B14
Apache Spark with IMS and DB2 data

Denis Gaebler, IBM Germany

gaebler@de.ibm.com

What is Apache Spark?

IMS Data with Apache Spark

Spark analytics in IMS applications

Demo

Summary

What is Apache Spark?

Spark - Definition from Wikipedia

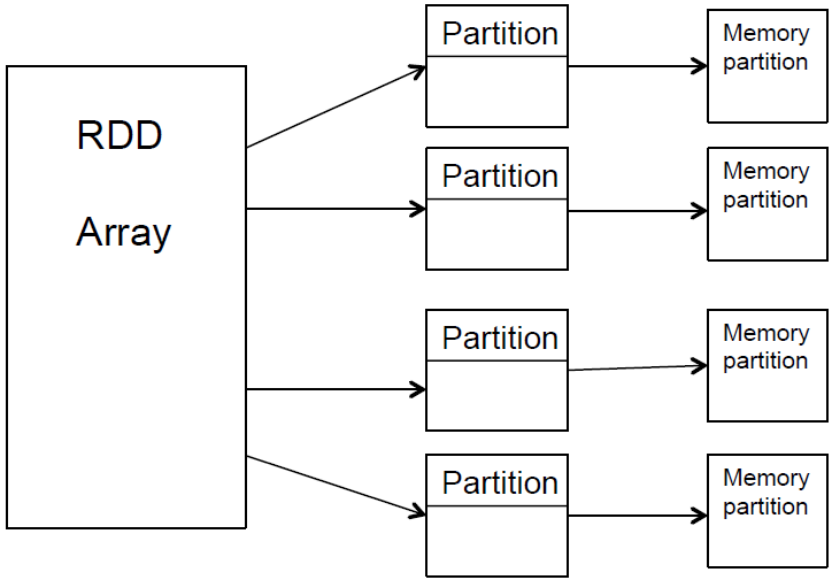
- Open source cluster computing framework
- Originally developed by University of Berkeley, then donated to Apache Software Foundation
- Provides interface for programming entire clusters with implicit parallelism and fault-tolerance

What is Apache Spark?

- **Addressing limitations of Hadoop MapReduce programming model**
  - –No iterative programming, latency issues, ...
- **Using a fault-tolerant abstraction for in-memory cluster computing**
  - –Resilient Distributed Datasets (RDDs)
- **Can be deployed on different cluster managers**
  - –YARN, MESOS, standalone
- **Supports a number of languages**
  - –Java, Scala, Python, SQL, R
- **Comes with a variety of specialized libraries**
  - –SQL, ML, Streaming, Graph
- **Enables additional use cases, user roles, and tasks**
  - –E.g. data scientist

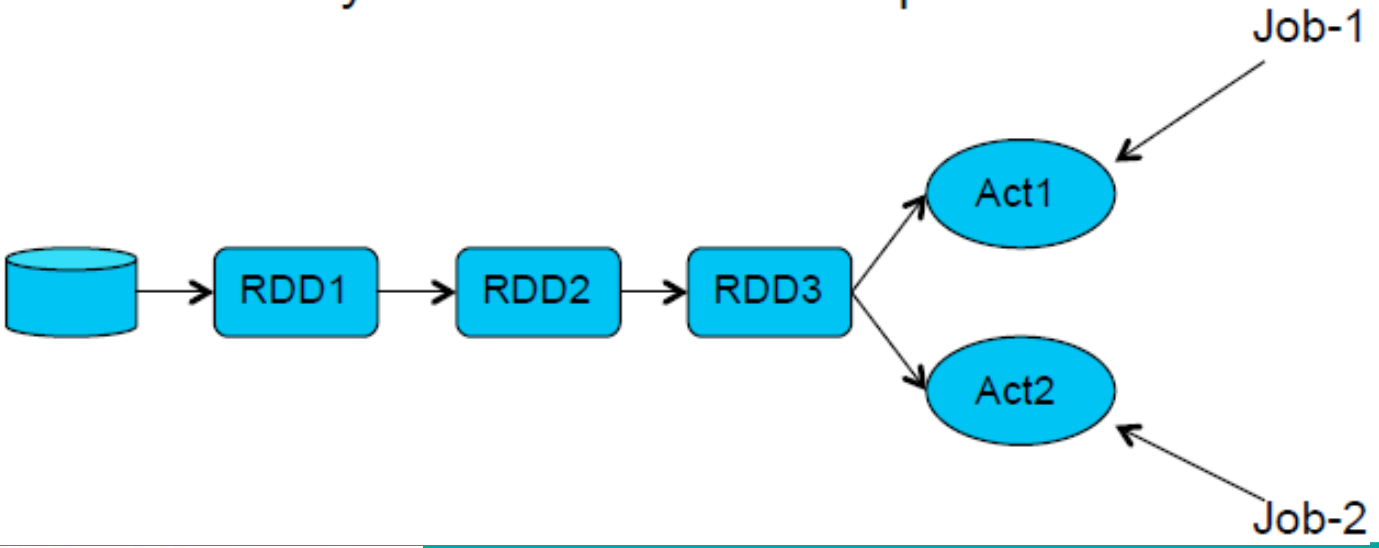
Resilient Distributed Dataset (RDD)

- **Key idea: write programs in terms of transformations on distributed datasets**
- **RDDs are immutable**
  - –Modifications create new RDDs
- **Holds references to partition objects**
- **Each partition is a subset of the overall data**
- **Partitions are assigned to nodes on the cluster**
- **Partitions are in memory by default**
- **RDDs keep information on their lineage**



Spark Programming Model

- **Operations on RDDs (datasets)**
  - –Transformation
  - –Action
- **Transformations use lazy evaluation**
  - –Executed only if an action requires it
- **An application consist of a directed acyclic graph (DAG)**
  - –Each action results in a separate batch job
  - –Parallelism is determined by the number of RDD partitions



What can you do with Spark programming
Some examples to get an idea based on Scala syntax

- **val textFiles = spark.wholeTextFiles("somedir")**
  // Read text files from "somedir" into an RDD of (filename, content) pairs.
- **val contents = textFiles.map(_._2)**
  // Throw away the filenames.
- **val tokens = contents.flatMap(_.split(" "))**
  // Split each file into a list of tokens (words).
- **val wordFreq = tokens.map((_, 1)).reduceByKey(_ + _)**
  // Add a count of one to each token, then sum the counts per word type.
- **wordFreq.map(x => (x._2, x._1)).top(10)**
  // Get the top 10 words. Swap word and count to sort by count.

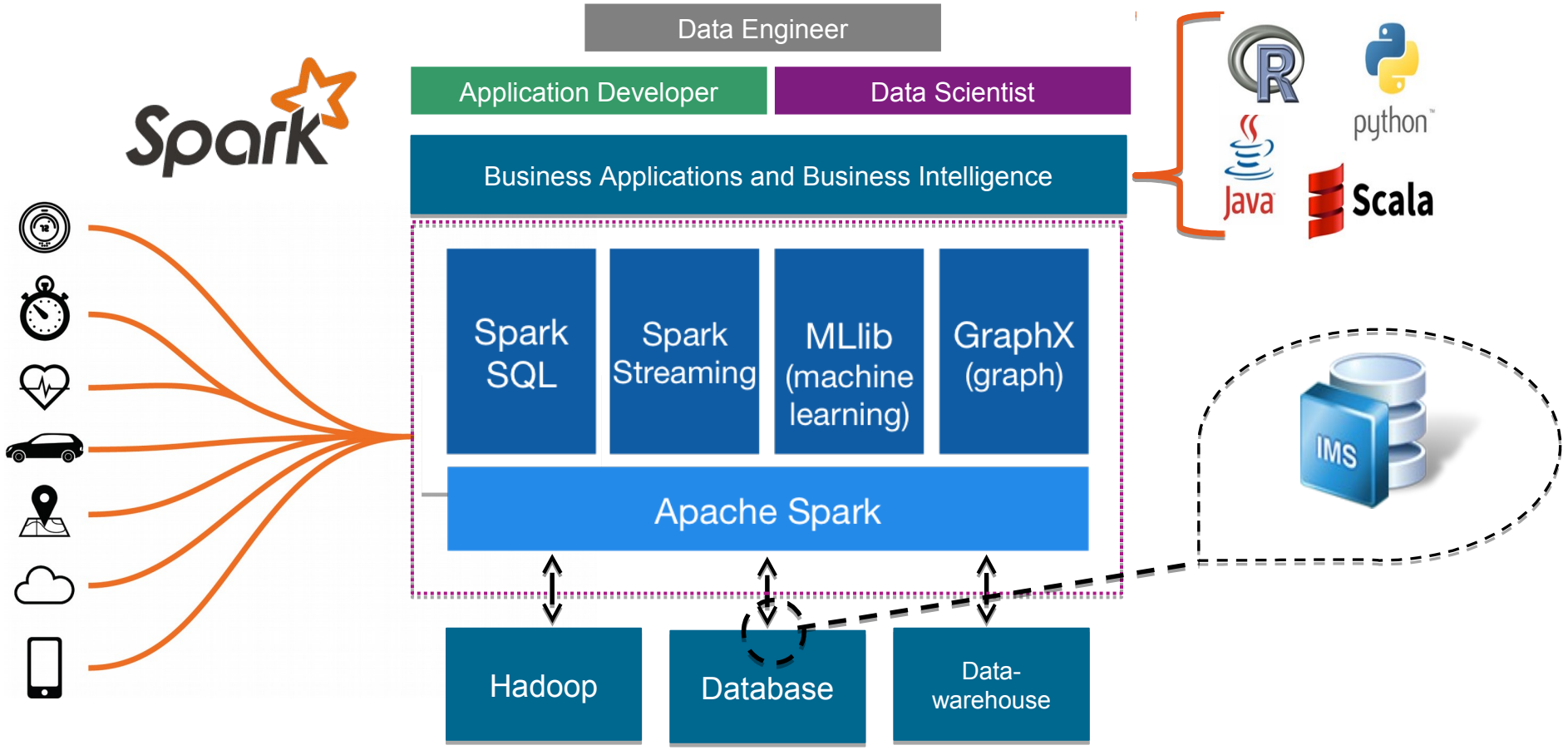
IMS Data with Apache Spark

IMS JDBC Driver, IMS Open Database with Spark

IMS and DB2 got the data, so what?! Analytics requirements

- Multiple solutions to access data, IBM and ISVs, free or charge
- Big data and analytics with unstructured data
- Join data from IMS and DB2 databases
- Assume Spark PoCs have been done in other departments, now they need real data
- Spark with Scala as the programming language is used
- Local Spark installation with Scala Shell, Scala Eclipse environment, Spark Driver program with Spark Cluster
- Spark analytics from z/OS Batch and Online

Spark Integration with IMS data



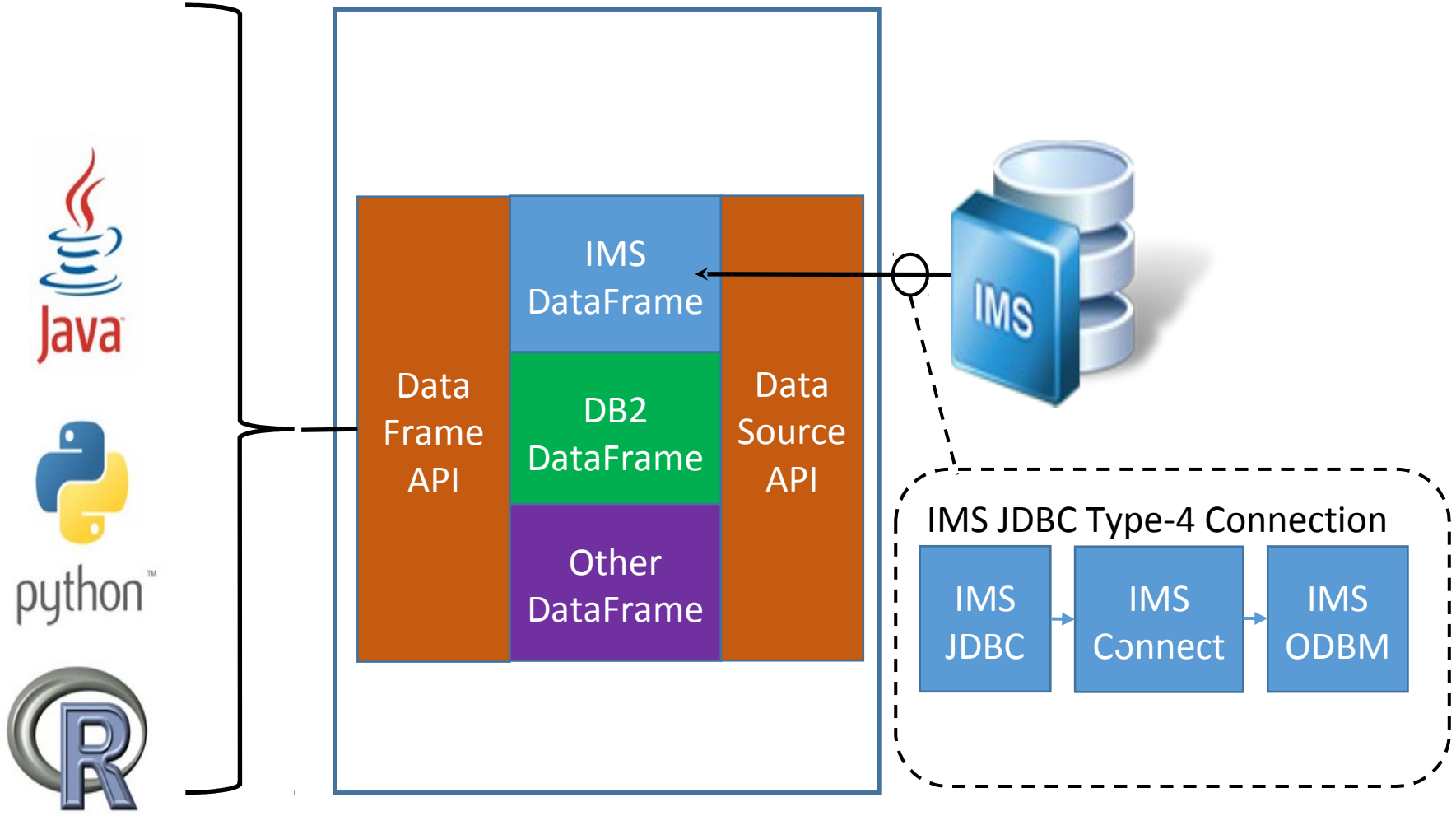
How to use relational data with Spark?

- Apache Spark comes with DataFrames support
- DataFrames API allows to use standard JDBC drivers such as DB2 Universal Database Driver and IMS Universal Database Driver
- Using it is as simple as adding the JDBC Driver to the Apache Spark classpath
- Scala sample for Type 4 JDBC:

```
val stokStat = sqlContext.load("jdbc", Map("url" ->
"jdbc:ims://172.16.36.226:5559/class://com.ibm.ims.db.databaseviews.DFSSAM09Datab
aseView:user=gaebler;password=password;","driver" ->
"com.ibm.ims.jdbc.IMSDriver","dbtable" -> "STOKSTAT"))
```

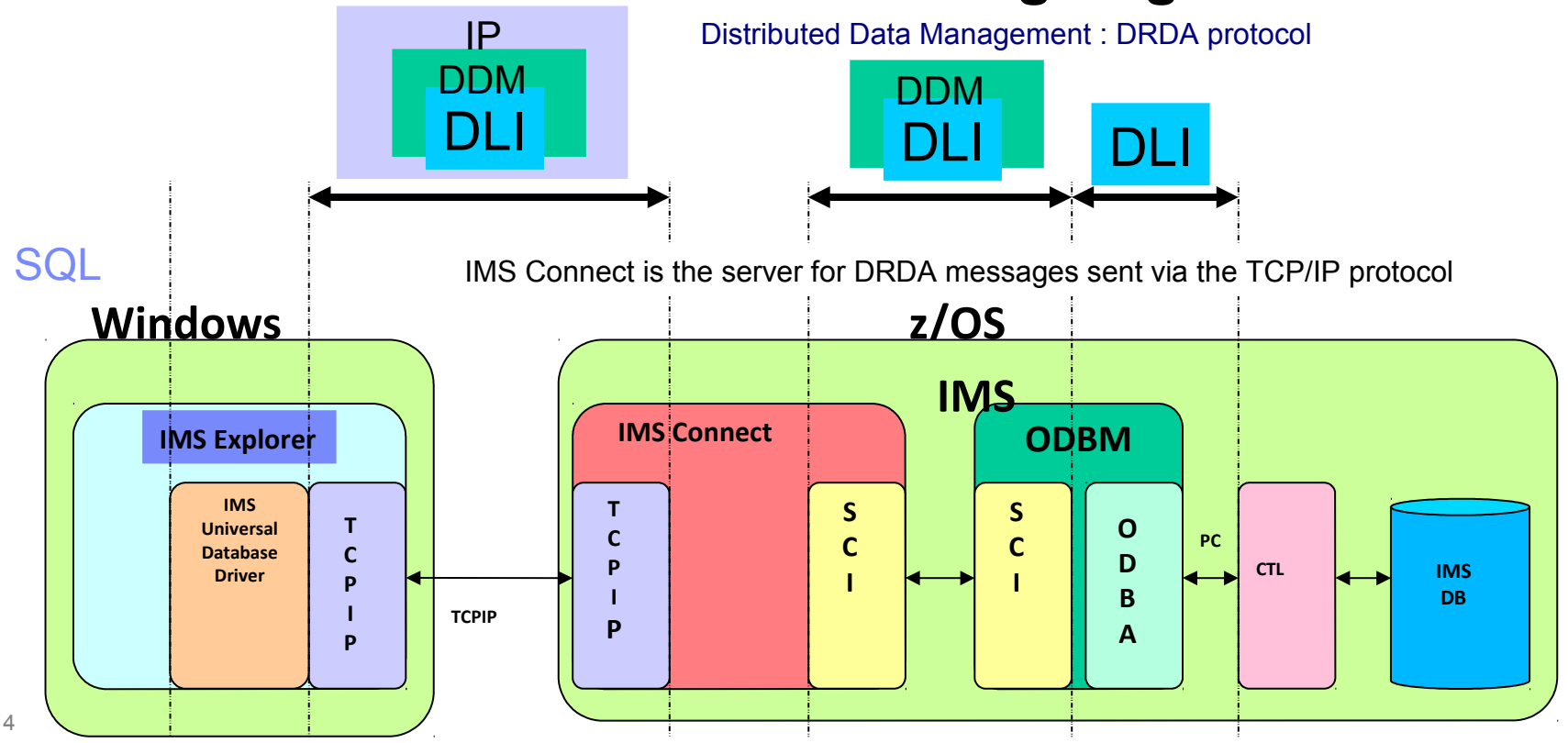
- For IMS PROCOPT=GO and for DB2 uncommited read recommended to avoid locking from lots of Spark jobs

Spark using IMS data and IMS Open Database



Technical Prerequisites

- DB2 just requires the DB2 Universal Driver and DDF
- IMS requires Open Database Infrasturcture (IMS V13+ and PTF for APAR PI47263, no retrofit to IMS V12)
- IMS DB Metadata either in the Catalog or generated in Java

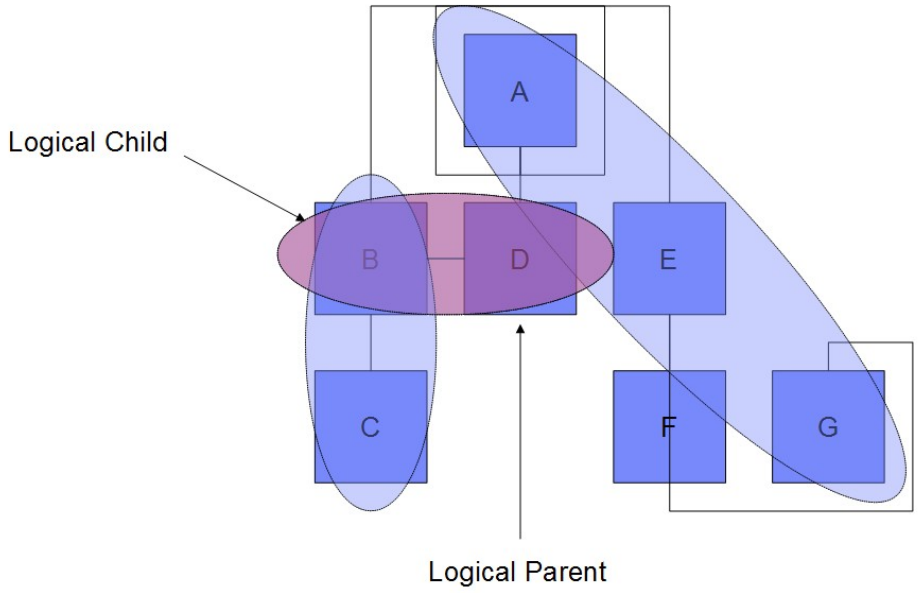


Spark for the advanced user

- One dataframe represents one IMS segment or IMS SQL resultset
- One dataframe represents one DB2 table or DB2 SQL resultset
- Join of two or more data sources with Spark possible
  Thats done by joining dataframes
  Result of a join is a new dataframe
- Analytics possible for those joined data sources
- Spark allows for caching and reusing the results which reduces the number of SQLs issued against the database

Spark for the advanced user...

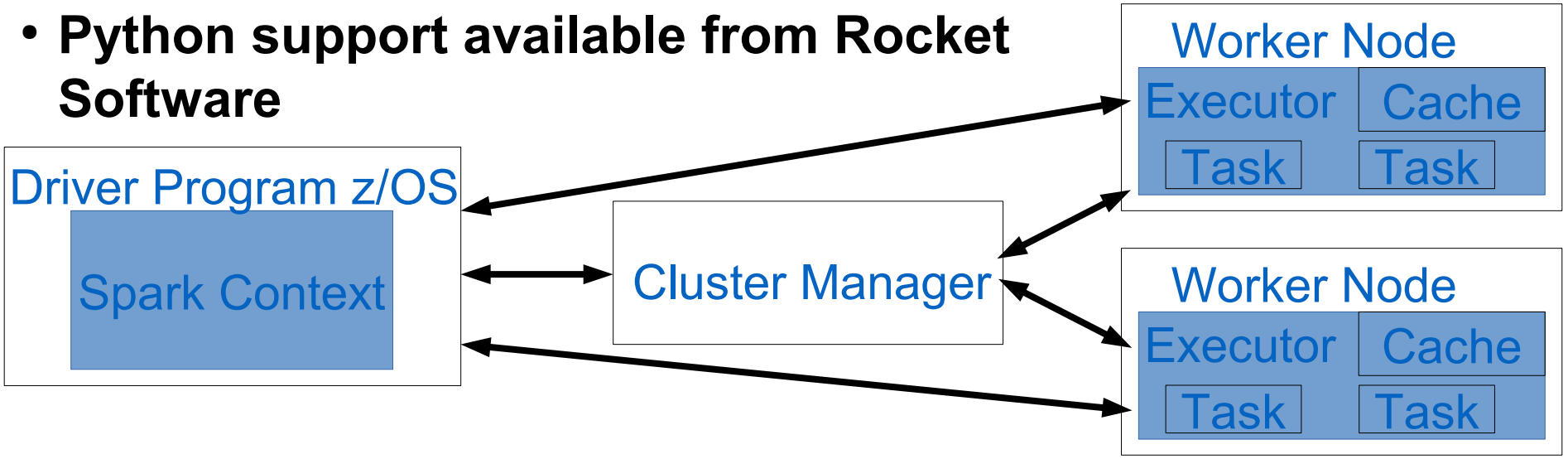
- **What is not possible with the IMS JDBC driver, is possible with Spark**
  Joins can be done for multiple IMS Segments that are in different hierarchic paths
  Joins can be done for multiple IMS databases that are separate PCBs in PSBs



Spark analytics in IMS applications

Spark APIs

- **Possible code available from PoCs or first projects on the distributed side**
  Scala API and Java API for Spark job submit can be used from IMS applications
- **Java API for Spark can be invoked with COBOL/Java interoperability**
  Scala API also possible
- **Python support available from Rocket Software**



Scala in the z/OS JVM

- Scala runs on the z/OS JVM
- So all environments that use the z/OS JVM can also invoke Scala applications
- That includes IMS JBP, JMP
- COBOL or PL/I to Scala Interoperability possible
- Export of compiled Scala code is a .jar that includes a class with a main method
  Can be executed on any JVM
- Make sure the classpath is complete, in tests most problems were due to incomplete classpaths

Scala IDE

IMSAccessSampleSimple.scala

```
package com.ibm.imstest

import org.apache.spark.SparkConf


object IMSAccessSampleSimple extends App {
try {
  val conf = new SparkConf()
  .setMaster("local[1]")
  .setAppName("GetStokStat")
  .set("spark.executor.memory", "1g")
  .set("spark.io.compression.codec", "lzf")

  val sc = new SparkContext(conf)

  val sqlContext = new SQLContext(sc)

  val optionsStokStat = Map[String, String]();
  optionsStokStat.put("driver", "com.ibm.ims.jdbc.IMSDriver");
  optionsStokStat.put("url", "jdbc:ims://172.16.36.226:5559/class://com.ibm.ims.db.databaseviews.DFSSAM09DatabaseView:user=gaebl
  //optionsStokStat.put("url", "jdbc:ims:class://com.ibm.ims.db.databaseviews.DFSSAM09DatabaseView");
  optionsStokStat.put("dbtable", "STOKSTAT");

  val stokStat = sqlContext.read.format("jdbc").options(optionsStokStat).load();

  stokStat.show();

  val optionsPartRoot = Map[String, String]();
  optionsStokStat.put("driver", "com.ibm.ims.jdbc.IMSDriver");
  optionsPartRoot.put("url", "jdbc:ims://172.16.36.226:5559/class://com.ibm.ims.db.databaseviews.DFSSAM09DatabaseView:user=gaebl
//  optionsPartRoot.put("url", "jdbc:ims:class://com.ibm.ims.db.databaseviews.DFSSAM09DatabaseView");
  optionsPartRoot.put("dbtable", "PARTROOT");

  val partRoot = sqlContext.read.format("jdbc").options(optionsPartRoot).load();

  partRoot.show(100);

  val joined = partRoot.join(stokStat, partRoot("PARTKEY") === stokStat("PARTROOT_PARTKEY"));

  joined.show()
  joined.cache()
} catch {
```

DEMO

Summary

Possible Use Cases

- Most companies have processes to extract mainframe data and send it to distributed platforms
- Usually thats unload/extract and ftp
- Combine and analyze the data outside the mainframe
- No sending back of the data required
- Think about replacing those processes with Spark
- Additional benefits through analytics possibilities of Spark
- Frameworks on top of Spark allow sophisticated visualization

Summary

- Apache Spark comes with DataFrames support
- DataFrames API allows to use standard JDBC drivers
- For DB2 you need DDF and for IMS the Open Database Infrastructure
- IMS Catalog is a strong surplus
- Get in touch with the PoCs and Users using Spark and advertise the access to IMS data