Introducing big data and Cassandra

Apache Cassandra:
Core Concepts, Skills, and Tools

Leo Schuman, Joe Chu

Learning Objectives

- **Understand big data**
- Describe Cassandra's history
- Understand common use cases
- Survey the Cassandra architecture

What are big data systems?

DATASTAX

- Applications involving the "three V's"
  - Volume: gigabytes, growing to terabytes and beyond
  - Velocity: sensor data, click streams, financial transactions
  - Variety: data must be ingested from many different formats
- Characteristics requiring
  - multi-region availability
  - very fast and reliable response
  - no single point of failure

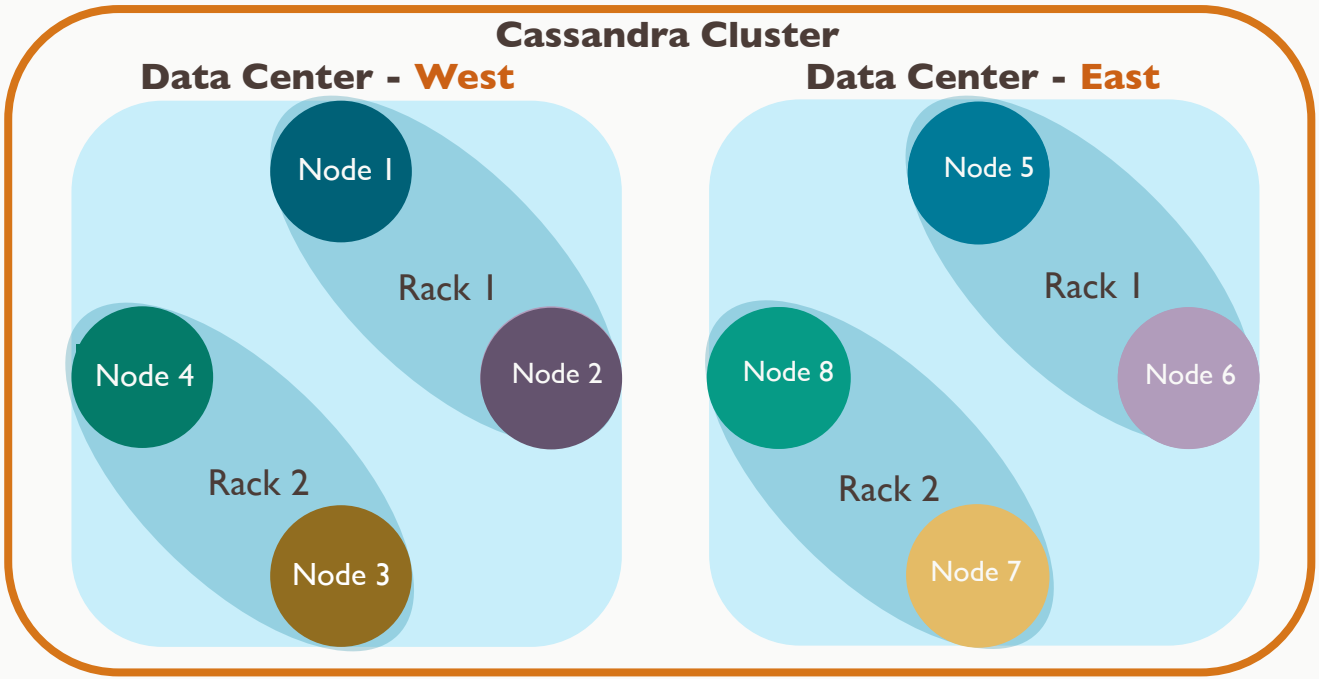
Why not relational data?

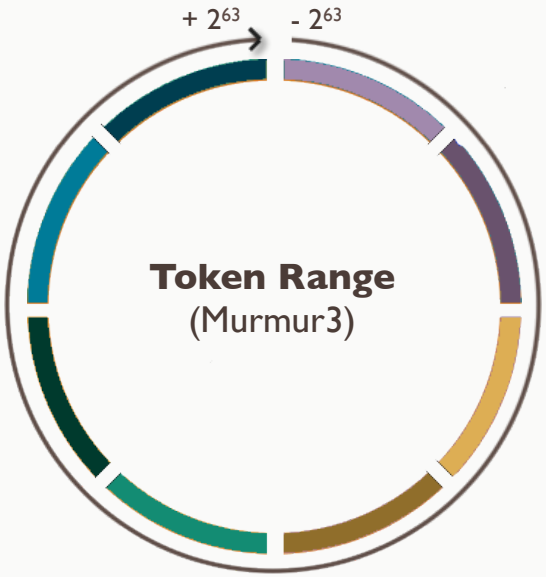
DATASTAX

- Relational model provides
  - Normalized table schema
  - Cross table joins
  - ACID compliance
- But, at very high cost
  - Big data table joins – *billions* of rows, or more – require massive overhead
  - Sharding tables across systems is complex and fragile
- Modern applications have different priorities
  - Needs for speed and availability trump "always on" consistency
  - Commodity server racks trump massive high-end systems
  - Real world need for transactional guarantees is limited

What strategies help manage big data?

- Distribute data across nodes
- Relax consistency requirements
- Relax schema requirements
- Optimize data to suit actual needs





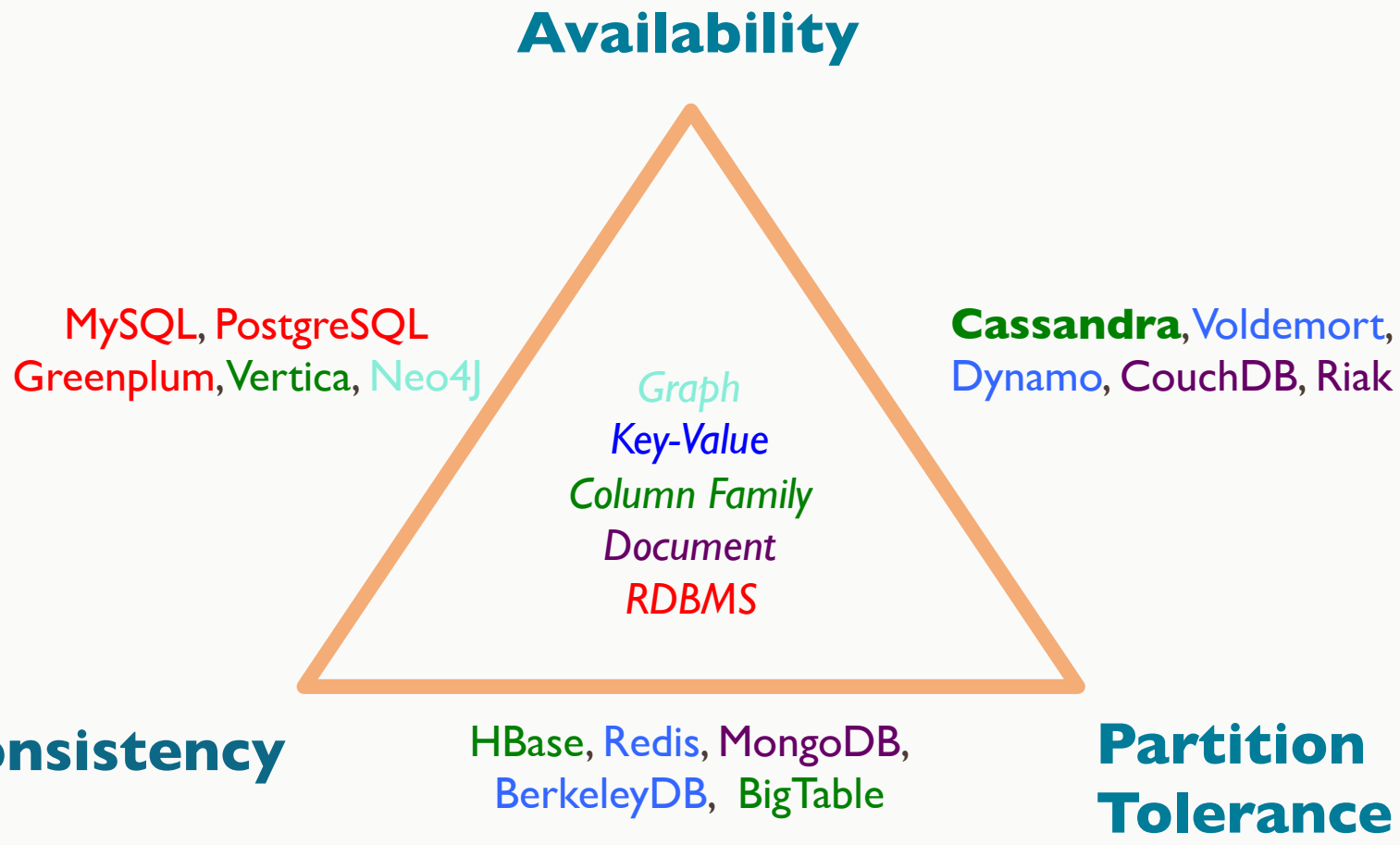
What is the NoSQL landscape?

DATASTAX

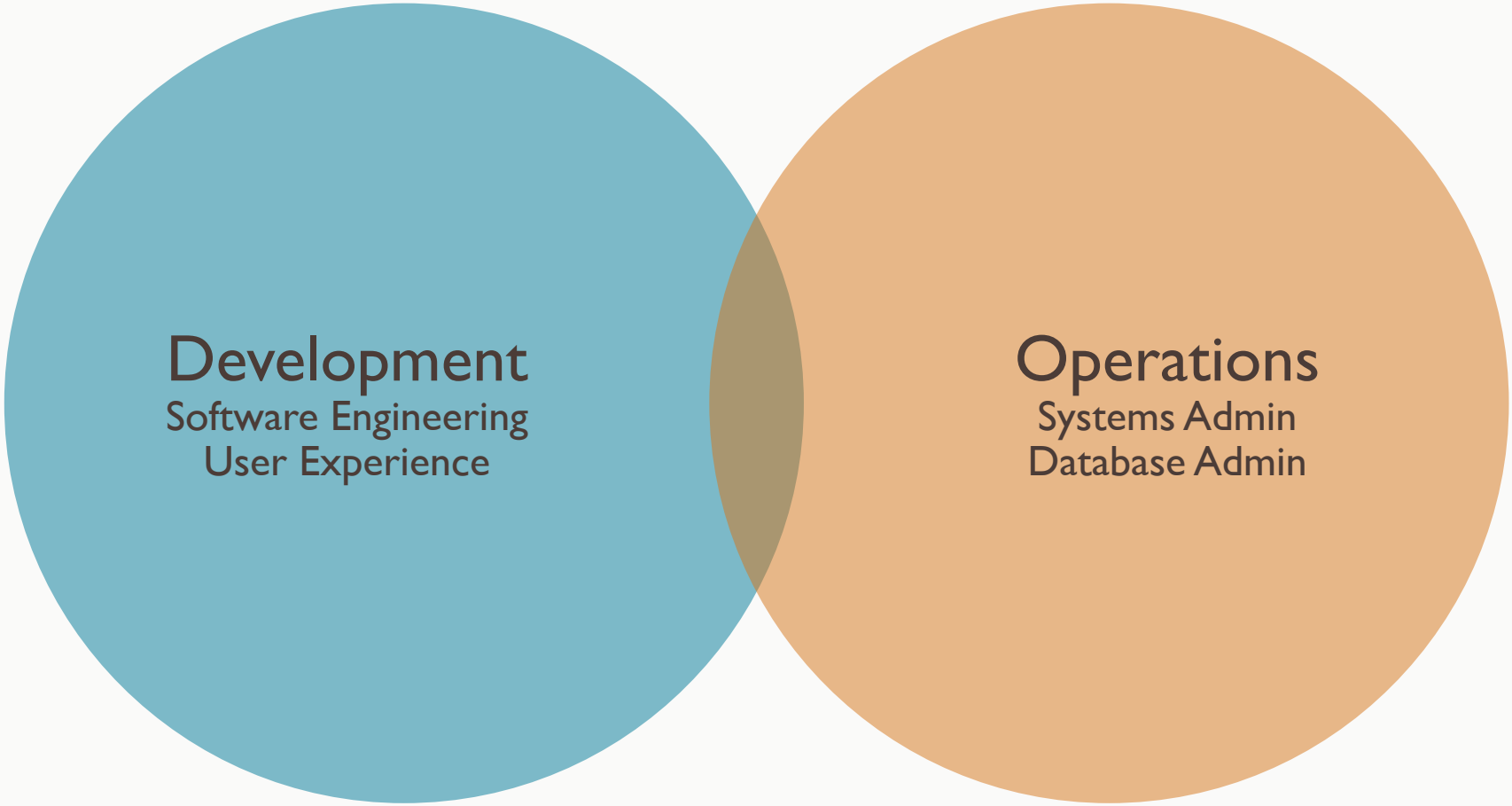
- Four broad classes of non-relational database
  - Graph: data elements each relate to *n* others in a graph/network
  - Key-Value: keys map to arbitrary values of any data type
  - Document: document sets (JSON) queryable in whole or part
  - Column Family: keys mapped to sets of *n*-number of typed columns
- Three key factors help navigate the landscape
  - Consistency: do you get identical results, regardless which node is queried?
  - Availability: can the cluster respond to *very* high write and read volumes?
  - Partition Tolerance: is the cluster still available when part of it goes dark?

What is the CAP Theorem?

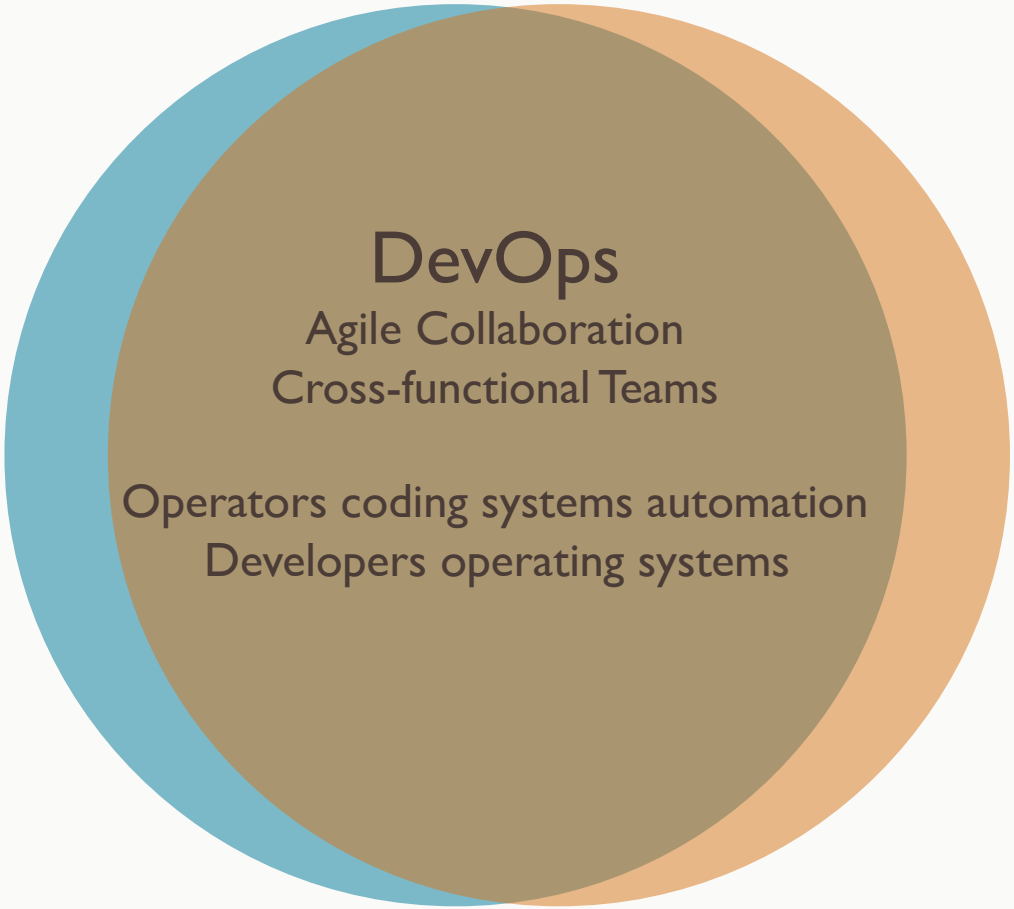
- In distributed systems, *consistency*, *availability*, *and partition tolerance* exist in a mutually dependent relationship. Pick any two.



What does "DevOps" mean?



What does "DevOps" mean?

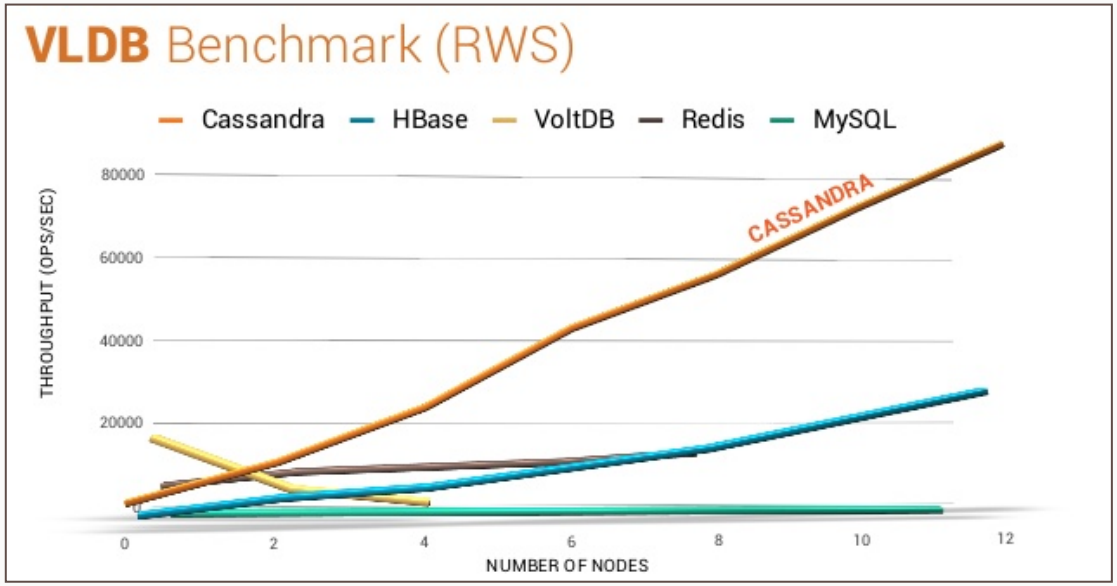


Learning Objectives

- Understand big data
- **Describe Cassandra**
- Understand common use cases
- Survey the Cassandra architecture

What is Cassandra?

- Massively linearly scalable NoSQL database
  - Fully distributed, with no single point of failure
  - Free and open source, with deep developer support
  - Highly performant, with near-linear *horizontal* scaling in proper use cases



What is Cassandra?

- No single point of failure, due to horizontal scaling
  - *horizontal scaling*: add commodity hardware to a cluster
  - *vertical scaling*: add RAM and CPUs to a specialized high performance box

How has Cassandra evolved?

DATASTAX

- Core technologies
  - Google BigTable: foundation of the storage model
  - Amazon Dynamo: foundation of the distribution backbone
  - Facebook: integrated BigTable and Dynamo, then released as Cassandra
- Rapid evolution
  - 0.6 – April 2010
  - 0.7 – January 2011
  - 0.8 – June 2011
  - 1.0 – October 2011
  - 1.1 – April 2012
  - 1.2 – January 2013
  - 2.0 – September 2013
  - 2.1 – September 2014
- Top-level Apache project since 2010

How does Cassandra model data?

DATASTAX

- Cassandra Query Language (CQL)
  - Provides a familiar, row-column, SQL-like approach
    - CREATE, ALTER, DROP
    - SELECT, INSERT, UPDATE, DELETE
  - Replaced the complex, storage-oriented Thrift API used in prior versions
  - Provides clear schema definitions in a flexible (NoSQL) schema context

```
CREATE TABLE Performer (
  name VARCHAR,
  type VARCHAR,
  country VARCHAR,
  style VARCHAR,
  born INT,
  died INT,
  PRIMARY KEY (name)
);
```

Learning Objectives

- Understand big data
- Describe Cassandra's history
- **Understand common use cases**
- Survey the Cassandra architecture

When is Cassandra the best solution?

DATASTAX

- Cassandra excels when you need
  - No single point of failure
  - Real-time writes with live operational data analysis
  - Flexible, easily altered data models
  - Near-linear horizontal scaling across commodity servers
  - Reliable replication across distributed data centers
  - Clearly defined table schema in a NoSQL environment

When is Cassandra not the best solution?

DATASTAX

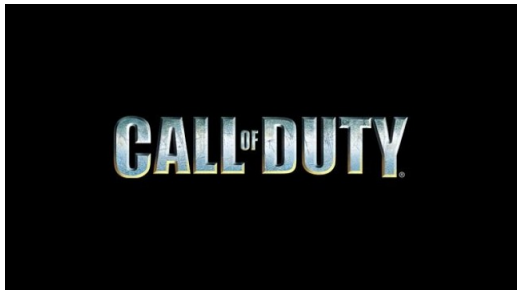
- Traditional RDBMS excels when you need
  - ACID-compliant transactions, with rollback (e.g., bank transfers)
  - Justification for high-end hardware

What are common Cassandra use cases?

DATASTAX

- Cassandra is particularly useful for
  - Playlists and collections (such as Spotify)
  - Personalization and recommendation engines (such as Ebay)
  - Messaging (such as Instagram)
  - Fraud detection (such as Barracuda)
  - Sensor data (such as Zonar)
- Many, many functional and industry use cases available
  - http://planetcassandra.org/functional-use-cases/

Who is using Cassandra?



for more, see PlanetCassandra.org/companies/

Summary

- "Big data" requires very high availability, tolerance, and response
- Manage with relaxed consistency and optimization to actual need
- There are four categories of NoSQL systems: graph, key-value, document, and column family
- Cassandra is a column-family system, offering near-linear scaling across commodity clusters
- Developer and Operator roles are merging into "DevOps"
- Cassandra has its roots in Amazon Dynamo and Google BigTable technology
- Data is modeled using *Cassandra Query Language (CQL)*
- Cassandra excels with flexible, real-time data ingestion and analysis with no single point of failure across commodity hardware clusters

Review Questions

- What do consistency, availability, and partition tolerance mean?
- Where does Cassandra fit within the CAP theorem?
- What are the technological roots of Cassandra?
- What technology does Cassandra use to model data?
- What use cases are great fits for Cassandra?