DATASTAX

# A Look at Apache Cassandra…

# How Many Have Heard of…

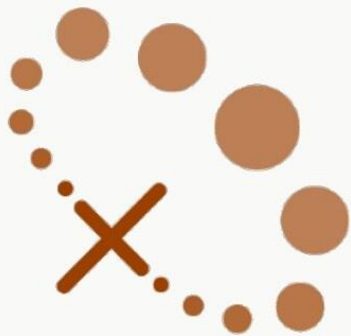
DATASTAX

- DataStax
- Apache Cassandra

# **What is** Apache Cassandra?

Apache Cassandra™ is a massively scalable NoSQL database.

Cassandra is designed to handle big data workloads across multiple data centers with no single point of failure, providing enterprises with continuous availability without compromising performance.



# Customer Industries

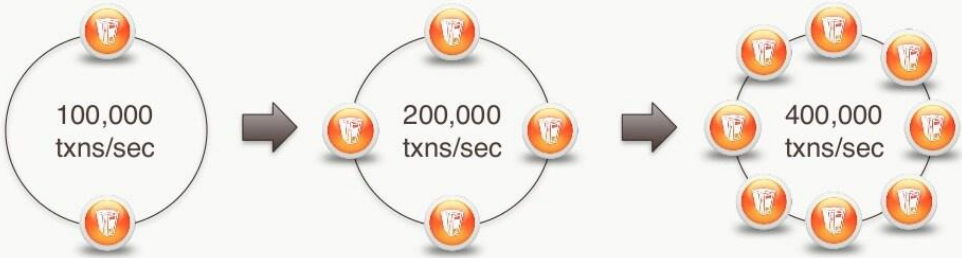
DATASTAX

- Financial
- eCommerce
- Marketing
- Recommendation Engines
- Health Sciences
- Fraud Detection
- Sensor Data
- Online Games

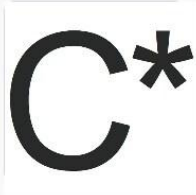
# **Why use** Cassandra?

DATASTAX

- Masterless architecture.
- Continuous availability.
- Multi-data center and cloud availability zone support.
- Flexible data model.
- Linear scale performance.
- Operationally simple.
- CQL – SQL-like language.



# History of Cassandra



- Amazon Dynamo partitioning and replication
- Log-structured ColumnFamily data model similar to Bigtable's

# Cassandra Terminology

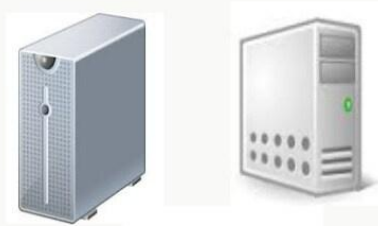
DATASTAX

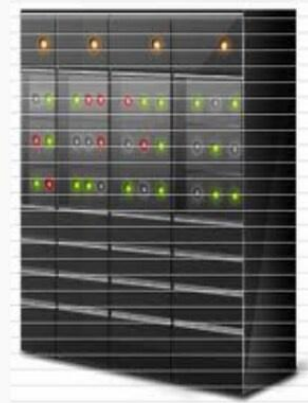
- Node (Vnode)
- Cluster
- Datacenter
- Partition
- Gossip
- Snitch
- Replication Factor
- Consistency

# Cassandra Node

DATASTAX

**A computer server running Cassandra**

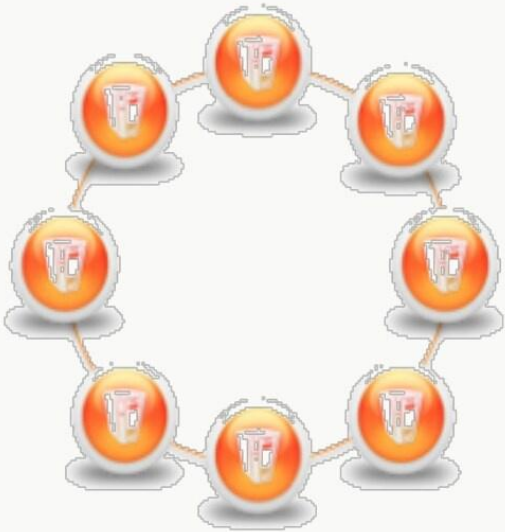




# Cassandra Cluster

DATASTAX

**A group of Cassandra nodes working together**

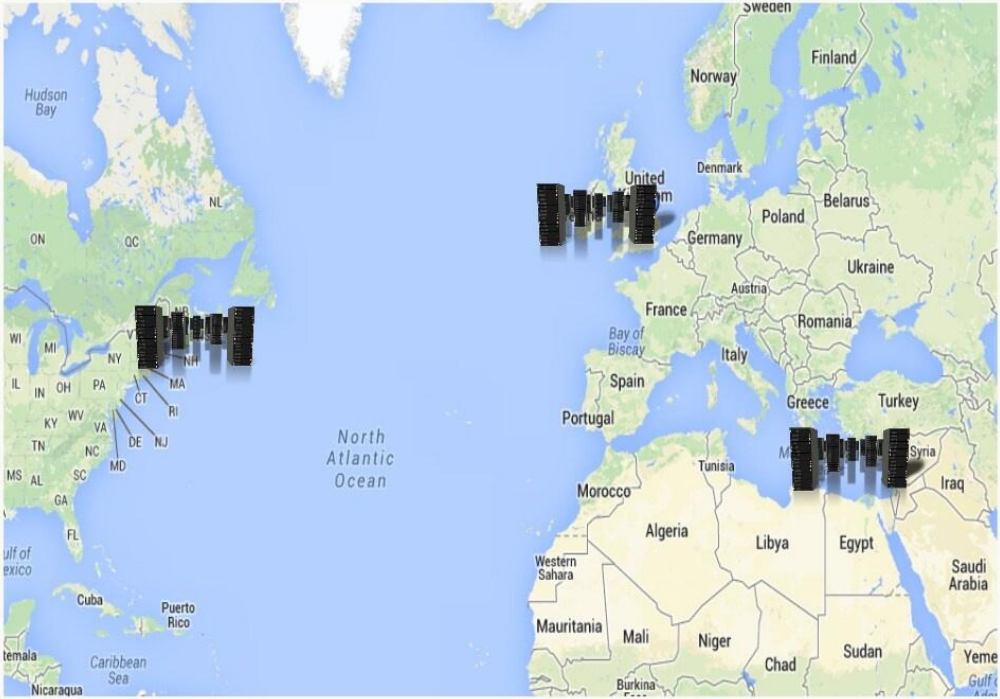


# Cassandra Datacenter

DATASTAX

# Logical Datacenter

# Physical Datacenter



# Cloud-based Datacenters

DATASTAX

# Cassandra Cluster

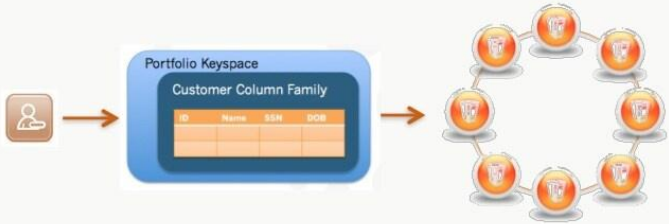
DATASTAX

**Data is evenly distributed around the nodes in a cluster**

# **Overview of Data** Partitioning in Cassandra

There are two basic data partitioning strategies:

1. **Random partitioning** – this is the default and recommended strategy. Partitions data as evenly as possible across all nodes using a hash of every column family row key
2. **Ordered partitioning** – stores column family row keys in sorted order across the nodes in a database cluster



# Data Distribution and Partitioning

DATASTAX

- Each node “owns” a set of tokens
  - A node’s token range is manually configured or randomly assigned when the node joins the cluster
- A partition key is defined for each table
- The partitioner applies a hash function to convert a partition key to a token. This determines which node(s) store that piece of data.
- By default, Cassandra users the Murmur3Partitioner

# What is a Hash Algorithm?

DATASTAX

- A function used to map data of arbitrary size to data of fixed size
- Slight differences in input produce large differences in output

| Input | MurmurHash |
|---|---|
| *1* | *8213365047359667313* |
| *2* | *5293579765126103566* |
| *3* | *-155496620801056360* |
| *4* | *-663977588974966463* |
| *5* | *958005880272148645* |

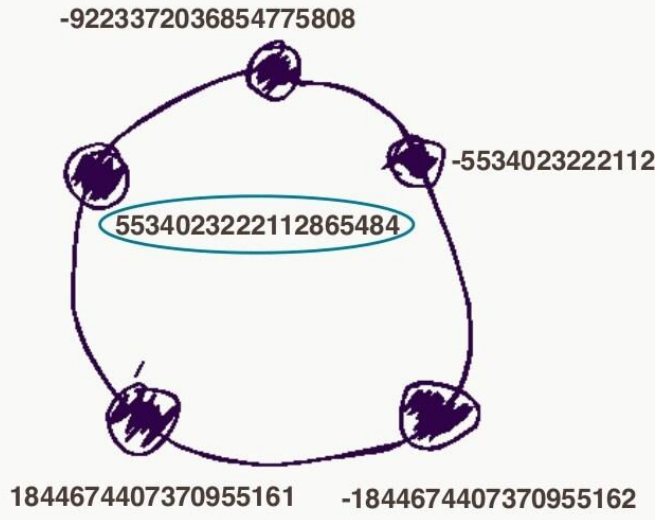
The Murmur3Partitioner uses the MurmurHash function. This hashing function creates a 64-bit hash value of the row key. The possible range of hash values is from $-2^{63}$ to $+2^{63}$.

# Data Distribution

DATASTAX

**Each node is configured with an initial token.**

**The initial token determines the token range owned by the node.**



# Data Distribution

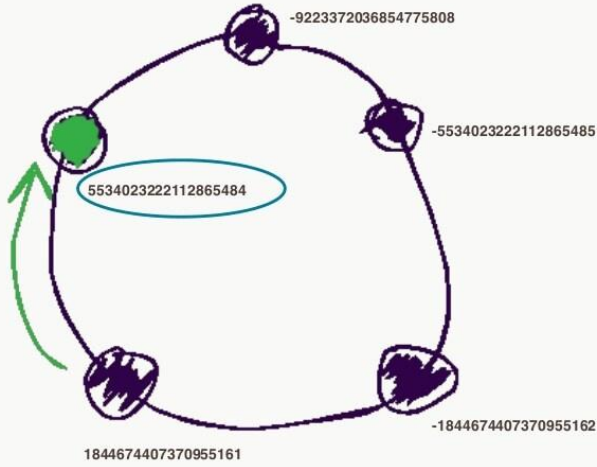
DATASTAX

**This node owns the token range:**

**1844674407370955162**

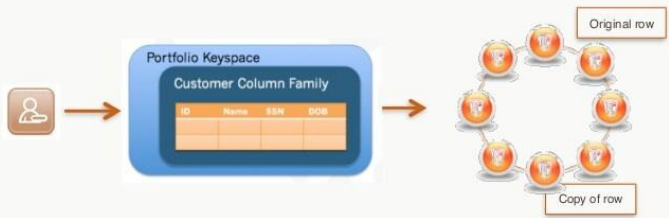
**To**

**5534023222112865484**



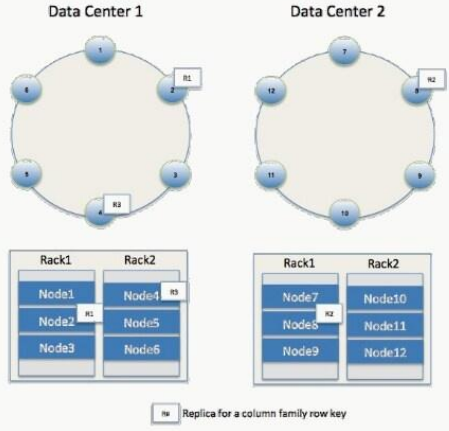
# Overview of Replication in Cassandra

- Replication is controlled by what is called the *replication factor*. A replication factor of 1 means there is only one copy of a row in a cluster. A replication factor of 2 means there are two copies of a row stored in a cluster
- Replication is controlled at the keyspace level in Cassandra



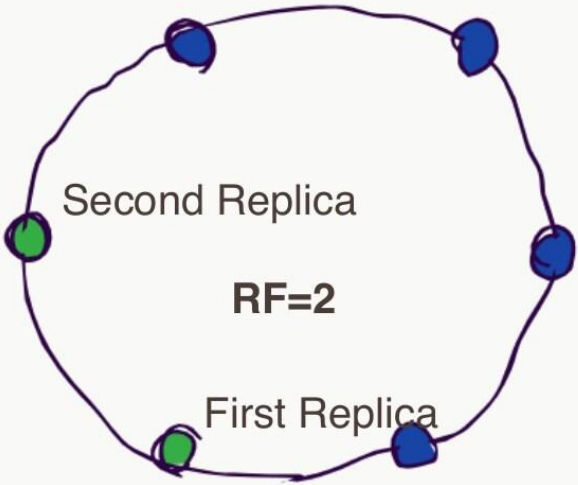
# **Cassandra** Replication Strategies

- **Simple Strategy**
- **Network Topology Strategy**:



# Simple Topology – Single Datacenter

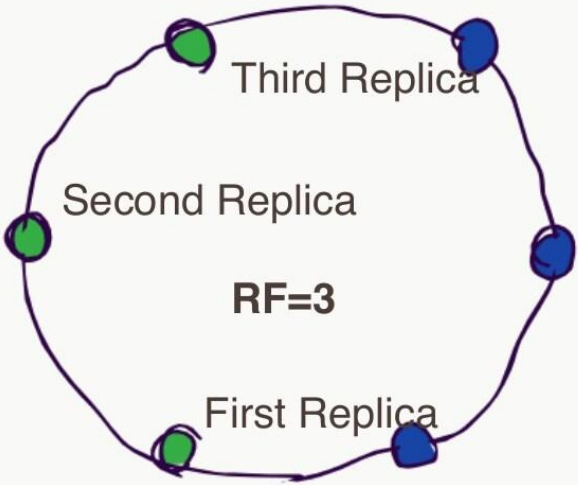
DATASTAX




**{ 'class' : 'SimpleStrategy', 'replication_factor' : 2 };**

# Simple Topology – Single Datacenter

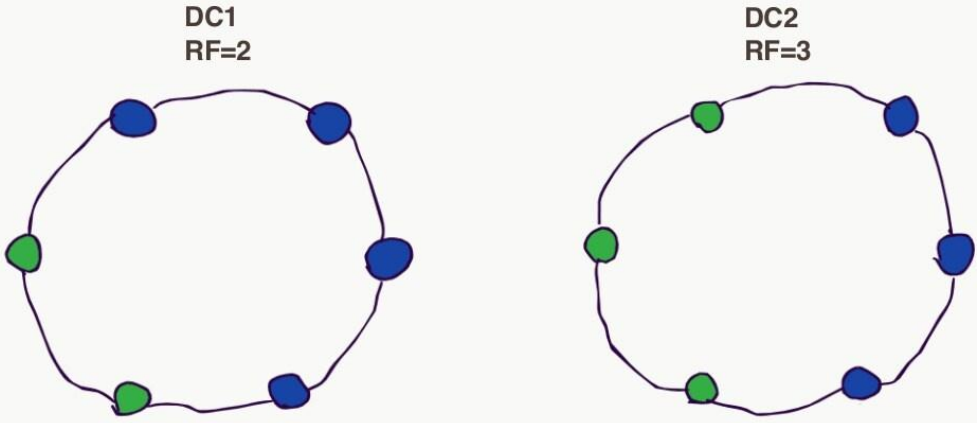
DATASTAX




```
{ 'class' : 'SimpleStrategy', 'replication_factor' : 3 };
```

# Network Topology – Multiple Datacenters

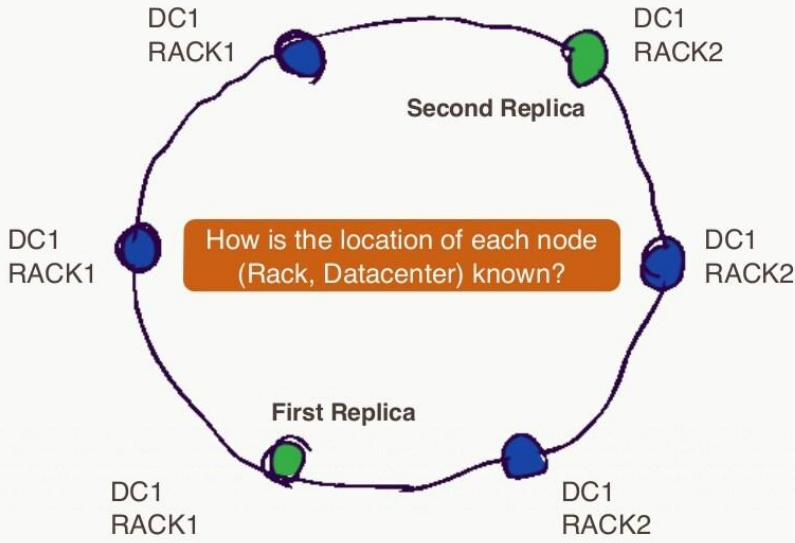
DATASTAX




```
CREATE KEYSPACE Test
  WITH REPLICATION = {'class' : 'NetworkTopologyStrategy', 'DC1' : 2, 'DC2' : 3};
```

# Network Topology – Rack Awareness

DATASTAX



# Snitches

DATASTAX

- A snitch determines which data centers and racks are written to and read from.
- Snitches inform Cassandra about the network topology so that requests are routed efficiently and allows Cassandra to distribute replicas by grouping machines into data centers and racks.
- Cassandra does its best not to have more than one replica on the same rack.

# Gossip = Internode Communications

DATASTAX

- Gossip is a peer-to-peer communication protocol in which nodes periodically (every second) exchange information about themselves and about other nodes they know about.
- Cassandra uses gossip to discover location and state information about the other nodes participating in a Cassandra cluster

# Vnodes (Virtual Nodes)

DATASTAX

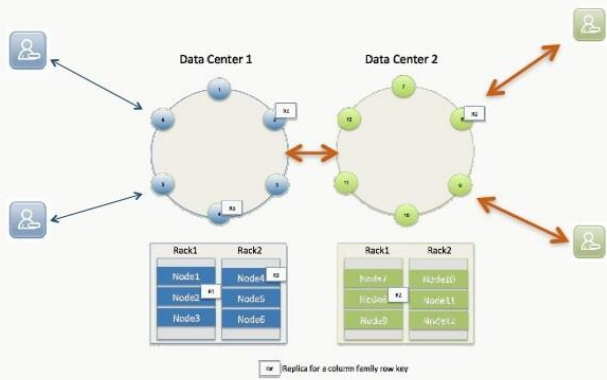
Instead of each node owning a single token range, Vnodes divide each node into many ranges (256).

Vnodes simplify many tasks in Cassandra:

- You no longer have to calculate and assign tokens to each node.
- Rebalancing a cluster is no longer necessary when adding or removing nodes.
- Rebuilding a dead node is faster because it involves every other node in the cluster.
- Improves the use of heterogeneous machines in a cluster. You can assign a proportional number of vnodes to smaller and larger machines.

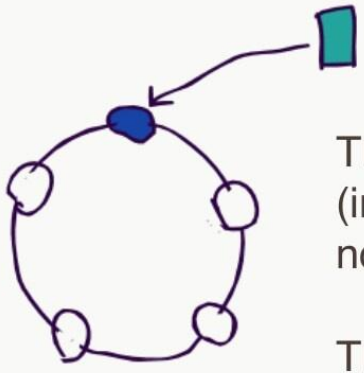
# Reading and Writing to Cassandra Nodes

- Cassandra has a 'location independence' architecture, which allows any user to connect to any node in any data center and read/write the data they need
- All writes being partitioned and replicated for them automatically throughout the cluster



# Writing Data

DATASTAX



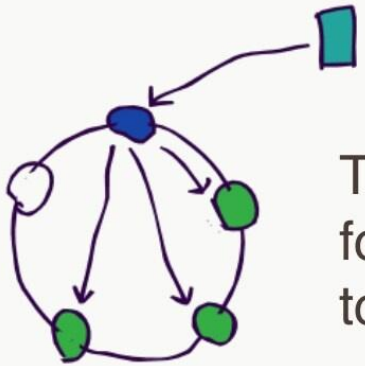
**RF=3**

The client sends a mutation (insert/update/delete) to a node in the cluster.

That node serves as the coordinator for this transaction

# Writing Data

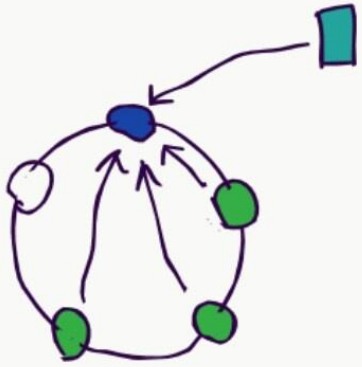
DATASTAX



The coordinator forwards the update to all replicas.

**RF=3**

# Writing Data

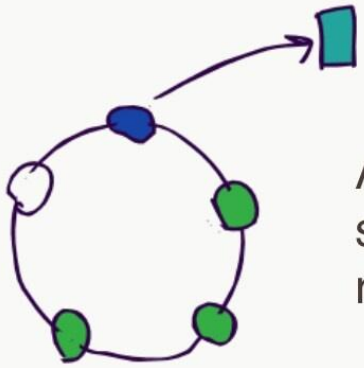


The replicas acknowledge that data was written.

**RF=3**

# Writing Data

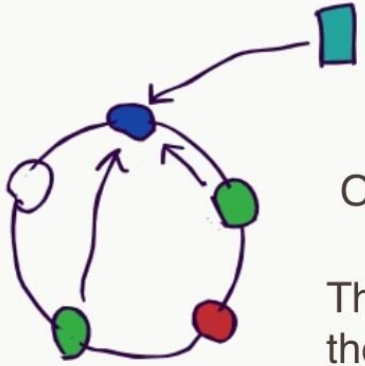
DATASTAX



And the coordinator sends a successful response to the client.

**RF=3**

# What if a node is down?



RF=3

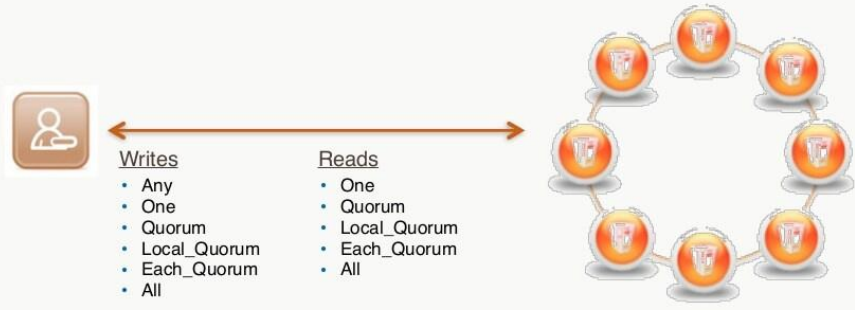
Only two nodes respond.

The client gets to choose if the write was successful.

**Write Consistency (WC)**

# Tunable Data Consistency

- Choose between strong and eventual consistency (one to all responding) depending on the need
- Can be done on a per-operation basis, and for both reads and writes
- Handles multi-data center operations



# Consistency

DATASTAX

- **ANY**

Returns data from any of the replica.

- **QUORUM**

Returns the most recent data from the majority of replicas.

- **LOCAL QUORUM**

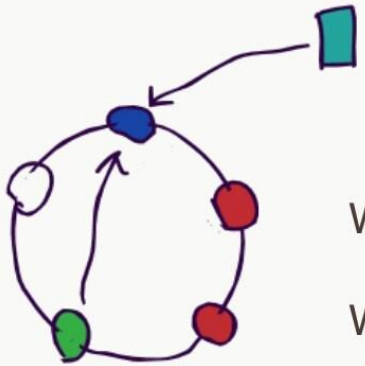
Returns the most recent data from the majority of local replicas.

- **ALL**

Returns the most recent data from all replicas.

# What if two nodes are down?

DATASTAX



RF=3

WC = QUORUM

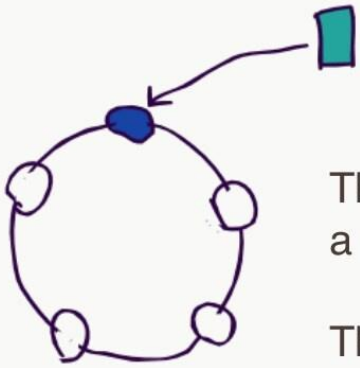
Will this write succeed?

**NO**

Failed to write a majority of replicas.

# Reading Data

DATASTAX

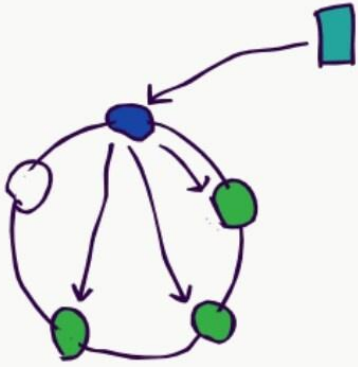


**RF=3**

The client sends a query to a node in the cluster.

That node serves as the coordinator.

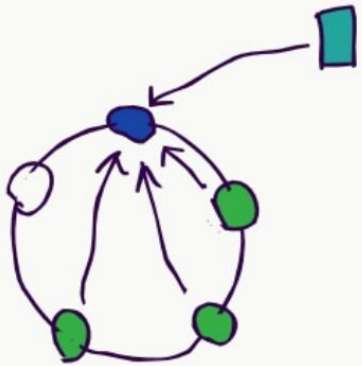
# Reading Data



RF=3

The coordinator forwards the query to all replicas.

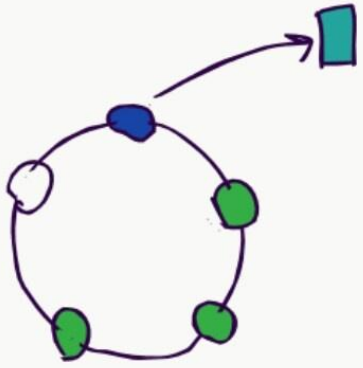
# Reading Data



The replicas respond with data.

**RF=3**

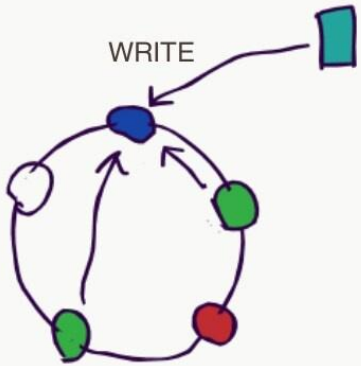
# Reading Data



RF=3

And the coordinator returns the data to the client.

# What if the nodes disagree?

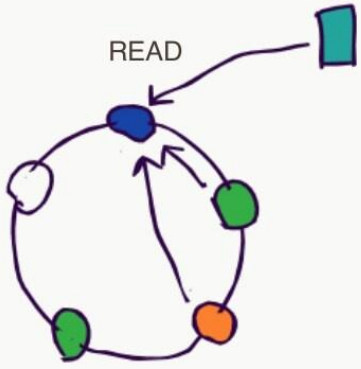



**RF=3**

Data was written with QUORUM when one node was down.

The write was successful, but that node missed the update.

# What if the nodes disagree?

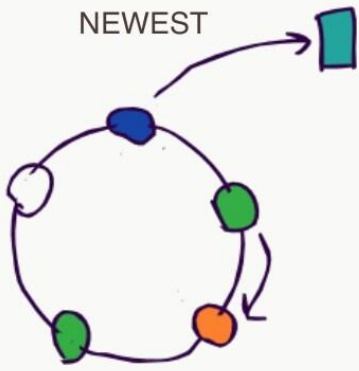



**RF=3**

Now the node is back online, and it responds to a read request.

It has older data than the other replicas.

# What if the nodes disagree?




**RF=3**

The coordinator resolves the discrepancy and sends the newest data to the client.

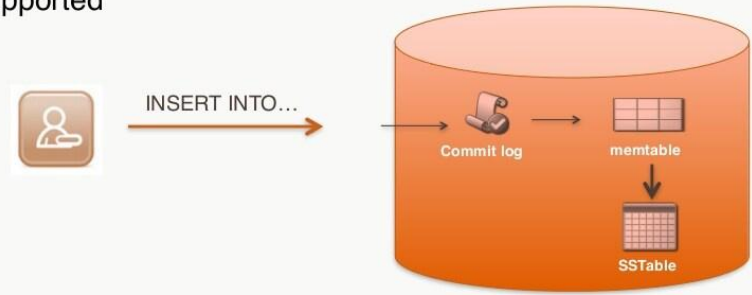
**READ REPAIR**

The coordinator also notifies the “out of date” node that it has old data.

The “out of date” node receives updated data from another replica.

# Writes (what happens within each node)

- Data is first written to a commit log for durability. Your data is safe in Cassandra
- Then written to a memtable in memory
- Once the memtable becomes full, it is flushed to an SSTable (sorted strings table)
- Writes are atomic at the row level; all columns are written or updated, or none are. RDBMS-styled transactions are not supported

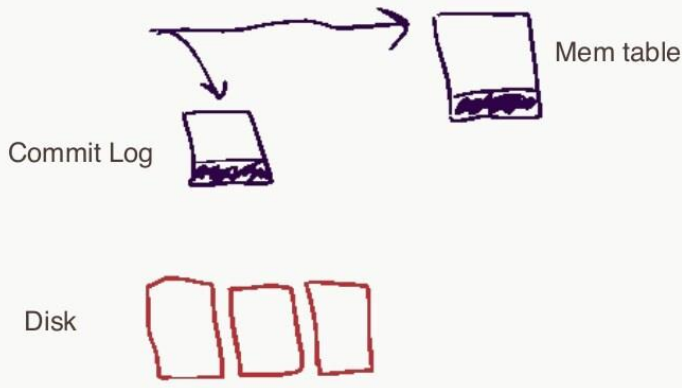



Cassandra is known for being the fastest database in the industry where write operations are concerned.

# SSTables


Cassandra writes to a commit log and to memory. When the memtable is full, data is flushed to disk, creating an SSTable.

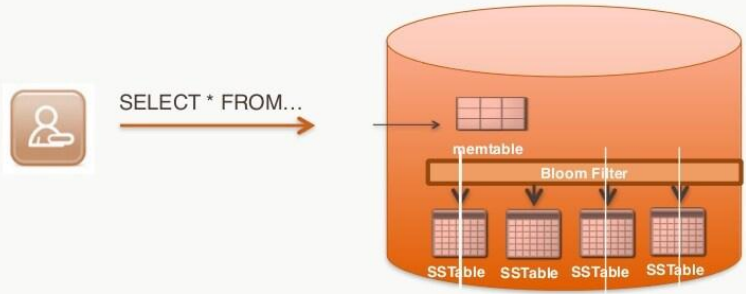
# Compaction




SSTables are cleaned up using compaction. Compaction creates a new combined SSTable and deletes the old ones.

# Reads (what happens within each node)

- Depending on the frequency of inserts and updates, a record will exist in multiple places. Each place must be read to retrieve the entire record.
- Data is read from the memtable in memory.
- Multiple SSTables may also be read.
- Bloom filters prevent excessive reading of SSTables.



# DataStax Distinctives

# What is DataStax Enterprise?

DATASTAX

Certified Production Cassandra
Multi-Workload/Use Case Capable
Integrated OLTP, Analytics, Search


Strong Data Protection
In-Memory OLTP/Analytics
Point-and-Click/Automated Mgmt

# Cassandra Clients – API & Native Driver

DATASTAX

- CQL (Cassandra Query Language) is the primary API
- Clients that use the native driver also have access to various policies that enable the client to intelligently route requests as required.
- DataStax drivers: Java, Python, C#, C++, Ruby, Node.js (much more to come and in the community)
- This includes:
  - Load Balancing
    - Data Centre Aware
    - Latency Aware
    - Token Aware
  - Reconnection policies
  - Retry policies
    - Downgrading Consistency
    - Plus others..


- http://www.datastax.com/download/clientdrivers

# Enterprise Security

# **Security** in Cassandra

DATASTAX

FEATURES

**Internal Authentication**
Manages login IDs and passwords inside the database

BENEFITS

+ Ensures only authorized users can access a database system using internal validation
+ Simple to implement and easy to understand
+ No learning curve from the relational world

**Object Permission Management**
controls who has access to what and who can do what in the database

+ Provides granular based control over who can add/change/delete/read data
+ Uses familiar GRANT/ REVOKE from relational systems
+ No learning curve

**Client to Node Encryption**
protects data in flight to and from a database cluster

+ Ensures data cannot be captured/stolen in route to a server
+ Data is safe both in flight from/to a database and on the database; complete coverage is ensured

# **Advanced Security** in DataStax Enterprise

DATASTAX

FEATURES

**External Authentication**
uses external security software systems to control security

BENEFITS

+ Only authorized users have access to a database system using external validation
+ Uses most trusted external security systems (Kerberos, LDAP, AD), mainstays in government and finance
+ Single sign on to all data domains

**Transparent Data Encryption**
encrypts data at rest

+ Protects sensitive data at rest from theft and from being read at the file system level
+ No changes needed at application level

PCI DSS COMPLIANT

**Data Auditing**
provides trail of who did and looked at what/when

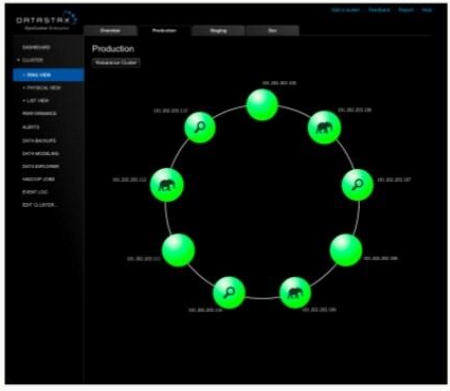
+ Supplies admins with an audit trail of all accesses and changes
+ Granular control to audit only what's needed
+ Uses log4j interface to ensure performance and efficient audit operations

# OpsCenter

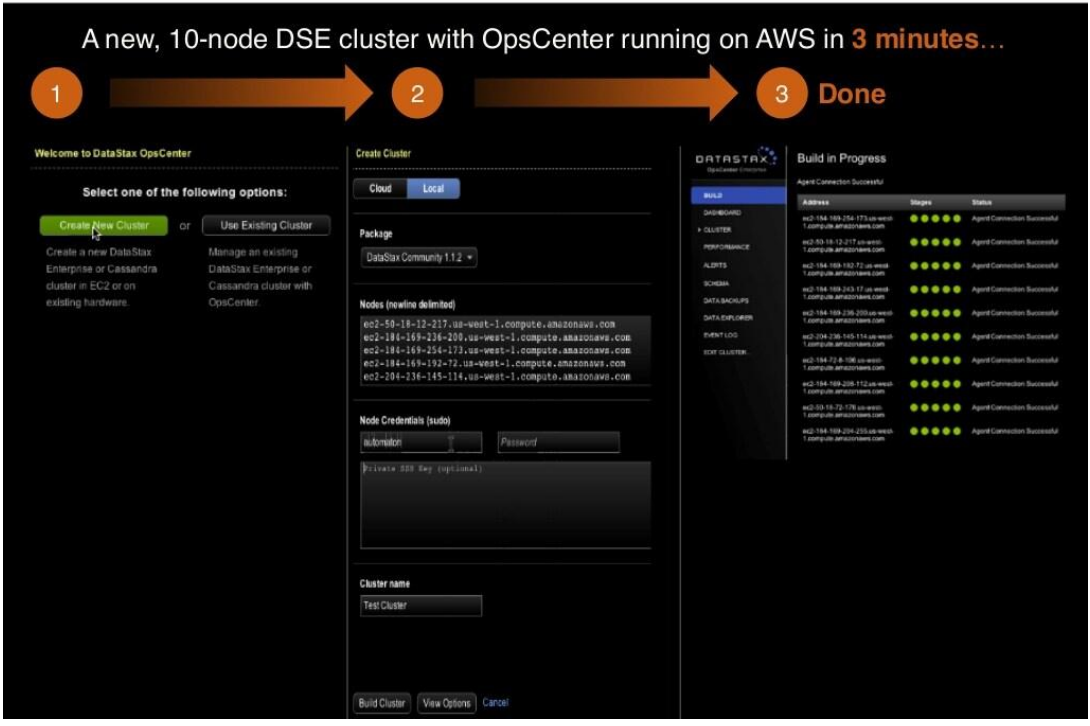
# **DataStax** OpsCenter

DATASTAX

- Visual, browser-based user interface negates need to install client software
- Administration tasks carried out in point-and-click fashion
- Allows for visual rebalance of data across a cluster when new nodes are added
- Contains proactive alerts that warn of impending issues.
- Built-in external notification abilities
- Visually perform and schedule backup operations



# DataStax OpsCenter



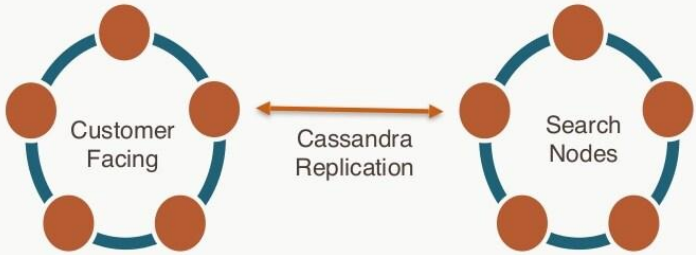
# Integrated Search

# Enterprise Search

DATASTAX

- Built-in enterprise search on Cassandra data via Solr integration
- Facets, Filtering, Geospatial search, Text Analysis, etc.
- Near real-time search operations
- Search queries from CQL and REST/Solr
- Solr shortcomings:
  - No bottleneck. Client can read/write to any Solr node.
  - Search index partitioning and replication for scalability and availability.
  - Multi-DC support
  - Data durability (Solr lacks write-ahead log, data can be lost)

Apache Solr

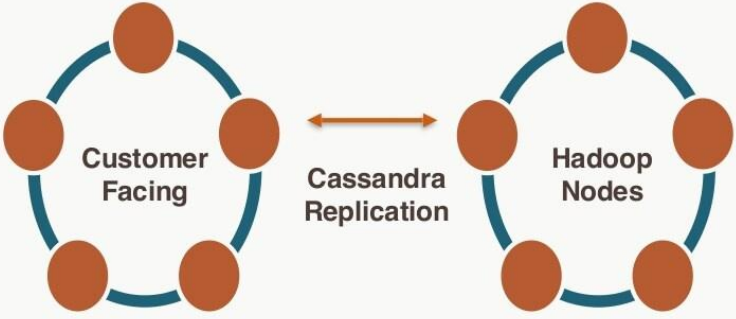


# Integrated Batch Analytics

# Batch Analytics - Hadoop

DATASTAX


- Integrated Hadoop 1.0.4
- CFS (Cassandra File System) , no HDFS
- No Single Point of failure
- No Hadoop complexity – every node is built the same
- Hive / Pig / Sqoop / Mahout



# External Batch Analytics

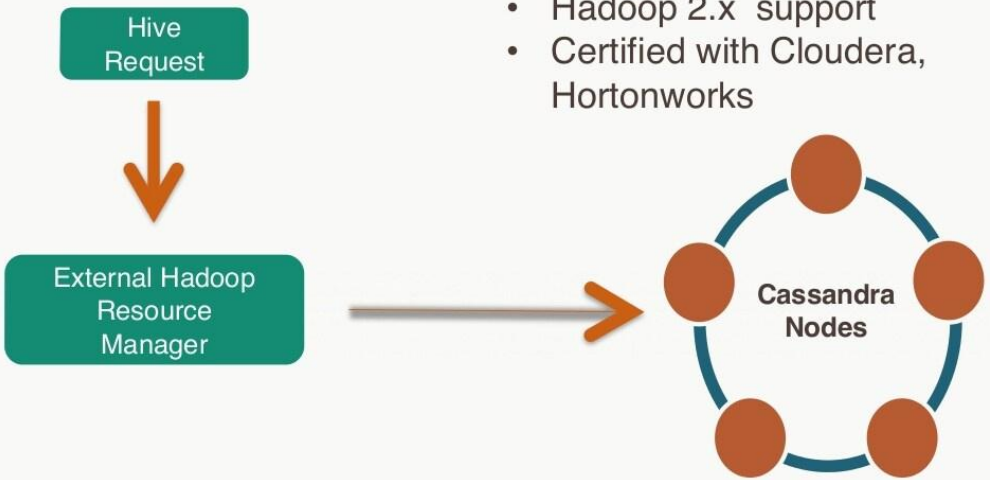
# External Batch Analytics - **BYOH**

DATASTAX

## Bring Your Own Hadoop

cloudera Ask Bigger Questions  Hortonworks

- Analytics from external Hadoop distribution
- Hadoop 2.x support
- Certified with Cloudera, Hortonworks

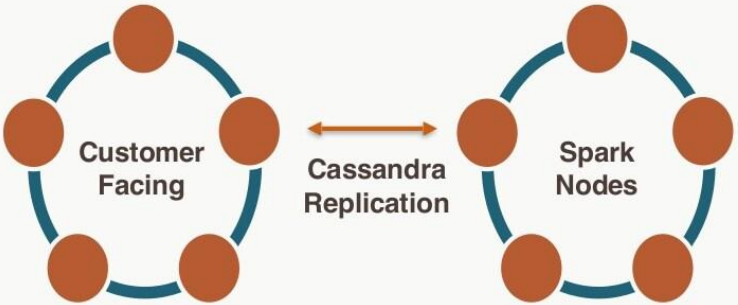


# Integrated Near Real-Time Analytics

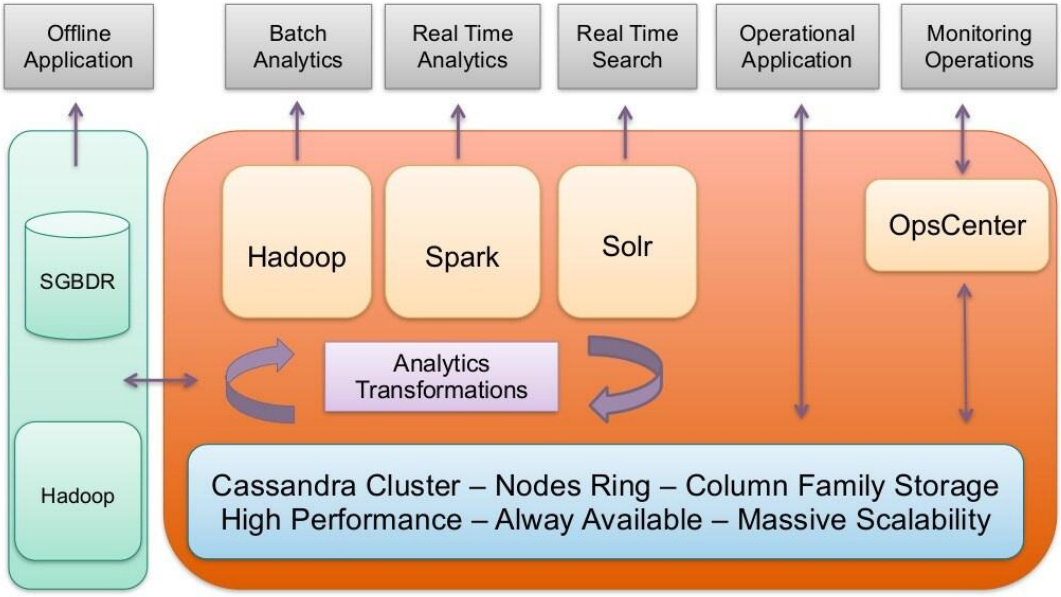
# Real-Time Analytics - Spark

DATASTAX

- Tight integration with Cassandra
- Real-time Streaming
- Distributed Processing
- “In-memory Map/Reduce”, multi-thread, best for iterations
- GraphX, MLLib, SparkSQL, Shark (Hive SQL like)
- DataStax / Databricks partnership
- **10x – 100x speed of MapReduce**



# DataStax Enterprise Products




External Hadoop Distribution
Cloudera, Hortonworks

DataStax Cassandra Enterprise

# How to test DataStax Cassandra ?

DATASTAX

- **CCM**, A development tool for creating local Cassandra clusters
  - http://www.datastax.com/dev/blog/ccm-a-development-tool-for-creating-local-cassandra-clusters
  - https://github.com/pcmanus/ccm
- Using **Vagrant** for Local Cassandra Development (VM)
  - http://www.cantoni.org/2014/08/26/vagrant-cassandra
  - https://github.com/bcantoni/vagrant-cassandra
- **Sandbox**
  - http://www.datastax.com/download#dl-sandbox
- **Cloud** : Amazon AWS, Google Cloud, Microsoft Azure
- DataStax Code **Samples**
  - https://github.com/DataStaxCodeSamples/

# Find out more

DATASTAX

- DataStax: http://www.datastax.com
- Getting Started: http://www.datastax.com/documentation/gettingstarted/index.html
- Training: http://www.datatstax.com/training
- Downloads: http://www.datastax.com/download
- Documentation: http://www.datastax.com/docs
- Developer Blog: http://www.datastax.com/dev/blog
- Community Site: http://planetcassandra.org
- Webinars: http://planetcassandra.org/Learn/CassandraCommunityWebinars
- Summit Talks: http://planetcassandra.org/Learn/CassandraSummit

DATASTAX

# Thank You !

# Q&A