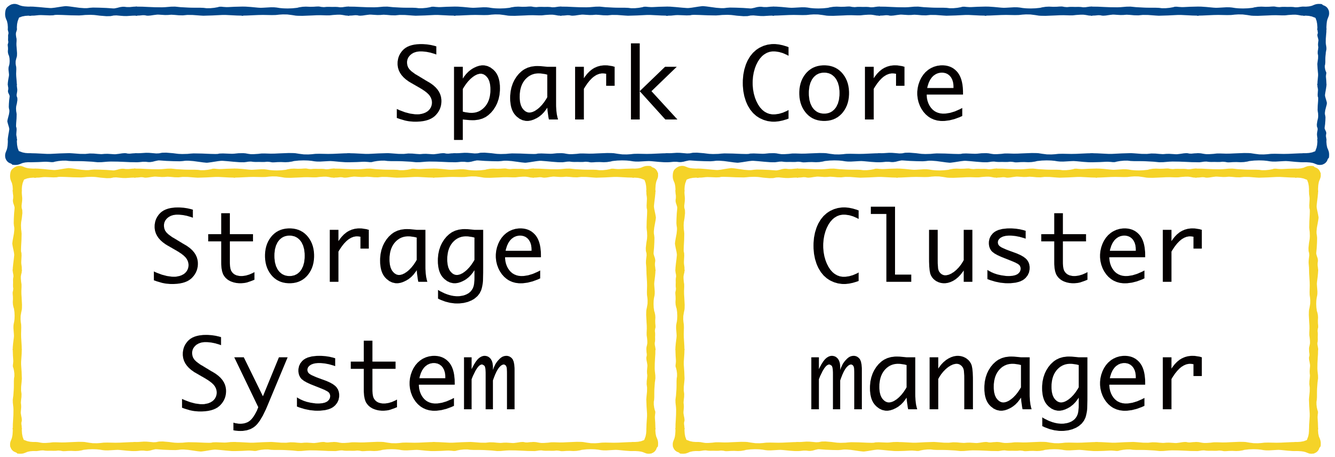
SPARK SQL

Recap

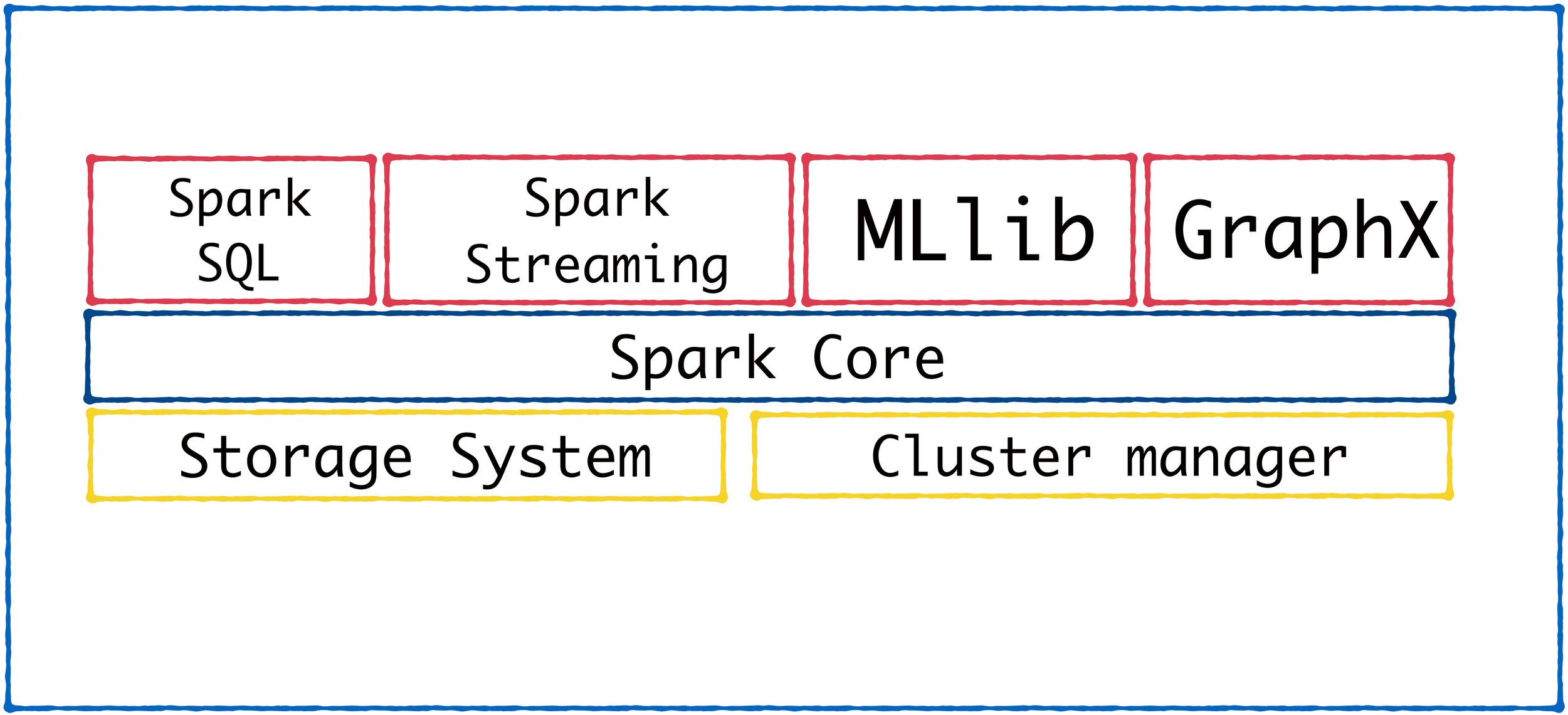
APACHE SPARK

Spark comes with some additional packages that make it truly general - purpose



Recap

APACHE SPARK



Recap

APACHE SPARK

Spark SQL

Spark SQL provides an SQL interface for Spark

Recap

APACHE SPARK

Spark SQL

Lot's of folks are familiar with SQL and like to use to express data manipulation

Recap

APACHE SPARK

Spark SQL

With Spark SQL, folks can use SQL to work with large amounts of data stored in a cluster

Recap

APACHE SPARK

Spark SQL

You can use the familiar SQL and still get all the in-memory performance benefits of Spark

Spark SQL

Let's say you were performing some complex data manipulations in a program

Spark SQL

You could load data in table form into memory

using a special type of RDD called a DataFrame

DataFrame

Spark SQL

DataFrames are like in-memory database tables

You can use SQL statements to query and manipulate them

DataFrame

Spark SQL

DataFrames are also like RDDs

You can use most of the usual transformations and actions with some modifications

Spark
SQL

Say you wanted to work with some Twitter data

Twitter data

Spark SQL

datahub has a dataset with tweets from the Presidential election in 2012

Twitter data

Spark SQL

datahub
The easy way to get, use and share data

Datasets Organizations About Blog Help Search

/ Organizations / Kingmolnar / **Twitter 2012 Presidential ...**

Twitter 2012 Presidential Election

Followers
1

Organization

Dataset Groups Activity Stream Related

Twitter 2012 Presidential Election

This data-set contains over 170,000,000 tweets collected during 3 months leading up to the 201 presidential elections.

Format

Twitter data

Spark SQL

The records are stored in JSON form

{ "created_at": "Fri Dec 21 22:53:46 +0000 2012", "id": 282257583836889090, "id_str": "282257583836889090", "text": "Obama meets Reid on fiscal cliff, plans remarks http://t.co/Nuec5Wh6", "source": "twitterfeed", "truncated": false, "in_reply_to_status_id": null, "in_reply_to_status_id_str": null, "in_reply_to_user_id": null, "in_reply_to_user_id_str": null, "in_reply_to_screen_name": null, "user": { "id": 27703934, "id_str": "27703934", "name": "Washington Examiner", "screen_name": "washexaminer", "location": "Washington DC", "url": "http://www.washingtonexaminer.com", "description": "", "protected": false, "followers_count": 1995, "friends_count": 67, "listed_count": 94, "created_at": "Mon Mar 30 18:56:55 +0000 2009", "favourites_count": 0, "utc_offset": -18000, "time_zone": "Eastern Time (US & Canada)", "geo_enabled": true, "verified": false, "statuses_count": 42208, "lang": "en", "contributors_enabled": false, "is_translator": false, "profile_background_color": "FFFFFF", "profile_background_image_url": "http://a0.twimg.com/profile_background_images/553522199/twitter-design-Sections1.jpg", "profile_background_image_url_https": "https://si0.twimg.com/profile_background_images/553522199/twitter-design-Sections1.jpg", "profile_background_tile": false, "profile_image_url": "http://a0.twimg.com/profile_images/2223644256/logo-social_normal.gif", "profile_image_url_https": "https://si0.twimg.com/profile_images/2223644256/logo-social_normal.gif", "profile_link_color": "CA0613", "profile_sidebar_border_color": "C0DEED", "profile_sidebar_fill_color": "DDEEF6", "profile_text_color": "333333", "profile_use_background_image": true, "default_profile": false, "default_profile_image": false, "following": null, "follow_request_sent": null, "notifications": null }, "geo": null, "coordinates": null, "place": null, "contributors": null, "retweet_count": 0, "entities": { "hashtags": [], "urls": [{ "url": "http://t.co/Nuec5Wh6", "expanded_url": "http://bit.ly/UWIjMN", "display_url": "bit.ly/UWIjMN", "indices": [48, 68] }], "user_mentions": [] }, "favorited": false, "retweeted": false, "possibly_sensitive": false, "lang": "en" }

Twitter data

Spark SQL

We can treat the JSON file as any other text file

```
val twitterData=sc.textFile(twitterPath)
```

Twitter data

Spark SQL

```
import play.api.libs.json._
```

```
val twitterData=sc.textFile(twitterPath).map(x => Json.parse(x))
```

We can then use a JSON parser library to parse the data

Twitter data

Spark SQL

```
import play.api.libs.json._
```

```
val twitterData=sc.textFile(twitterPath).map(x => Json.parse(x))
```

twitterData is an RDD where each record is JSObject

```
twitterData.first()\"text"
```

```
"Obama vies for health care edge in Florida - h
```

Twitter data

Spark SQL

```
import play.api.libs.json._
```

```
val twitterData=sc.textFile(twitterPath).map(x => Json.parse(x))
```

Each record has all details of a single tweet

```
twitterData.first()\"text"
```

```
"Obama vies for health care edge in Florida - h
```

Twitter data

Spark SQL

```
import play.api.libs.json._
```

```
val twitterData=sc.textFile(twitterPath).map(x => Json.parse(x))
```

```
twitterData.first( \"text"
```

```
"Obama vies for health care edge in Florida - h
```

You can access fields using the \ operator

Twitter data

Spark SQL

```
val twitterData
```

You can manipulate this using the usual transformations and actions

map, reduce, aggregate etc

Twitter data

Spark SQL

```
twitterData.filter(x => (x\"user"\"screen_name").toString.contains("realDonaldTrump")).m
```

If you wanted to look at a sample of tweets from Donald Trump during this period

Twitter data

Spark SQL

```
ta.filter(x => (x\"user"\"screen_name").toString.contains("realDonaldTrump"))
```

First, filter tweets by Donald Trump

Twitter data

Spark SQL

```
rump")).map(x => x\"text").take(10)
```

Extract the text from those tweets

Twitter data

Spark SQL

```
ap(x => x\"text").take(10)
```

A sample of tweets

Twitter data

Spark SQL

```
twitterData.filter(x => (x\"user"\"screen_name").toString.contains("realDonaldTrump")).map(x => x\"text").take(10)
```

Many folks are comfortable with querying data using SQL

Twitter data

Spark SQL

```
twitterData.filter(x => (x\"user"\"screen_name").toString.contains("realDonaldTrump")).map(x => x\"text").take(10)
```

If twitterData were a database table, this same operation could be expressed as

```
SELECT text FROM twitterData
WHERE screen_name='realDonaldTrump'
LIMIT 10
```

Twitter data

Spark SQL

```
twitterData.filter(x => (x\"user"\"screen_name").toString.contains("realDonaldTrump")).map(x => x\"text").take(10)
```

With Spark SQL and DataFrames, you can do exactly this!

```
SELECT text FROM twitterData
WHERE screen_name='realDonaldTrump'
LIMIT 10
```

Twitter data

Spark SQL

We normally load data into RDDs using a SparkContext

```
val twitterData=sc.textFile(twitterPath)
```

To load data into a DataFrame we need an SQLContext

Twitter data

SQLContext

Spark SQL

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.Row
```

Import SQLContext from spark.sql

Twitter data

SQLContext

Spark SQL

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.Row
```

```
val sqlC= new SQLContext(sc)
```

Use the SparkContext to set up the SQLContext

Twitter data

SQLContext

Spark SQL

```
import org.apache.spark.sql.SQLContext
import org.apache.spark.sql.Row
```

```
val sqlC= new SQLContext(sc)
```

```
val twitterTable=sqlC.read.json(twitterPath)
```

The SQLContext has a method to directly read JSON files

Twitter data

SQLContext

Spark SQL

```
twitterTable:
```

While the data is loaded, SQL Context will infer the schema of the table

Twitter data

SQLContext

Spark SQL

```
twitterTable.printSchema()
```

```
root
 |-- contributors: array (nullable = true)
 |     |-- element: long (containsNull = true)
 |-- coordinates: struct (nullable = true)
 |     |-- coordinates: array (nullable = true)
 |     |     |-- element: double (containsNull = true)
 |     |-- type: string (nullable = true)
 |-- created_at: string (nullable = true)
 |-- entities: struct (nullable = true)
 |     |-- hashtags: array (nullable = true)
 |     |     |-- element: struct (containsNull = true)
 |     |     |     |-- indices: array (nullable = true)
 |     |     |     |     |-- element: long (containsNull = true)
 |     |     |     |-- text: string (nullable = true)
 |     |-- media: array (nullable = true)
 |     |     |-- element: struct (containsNull = true)
 |     |     |     |-- display_url: string (nullable = true)
 |     |     |     |-- expanded_url: string (nullable = true)
```

Twitter data

SQLContext

Spark SQL

```
twitterTable
```

The data is loaded into an in-memory table i.e. DataFrame

Twitter data

SQLContext

Spark SQL

```
twitterTable.getClass
```

```
class org.apache.spark.sql.DataFrame
```

Twitter data

SQLContext

Spark SQL

```
twitterTable:
```

You can use SQL statements to query this DataFrame

Twitter data

SQLContext

Spark SQL

```
twitterTable:
```

The SQL statements will be executed by the SQLContext

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
```

First you need to register this DataFrame as a table with the SQLContext

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
```

This is just a bit of setup before you can use Spark SQL with twitterTable

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
```

This is the alias for the Dataframe in Spark SQL

Twitter data

Spark SQL

SQLContext

```
twitterTable.registerTempTable("twitterTable")
```

You can give any name here

Just use the same name in your SQL statements

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
sqlC.sql("Select text, user.screen_name from twitterTable where user.screen_name='realDonaldT
```

The SQLContext has an SQL method to which you'll give your SQL query

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
```

```
sqlC.sql("Select text, user.screen_name from twitterTable where user.screen
```

This is just like a Transformation

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
```

```
sqlC.sql("Select text, user.screen_name from twitterTable where user.screen
```

The output is another DataFrame

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
```

```
mp' limit 10").collect()
```

Just like with RDDs you need to use an action to fetch the results

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
```

```
sqlC.sql("Select text, user.screen_name from twitte
```

A DataFrame is like a special kind of RDD

Twitter data

SQLContext

Spark SQL

```
twitterTable.registerTempTable("twitterTable")
```

```
sqlC.sql("Select text, user.screen_name from twitte
```

You can still apply many of your regular transformations and actions

Twitter data

Spark SQL

DataFrames are made up of Row objects

Twitter data

Spark SQL

DataFrames are made up of Row objects

Rows are made up of Columns

You can use the field names to extract an RDD to represent 1 Column

Twitter data

Spark SQL

```
val trumpTweets = sqlC.sql("Select text, user.screen_name,entities from twitterTable where user.screen
```

Create a DataFrame with only Trump's tweets

Twitter data

Spark SQL

```
trumpTweets.select("text").take(10)
```

```
Array([Obama asked a 7 yr old for his birth certificate. He's "in your face" because the Republicans dropped the bal
l. (cont) http://t.co/FufZD79U], [Obama is taunting the Republicans on the birther issue. They should call his bluff
 &amp;amp; demand the REAL facts. He (cont) http://t.co/NWmVp06e], ["President Obama is the greatest hoax ever perpetrat
d on the American people"
--Clint Eastwood])
```

Gives you an RDD with only the text column. On this you can use regular transformations etc

Twitter data

Spark SQL

As you can see, DataFrames are very versatile!

Use SQL manipulations and Scala functions on the very same dataset

Twitter data

Spark SQL

With DataFrames you can

Use SQL manipulations

Scala functions

RDD Transformations and Actions

On the same dataset

In the same program