



DataStax Enterprise Analytics

Rob Murphy & Aaron Regis

14th June 2017

Agenda

1	Introduction DSE Analytics
2	Hands-on DSE Analytics

DSE Analytics

Data extraction, transformation and load (ETL)

Cross-table operations, JOIN, UNION

Ad-Hoc queries

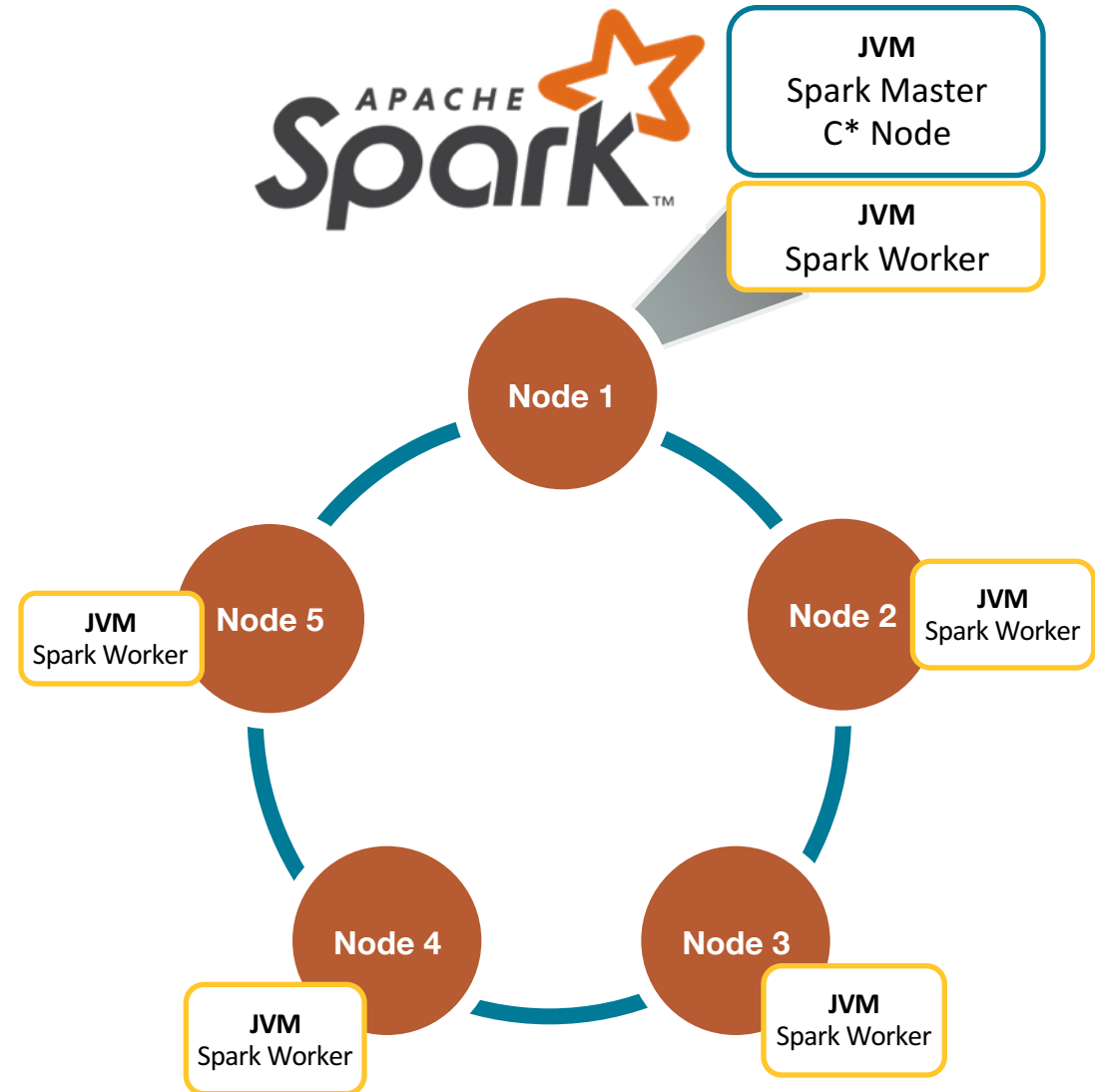
Complex analytics e.g. machine learning

Streaming processing



Architecture overview

- HA for Spark Master
- All nodes are worker nodes
- No extra software needed
e.g. Zookeeper, Yarn



Spark data model RDD

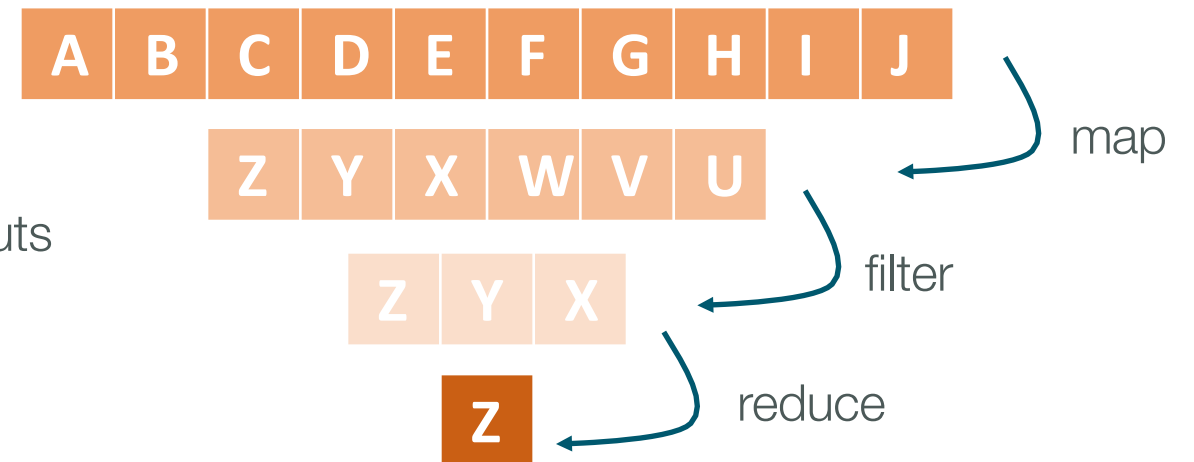
RDD = Resilient Distributed Dataset

A collection with following qualities:

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

Immutable In and Outputs

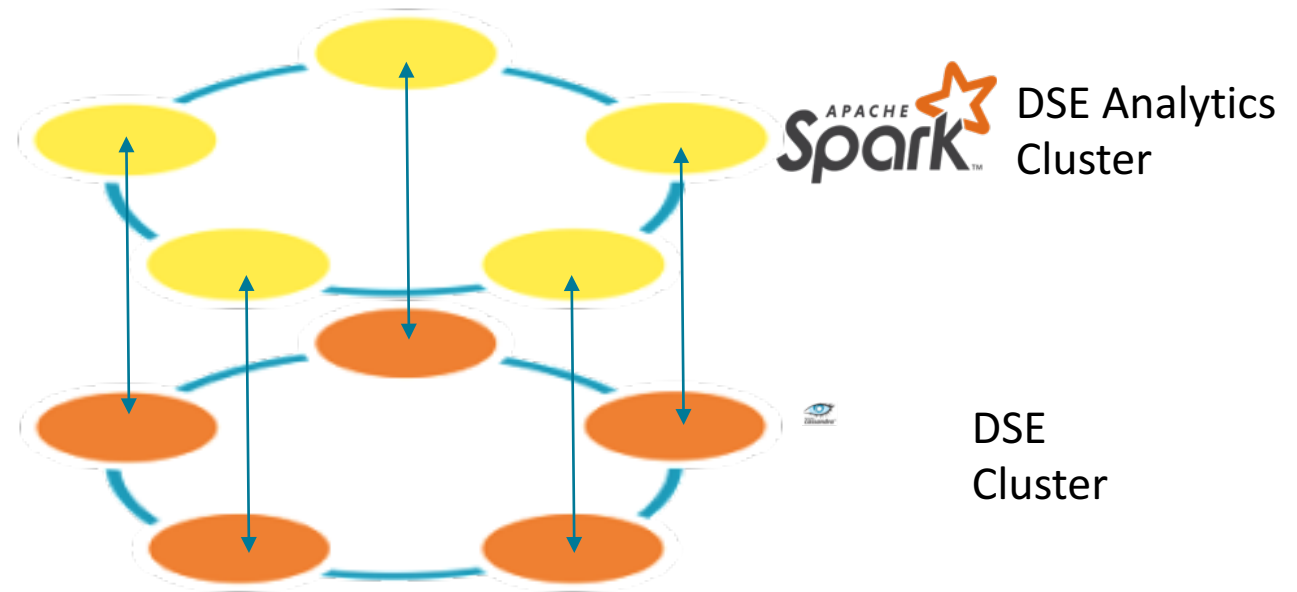
Partitioned RDD



Transformations are state less

Token Ranges Aware

- Simple provisioning and deployment
- Data locality: less network hops
- Pushdown Predicates
- Caching



Submit Spark Jobs to DSE Analytics

Spark Master: Allocate resources for app

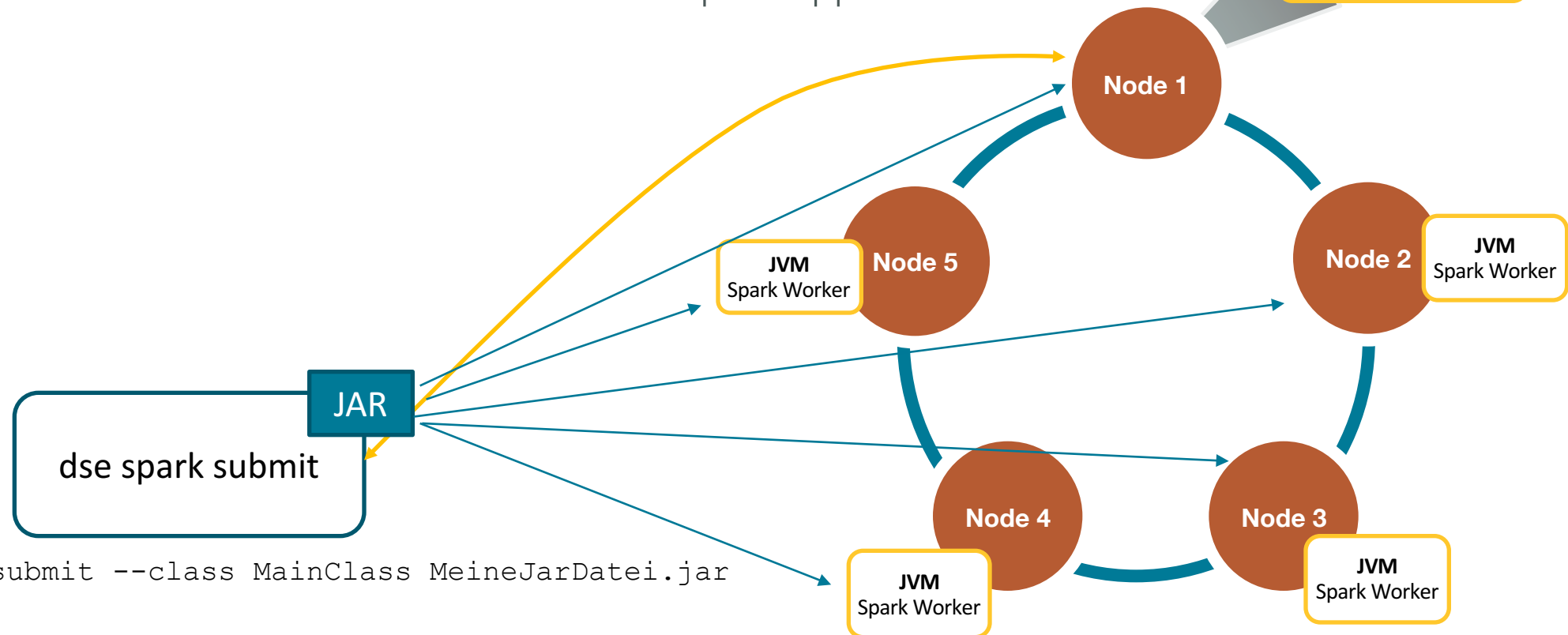
Spark Worker: Starts and manages the executors

Spark Executor: Covers all main tasks of the Spark App



JVM
Spark Master
C* Node

JVM
Spark Worker



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

Features of DSE Search

- Apache Cassandra™ tables exposed as
 - Apache Spark™ RDDs
 - Apache Spark™ DataFrames
- Load data from Apache Cassandra™ to Apache Spark™
- Write data from Apache Spark™ to Apache Cassandra™
- Object mapper : Mapping of C* tables and rows to Scala / Java objects
- All Apache Cassandra™ types supported and converted to Scala / Java types
- Server side data selection
- Virtual Nodes support
- Data Locality awareness
- Scala, Python and Java APIs

DSE Spark REPL

```
$ dse spark
```

```
scala> val table = sc.cassandraTable[CassandraRow](  
retailer', 'sales').select("name", "item").where("name= ?", "gregg")
```

```
table: com.datastax.spark.connector.rdd.CassandraTableScanRDD [com.datastax.spark.connector.CassandraRow] =  
CassandraTableScanRDD[5] at RDD at CassandraRDD.scala:15
```

```
// With collect the data gets read from Cassandra
```

```
scala> table.collect()
```

```
res1: Array[com.datastax.spark.connector.CassandraRow] = Array(CassandraRow{name: gregg, item: PlayStation 4},  
CassandraRow{name: gregg, item: iMac}, CassandraRow{name: gregg, item: Microsoft Xbox})
```

```
scala> table.collect().foreach(println);
```

```
CassandraRow{name: gregg, item: PlayStation 4}
```

```
CassandraRow{name: gregg, item: iMac}
```

```
CassandraRow{name: gregg, item: Microsoft Xbox}
```

Spark SQL

Save to DSE

```
sc.parallelize(Seq(A,B,C)).saveToCassandra("demo", "collections")
```

Spark RDD JOIN with NOSQL!

```
churnRateRdd.join(custJourney).saveToCassandra("customerESxty", "churns")
```

Use Spark SQL to select the database

```
val track_count_by_year = sqlContext.sql("select 'dummy' as dummy, album_year as year, count(*) as track_count from tmp_tracks_by_album group by album_year")
```

DataStax Enterprise Max – Apache Spark™ Special Features

- Easy setup and config
 - No need to setup a separate Apache Spark™ cluster
 - No need to tweak classpaths or config files
- High availability of Apache Spark™ Master
- Enterprise security
 - Password / Kerberos / LDAP authentication
 - SSL for all Apache Spark™ to Apache Cassandra™ connections
- Included support of Apache Spark™ in DSE Max subscription

Lab 6 : Hands-on DSE Analytics

Thank You!