Fundamentals of Scalable Data Science

Week Two

Data Storage Solutions

	SQL	NoSQL	ObjectStorage
Storage cost	high	low	Very low
Scalability	Low	high	Very high
Query speed	high	low	Very low
Flexibility	low	high	Very low

SQL: a low amount of data, a very stable schema.

NoSQL: amount of data increases, continuously changing schemas.

ObjectStorage: high amounts of data, very unstable schemas, even including audio, images and video data.

Schema:

- schema is basically all the names and types (and additional properties like nullable, ...) of a relation (database table)
- static schema: you have in relational databases where u define a table with a schema
- in NoSQL you have collections of documents, mostly JSON or key-value store, each JSON document can have a different schema
- in this course we are creating a SQL view on a nosql collection containing JSON document with different schemas, so you get the UNION of the schemas and the values are NULL if the document doesn't contain them

SQL database

- Advantages

    - Well known, well established.
    - High integrity, high data normalization
    - Fast indexed access
    - Open standard
- Disadvantages
    - Change of schema needs DDL (data-definition language)
    - Hard to scale
    - High storage cost: need very reliable storage system and not tolerate temporary inaccessible storage.

NoSQL database

- Advantages
    - Dynamic schema, change of schema no problem
    - Low storage cost
    - Linearly scalable
- Disadvantages
    - No data normalization, no data integrity
    - Less established
    - Generally slower in access than SQL

ObjectStorage

- Advantages
    - Very low storage cost
    - Linearly estable
    - Seamless schema migration, schema-less

ApacheSpark - JVM

- Limited to resources of a single node;
- Want to use large compute clusters but don't want to take care of parallelizing of programs.
- Worker Nodes: single node JVM application, not involved in any computations but manage remote compute nodes. And remote compute nodes at JVM is

running and they are responsible for executing parallel version of analytics workflow.

- Two options for attaching to an ApacheSpark cluster.
    - An off node storage approach, where the third system is attached to the cluster using a fast network connection. Switching fabric: high out bandwidth is needed, guarantees the max network performance between the storage system and the worker nodes.
    - JBOD - just a bunch of disk/direct attach; HDFS: Hadoop Distributed File System. Combined storage capacity of all disks as one large virtual file system.

Q&A: consider a relational database installed on some server in a data center and ApacheSpark cluster in the same data center reading data from the database. Then Network Attached Storage (off-node storage approach) should be used. Reason: A remote relational database behaves basically like a remote file system. The only difference is that an additional data processing engine sites in between ApacheSpark and the data. But from a topological perspective this doesn't make a difference.

- RDD: Resilient Distributed Dataset
    - Distributed immutable collection or list of data
    - Only containing strings or double values and they are created from existing data from different sources like HDFS, ObjectStore, NoSQL, SQL
    - When main memory of all worker nodes is not sufficient, then data gets split to disk.
    - Only if the data is really needed for a certain computation, it is read from the underlying storage system, otherwise not.
- DSX: IBM Data Scientist Workbench
- Summary
    - ApacheSpark programs are implicitly parallel
    - No matter how much data process, the program always stays the same
    - RDD central API
    - Data and task are transparent

Q&A:

```
1  rdd = sc.parallelize(range(100))
2  rdd2 = range(100)
```

-
    - “Rdd” stored in main-memory of ApacheSparl worker nodes; “rdd2” stored on the local driver machine.
- len(range(n)) = sc.parallelize(range(n)).count()
- Support seamless modification of schemas: ObjectStorage, NoSQL
- Support dynamic scaling on storage: ObjectStorage, NoSQL
- Support normalization and integrity on data: SQL/Relational Databases

Programming language options on ApacheSpark

- Scala
    - Native to ApacheSpark
    - Complete API set is available
    - Runs fast
- Java
    - Not a data science programming language
    - Complete API set is available
    - De-factor standard in Enterprise IT
    - Language of “Hadoop”
- R
    - The data science programming language
    - Only subset of ApacheSpark API is available
    - De-factor standard in academic research
    - >8000 add on package
    - Plowing and charting
    - Slow
    - ApacheSpark programs when implemented in R are only slow when a lot of local executions within R libraries takes place
- Python
    - As widely used in Data Science as R
    - Only subset of ApacheSpark API is available
    - Nive plotting and charting but R is better here

- Can get slow
- Libraries only running on a single machine (pandas, scikit-learn, scipy)

	Scala	Java	R	Python
Spark API support	Complete	complete	Very limited	limited
Ease of use	low	Very low	high	Very high
speed	Very high	high	Very low	low
3rd party libs	few	few	many	many

Functional Programming basics

- Functional programming: every computation can be expressed as an anonymous function which is applied to a data set.
- Scala also supports procedural and object oriented programming paradigms.
- Python, R and Java also support some rudimentary FP principles.
- F(x)= x+1. In procedural or object-oriented programming paradigms, iterate over the list and then increment each element. But in functional programming, the computation basically

Q&A: Scala (not R or Python) can be used for GraphX, the ApacheSpark graph processing engine.

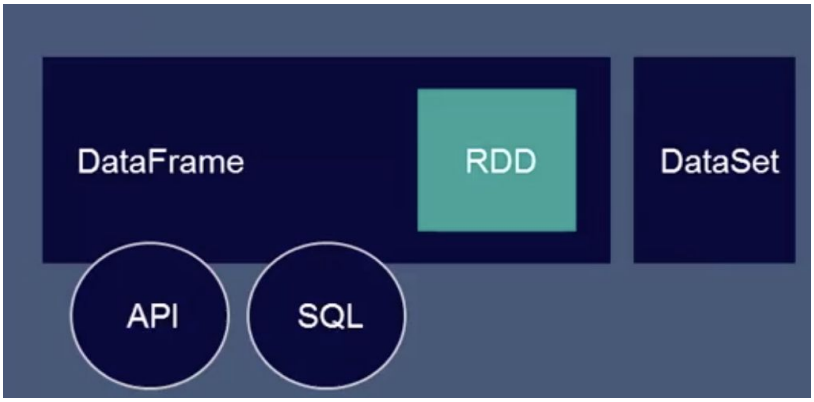
Introduction of Cloudant

- Horizontally scalable.

Q&A: What happens if in a globally distributed and replicated Cloudant environment writing to one replica fails? Writing to the rest of the replicas still is performed. Through usage of internally revision IDs the failed replica is eventually catching up and integrate this write operation into its own replica. That's called "eventual consistency".

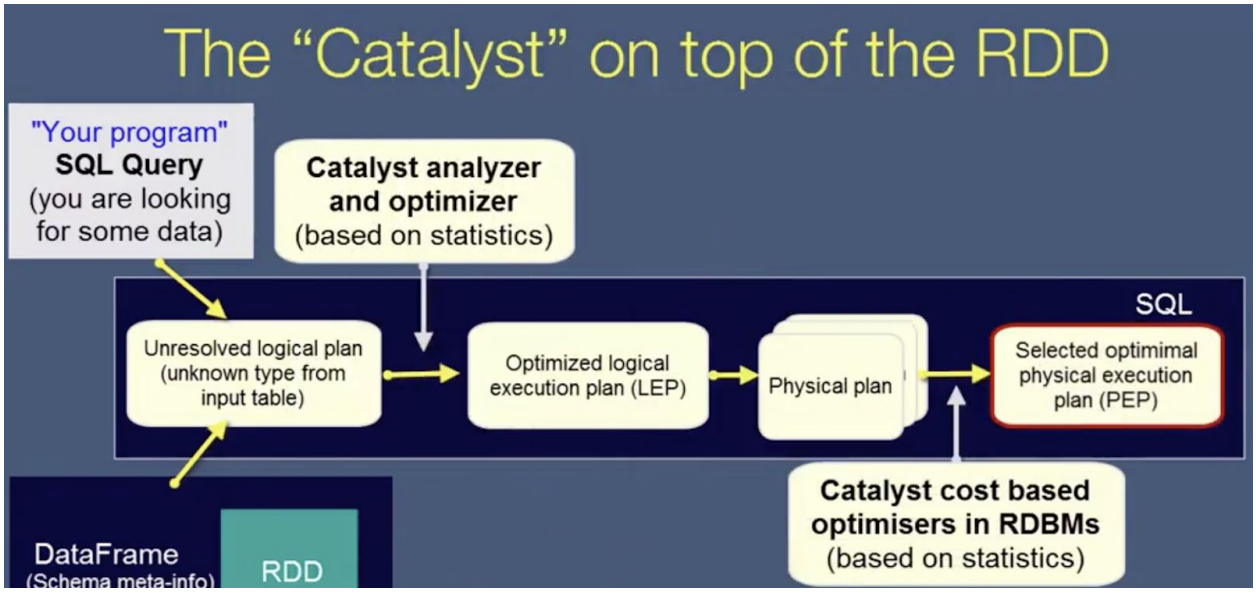
- Ideal for storage of IOT sensor data
- Extraordinary high ingestion rates possible
- Storage cost is relatively low
- Dynamic scaling to cope with a changing resource demand

ApacheSparkSQL



Q&A: which is faster between SQL and the DataFrame API? Same. Calls to the DataFrame API as well as SQL statements are converted to the exactly same internal representation, therefore it doesn't matter what you are using. In addition you can also mix and match those.

- RDD are schema less (schema on read)
- Schema made up by columns names, and columns data types
- An execution plan in form of a tree/ catalyst optimizer



- JVM (Java Virtual Machine) is a general purpose bytecode execution engine

- Using JVM objects and the internal JVM memory manager called Garbage Collector, induces some overhead. Since the GC might estimated wrong on optimal storage. But ApacheSpark knows better which part should be kept in memory and what should already be removed.
- Project Tungsten: improved memory management for improved performance
- ApacheSparkSQL is always preferred over usage of RDDs since the Catalyst Optimizer is doing a very good job optimizing the internal calls to the RDD API.
- ApacheSpark supports SQL via dataframe API
- Internally still RDDs are used
- Performance benefits through Catalyst & Tungsten
- The data is created on a LOT device sent through the IBM Watson LOT platform and stored in Cloudant. The database, as a service offering from IBM, for Apache CouchDB NoSQL database.

Overview of how the test data has been generated

- Flow 1 contains a NodeRED application to simulate IoT sensor data. This flow is running in the cloud. In a real scenario, where would this flow normally run? On a IoT Gateway/Device.

Q&A:

- Cloudant:
    - Cloudant is based on ApacheCouchDB;
    - Cloudant is a NoSQL database;
    - Cloudant is meant for storing JSON documents effectively;
    - BigCouch is a component between the client and a set of CouchDB services used for horizontal scaling
- How does the Catalyst optimizer work internally? A LEP(Logical Execution Plan) is created from a SQL AST. This LEP is transformed (optimised). Then multiple PEP's are created from the optimised LEP. Finally, based on cost based statistics an optimal PEP (Physical Execution Plan) is chosen to be executed.
- What's the advantage of using ApacheSparkSQL over RDDs?
    - Catalyst and Tungsten are able to optimise the execution, so are more likely to execute more quickly than if you would have implemented something equivalent using the RDD API.

- The API is simpler and doesn't require specific functional programming skills.

IBM Watson Studio

Week Three: Mathematical foundation of exploratory data analysis

Statistical moments

- 1st: mean / average / median

- 2nd: standard deviation / variance=deviation square

- 3rd: skewness
  - How asymmetric data is spread around the mean
  - Right tail: positivel; left tail: negative
  - $skewness = \frac{1}{n}\frac{\sum_{j=1}^{n}(x_j - \bar{\bar{x}})^3}{s^3}$, s is standard deviation
- 4th: kurtosis
  - Reports on the shape of the data
  - Higher: longer the tails of the distributions are
  - Indicates outlier content within the data. The more outliers are present in examples of fuel consumption where engines didn't fail, the higher the false positive rate gets when choosing a threshold value which is too low.
  - $kurtosis = \frac{1}{n}\frac{\sum_{i=1}^{n}(x_i - \bar{x})^4}{s^4}$

Covariance, covariance matrices, correlation

- Covariance:
  - $cov(X,Y) = \frac{1}{n}\sum_{i=1}^{n}(x_i - \bar{x})(y_i - \bar{y})$
  - X, Y are vectors
  - ```
    rddXY = rddX.zip(rddY)
    covXY = rddXY.map(lambda (x,y) : (x-meanX)*(y-meanY)).sum()/rddXY.count()
    covXY
    ```
- Correlation
  - $corr(X,Y) = \frac{cov(X,Y)}{\sigma_X \sigma_Y}$, sigma is standard deviation
  - Statistics.corr(dataset)
    ```

Multidimensional vector spaces

Week Four: Data Visualization

Plotting with ApacheSpark and python's matplotlib

- Sampling
  - Subset of original data
  - Due to inherent randomness, preserves most properties of original data
  - Reduces computational cost
- Box plot
  - Mean, standard deviation, skew, outliers

```
result = spark.sql("select voltage from washingflat where voltage is not null")
result_array = result.rdd.map(lambda row : row.voltage).sample(False,0.1).collect()
result_array
```

```
[237,
 232,
```

```
%matplotlib inline
```

```
import matplotlib.pyplot as plt
plt.boxplot(result_array)
plt.show()
```

What is the meaning of the following line?

```
1  %matplotlib inline
```

- Automatically sample the data so that the memory footprint is lower
- Please display all plots created by matplotlib as images under the cell

Correct

- Run charts

```
result = spark.sql("select voltage,ts from washingflat where voltage is not null order by ts asc")
result_rdd = result.rdd.sample(False,0.1).map(lambda row : (row.ts,row.voltage))
result_array_ts = result_rdd.map(lambda (ts,voltage): ts).collect()
result_array_voltage = result_rdd.map(lambda (ts,voltage): voltage).collect()
result_array_ts
```

```
plt.plot(result_array_ts,result_array_voltage)
plt.xlabel("time")
plt.ylabel("voltage")
plt.show()
```

Dimensionality reduction

PCA removes information of dimensions which have a high correlation with each other because information content of highly correlated dimension is lower as with weakly correlated ones.

PCA removes the most irrelevant dimensions by creating a lower dimensional data set with newly created columns but preserving distance properties between individual rows

$$loss = sse(D, PCA^{-1}(PCA(D)))$$

$$sse(X,Y) = \frac{1}{n}\sqrt{\sum_{i=1}^{n}(X_i - Y_i)^2}, X = X_1, X_2, .., X_N, X_n = x_n^1, x_n^2, .., x_n^d$$

PCA

Advantage using Spark Machine Learning over using scikit-learn: Spark Machine Learning runs in parallel on a cluster

model = pca.fit(features)

// calculate the transformation rules necessary to perform transformation.

model.transform(features) // transformation

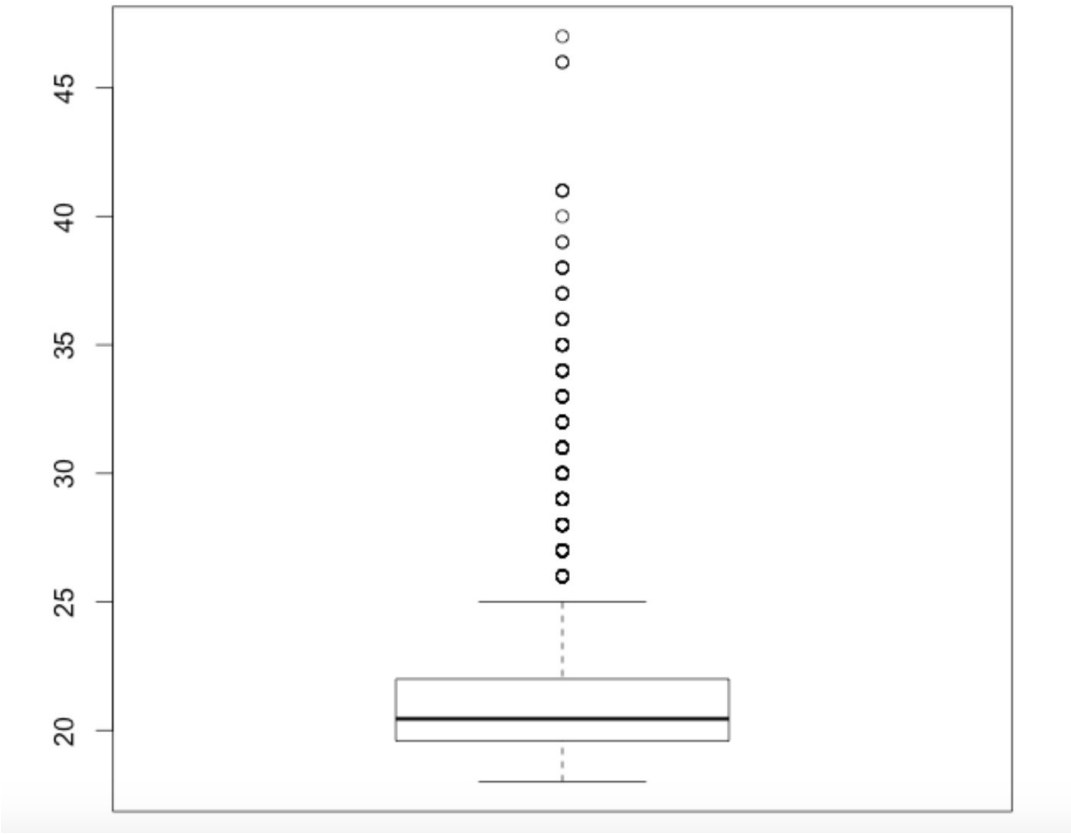
1. Among dimension reduction, what techniques exist to visualize more than three dimensions in a single plot?

- Color coding for discrete dimensions
- Using different symbols apart from simple points for discrete dimensions
- Add multiple lines to a run chart

2. What properties are true about PCA for dimension reduction?

- PCA is a linear transformation, this means most of the original properties of your data are preserved
- PCA transforms your dataset so that the first k dimensions have the lowest correlation among each other
- PCA dimensionality reduction is lossy

3. Plot



- This is a box plot
- There are more than 15 outliers present
- Kurtosis is >1
- Skew is >1
- >50% of all values are < 30
- Mean is >20