An Introduction to Cassandra Database

JHUG Meetup

An Introduction to Cassandra Database

Petros Kaklamanis

December 2016

GENERAL

What is Cassandra (I)

- Fast Distributed DB
- High Availability
- Near-Linear Horizontal Scalability

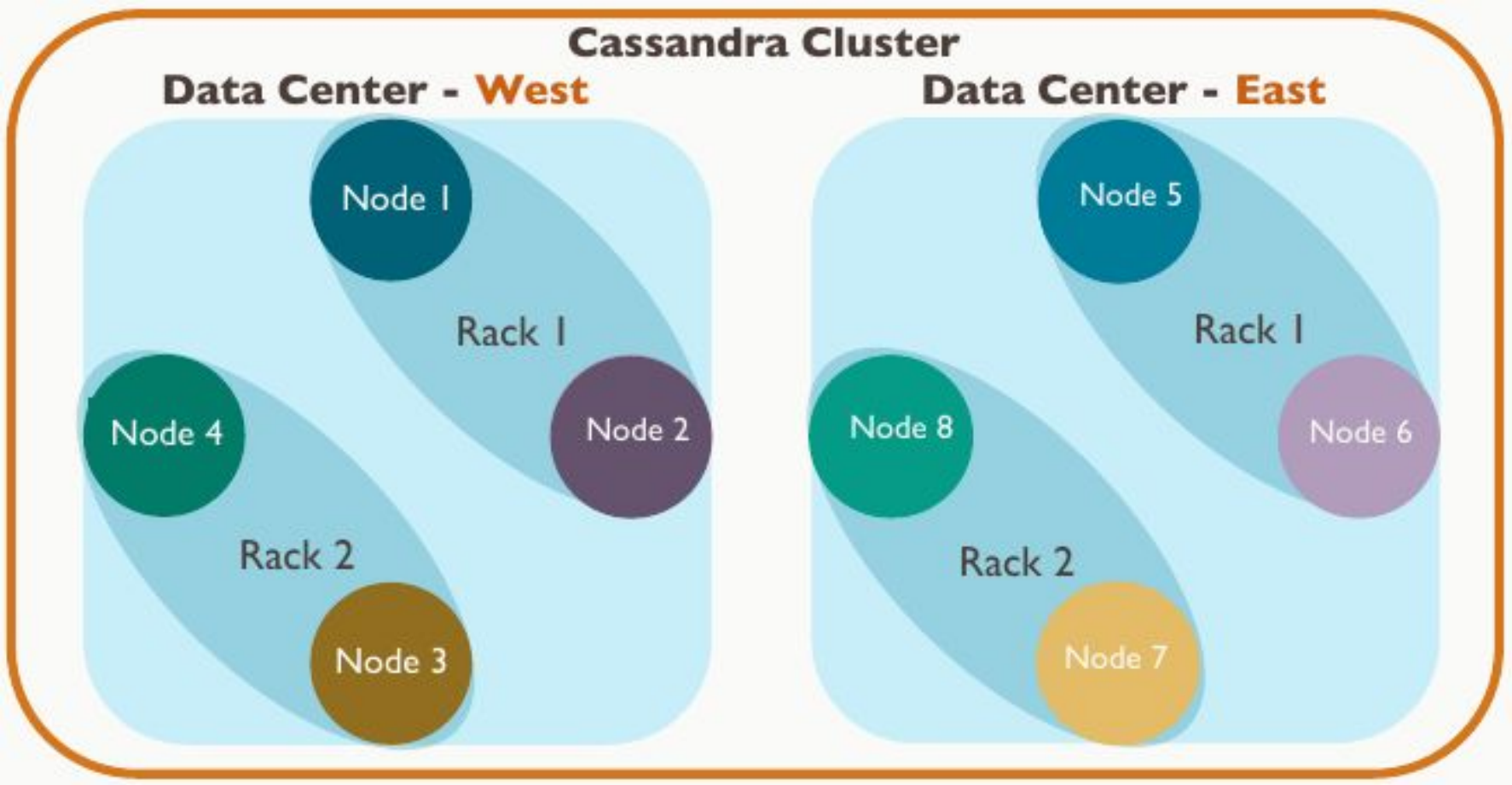
C*

What is Cassandra (II)

- Predictable Performance
- Fault Tolerance (Peer-to-Peer)
- Cannot replace RDBMS ad hoc
  - Data Model is different
  - Transaction mechanism is different
    - it is not ACID
- Current version is 3.9

- Google Big Table
  - Storage Model
- Amazon Dynamo
  - Distribution backbone
- Facebook integrated these two (2008)
  - Later released as Cassandra
  - Nowadays an Apache project

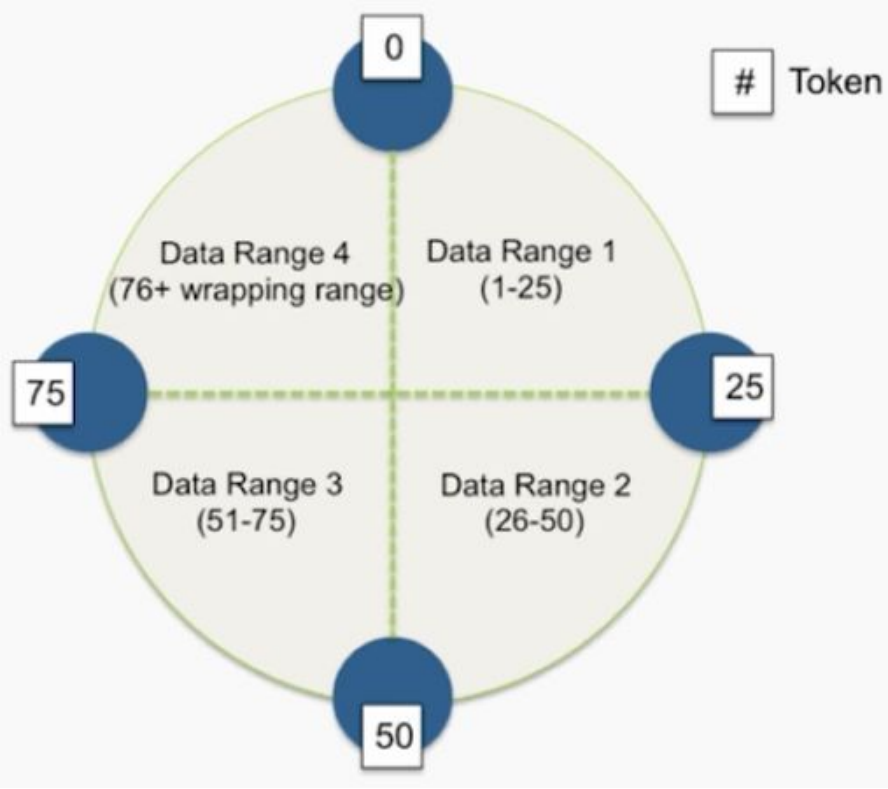
Structure (I)



C*

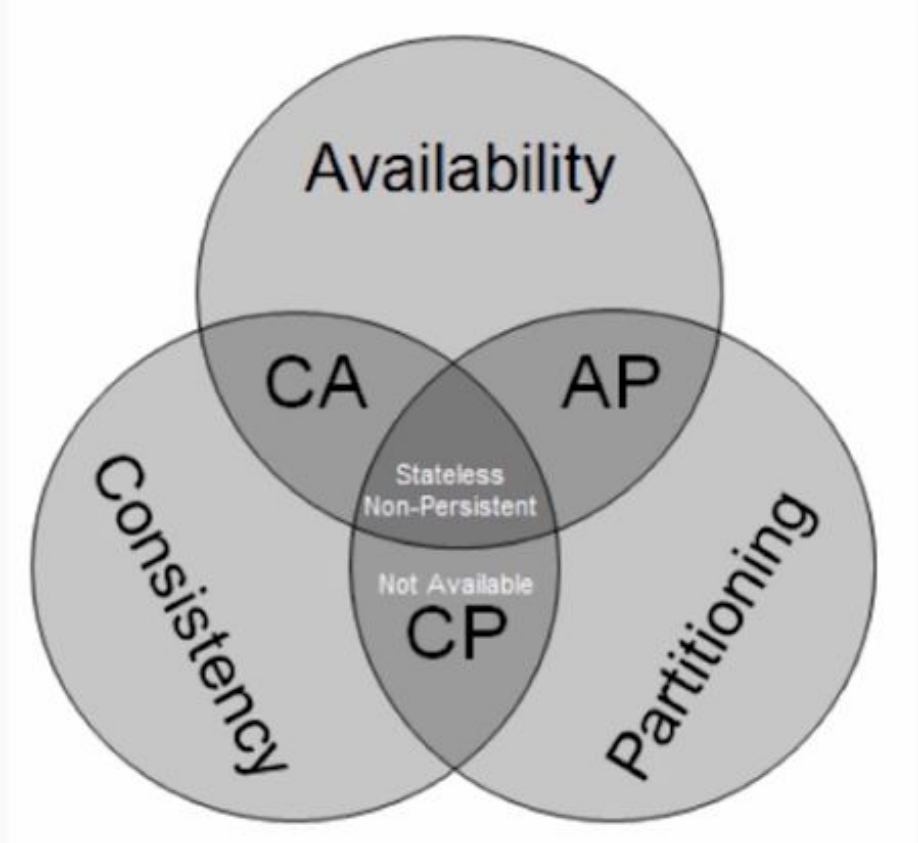
Structure (II)

- Hash Ring
- P2P
- Data partitioning
- Replication across peers



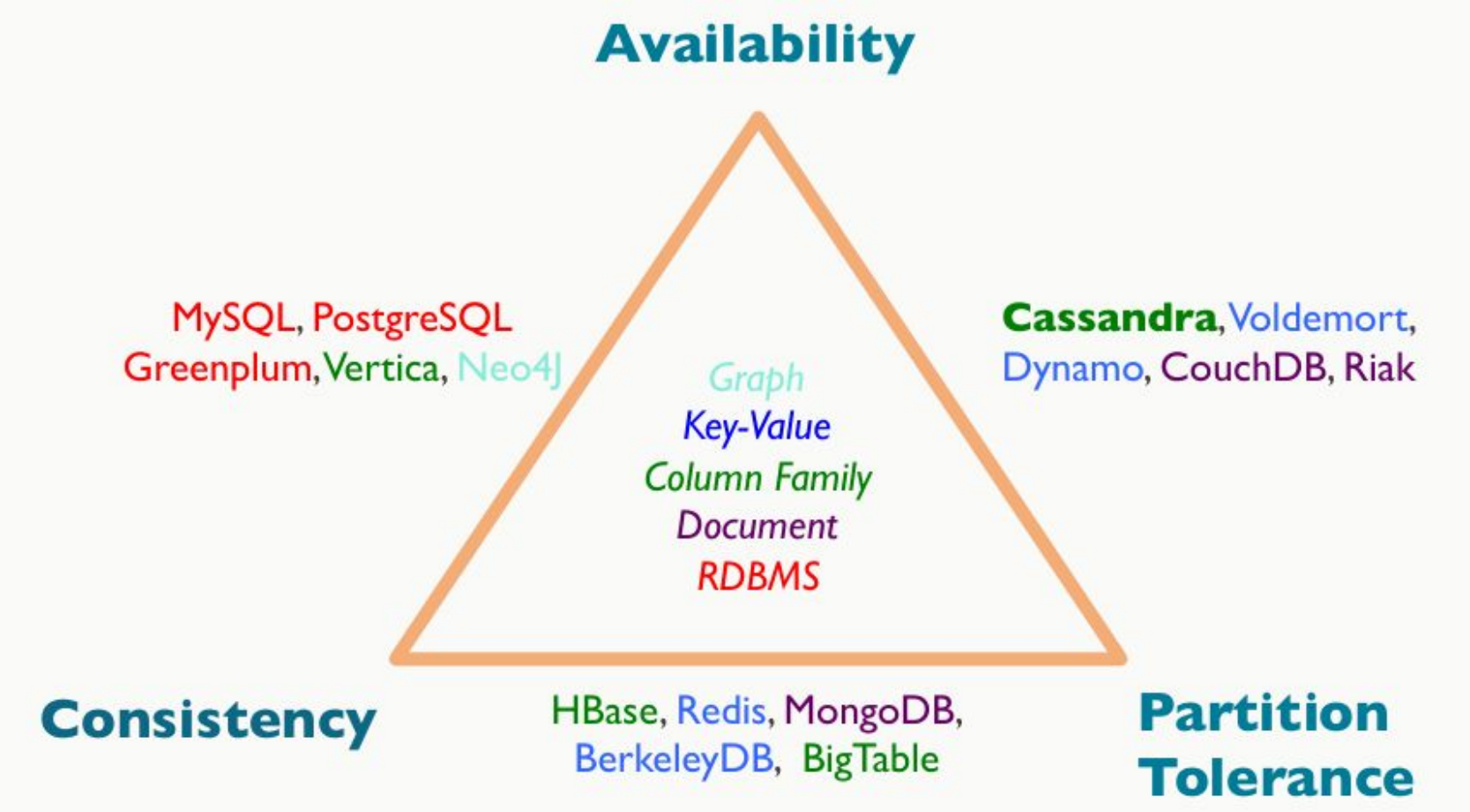
CAP theorem (I)

- Consistency Availability Partitioning trade-off
- Partitioning = Partition Tolerance = same network or not
- C* design choice:
  - A, P over C



C*

CAP theorem (II)



C*

Common Use Cases

- From: http://www.planetcassandra.org/apache-cassandra-use-cases/
- Product Catalog and Playlist
- Recommendation and Personalization
- Fraud Detection
- Messaging
- IOT and Sensor Data
- Marketing and Advertising
- Social Media and Networking

INTERNALS

Replication Factor

- Definition “how many copies of our data, do exist in a cluster” (RF)
- Data is always replicated
- RF is defined and configured for a KeySpace per Data Center
- A KeySpace is a “collection of Tables”

Multiple Data Centers

- DCs can be physical or logical
- Asynchronous replication to other DCs
- CREATE KEYSPACE hospital

  WITH REPLICATION = {

  'class' : 'SimpleStrategy',

  'replication_factor' : 3

  };

Consistency Level

- Definition “How many replicas respond Properly to a query” in order to consider the query successful
  - A query can be a Read or a Write
- Examples: ALL, QUORUM, ONE
- Consistency Level (CL) affects performance and availability (fault-tolerance)
- CL is configured per query
  - This enables using C* even in **CA**P mode

Consistency Level Details

- Several are available
- Defined per request, by default ONE

Name	Description	Usage
ANY (writes only)	Write to any node, and store *hinted handoff* if all nodes are down.	Highest availability and lowest consistency (writes)
ALL	Check all nodes. Fail if any is down.	Highest consistency and lowest availability
ONE (TWO, THREE)	Check closest node to coordinator.	Highest availability and lowest consistency (reads)
QUORUM	Check quorum of available nodes.	Balanced consistency and availability

Consistency Level Trade-Off

- Consistency Level ALL
  - **Consistent Read**,

    Highest latency, Lowest availability
- Consistency Level ONE
  - Maybe inconsistent Read,

    Lowest latency, **Highest availability**
- Consistency Level QUORUM
  - Consistent Read (if both Read/Write are QUORUM), Medium latency, Medium availability

Immediate Consistency (I)

- Immediate Consistency
  - Reads always return the most recent data
- We achieve this by configuring
  - CL per Read, Write
  - RF per KeySpace
- It must hold:
- Practically, does it worth it?
  - CL ONE is enough in most cases

Immediate Consistency (II)

Configuration examples for a Cluster with 4 Nodes:

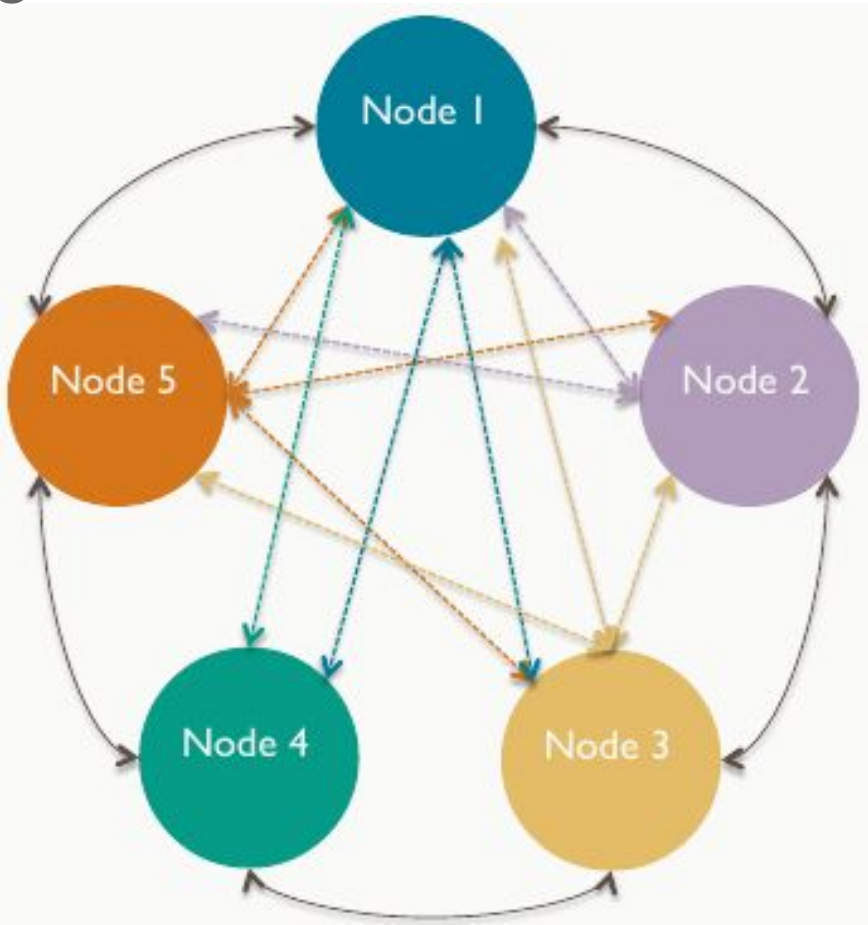
1. Frequent Read operations:
   a. RF = 3
   b. CL_{Read} = QUORUM, CL_{Write} = QUORUM
2. Frequent Write operations:
   a. RF = 3
   b. CL_{Read} = ALL, CL_{Write} = ONE

Cluster internal communication

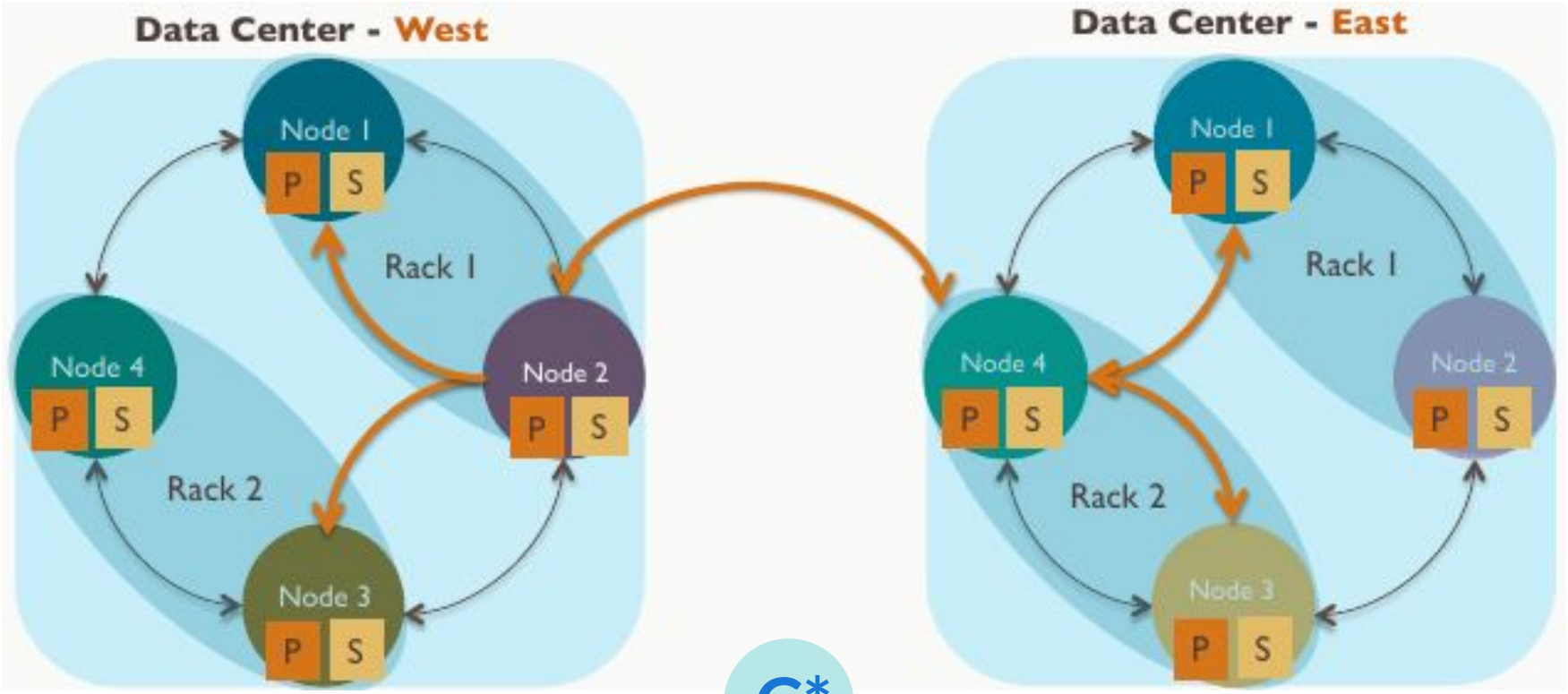
- Nodes continuously communicate and exchange information
- Two central mechanisms
  - Gossip
  - Snitch

Every one second, each Node contacts 1 to 3 others, sending and requesting timestamped updates about known Nodes

- states
- locations



This is how Nodes know about the rack and data center topology



CONFIGURATION FILES & TOOLS

- Requirements for CPU, RAM, HDD
- Operating System
- NTP – C* requires synchronized clocks
- Disable memory swaps
- Java: Oracle JDK
- Network configuration
- C* installation
- C* configuration

- Apache Cassandra
- DataStax Community Edition (DSC)
  - Additional tools for managing a Cluster
- DataStax Enterprise Edition (DSE)
  - More features than DSC, better for Analytics
  - Special program for start-ups
- www.planetcassandra.org/cassandra/

Configuration Files

- Located under **$CASSANDRA_HOME/conf/**
  - Example: dsc-cassandra-2.1.10/conf/
- Most important files:
  - cassandra.yaml
  - cassandra-env.sh
  - logback.xml
  - cassandra-rackdc.properties
  - cassandra-topology.properties

- Located under
  - **$CASSANDRA_HOME/bin/**
  - **$CASSANDRA_HOME/tools/**
- Tools
  - nodetool
  - cqlsh
  - cassandra-stress
  - sstable2json, json2sstable
  - Cassandra Cluster Management – CCM (DataStax)
  - DevCenter (DataStax)

CASSANDRA DATA MODEL

C* Data Model (I)

- Data is stored and organized in a Column Family
- A Column Family is comprised of Rows
- A Row is the smallest unit that stores related data

C* Data Model (II)

- A Partition (old name: RowKey) uniquely identifies a Row in a Column Family
- It stores data in Cells
- Cell parts
  - column name
  - column value
  - data creation timestamp
- Maximum cell size (column value)
  - 2 GB in theory
  - 100MB in praxis

C* Data Model (III)

1. A Table is a 2D view of a column family
   a. A table has Partitions
   b. A Partition may be a single row or multiple rows
2. A Partition Key uniquely identifies a Partition
   a. Can be composite
   b. It is hashed by the partitioner system to determine which Node will store it

C* Data Model (IV)

- A Primary Key uniquely identifies a row
  - Can be composite
  - It is comprised of two parts
    - the Partition Key
    - optionally, further columns

- Data Definition Language (DDL) describes Tables, Partition Keys, Primary Keys

Data in Clustering Columns (I)

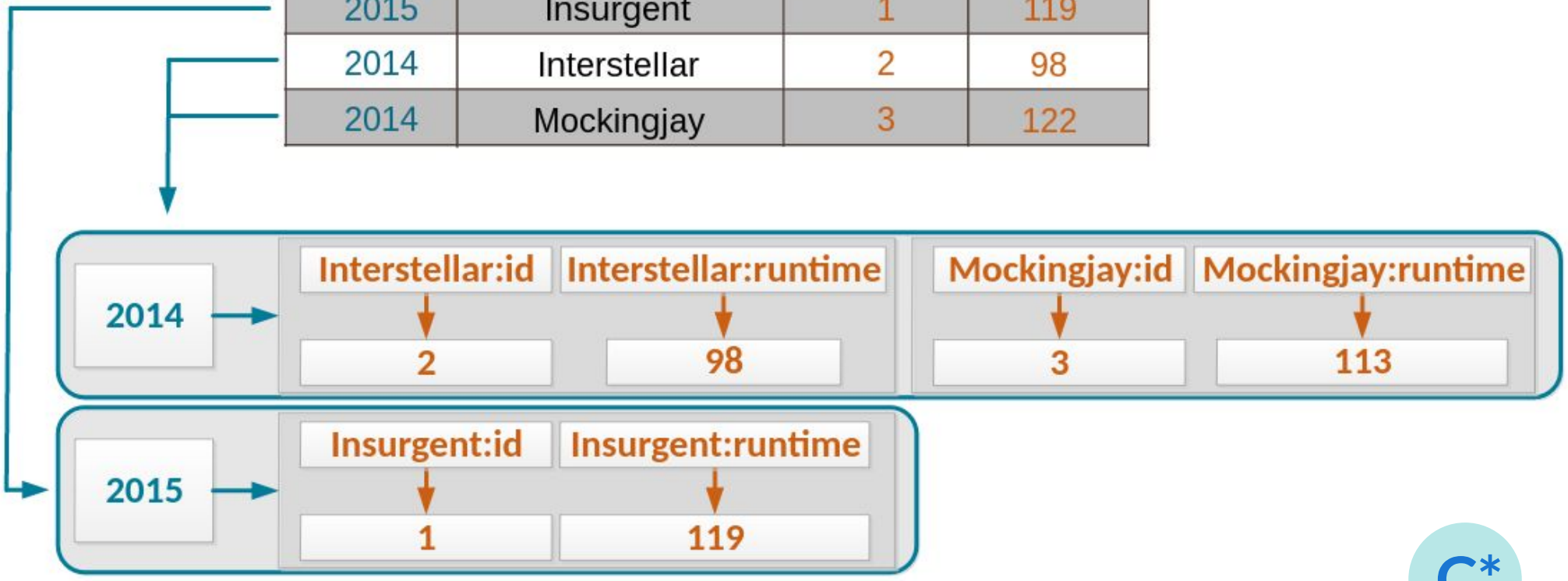
For table Videos below:

```
CREATE TABLE Videos
( id INT, name TEXT, year INT, runtime INT,
 PRIMARY KEY ((year), name)
);
```

year	name	id	runtime
2015	Insurgent	1	119
2014	Interstellar	2	98
2014	Mockingjay	3	122

Clustering columns divide Rows among partitions

year	name	id	runtime
2015	Insurgent	1	119
2014	Interstellar	2	98
2014	Mockingjay	3	122



CASSANDRA QUERY LANGUAGE

Cassandra Query Language

- Language for communicating with the C* DB
- Abbreviated as CQL
- Similar to SQL
- It can create, modify, delete tables and data

CQL Basic Data Types

CQL Type	Constants	Description
ASCII	strings	US-ASCII character string
BIGINT	integers	64-bit signed long
BLOB	blobs	Arbitrary bytes (no validation), expressed as hexadecimal
BOOLEAN	booleans	true or false
COUNTER	integers	Distributed counter value (64-bit long)
DECIMAL	integers, floats	Variable-precision decimal
DOUBLE	integers	64-bit IEEE-754 floating point
FLOAT	integers, floats	32-bit IEEE-754 floating point
INET	strings	IP address string in IPv4 or IPv6 format*
INT	integers	32-bit signed integer
LIST	n/a	A collection of one or more ordered elements
MAP	n/a	A JSON-style array of literals: { literal : literal, literal : literal ... }
SET	n/a	A collection of one or more elements
TEXT	strings	UTF-8 encoded string
TIMESTAMP	integers, strings	Date plus time, encoded as 8 bytes since epoch
TUPLE	n/a	Up to 32k fields
UUID	uuids	A UUID in standard UUID format
TIMEUUID	uuids	Type 1 UUID only (CQL 3)
VARCHAR	strings	UTF-8 encoded string
VARINT	integers	Arbitrary-precision integer

CQL: Create Table


```
CREATE TABLE cars_by_cost (
    - brand TEXT,       // part of Partition Key
    - model TEXT,       // part of Partition Key
    - cost DECIMAL,   // Clustering Key
    - merchant TEXT,
    - PRIMARY KEY ((brand, model), cost)
) WITH CLUSTERING ORDER BY (cost ASC);
```

CQL: Modify Table

- ALTER TABLE cars_by_cost ADD cc INT;

- ALTER TABLE cars_by_cost

  ALTER cc TYPE BIGINT;

  - Types must be compatible

- ALTER TABLE cars_by_cost DROP cc;

CQL: Remove or Empty Table

- To fully remove a Table:
- DROP TABLE cars_by_cost;

- To clear all data (delete all Partitions) from a Table – but spare the Table:
- TRUNCATE cars_by_cost;

CQL: Read Data (I)

- General syntax:

- **SELECT columns**
  FROM table
  WHERE relations
  ORDER BY clustering_column ASC/DESC
  LIMIT number
  ALLOW FILTERING;

CQL: Read Data (II)

- Typical cases
  - Beware, these examples include Anti-Patterns!

- SELECT brand, merchant FROM cars_by_cost;
  - Avoid retrieving all partitions and rows unless absolutely necessary

- SELECT * FROM cars_by_cost;
  - Avoid retrieving all columns unless necessary

CQL: Read Data (III)

- SELECT * FROM cars_by_cost
  WHERE brand = 'b' AND model = 'm';
  - To retrieve a partition, values for **all** partition columns are needed

- SELECT * FROM cars_by_cost
  WHERE brand = 'b' AND
  model = 'm' AND cost < 1000;
  - To retrieve a row, values for **all** partition and clustering columns (primary key) are needed

- Trick of ALLOW FILTERING
  - Allows scanning over all partitions and the predicate needs not give values for all partition columns
  - May lead to slow queries with large result set

CQL: Indexing to Read Data

- Used to allow queries on normal columns
- Two types of Indexes
  - Secondary Indexes
    - traditional
  - Custom Indexes: SASI
    - SSTable Attached Secondary Index
    - Since C* v3.4
- Indexes usage is NOT a spontaneous decision, but a well thought one

CQL: Secondary Indexes

For exact matches

- **CREATE INDEX** merchant_idx
  ON cars_by_cost (merchant);

  - SELECT * FROM cars_by_cost
    WHERE **merchant = 'm'**;
  - SELECT * FROM cars_by_cost
    WHERE brand = 'b' AND **merchant = 'm'**;

- **DROP INDEX** merchant_idx;

CQL: Custom/SASI Indexes (I)

For partial matches:

```
CREATE CUSTOM INDEX merchant_idx
ON cars_by_cost (merchant)
USING 'org.apache.cassandra.index.sasi.SASIIndex'
WITH OPTIONS = {
   'mode': 'CONTAINS',
   'analyzer_class': 'org.apache.cassandra.index.sasi
                      .analyzer.NonTokenizingAnalyzer',
   'case_sensitive': 'false'
};
```

CQL: Custom/SASI Indexes (II)

Query using a SASI Index:

- **SELECT** *

  FROM cars_by_cost

  WHERE merchant **LIKE** '%son%' ;

CQL: Additional Functions

- Aggregation related
  - count(), min(), max(), sum(), avg(), ...
- Time related
  - now(), dateof(), …
- Blob conversion related
  - bigintAsBlob, blobAsBigint, …
- User Defined Functions are also possible!
  - To be executed within C*, thus written in Java

CQL: Create (Insert) Data

- INSERT INTO

  cars_by_cost (brand, model, cost, merchant)

  VALUES ('volvo', 'xc90', 9999, 'daves');

- What does it do?
  - Creates non-existing partitions
  - **But also updates existing partitions**

CQL: Update Data

- UPDATE cars_by_cost

  SET merchant = 'pauls'

  WHERE brand = 'volvo'

  AND model = 'xc90'

  AND cost = 9999;
- What does it do?
  - Updates existing partitions
  - **But also creates non-existing partitions**

- Insert and Update have the notion of Upsert
  - Update or Insert
- Why?
  - Because of the way data is organized into Clustering columns

CQL: Delete Data (I)

- Deleting a Partition
- DELETE FROM cars_by_cost

  WHERE brand = 'b' AND model = 'm';

- Deleting a Row
- DELETE FROM cars_by_cost

  WHERE brand = 'b'

  AND model = 'm'

  AND cost = 1000;

CQL: Delete Data (II)

- Deleting (setting to NULL) a cell from a Row
- DELETE merchant FROM cars_by_cost

  WHERE brand = 'b'

  AND model = 'm'

  AND cost = 1000;
- To clear all data (delete all Partitions) from a Table – but spare the Table:
- TRUNCATE cars_by_cost;

CQL: Time-To-Live (TTL) (I)

- Used in INSERT and UPDATE commands
- Column values in commands with TTL are automatically marked as deleted after the specified amount of time has expired
- Any subsequent update of the column resets the TTL to the new value specified in the update

CQL: Time-To-Live (TTL) (II)

- Expressed in seconds

- INSERT INTO

  cars_by_cost (brand, model, cost, merchant)

  VALUES ('volvo', 'xc90', 9999, 'daves')

  USING TTL 86400;

ACID & TRANSACTIONS

To Be Presented...

… in a next Meetup :)

It is one of the most interesting “chapters”

DATA MODELING FOR CASSANDRA DB

To Be Presented...

… in a next Meetup :)

And it is what developers need most

MORE ABOUT CASSANDRA DB

Get Started (I)

- Apache C* - http://cassandra.apache.org/
- DataStax C* - http://www.datastax.com/
- DataStax C* Drivers and Tools

  https://academy.datastax.com/downloads/welcome
- DataStax and Apache C* Drivers -
  - DataStax
  - Apache

Get Started (II)

- DataStax and Apache CQL Documentation
  - http://docs.datastax.com/en/cql/3.3/cql/cqlIntro.html
  - http://cassandra.apache.org/doc/latest/cql/

- **DataStax Startup Program**

  http://www.datastax.com/datastax-enterprise-for-startups

- DataStax Community Service
  - http://www.planetcassandra.org/
- DataStax Cassandra Academy
  - https://academy.datastax.com/
- Stackoverflow Tags:
  - cassandra, datastax, datastax-enterprise
- Books
  - visit Amazon

Get Involved

- Meetup Groups

  http://www.meetup.com/

- Join a DataStax Group:

  https://www.meetup.com/Athens-Cassandra-Users/

References – Sources (I)

These (already mentioned) web pages:

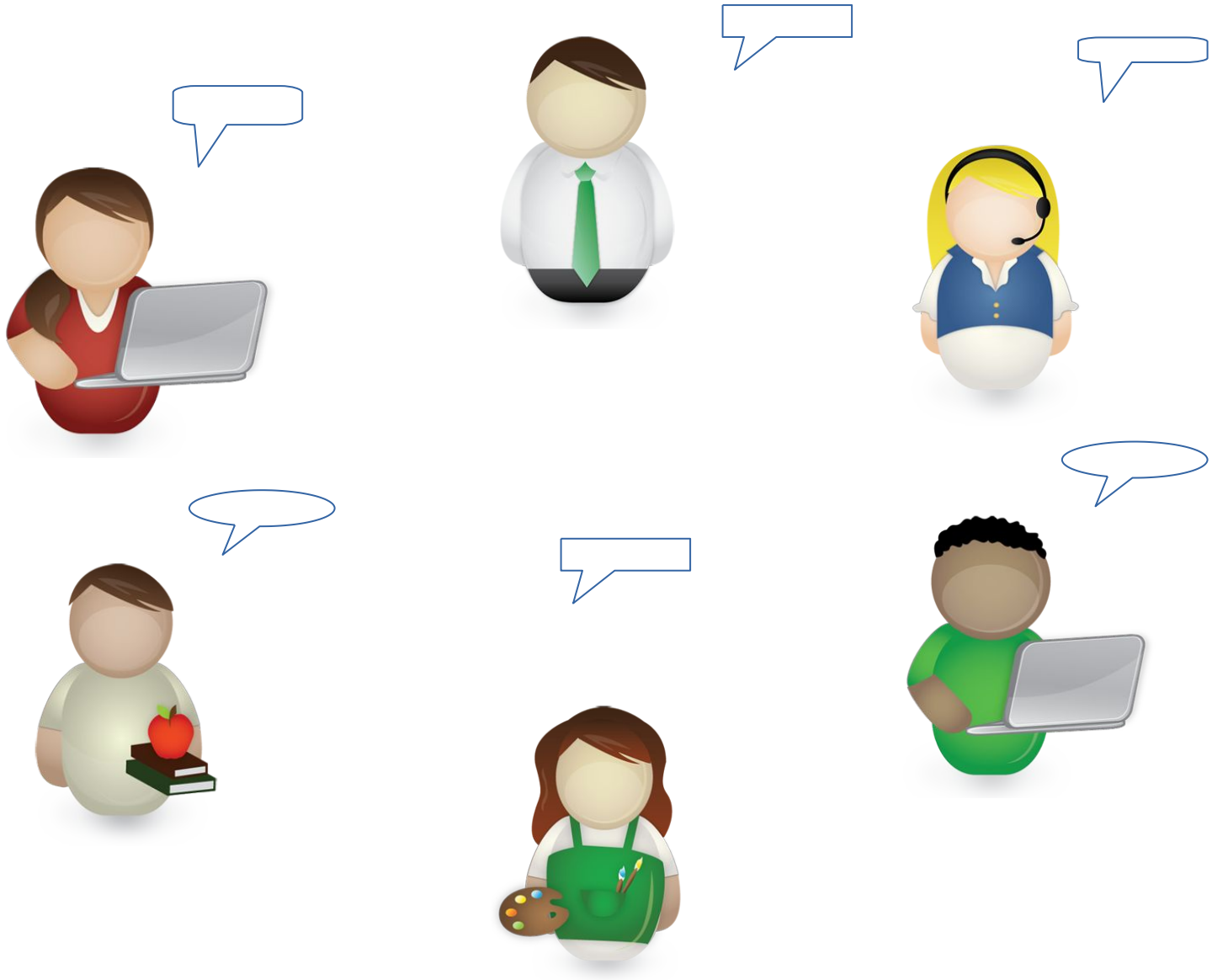
- http://cassandra.apache.org/
- http://www.datastax.com/
- http://docs.datastax.com/en/cql/3.3/cql/cqlIntro.html
- http://cassandra.apache.org/doc/latest/cql/
- https://academy.datastax.com/
- http://www.planetcassandra.org/

- These additional articles and web pages:
  - Leslie Lamport, "Time, clocks, and the ordering of events in a distributed system", http://research.microsoft.com/en-us/um/people/lamport/pubs/time-clocks.pdf
  - Mark Burgess, "Deconstructing the 'CAP theorem' for CM and DevOps", http://markburgess.org/blog_cap.html

References - Acknowledgments

Tables and Graphics in slides marked with
are copyrighted work of DATASTAX
and used here after their permission.
We really thank them for that.

Let's Talk!



THANK YOU !