CS 744: SPARK SQL

Shivaram Venkataraman

Fall 2019

ADMINISTRIVIA

- Assignment 2 grades this week
- Midterm details on Piazza
- Course Project Proposal comments

Bismarck

PS

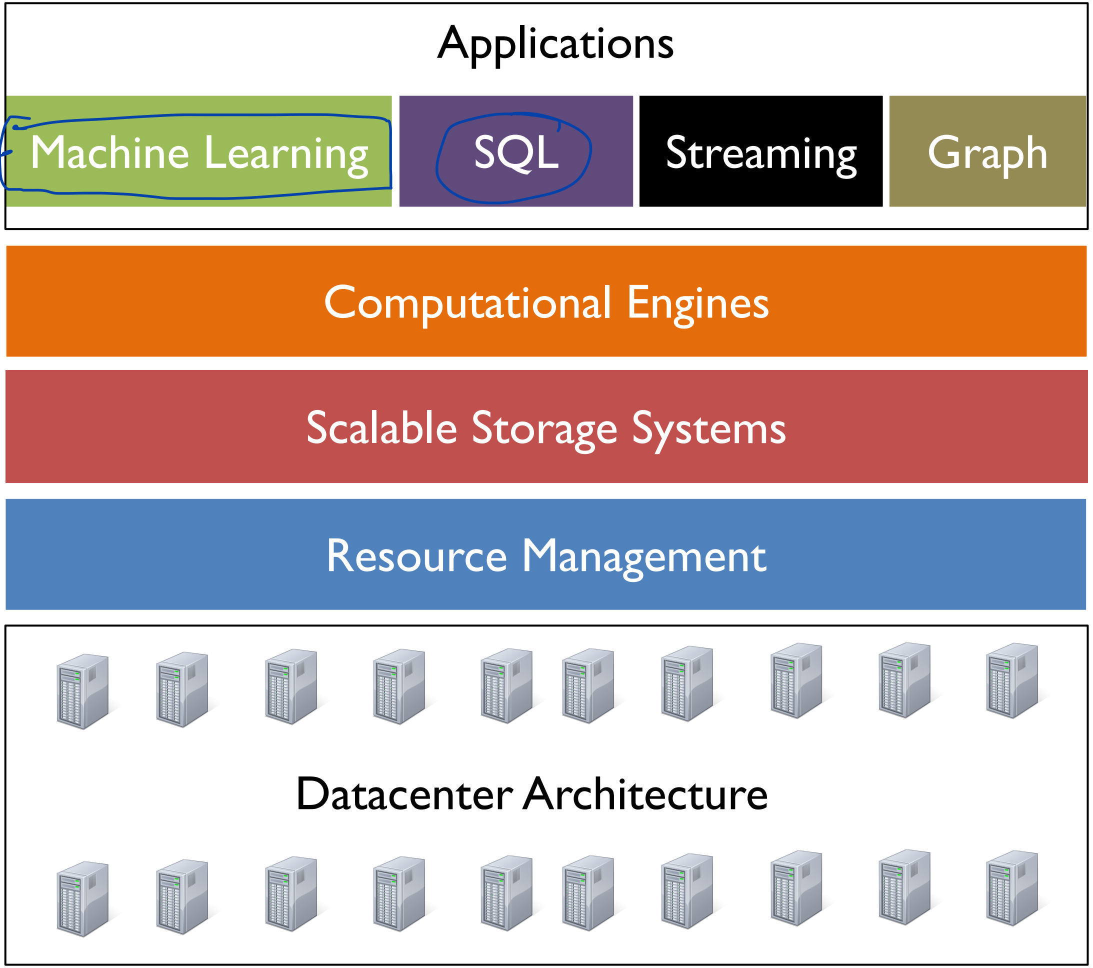
Tensorflow

Ray

GFS

Mesos

DRF



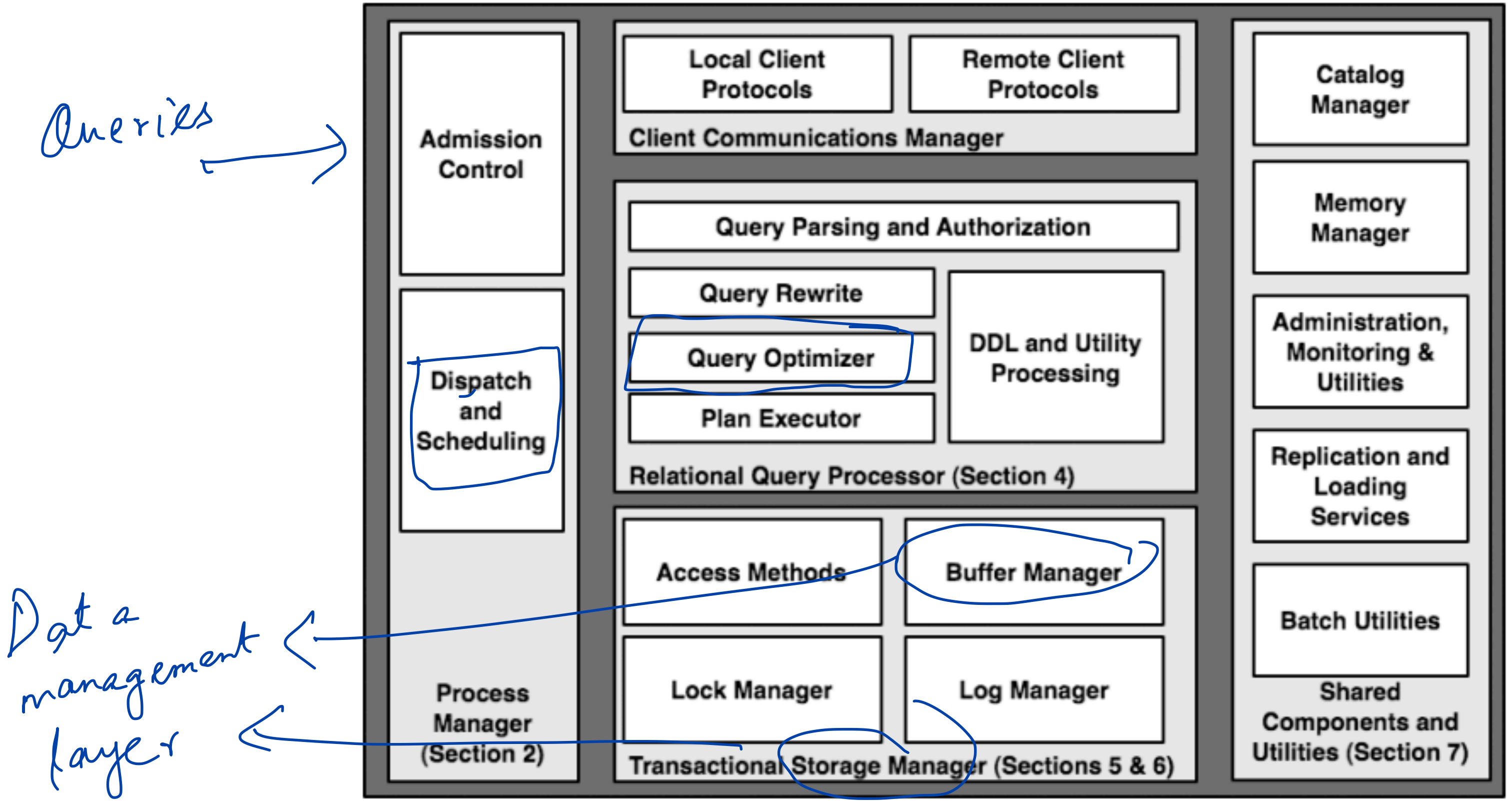
MapReduce

Spark

Gandiva

SQL: STRUCTURED QUERY LANGUAGE

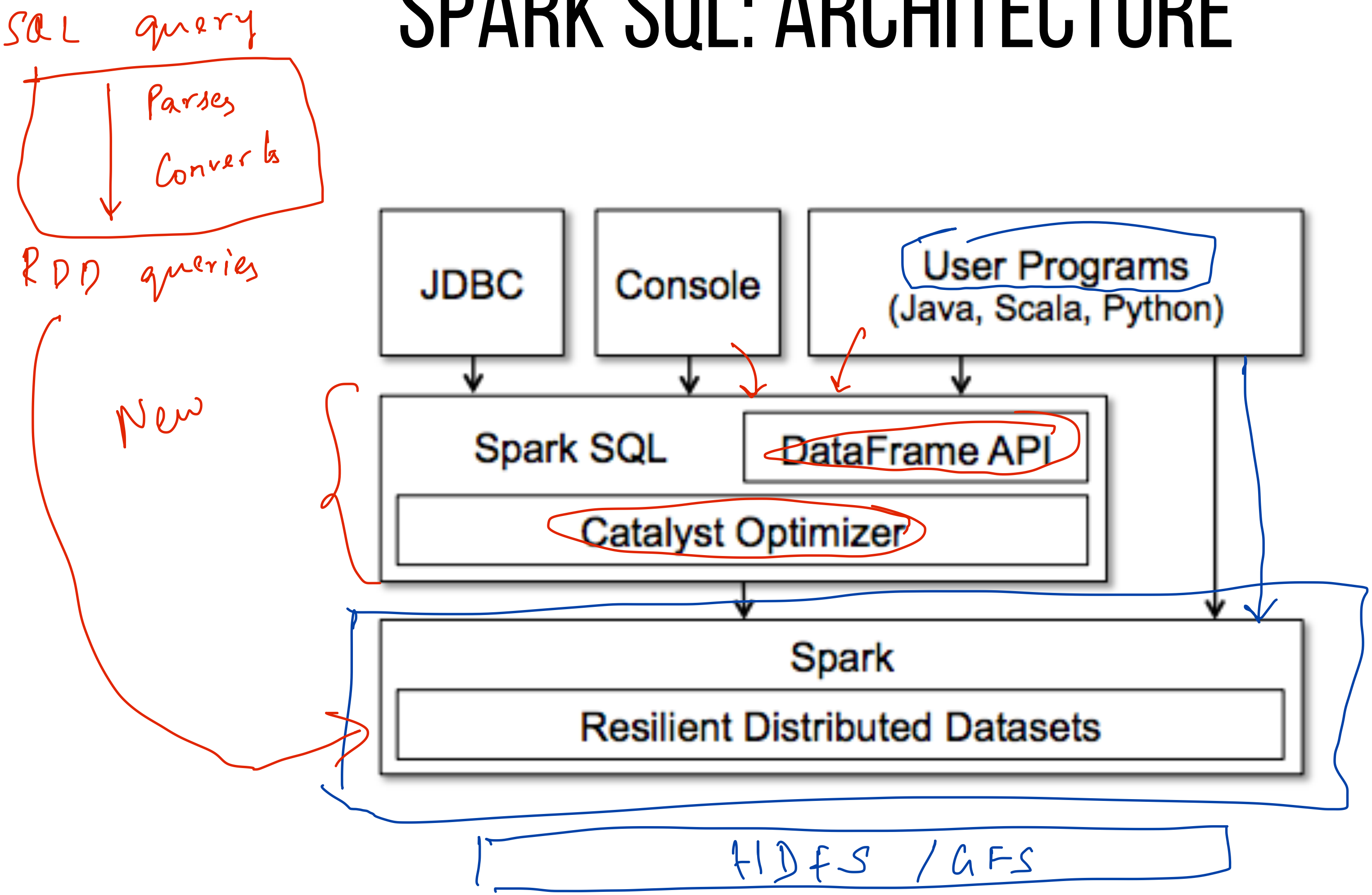
DATABASE SYSTEMS



SQL IN BIG DATA SYSTEMS

- Scale: How do we handle large datasets, clusters ?
- Wide-area: How do we handle queries across datacenters ?

SPARK SQL: ARCHITECTURE



DATAFRAME

name	age

Motivation: Understanding the structure of data

```
lines = sc.textFile("users")
csv = lines.map(x =>
    x.split(',') )
young = csv.filter(x =>
    x(1) < 21)
println(young.count())
```

How many (columns/entries) in one row

Type of the columns

PROCEDURAL VS. RELATIONAL

no parsing

```
lines = sc.textFile("users")
csv = lines.map(x =>
    x.split(','))
young = csv.filter(x =>
    x(1) < 21)
println(young.count())
```

Creates table
DataFrame

```
ctx = new HiveContext ()
users = ctx.table("users")
young = users.where(
    users("age") < 21)
println(young.count())
```

Column name

System can analyze expressions passed here

OPERATORS → EXPRESSIONS

Projection (select), Filter, Join, Aggregations take in Expressions

```
employees.join(dept,
   employees (“deptId") === dept ("id ")
)
```

restricted set of operators

Build up Abstract Syntax Tree (AST)

OTHER FEATURES

1. Debugging: Eager analysis of logical plans

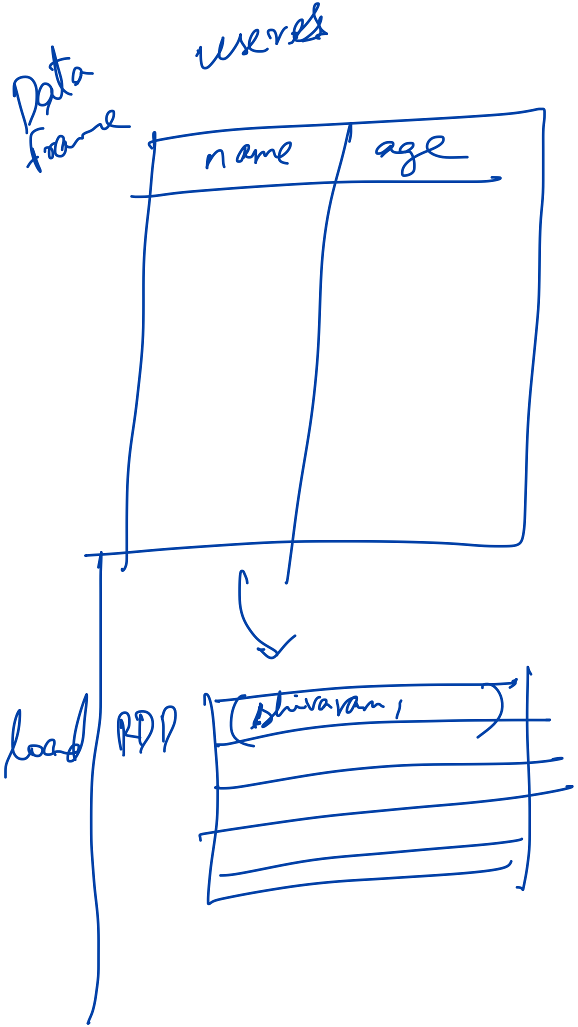
user("address") == "101"

2. Interoperability: Convert RDD to Dataframes

Relational : optimizations, lesser code

Procedural : ETL : extract, transform, load

custom code



OTHER FEATURES

DF : users : (String, Int)

Row

Array

100 rows

W0

W1

(Array Str , Array Int)

Int   Delta

Int	Delta
21	21
24	+3
22	-2
21	-1
23	+2

3. Caching: Columnar caching with compression

Parquet files from HDFS

↳ File format

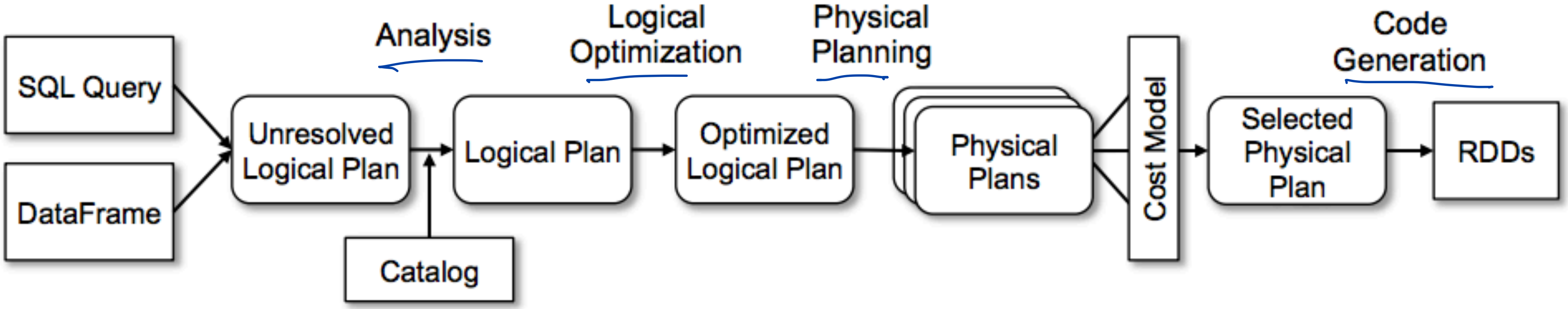
4. UDFs: Python or Scala functions

```
val model: LogisticRegressionModel = ...
ctx.udf. register (" predict", (x: Float , y: Float) =>
model.predict(Vector(x, y)))
ctx.sql (" SELECT predict(age , weight) FROM users ")
```

Better Integration with language feature

CATALYST

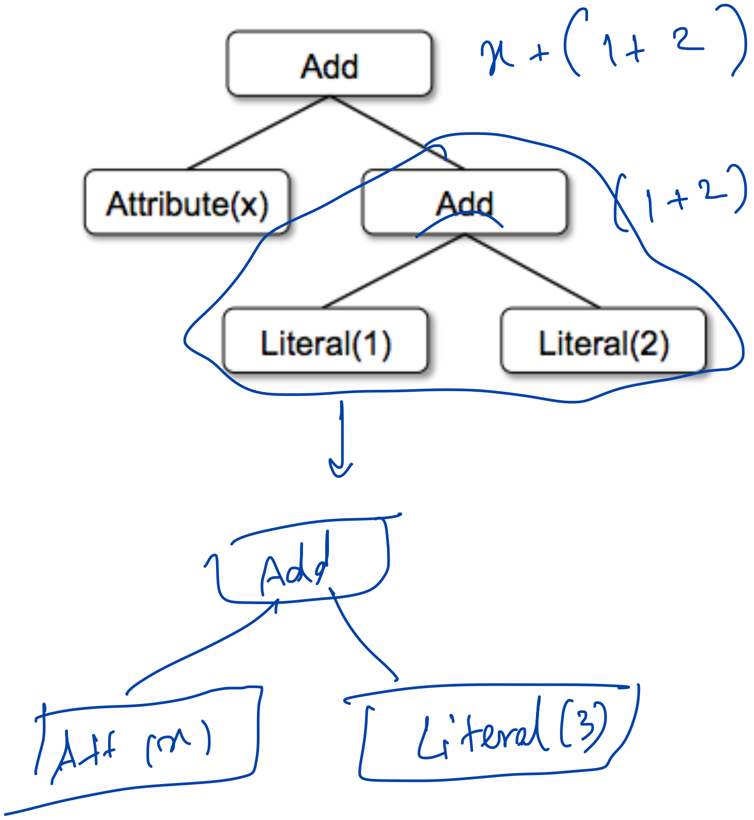
Goal: Extensibility to add new optimization rules



CATALYST DESIGN

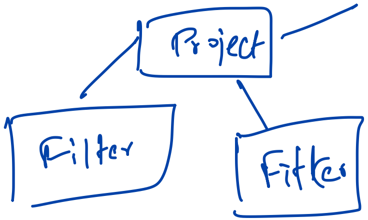
Library for representing trees and rules to manipulate them

```
tree. transform {
   case Add(Literal(c1),Literal(c2)) =>
Literal(c1+c2)
   case Add(left , Literal(0)) => left
   case Add(Literal(0), right) => right
}
```



C-store

LOGICAL, PHYSICAL PLANS



1. Analyzer: Lookup relations, map named attributes, propagate types
2. Logical Optimization

   a. Predicate pushdown: reduce num rows that need to be processed

   b. Project pruning: reduce cols that need to be processed

   Select name, age, UDF(name)
   from users
   where $\boxed{\text{age} < 21}$

3. Physical Planning

   → Join selection: Hash, Broadcast, Sort-merge → Cost based opt.

   → Pipelining filters, projections : map(Filter, Project)

CODE GENERATION

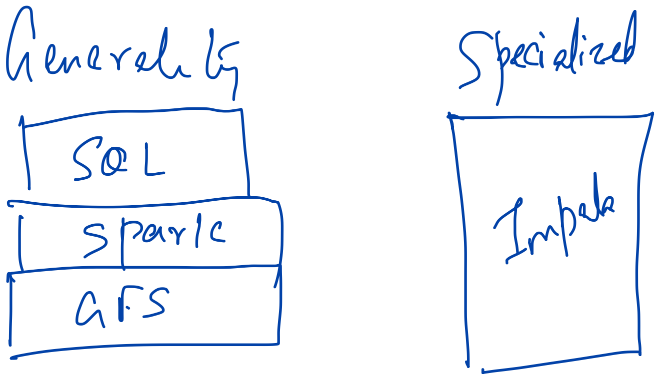
CPU bound when data is in-memory

Branches, virtual function calls etc.

```
def compile(node: Node ): AST = node match {
  case Literal(value) => q"$value"
  case Attribute (name) => q"row.get($name)"
  case Add(left, right) =>
    q"${compile(left)} + ${compile(right)}"
}
```

branch statements

EXTENSIONS



Data sources

- Define a `BaseRelation` that contains schema
- `TableScan` returns `RDD[Row]`
- Pruning / Filtering optimizations

User-Defined Types (UDTs)

- Support advanced analytics with e.g. Vector
- Users provide mapping from UDT to Catalyst Row

SUMMARY, TAKEAWAYS

Relational API

- Enables rich space of optimizations
- Easy to use, integration with Scala, Python

Catalyst Optimizer

- Extensible, rule-based optimizer
- Code generation for high-performance

Evolution of Spark API

DISCUSSION

https://forms.gle/r6DnV7wLGHjYmYd17

Does SparkSQL help ML workloads? Consider the MNIST code in your assignment. What parts of your code would benefit from SparkSQL and what parts would not?

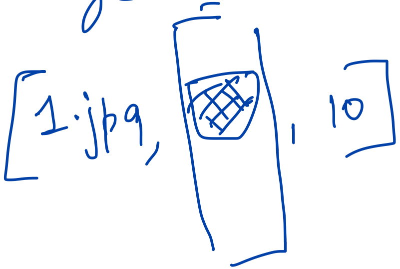
1. Things not in ML workload

- Pre-processing
- ETL
- Sampling

2. Batch operation on columns

↳ Cache misses on row vs. column storage

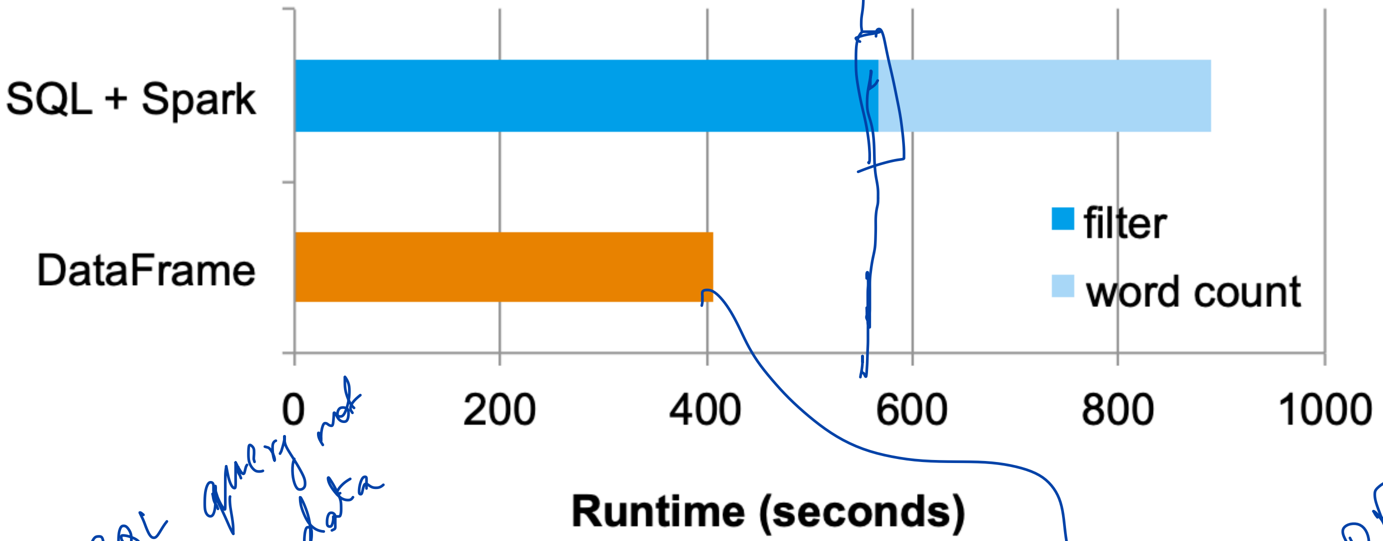
↳ Does it help compression



3. Eager analysis: Matrix sizes, Indexing

2 stage pipeline

storing data into FS



1 stage

1. SQL query not using column data

2. Query opt
↳ Total data saved to disks is smaller

run more queries on DF

What are some limitations of the Catalyst optimizer as described in the paper? Describe one or two ideas to improve the optimizer

2. Cost model

General purpose fns on DF

Improve Columnar storage ?

or

Choose storage format

NEXT STEPS

Next class: Wide-area SQL queries

Midterm coming up!

SCHEMA INFERENCE

Common data formats: JSON, CSV, semi-structured data

JSON schema inference

- Find most specific SparkSQL type that matches instances
  e.g. if tweet.loc.latitude are all 32-bit then it is a INT
- Fall back to STRING if unknown
- Implemented using a reduce over trees of types