SPARK AND COUCHBASE

AUGMENTING THE OPERATIONAL DATABASE WITH SPARK

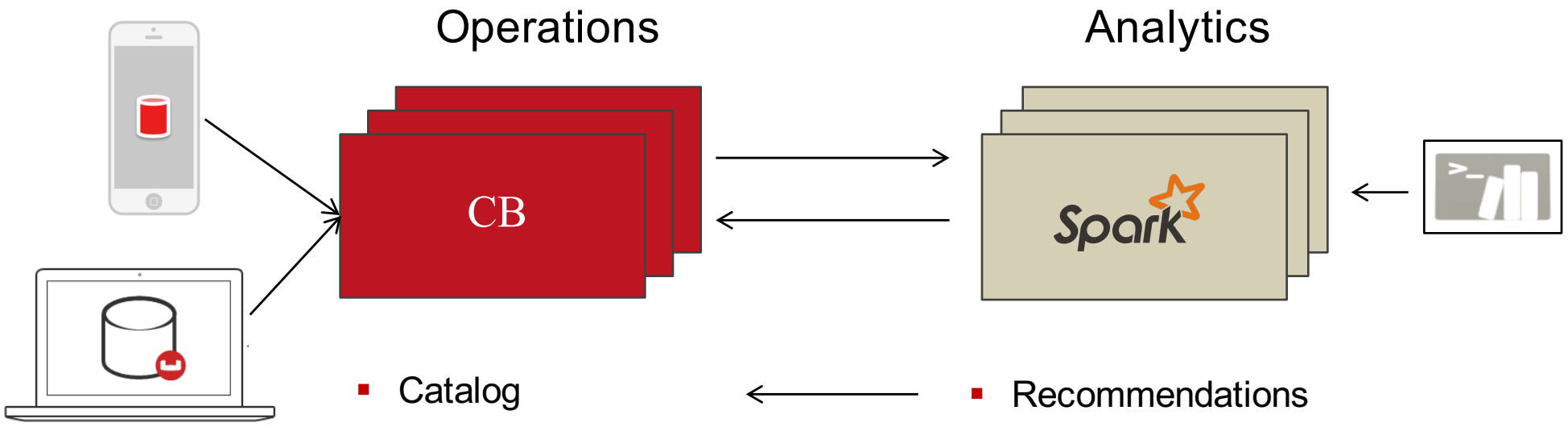
Michael Nitschinger
Couchbase

Couchbase

Overview & Use-Cases

WHY SPARK AND COUCHBASE

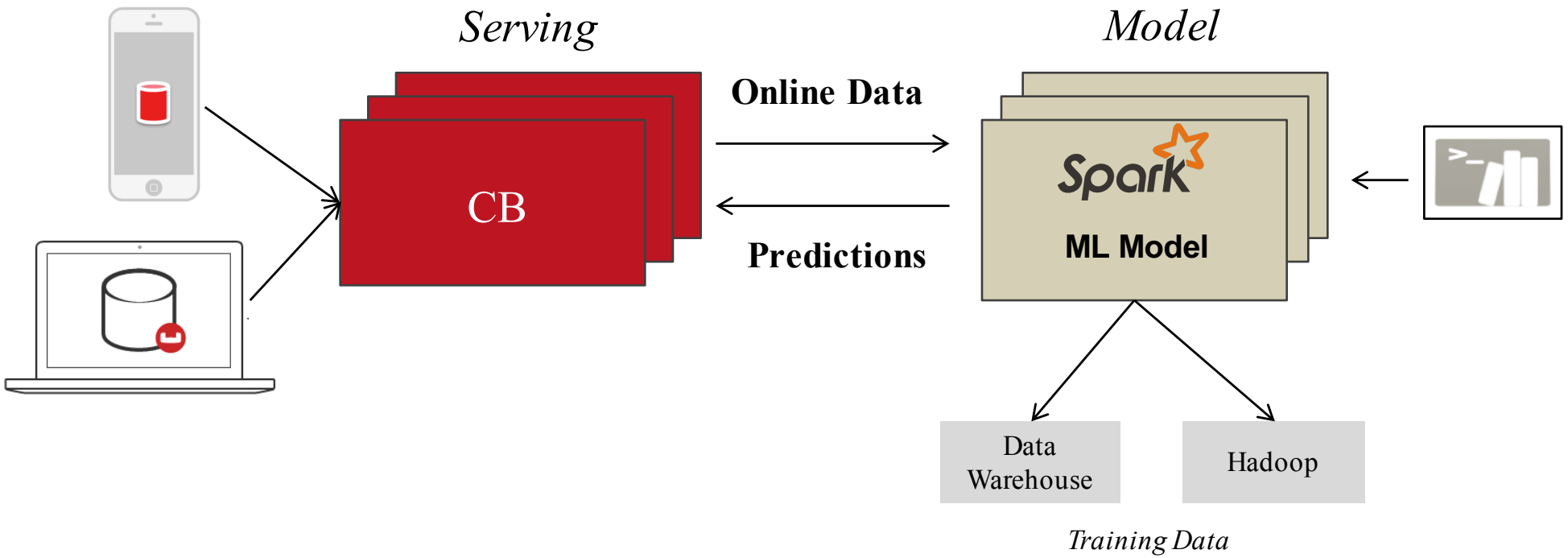
Use Cases

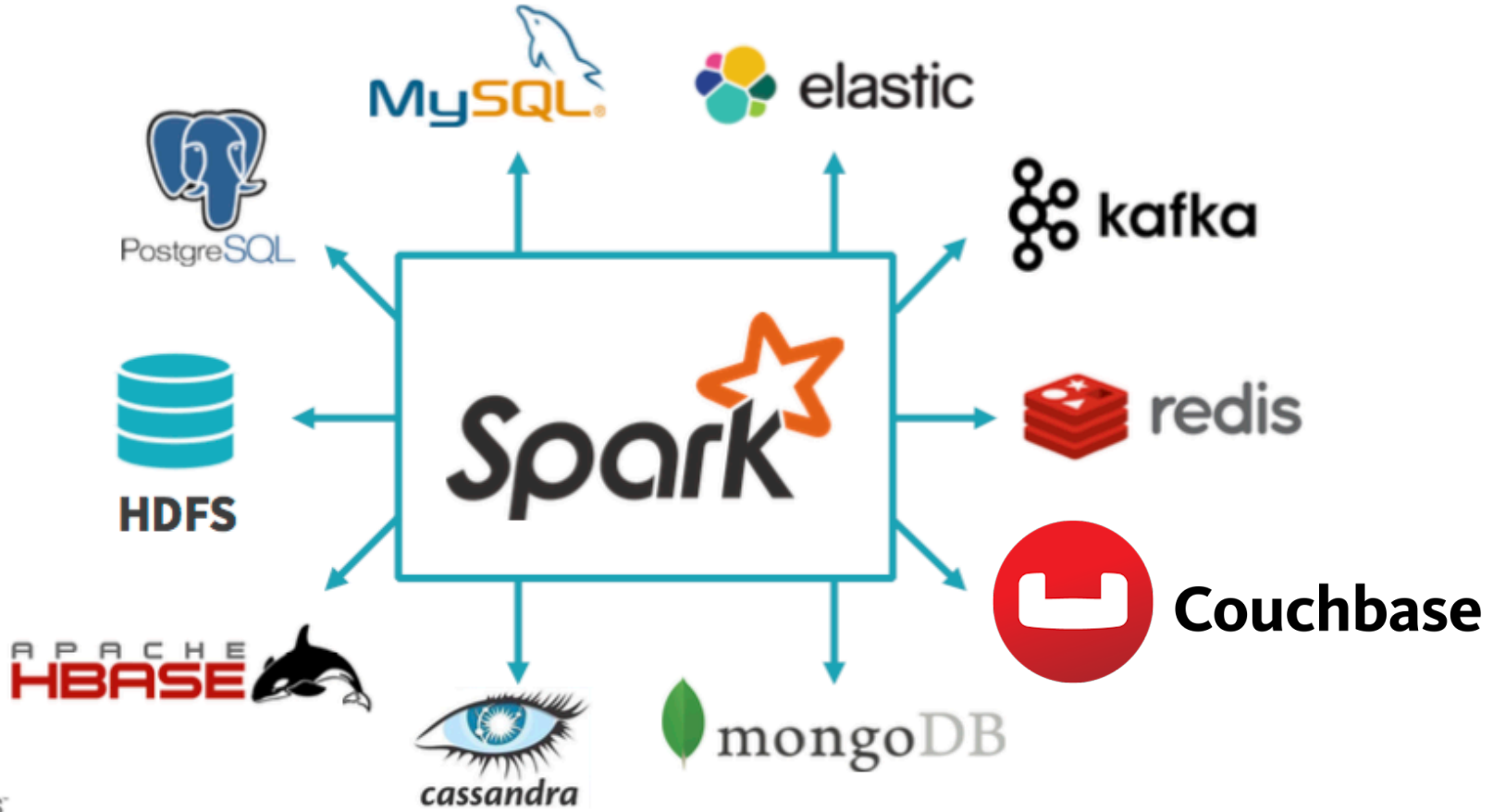


- Catalog
- Personalization
- Mobile applications

- Recommendations
- Predictive analytics
- Fraud detection

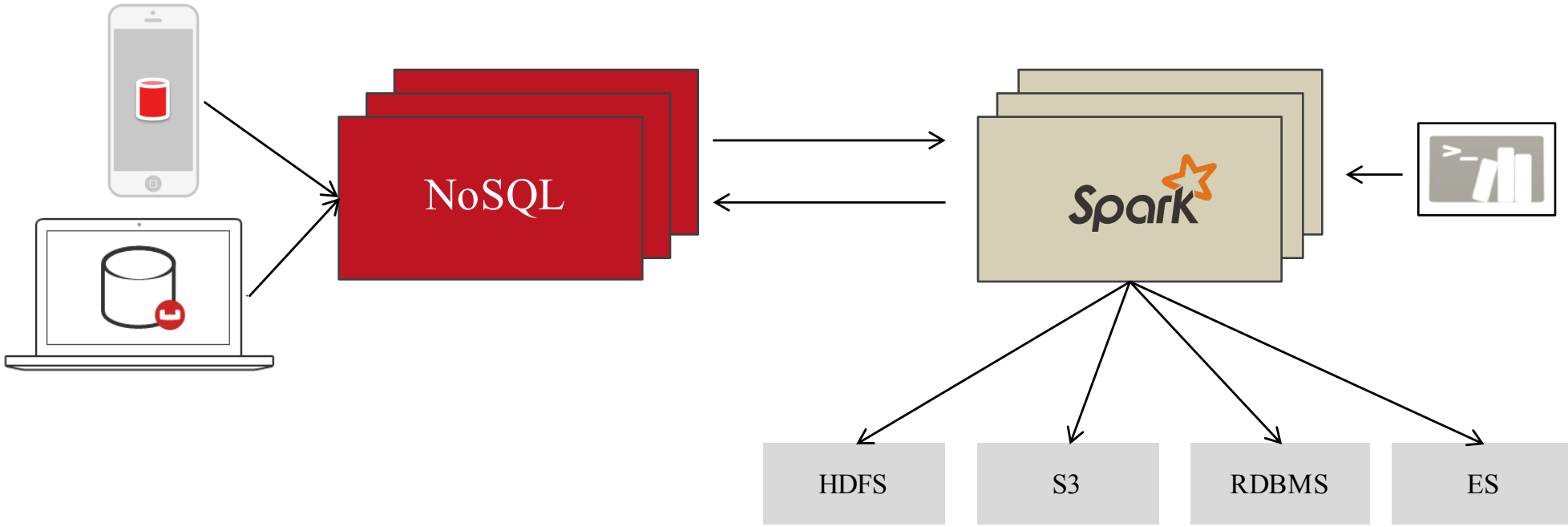
Use Case: Operationalize Analytics / ML



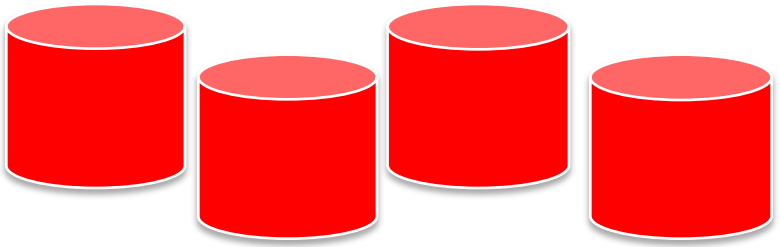


Adapted from: Databricks – Not Your Father's Database https://www.brighttalk.com/webcast/12891/196891

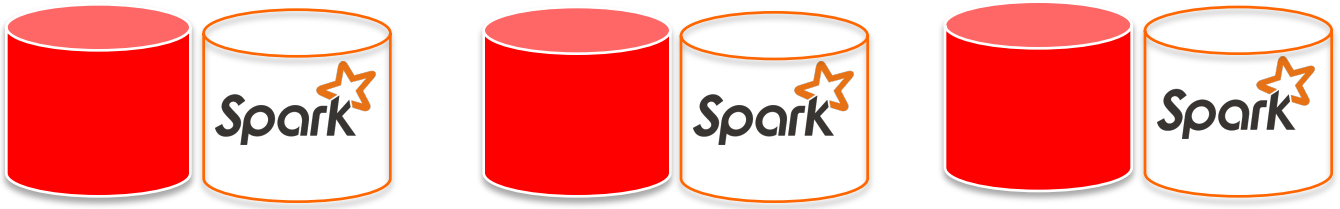
Use Case: Data Integration



Standalone Deployment



Side-By-Side Deployment



From Spark to Couchbase and Back Again

ACCESS PATTERNS

Key-Value

Fetch/Store by Document ID

Key-Value

Fetch/Store by Document ID

N1QL Query

Fetch by Criteria “SQL”

Key-Value

Fetch/Store by Document ID

N1QL Query

Fetch by Criteria “SQL”

Map-Reduce Views

Materialized Indexes (Aggregation)

Key-Value

Fetch/Store by Document ID

N1QL Query

Fetch by Criteria “SQL”

Map-Reduce Views

Materialized Indexes (Aggregation)

Streaming

Mutation Streams For Processing

Key-Value

Fetch/Store by Document ID

N1QL Query

Fetch by Criteria “SQL”

Map-Reduce Views

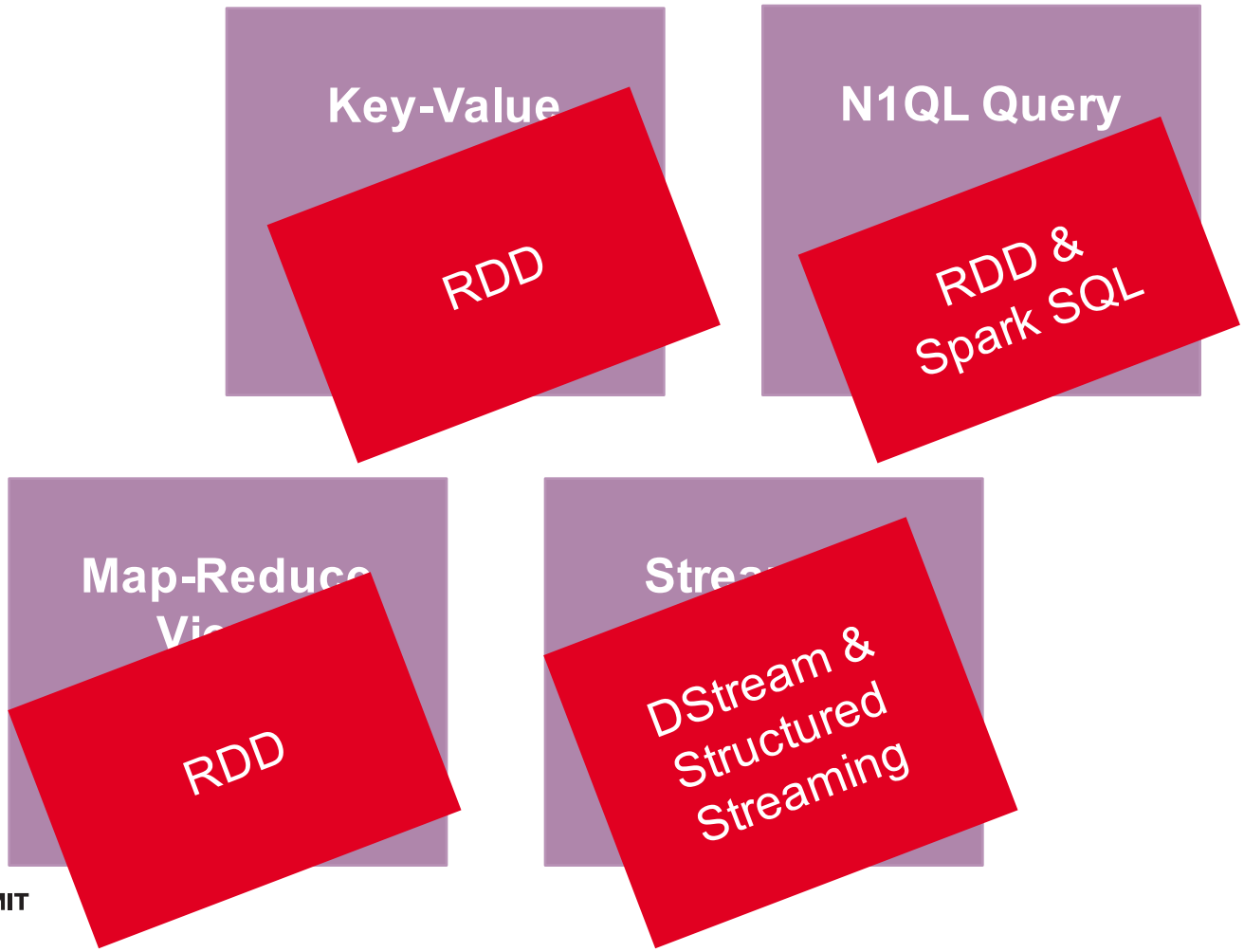
Materialized Indexes (Aggregation)

Streaming

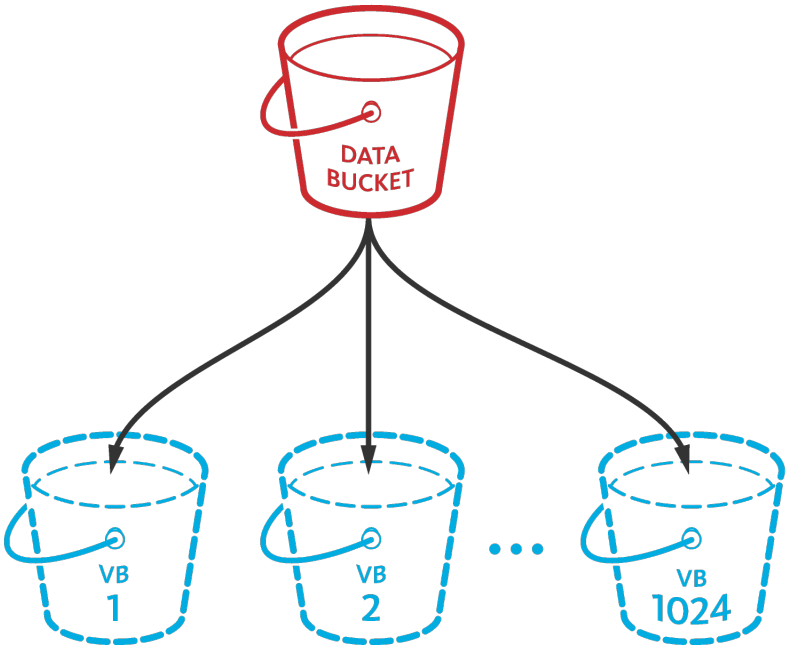
Mutation Streams For Processing

Full Text

Search on Freeform Text


Key-Value
RDD
N1QL Query
RDD &
Spark SQL
Map-Reduce
Vie
RDD
Strea
DStream &
Structured
Streaming

Couchbase Data Partitioning



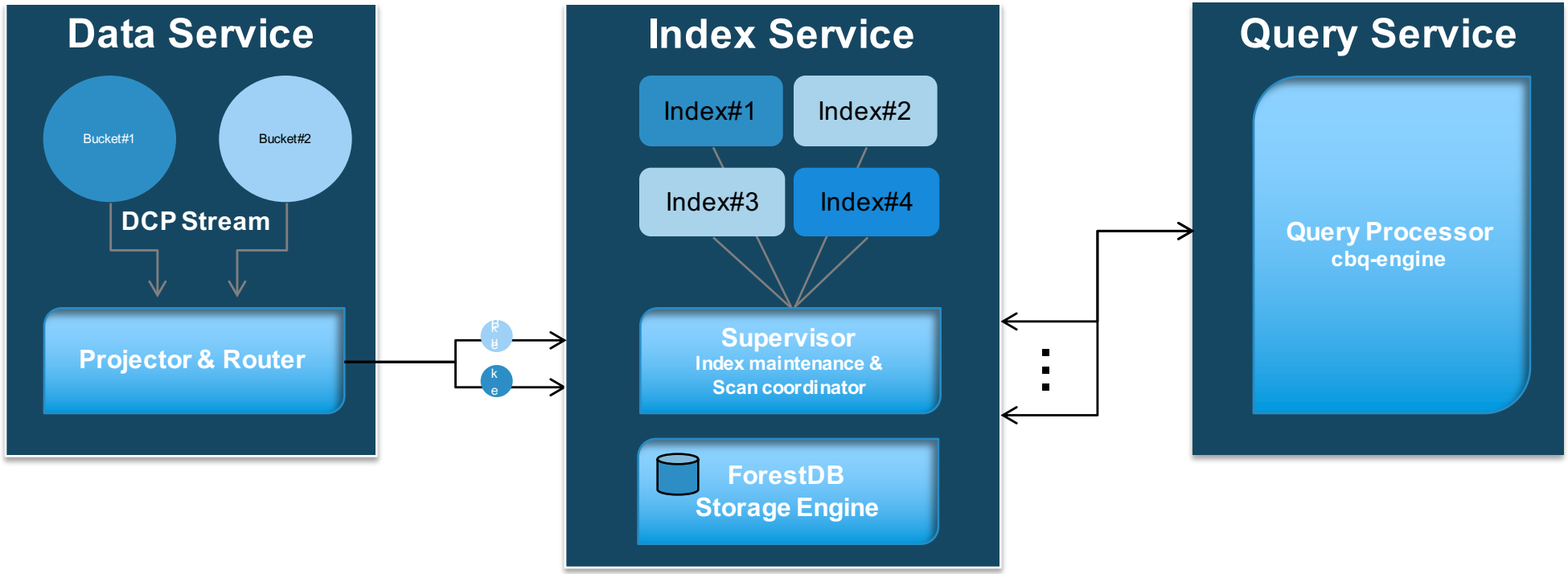
Data Locality

- RDD Location Hints based on the Cluster Map
- Not available for N1QL or Views
  - Round robin - can't give location hints
  - Back end is scatter gather with 1 node responding

N1QL Query

- N1QL is a **SQL** service with **JSON** extensions
- Uses Couchbase's Global Secondary Indexes
- Can run on any nodes within the cluster
- Nodes with differing services can be added and removed as needed on the fly

Couchbase Query Architecture



Spark SQL Sources

TableScan

Scan all of the data and return it

PrunedScan

Scan an index that matches only relevant data to the query at hand.

PrunedFilteredScan

Scan an index that matches only relevant data to the query at hand.

Predicate Conversion

```
filter match {
  case EqualTo(attr, value) => s" ${attrToFilter(attr)} = " + valueToFilter(value)
  case GreaterThan(attr, value) => s" ${attrToFilter(attr)} > " + valueToFilter(value)
  case GreaterThanOrEqual(attr, value) => s" ${attrToFilter(attr)} >= " + valueToFilter(value)
  case LessThan(attr, value) => s" ${attrToFilter(attr)} < " + valueToFilter(value)
  case LessThanOrEqual(attr, value) => s" ${attrToFilter(attr)} <= " + valueToFilter(value)
  case IsNull(attr) => s" ${attrToFilter(attr)} IS NULL"
  case IsNotNull(attr) => s" ${attrToFilter(attr)} IS NOT NULL"
  case StringContains(attr, value) => s" CONTAINS(${attrToFilter(attr)}, '$value')"
  case StringStartsWith(attr, value) =>
    s" ${attrToFilter(attr)} LIKE '" + escapeForLike(value) + "%'"
```

Schema Inference

```
val spark = SparkSession
    .builder()
    .config("spark.couchbase.nodes", "127.0.0.1")
    .config("spark.couchbase.bucket.travel-sample", "")
    .getOrCreate()

val airlines = spark.read.couchbase(EqualTo("type", "airline"))

airlines
    .select("name", "callsign")
    .sort(airlines("callsign").desc)
    .show(10)
```

Schema Inference

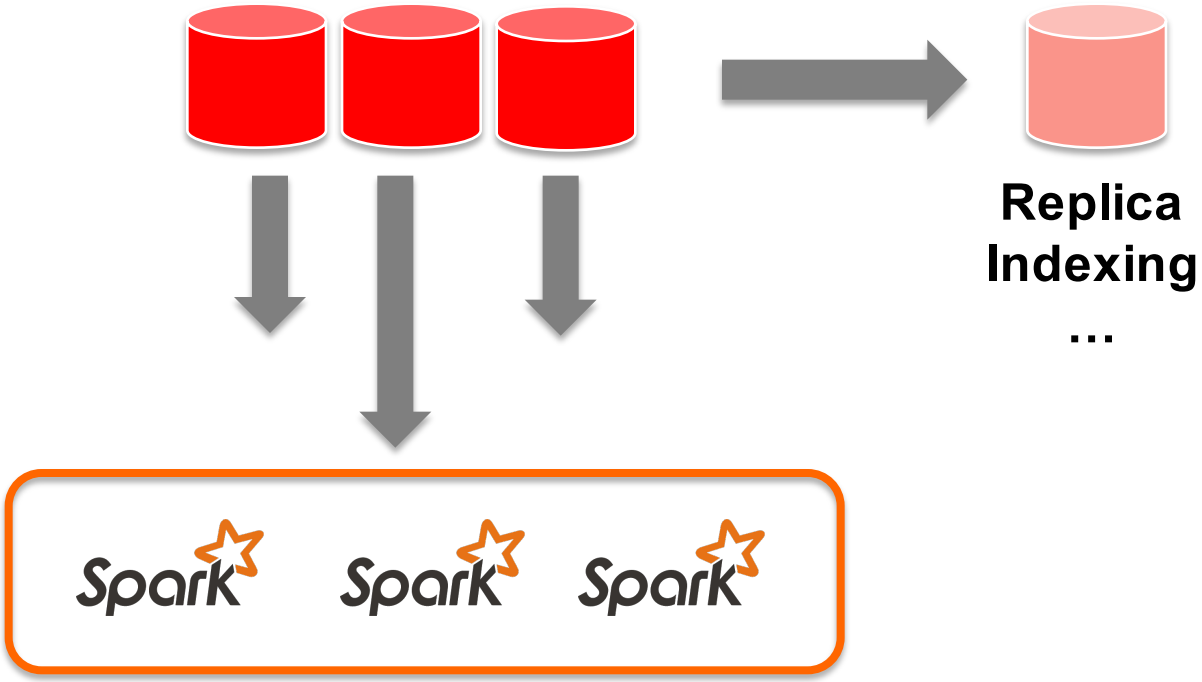
```
N1QLRelation:28 - Inferring schema from bucket travel-sample with query 'SELECT META(`travel-sample`).id as `META_ID`, `travel-sample`.* FROM `travel-sample` WHERE `type` = 'airline' LIMIT 1000'

N1QLRelation:28 - Executing generated query: 'SELECT `name`,`callsign` FROM `travel-sample` WHERE `type` = 'airline''
```

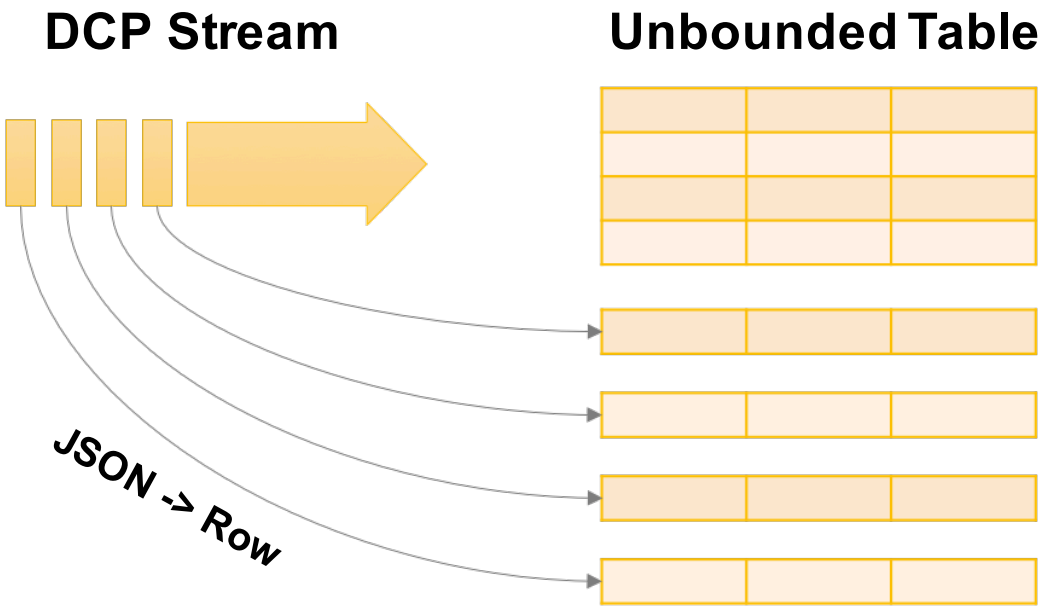
Schema Inference

```
root
 |-- META_ID: string (nullable = true)
 |-- callsign: string (nullable = true)
 |-- country: string (nullable = true)
 |-- iata: string (nullable = true)
 |-- icao: string (nullable = true)
 |-- id: long (nullable = true)
 |-- name: string (nullable = true)
 |-- type: string (nullable = true)
```

DCP and Spark Streaming



Structured Streaming Source



Adapted from https://databricks.com/blog/2016/07/28/structured-streaming-in-apache-spark.html

(Un)Structured Streaming?

```
val schema = StructType(
        StructField("META_ID", StringType) ::
        StructField("type", StringType) ::
        StructField("name", StringType) :: Nil
)
```

Structured Streaming Source

```
val records = spark.readStream
        .format("com.couchbase.spark.sql")
        .schema(schema)
        .load()
```

Structured Streaming Sink

```
val query = wordCounts.writeStream
  .outputMode("complete")
  .option("checkpointLocation", "mycheckpointlocation")
  .option("idField", "value")
  .format("com.couchbase.spark.sql")
  .start()

query.awaitTermination()
```

Couchbase Spark Connector 1.2.1

- Spark 1.6.x support, including Datasets
- DCP Flow Control
- Enhanced Java APIs

Couchbase Spark Connector 2.0.0

- Spark 2.0.x Support
- Enhanced DCP Client
- Experimental Structured Streaming

Resources

- **Spark Packages** https://spark-packages.org/package/couchbase/couchbase-spark-connector
- **Docs** http://docs.couchbase.com
- **Source** https://github.com/couchbase/couchbase-spark-connector
- **Bugs** https://issues.couchbase.com/browse/SPARKC

THANK YOU.

Michael Nitschinger
@daschl
michael.nitschinger@couchbase.com