DataStax Enterprise Analytics

Negib Marhoul
Solution Engineer

Tuesday, July 11, 2017

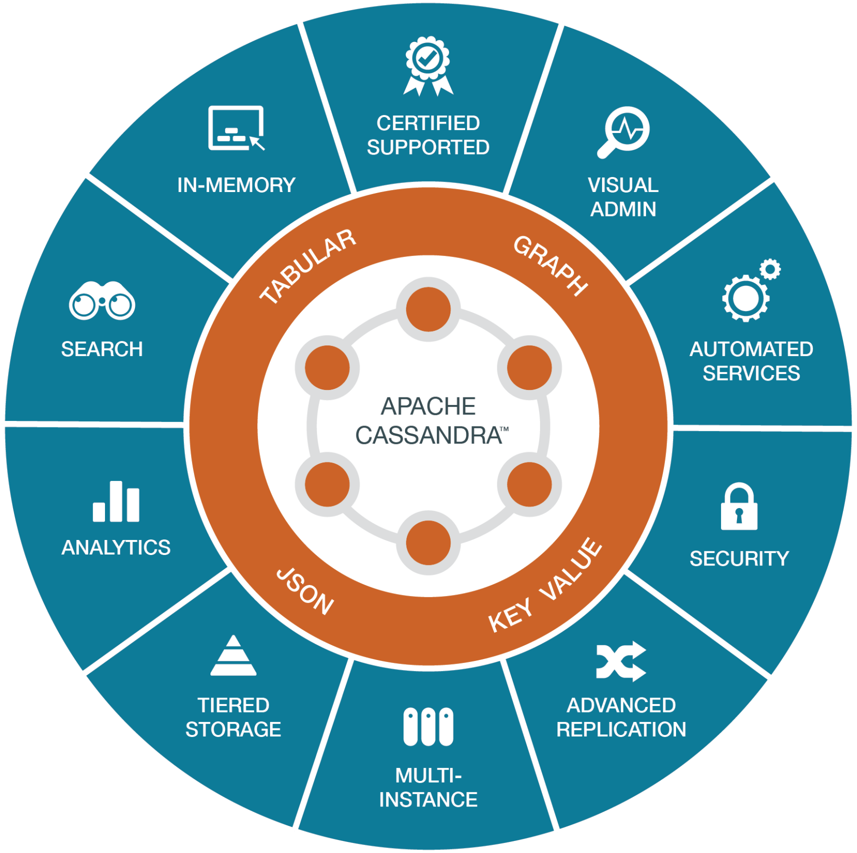
The power behind the moment.

Agenda

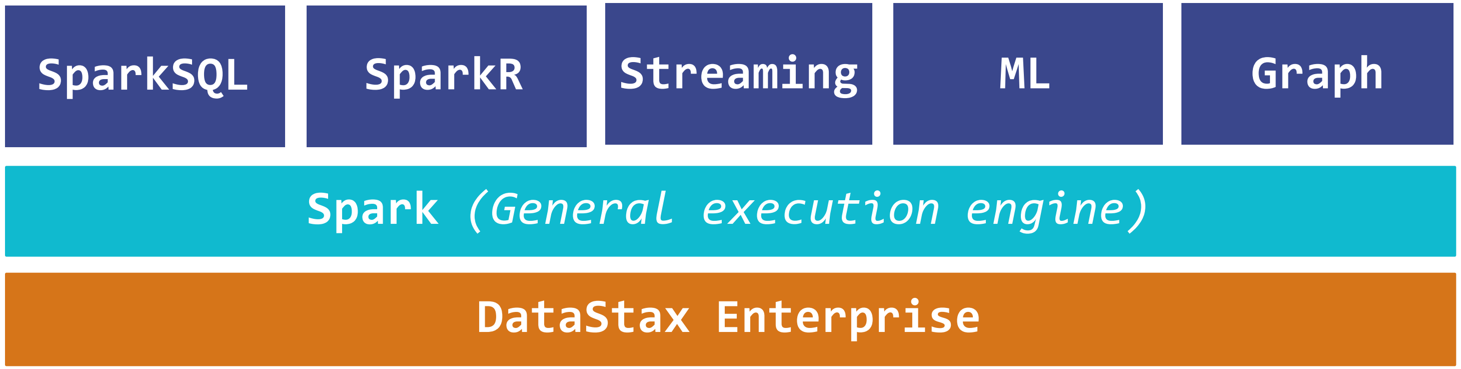
1	Introduction DSE Analytics
2	Hands-on DSE Analytics

Advanced Analytics with DATASTAX Enterprise

- DataStax Enterprise integrates real-time and batch operational analytics capabilities with an **enhanced version of Apache Spark™**
- Enables ad-hoc reporting
- Personalization
- Predictive Analytics
- Process real-time streams of data



Leveraging Apache Spark™ through DataStax Enterprise Analytics



SparkSQL

SparkR

Streaming

ML

Graph

Spark *(General execution engine)*

DataStax Enterprise

- Data model independent queries
- Cross-table operations (JOIN, UNION, etc.)
- Complex analytics (e.g. machine learning)
- Data transformation, aggregation, etc.
- Stream processing

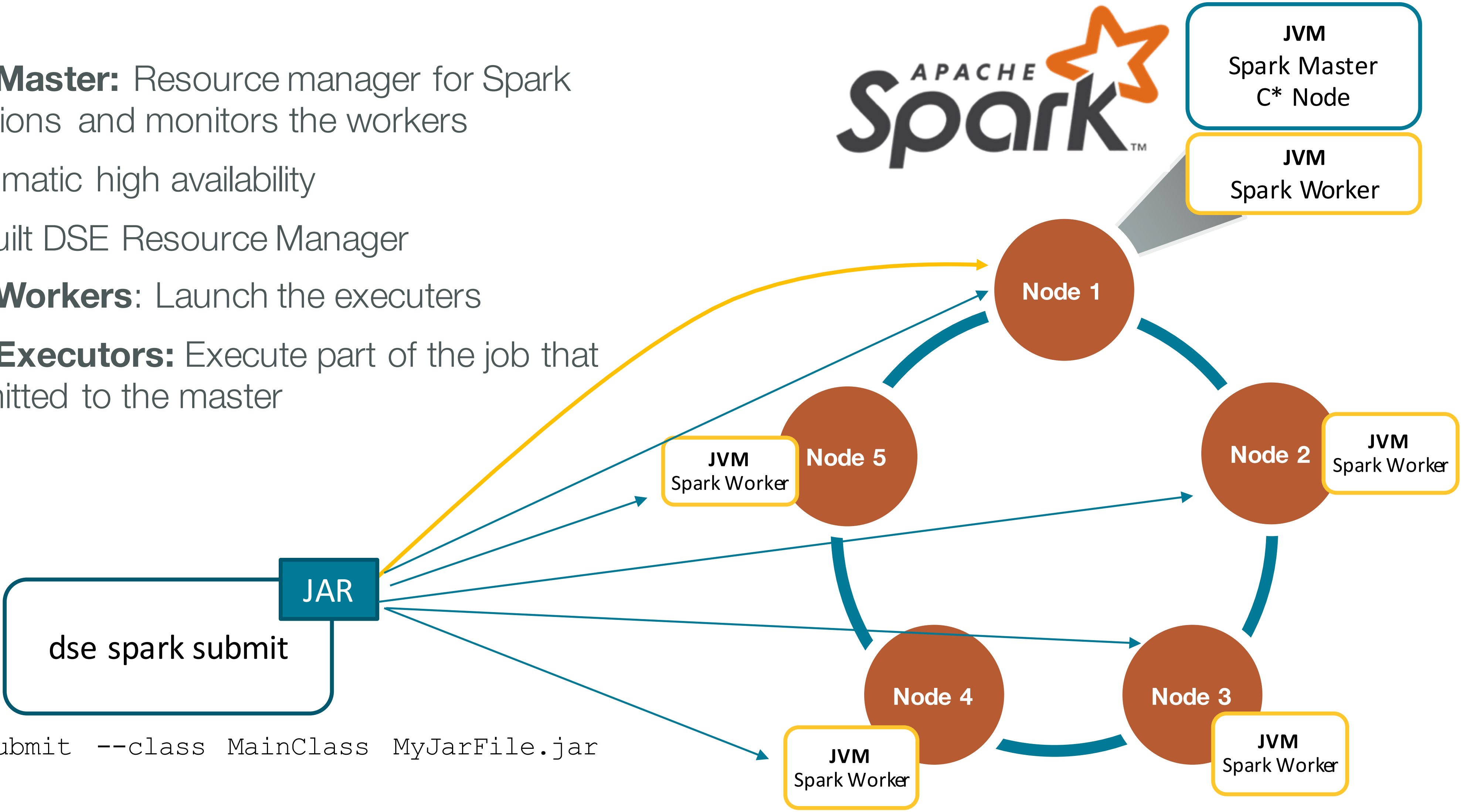
DSE Analytics Platform

Spark Master: Resource manager for Spark applications and monitors the workers

- Automatic high availability
- In-built DSE Resource Manager

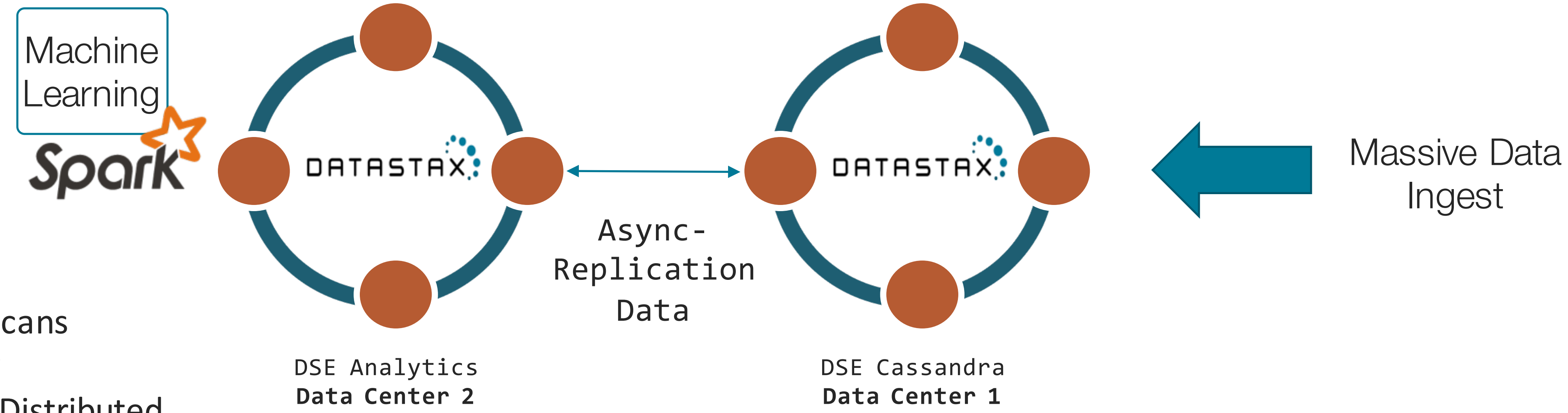
Spark Workers: Launch the executers

Spark Executors: Execute part of the job that is submitted to the master



```
dse spark-submit --class MainClass MyJarFile.jar
```

In-built Replication and Multi-Workload Separation



Sequential Scans
Data locality
Parallel and Distributed

```
CREATE KEYSPACE smart_meter WITH replication =
    {'class': 'NetworkTopologyStrategy', 'DC1': '3', 'DC2': '1'};
```

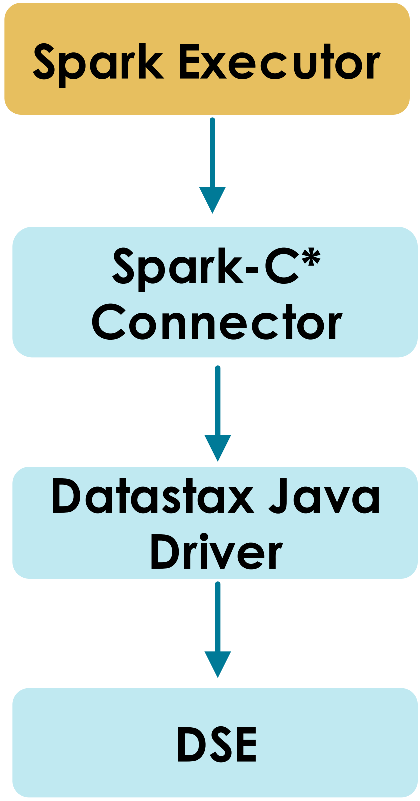
Database Access with DataStax Driver

- DataStax Cassandra Spark driver
  - Implemented mostly in Scala
  - Scala + Java APIs
  - Does automatic type conversions

```
// Spark connection options
val conf = new SparkConf(true)
     .set("spark.cassandra.connection.host", "127.0.0.1")
     .set("spark.cassandra.auth.username", "cassandra")
     .set("spark.cassandra.auth.password", "cassandra")
val sc = new SparkContext("spark://127.0.0.1:7077", "myapp", conf)


// Read from DSE and add partitioner with primary key
val rdd = sc.cassandraTable("my_keyspace", "my_table").byKey("pk","cc")


// Save to DSE
rdd.saveToCassandra("my_keyspace", "my_table", SomeColumns("key", "value"))
```



Spark Executor

Spark-C* Connector

Datastax Java Driver

DSE

Spark data model RDD

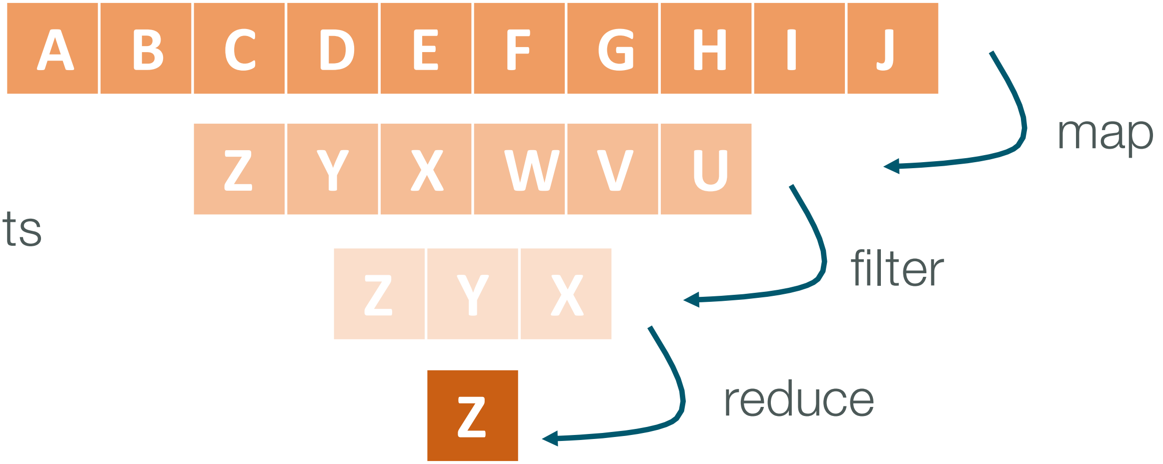
RDD = Resilient Distributed Dataset

A collection with following qualities:

- immutable
- iterable
- serializable
- distributed
- parallel
- lazy

Immutable In and Outputs

Partitioned RDD



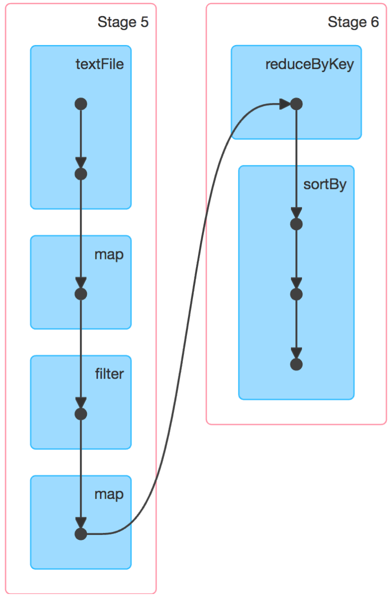
Transformations are state less

RDD – Resilient Distributed Dataset

- Sparks RDD is the Data abstraction layer for the distributed data processing
- RDDs are **stateless , immutable and partitioned data collection**s, which are distributed across many machines (cluster)

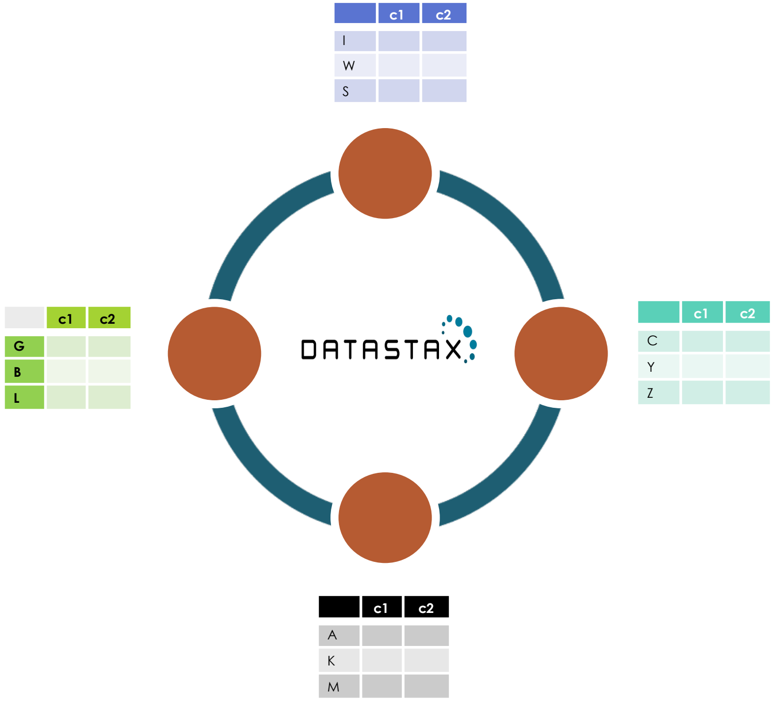
```
myRDD.filter(_._3 == "Hessen")
     .map(record => (record._5,1)).
     .reduceByKey( (a,b) => (a + b)).
     .sortBy(-_._2).
     .take(10)
```

Directed Acyclic Graph (DAG)



- **Resilience** : Spark's RDDs dependencies address fault tolerance by using a **lineage graph**
- **Lazy Evaluation**: transformations performed on RDDs without actually spending compute time on them
- **RDD functions and data structure are opaque** to Spark => general-purpose compute engine

Data Locality

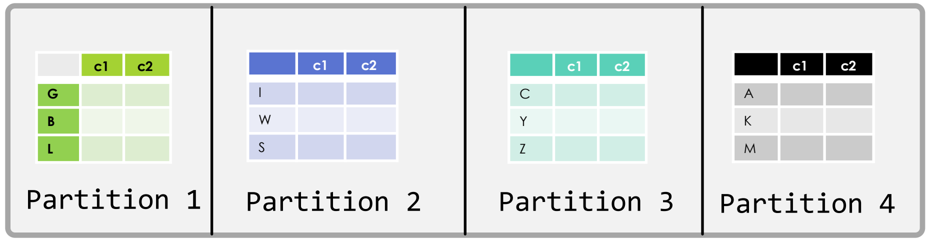


- DSE Analytics respects data locality
- No need for ETL between separated clusters
- Spark Master HA

Every Spark task uses a CQL-like query to fetch data for a given token range:

```
SELECT "key", "value"FROM "keyspace"."table“
    WHERE
        token("key") >   384023840238403 AND
        token("key") <= 38402992849280
ALLOW FILTERING
```

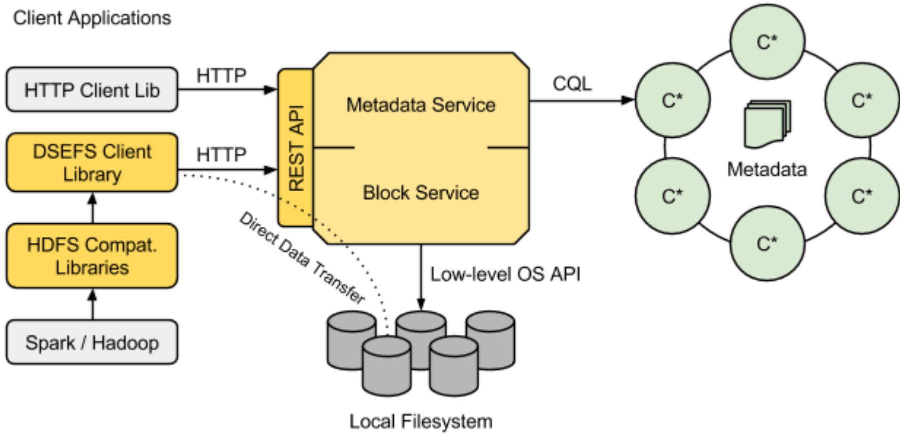
In Memory: Distributed on all available nodes



DSEFS – DataStax Enterprise Filesystem

DSEFS is a fault-tolerant, general purpose, distributed file system
For data ingestion, data staging, and state management for Spark Streaming

- Support of operational analytics and as a drop-in replacement for HDFS
- DSEFS stores data blocks outside of Cassandra tables
- No compaction and no writing to commit log
- No master or leader node, Metadata is stored in DSE tables for fault-tolerant
- No dependency to Zookeeper
- DSEFS user authentication and Linux-like file system permissions
- Deleting files is fast and space is reclaimed immediately



References

https://www.datastax.com/dev/blog/from-cfs-to-dsefs
https://www.datastax.com/2017/04/whats-new-in-datastax-enterprise-analytics-5-1

Pushdown Predicate and Integration with DSE Search

- SearchAnalytics mode allow you to create analytics queries that use DSE Search indexes
- improves performance by reducing the amount of data that is processed

Push Down Predicate with DSE Search solr query

```
val table = sc.cassandraTable("states_statistics","de_zip_code")
val result = table.select("zip","city")
.where("solr_query='cite:He*'")
.take(10)
```

Push Down Predicate with DataFrames

```
val table1 = spark.read.cassandraFormat("department","hr"))
.load()
```

DataFrames and Datasets

- Higher Level API of structured distributed data
- DataSets are structured, typesafe objects
- DataFrames equivalent to tables in relational DBs
- Uses Query optimizations and predicate pushdown
- Enables better optimizations through Spark
- Faster serialization and less memory consuming

Scala query

```
spark.table("zip").
    filter("state = 'Hessen'").
    groupBy("city").
    count().
    orderBy(desc("count")).
    limit(10).show()
```

SQL Query

```
spark.sql("select count(zip) z, city c
         FROM zip
         WHERE State = 'Hessen'
         GROUP BY city
         ORDER BY z desc
         LIMIT 10").show()
```

Apache Spark @Scale: A 60 TB+ production use case

https://code.facebook.com/posts/1671373793181703/apache-spark-scale-a-60-tb-production-use-case/

Vielen Dank!