APACHE CASSANDRA

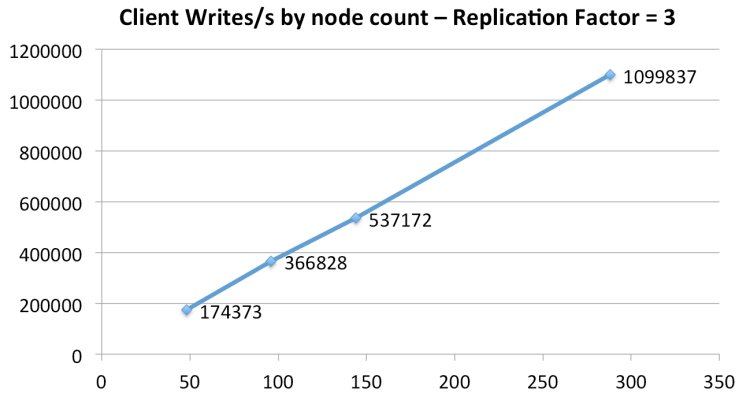
Vijay Dialani
Associate Professor,
Department of Computer Science
Boise State University

About Apache Cassandra

- Linear scalability
- Fault-tolerance on commodity hardware or cloud infrastructure
- Support for replication across multiple datacenters is best-in-class
- Provides lower latency for users
- Can survive regional outages.

Scalable Write Performance

Scale-Up Linearity



NETFLIX

Workload Distribution

Per Node Activity

Per Node	48 Nodes	96 Nodes	144 Nodes	288 Nodes
Per Server Writes/s	10,900 w/s	11,460 w/s	11,900 w/s	11,456 w/s
Mean Server Latency	0.0117 ms	0.0134 ms	0.0148 ms	0.0139 ms
Mean CPU %Busy	74.4 %	75.4 %	72.5 %	81.5 %
Disk Read	5,600 KB/s	4,590 KB/s	4,060 KB/s	4,280 KB/s
Disk Write	12,800 KB/s	11,590 KB/s	10,380 KB/s	10,080 KB/s
Network Read	22,460 KB/s	23,610 KB/s	21,390 KB/s	23,640 KB/s
Network Write	18,600 KB/s	19,600 KB/s	17,810 KB/s	19,770 KB/s

Node specification – Xen Virtual Images, AWS US East, three zones

- Cassandra 0.8.6, CentOS, SunJDK6
- AWS EC2 m1 Extra Large – Standard price $ 0.68/Hour
- 15 GB RAM, 4 Cores, 1Gbit network
- 4 internal disks (total 1.6TB, striped together, md, XFS)

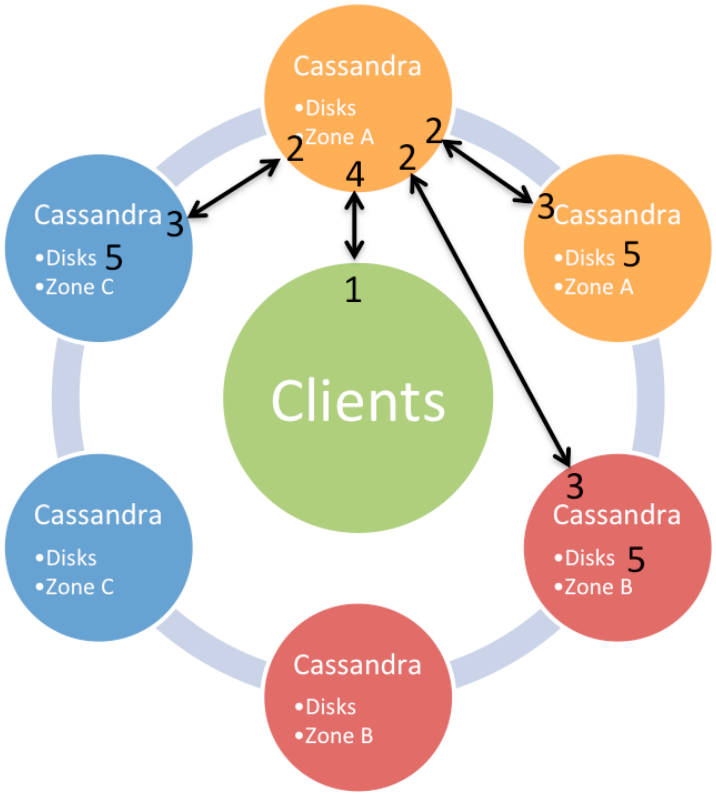
NETFLIX

NETFLIX setup of Apache Cassandra

Cassandra Write Data Flows

Single Region, Multiple Availability Zone

1. Client Writes to any Cassandra Node
2. Coordinator Node replicates to nodes and Zones
3. Nodes return ack to coordinator
4. Coordinator returns ack to client
5. Data written to internal commit log disk



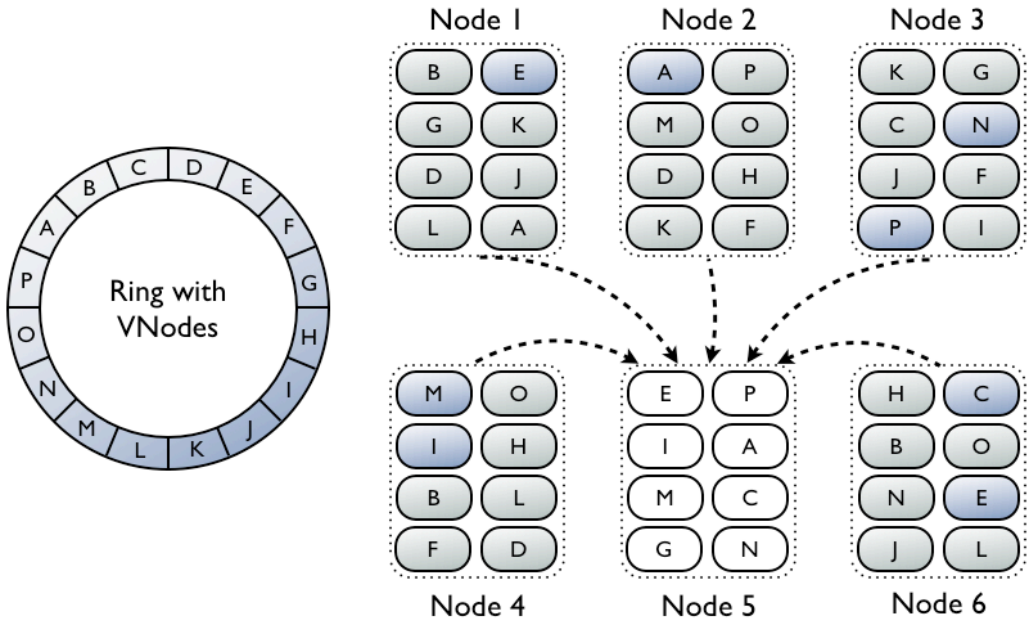
If a node goes offline, hinted handoff completes the write when the node comes back up.

Requests can choose to wait for one node, a quorum, or all nodes to ack the write

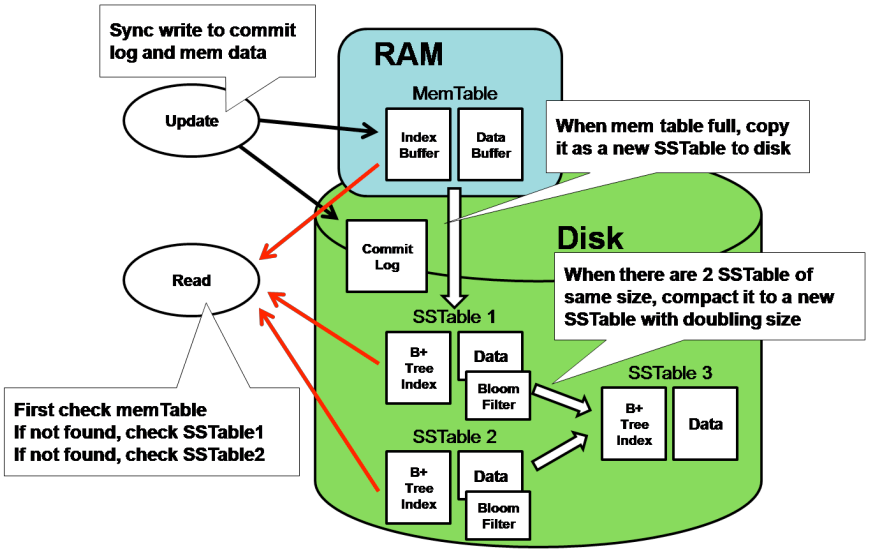
SSTable disk writes and compactions occur asynchronously

NETFLIX

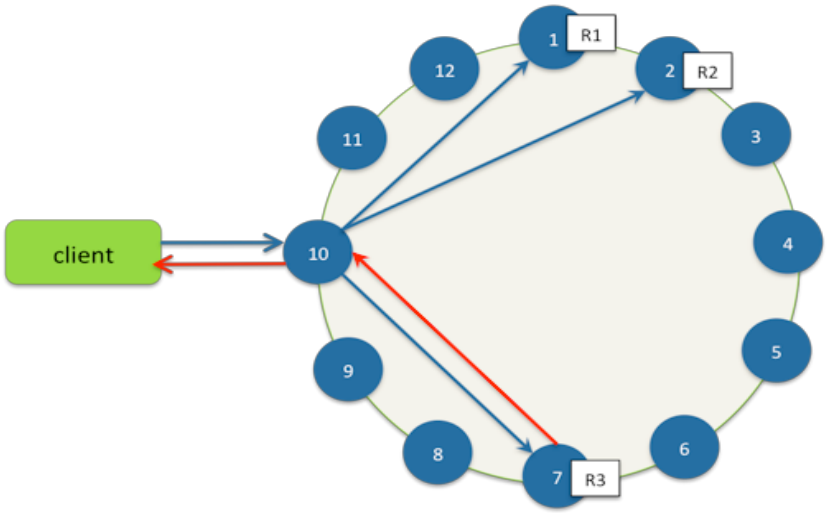
Cassandra V-Node Architecture



Single Node Architecture



Topology aware Client



APACHE CASSANDRA

Cassandra Cluster (nodes configuration - cassandra-topology.properties)

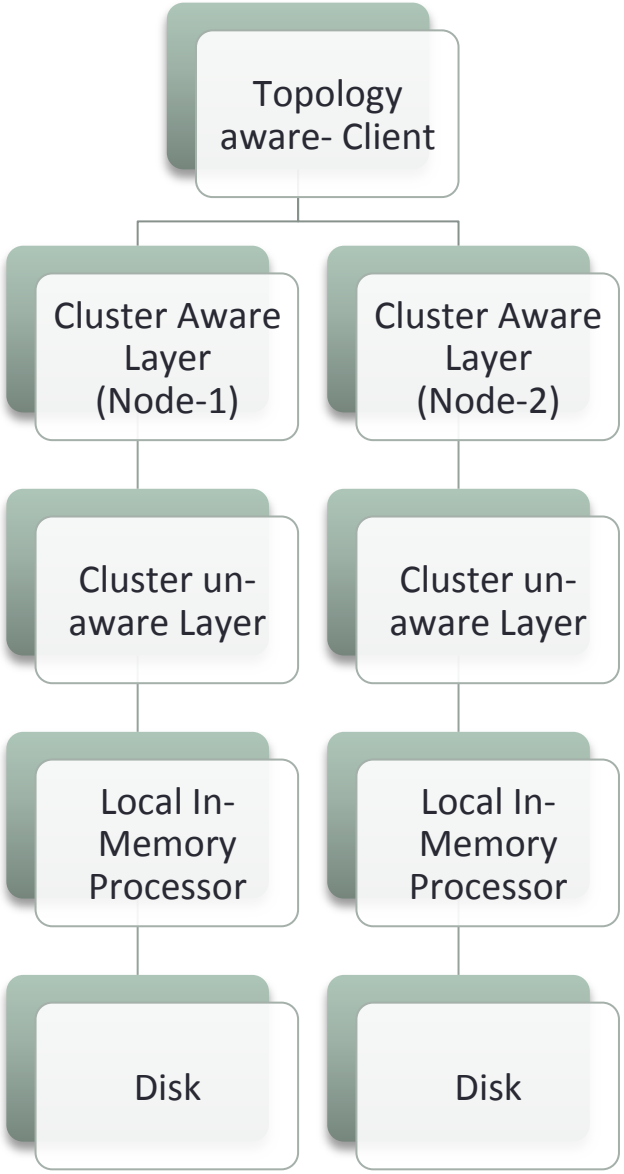
- # Cassandra Node IP=Data Center:Rack
- 192.168.1.100=DC1:RAC1
- 192.168.2.200=DC2:RAC2

- 10.0.0.10=DC1:RAC1
- 10.0.0.11=DC1:RAC1
- 10.0.0.12=DC1:RAC2

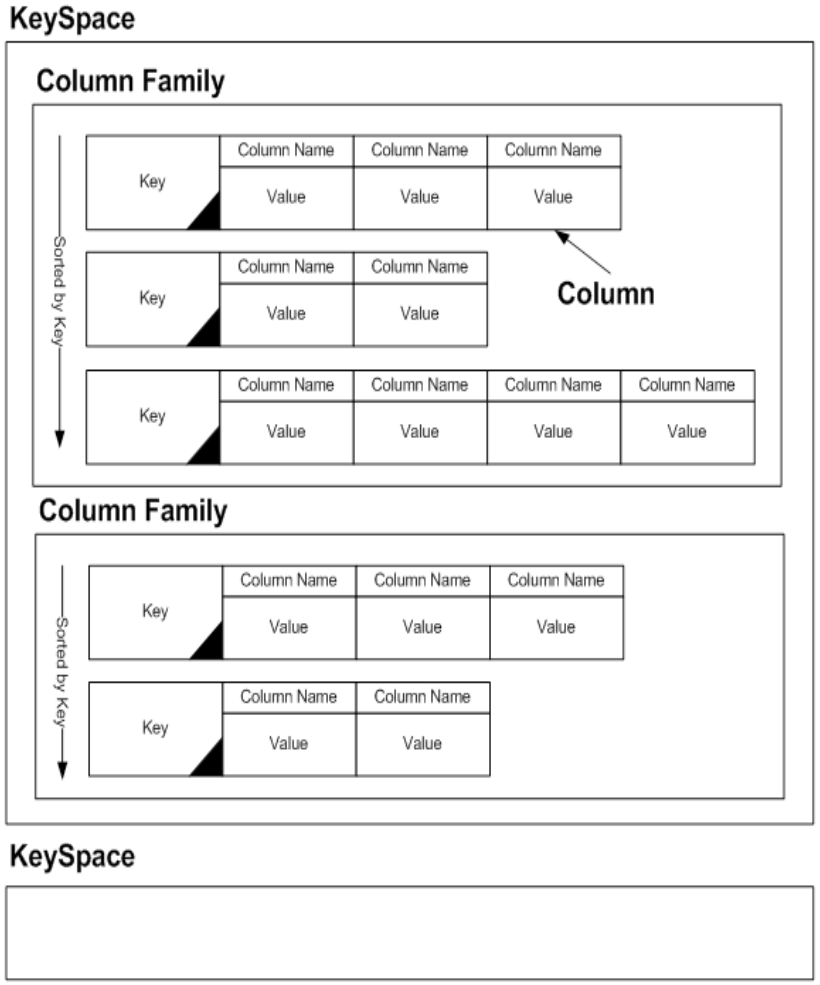
Configuring Cluster Properties – cassandra-rackdc.properties

```
# These properties are used with GossipingPropertyFileSnitch and will
# indicate the rack and dc for this node
dc=DC1
rack=RAC1

# Add a suffix to a datacenter name. Used by the Ec2Snitch and Ec2MultiRegionSnitch
# to append a string to the EC2 region name.
#dc_suffix=

# Uncomment the following line to make this snitch prefer the internal ip when possible, as
the Ec2MultiRegionSnitch does.
# prefer_local=true
```

Cassandra Architecture



KEYSPACES and Column Families



Cassandra Query Language

- Subset of SQL
- Allows create, alter, drop and rename – Keyspaces, Column Families, Columns
- Supports select, insert, update, delete on Rows/RowCollections
- Extends SQL to include replication specific constructs
- Does not support Joins

Example - Key Space and Column Family creation

```
CREATE KEYSPACE Excalibur WITH replication =
{'class': 'NetworkTopologyStrategy', 'DC1' : 1, 'DC2' : 3}
AND durable_writes = false;

USE KEYSPACE Excalibur;

CREATE TABLE monkeySpecies ( species text PRIMARY KEY, common_name text,
population varint, average_size int ) WITH comment='Important biological
records' AND read_repair_chance = 1.0;
```

Note that the CREATE COLUMNFAMILY syntax is supported as an alias for CREATE TABLE

QUESTIONS AND ANSWERS