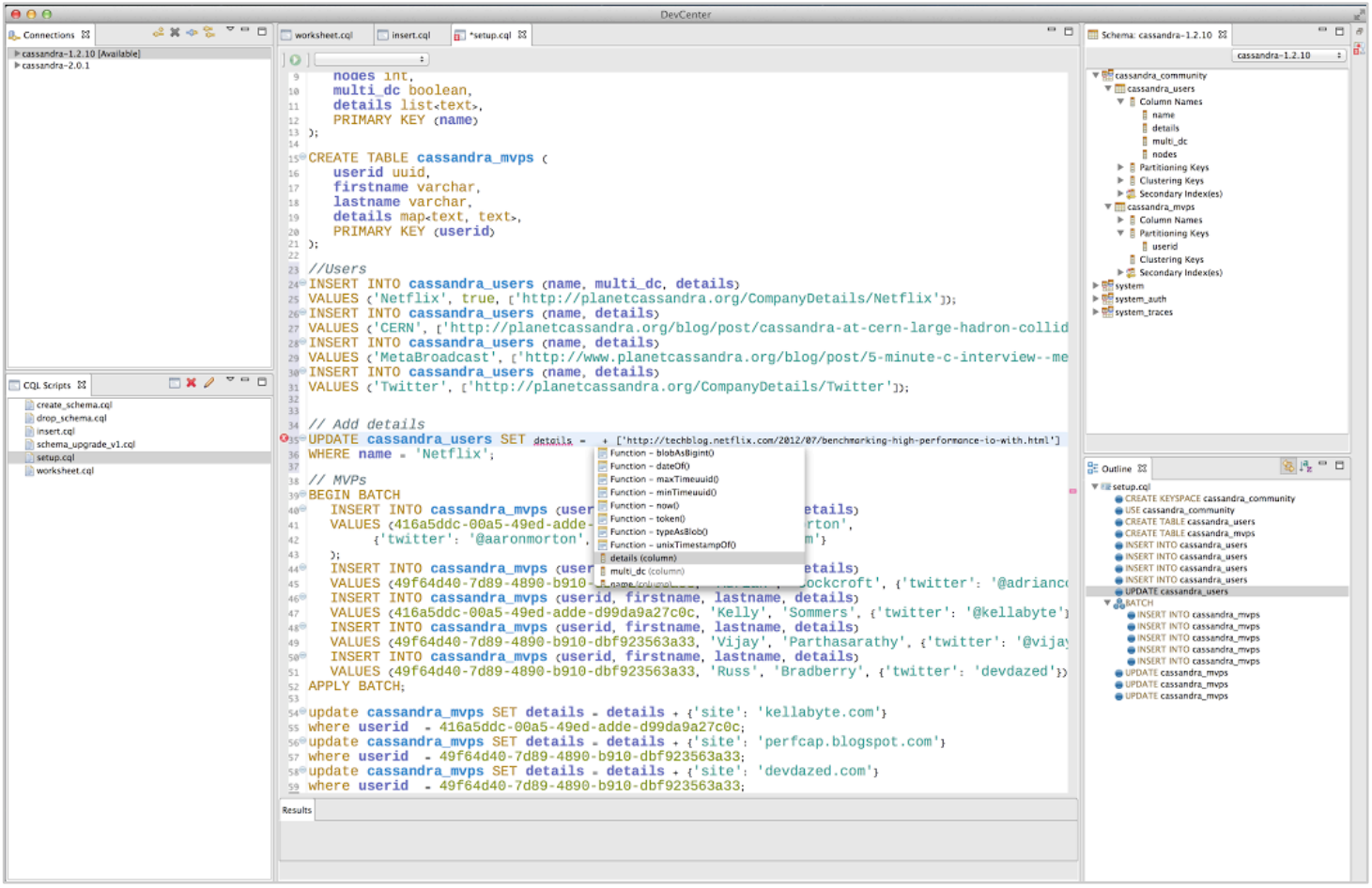
Cassandra Query Language

CQL - DevCenter

DATASTAX

A SQL-like query language for communicating with Cassandra

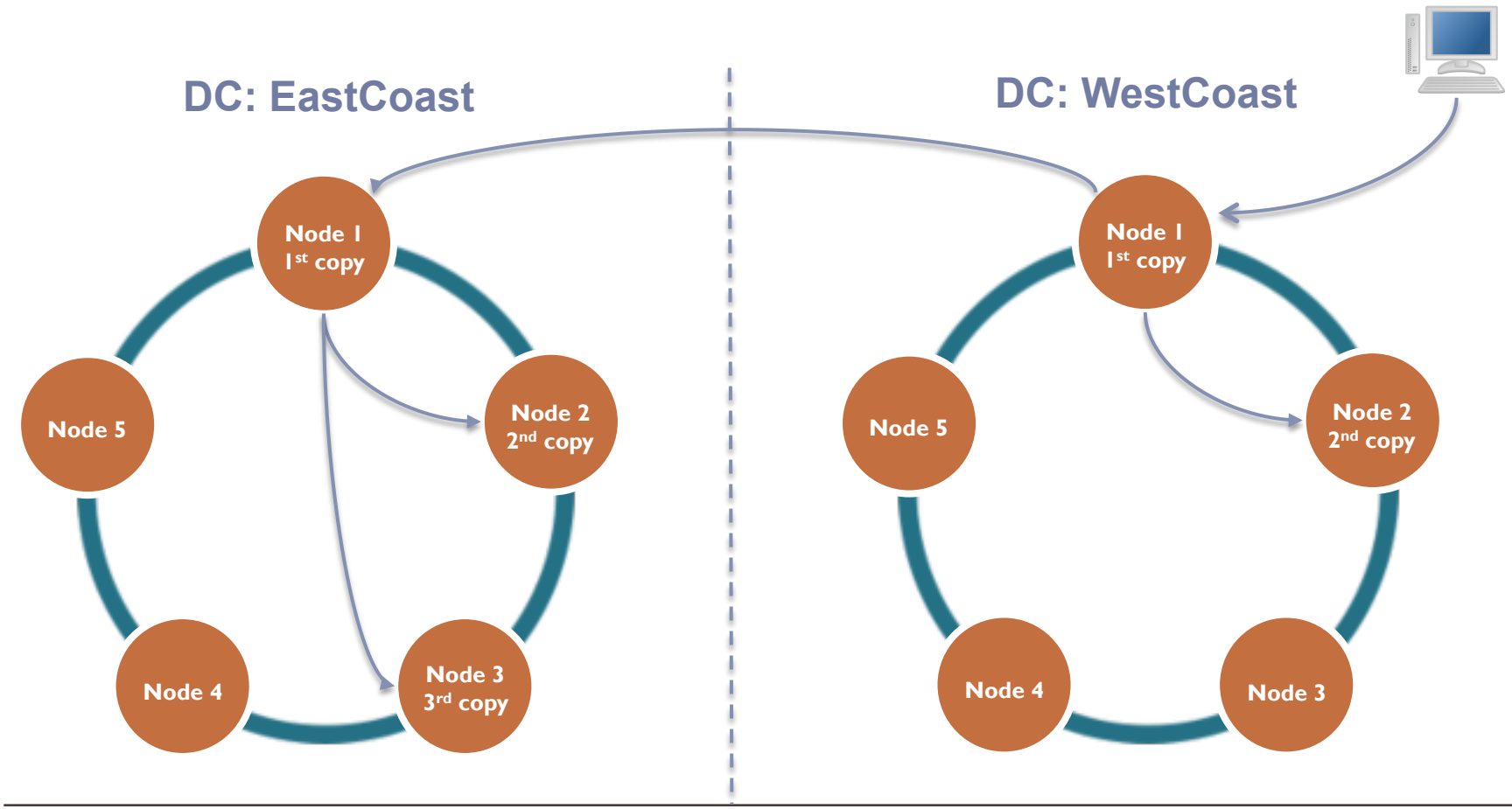
DataStax DevCenter – a free, visual query tool for creating and running CQL statements against Cassandra and DataStax Enterprise.



CQL - Create Keyspace

DATASTAX

```
CREATE KEYSPACE demo WITH REPLICATION =
{'class' : 'NetworkTopologyStrategy', 'EastCoast': 3,
'WestCoast': 2);
```



CQL - Basics

DATASTAX

```
CREATE TABLE users (
    username text,
    password text,
    create_date timestamp,
    PRIMARY KEY (username, create_date desc);
```

```
INSERT INTO users (username, password, create_date)
VALUES ('caroline', 'password1234', '2014-06-01 07:01:00');
```

```
SELECT * FROM users WHERE username = 'caroline' AND
create_date = '2014-06-01 07:01:00';
```

Predicates
On the **partition key:** = and IN
On the **cluster columns:** <, <=, =, >=, >, IN

Collection Data Types

DATASTAX

CQL supports having columns that contain collections of data.

The collection types include:
Set, List and Map.

```
CREATE TABLE users (
    username text,
    set_example set<text>,
    list_example list<text>,
    map_example map<int,text>,
    PRIMARY KEY (username)
);
```

Favor sets over list – better performance

Plus much more…

DATASTAX

Light Weight Transactions

```
INSERT INTO customer_account (customerID, customer_email)
VALUES (‘LauraS’, ‘lauras@gmail.com’) IF NOT EXISTS;

UPDATE customer_account SET customer_email=’laurass@gmail.com’
IF customer_email=’lauras@gmail.com’;
```

Counters

```
UPDATE UserActions SET total = total + 2
WHERE user = 123 AND action = ’xyz';
```

Time to live (TTL)

```
INSERT INTO users (id, first, last) VALUES (‘abc123’, ‘abe’,
‘lincoln’) USING TTL 3600;
```

Batch Statements

```
BEGIN BATCH
  INSERT INTO users (userID, password, name) VALUES ('user2',
'ch@ngem3b', 'second user')
  UPDATE users SET password = 'ps22dhds' WHERE userID =
'user2'
  INSERT INTO users (userID, password) VALUES ('user3',
'ch@ngem3c')
  DELETE name FROM users WHERE userID = 'user2’
APPLY BATCH;
```

DATASTAX

JAVA CODE EXAMPLES

DataStax Java Driver

DATASTAX

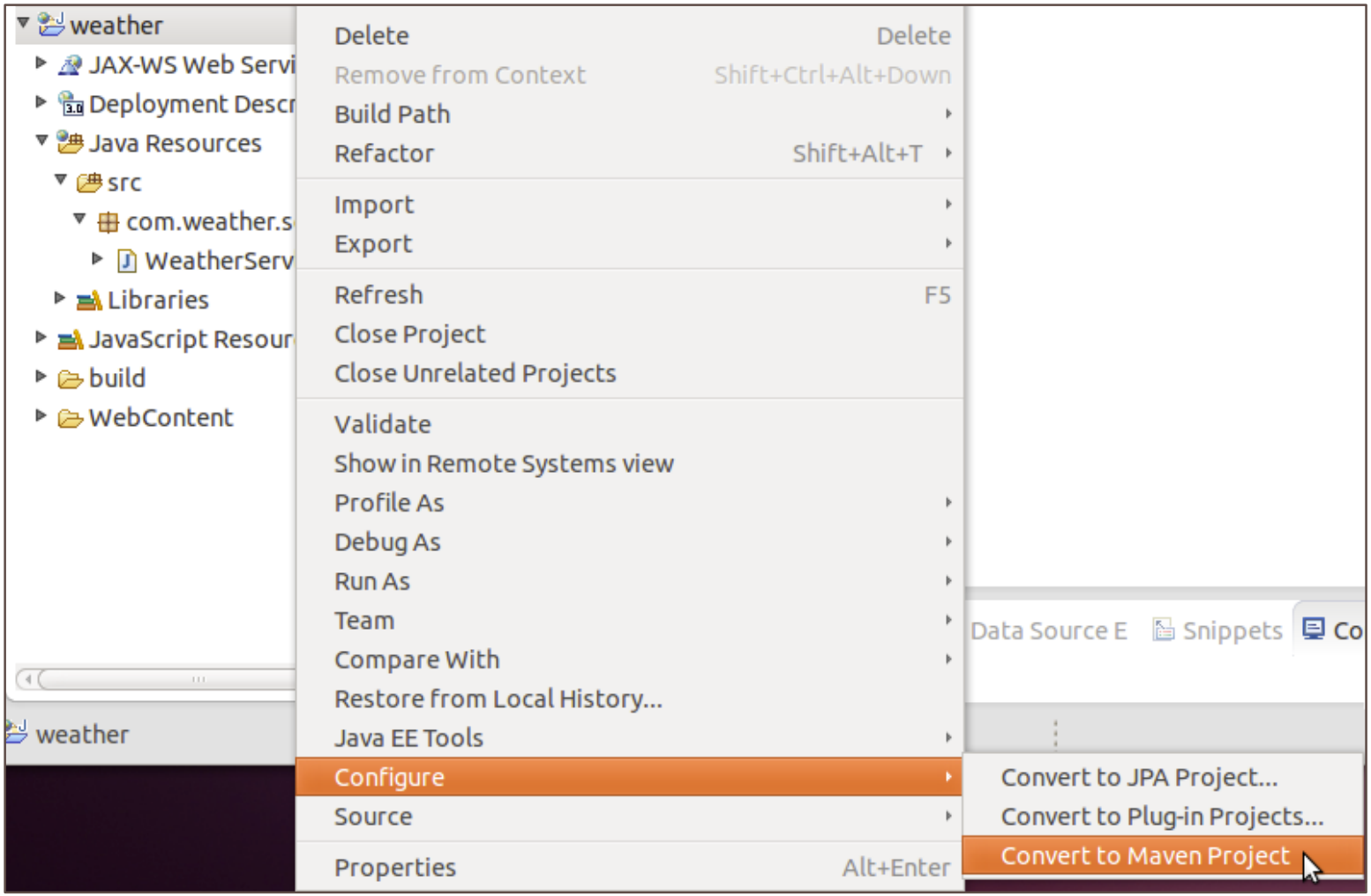
- Written for CQL 3.0
- Uses the binary protocol introduced in Cassandra 1.2
- Uses Netty to provide an asynchronous architecture
- Can do asynchronous or synchronous queries
- Has connection pooling
- Has node discovery and load balancing

http://www.datastax.com/download

Add .JAR Files to Project

DATASTAX

Easiest way is to do this with Maven, which is a software project management tool



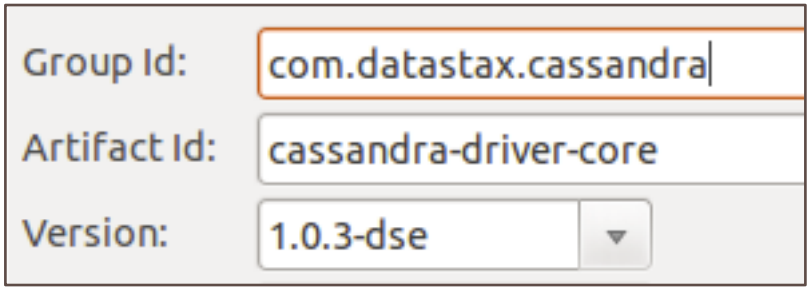
Add .JAR Files to Project

DATASTAX

In the pom.xml file, select the **Dependencies** tab

Click the **Add…** button in the left column

Enter the DataStax Java driver info




```
<dependency>
  <groupId>com.datastax.cassandra</groupId>
  <artifactId>cassandra-driver-core</artifactId>
  <version>1.0.3-dse</version>
</dependency>
```

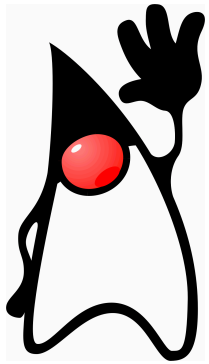
Connect & Write

DATASTAX

```
Cluster cluster = Cluster.builder()
      .addContactPoints("10.158.02.40", "10.158.02.44")
      .build();


Session session = cluster.connect("demo");


session.execute(
   "INSERT INTO users (username, password) ”
   + "VALUES(‘caroline’, ‘password1234’)"
);
```



Note: Cluster and Session objects should be long-lived and re-used

Read from Table

DATASTAX

```
ResultSet rs = session.execute("SELECT * FROM users");


List<Row> rows = rs.all();


for (Row row : rows) {
    String userName = row.getString("username");
    String password = row.getString("password");
}
```

Asynchronous Read

DATASTAX

```
ResultSetFuture future = session.executeAsync(
                "SELECT * FROM users");


for (Row row : future.get()) {
    String userName = row.getString("username");
    String password = row.getString("password");
}
```

Note: The future returned implements Guava's ListenableFuture interface. This means you can use all Guava's Futures[1] methods!

[1]http://docs.guava-libraries.googlecode.com/git/javadoc/com/google/common/util/concurrent/Futures.html

Read with Callbacks

DATASTAX

```
final ResultSetFuture future =
      session.executeAsync("SELECT * FROM users");


future.addListener(new Runnable() {


   public void run() {
      for (Row row : future.get()) {
         String userName = row.getString("username");
         String password = row.getString("password");
      }
   }

}, executor);
```

Parallelize Calls

DATASTAX

```
int queryCount = 99;

List<ResultSetFuture> futures = new
ArrayList<ResultSetFuture>();

for (int i=0; i<queryCount; i++) {
   futures.add(
      session.executeAsync("SELECT * FROM users "
            +"WHERE username = '"+i+"'"));
}

for(ResultSetFuture future : futures) {
   for (Row row : future.getUninterruptibly()) {
      //do something
   }
}
```

Prepared Statements

DATASTAX

```
PreparedStatement statement = session.prepare(
      "INSERT INTO users (username, password) "
      + "VALUES (?, ?)");


BoundStatement bs = statement.bind();


bs.setString("username", "caroline");
bs.setString("password", "password1234");


session.execute(bs);
```

Query Builder

DATASTAX

```
Query query = QueryBuilder
        .select()
        .all()
        .from("demo", "users")
        .where(eq("username", "caroline"));


ResultSet rs = session.execute(query);
```

Load Balancing

DATASTAX

Determine which node will next be contacted once a connection to a cluster has been established

```
Cluster cluster = Cluster.builder()
        .addContactPoints("10.158.02.40","10.158.02.44")
        .withLoadBalancingPolicy(
           new DCAwareRoundRobinPolicy("DC1"))
        .build();
```

Name of the local DC

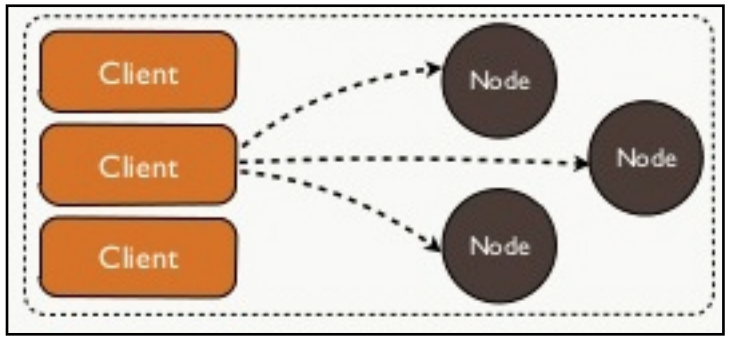
Policies are:

- **RoundRobinPolicy**
- **DCAwareRoundRobinPolicy** (default)
- **TokenAwarePolicy**

RoundRobinPolicy

DATASTAX

- Not data-center aware
- Each subsequent request after initial connection to the cluster goes to the next node in the cluster

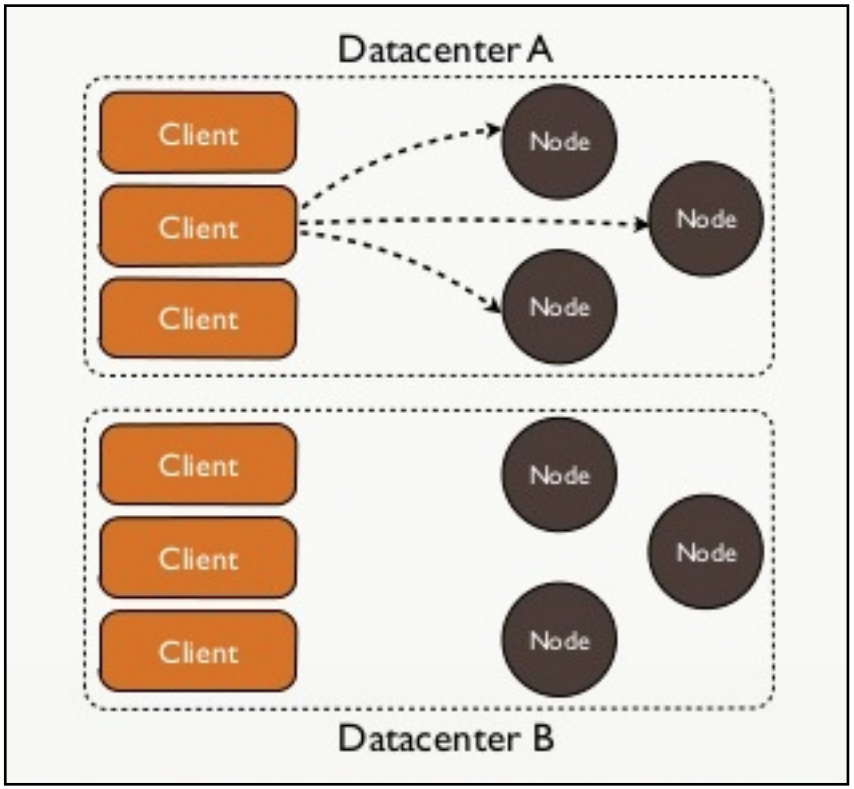


- If the node that is serving as the coordinator fails during a request, the next node is used

DCAwareRoundRobinPolicy

DATASTAX

- Is data center aware
- Does a round robin within the local data center
- Only goes to another
  data center if there is
  not a node available
  to be coordinator in
  the local data center



TokenAwarePolicy

DATASTAX

- Is aware of where the replicas for a given token live
- Instead of round robin, the client chooses the node that contains the primary replica to be the chosen coordinator
- Avoids unnecessary time taken to go to any node to have it serve as coordinator to then contact the nodes with the replicas

Additional Information & Support

DATASTAX

- Community Site

  (http://planetcassandra.org)
- Documentation

  (http://www.datastax.com/docs)
- Downloads

  (http://www.datastax.com/download)
- Getting Started

  (http://www.datastax.com/documentation/gettingstarted/index.html)
- DataStax

  (http://www.datastax.com)

DATASTAX

ABOUT DATASTAX

About DataStax

DATASTAX

300+

Employees

Founded in April 2010

Santa Clara, Austin, New York, London

30

Percent

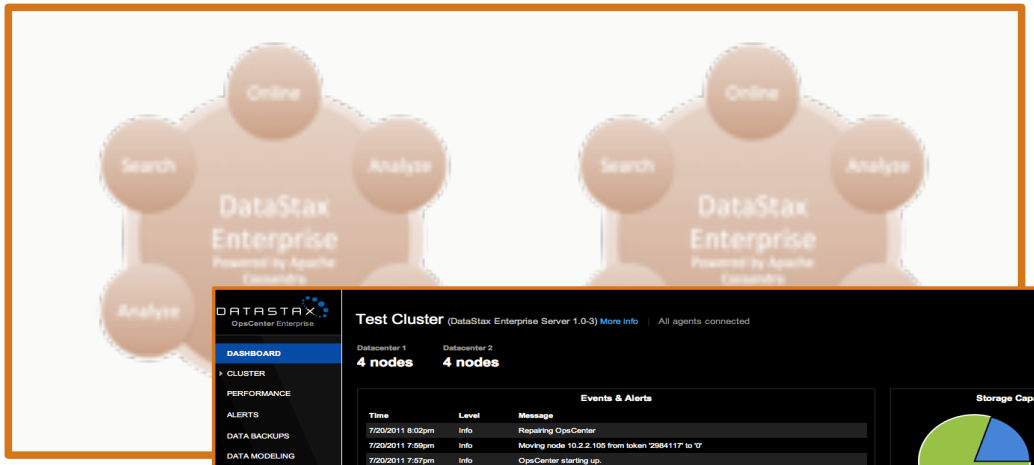
FORTUNE 100

500+

Customers

comcast ebay Adobe travelocity intuit accenture nest SAFEWAY

DataStax delivers Apache Cassandra to the Enterprise



Certified / Enterprise-ready Cassandra

Visual Management & Monitoring Tools

24x7 Support & Training

DATASTAX


CASSANDRASUMMIT2014
September 10 - 11 | #CassandraSummit
SAN FRANCISCO
WORLD'S LARGEST GATHERING
OF CASSANDRA DEVELOPERS.
FREE ADMISSION.
SERIOUSLY.

DSE 4.5

DATASTAX

Thank You!

cgilmore@datastax.com