Spark I

Big Data Management

UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH

DTIM
www.essi.upc.edu/dtim

Knowledge objectives

1. Name the main Spark contributions and characteristics
2. Compare MapReduce and Spark
3. Define a dataframe
4. Distinguish dataframe from relation and matrix
5. Distinguish Spark and Pandas dataframe
6. Enumerate some abstraction on top of Spark

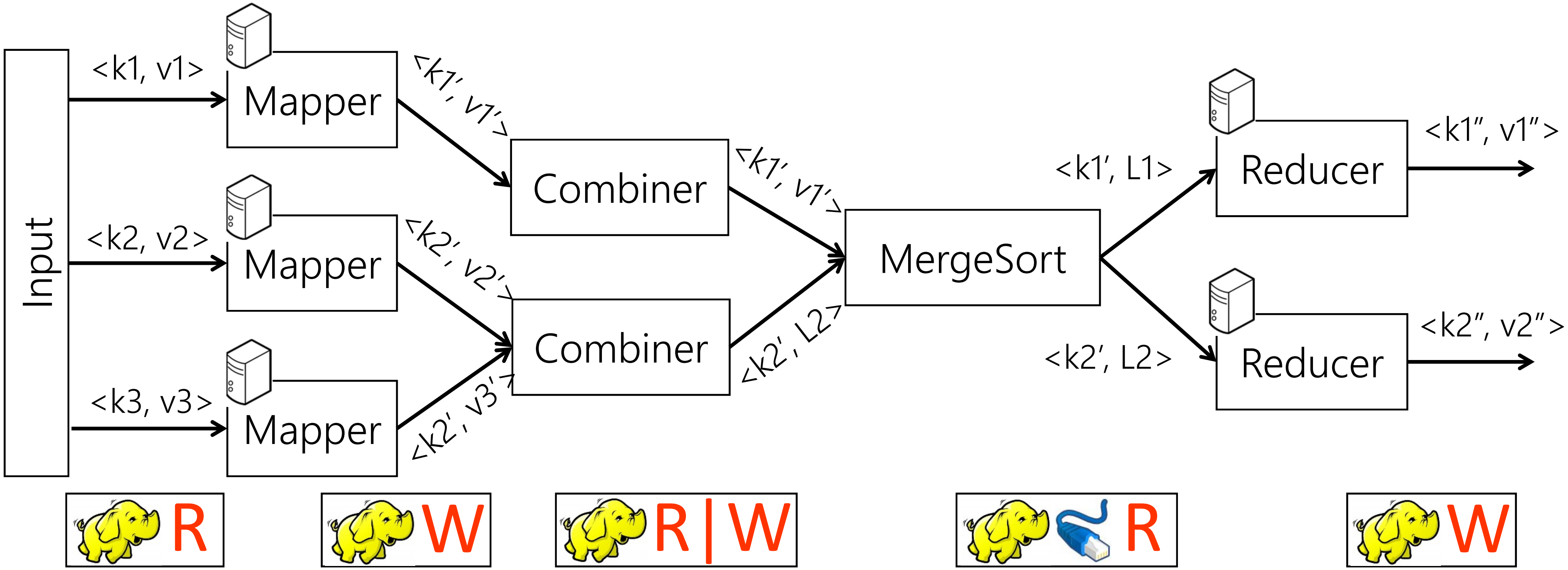
Application Objectives

- Provide the Spark pseudo-code for a simple problem using dataframes

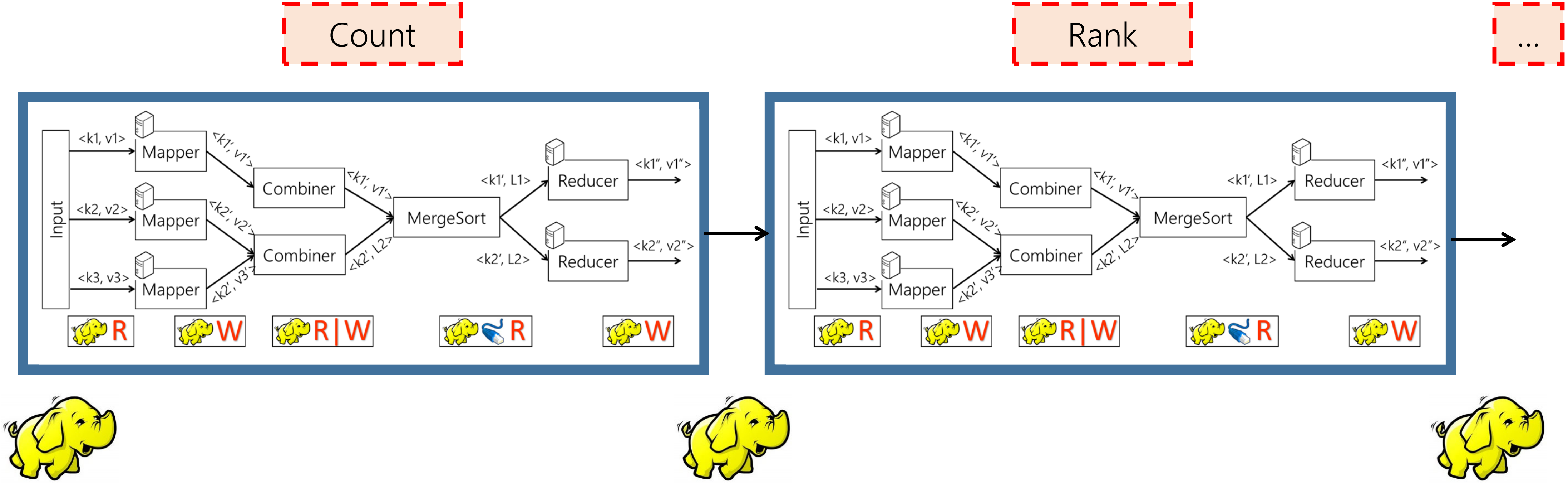
Background

MapReduce limitations

MapReduce intra-job coordination



MapReduce inter-job coordination



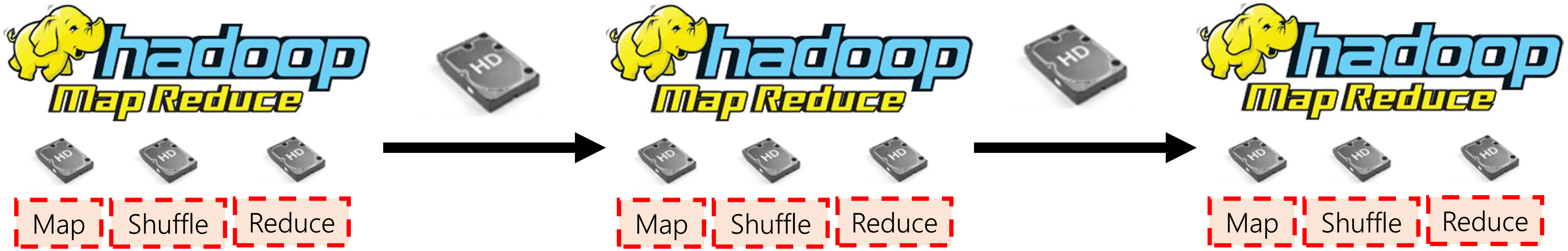
MapReduce limitations

Spark is not in memory processing. It also uses the disk but

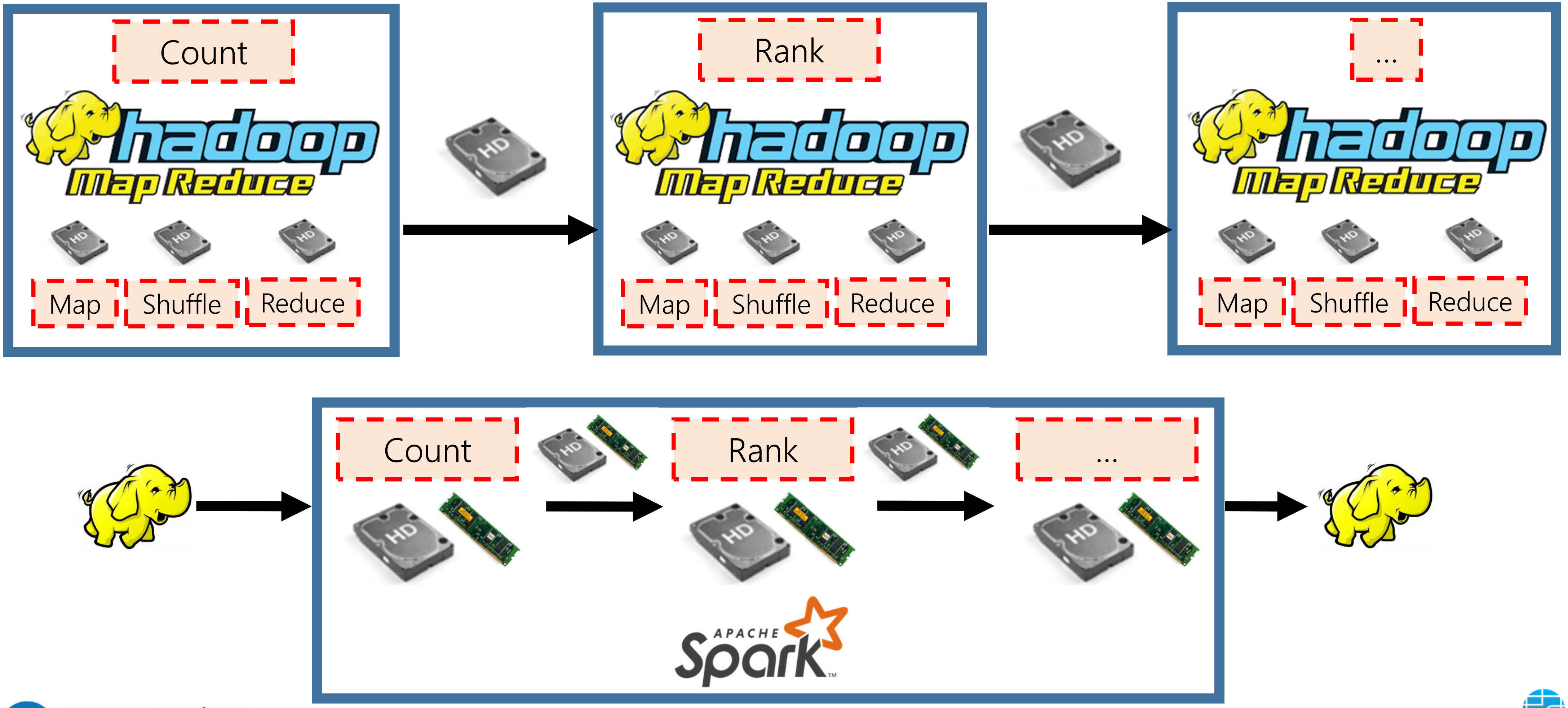
1. No need to chain multiple jobs
2. In one job, we can implement different operators like counting, ranking
3. So, we only read once at HDFS at the beginning of the long job and write it at the end of job.
4. Has choice between memory and disk. Tries to avoid disk as much as possible. This is the reason Spark is superior to Map Reduce.

- Coordination between phases using DFS
  - Map, Shuffle, Reduce
- Coordination between jobs using DFS
  - Count, rank, aggregate, …

Constantly using disk again and again



Main memory coordination in Spark



Count

Map

Shuffle

Reduce

Rank

Map

Shuffle

Reduce

…

Map

Shuffle

Reduce

Count

Rank

…

Dataframes

Problems of relational tables in data exploration

- Schema needs to be defined before examining the data
- Not well-structured data is difficult to query
- Generating queries requires familiarity with the schema
- Complex declarative queries are hard to debug
  - SQL was not conceived for REPL (Read, Evaluate, Print Loop)

Characteristics of dataframes

- First introduced to S in 1990
- Symmetrical treatment of rows and columns
  - Both can be referenced explicitly
    - By position (data is ordered row- and column-wise)
    - By name
- Data has to adhere to a schema
  - Defined at runtime
    - Useful for data cleaning
- A variety of operations
  - Relational-like (e.g., filter, join)
  - Spreadsheet-like (e.g., pivot)
  - Linear algebra (e.g., multiply)
- Incrementally composable query syntax
- Native embedding in an imperative language

Relation, Dataframe and Matrix

Rows unnamed unordered

Rows named ordered

Relation

R

T_1	…	T_n
A_1	…	A_n

Original dataframe

	T_1	T_2	…	T_n
	$1/A_1$	$2/A_2$	…	n/A_n
$1/r_1$	x	“x”	…	T
$2/r_2$	y	“y”	…	F
…	…	…		…
m/r_m	z	None	…	T

Matrix

Numeric

$$\begin{array}{c|cccc} & 1 & 2 & \dots & n \\ \hline 1 & a_{11} & a_{12} & \dots & a_{1n} \\ 2 & a_{21} & a_{22} & \dots & a_{2n} \\ \dots & \dots & \dots & & \dots \\ m & a_{m1} & a_{m2} & \dots & a_{mn} \end{array}$$

Spark dataframe

T_1	T_2	…	T_n
$1/A_1$	$2/A_2$	…	n/A_n
x	“x”	…	T
y	“y”	…	F
…	…		…
z	None	…	T

No row identifier no order

Spark Dataframe definition

"A Dataset is a strongly typed collection of domain-specific objects that can be transformed in parallel using functional or relational operations."

"A Dataframe is an immutable collection of data organized into named columns, potentially distributed in the nodes of a cluster. It is implemented as an indexed Dataset of Rows."

- Resembles a Relational table
- Row class does not fix a schema at compile time, but at execution time
  - Uses StructType
  - Allows to infer schemas from the file (e.g., CSV or JSON)
- Can be partitioned and distributed
  - Implemented on top of Resilient Distributed Datasets (RDD)

https://spark.apache.org/docs/latest/api/java/org/apache/spark/sql/Dataset.html
https://spark.apache.org/docs/latest/api/java/org/apache/spark/sql/Row.html
https://spark.apache.org/docs/latest/api/java/org/apache/spark/sql/types/StructType.html

Dataframe vs Matrix/Array/Tensor

Dataframe	Matrix
Heterogeneously typed	Homogeneously typed
Both numeric and non-numeric types	Only numeric types
Explicit column names (also row in Pandas)	No names at all
Supports relational algebra	Does not support relational algebra

Filter, Project, Join

D. Petersohn, et al. *Towards Scalable Dataframe Systems. Proc. VLDB Endow. 13(11), 2020*

Dataframe vs Relation/Table

can point to position of row

primary key different case

<table>
<tr><th>Pandas Dataframe</th><th>Spark Dataframe</th><th>Relation</th></tr>
<tr><td>Ordered</td><td colspan="2">Unordered</td></tr>
<tr><td>Named rows</td><td colspan="2">Unnamed rows</td></tr>
<tr><td colspan="2">Lazily-induced schema</td><td>Rigid schema</td></tr>
<tr><td>Column-row symmetry</td><td colspan="2">Columns and rows are different</td></tr>
<tr><td>Supports linear algebra</td><td colspan="2">Does not support linear algebra</td></tr>
</table>

Mathematical operation on vector matrix

D. Petersohn, et al. *Towards Scalable Dataframe Systems. Proc. VLDB Endow. 13(11), 2020*

Dataframe implementations

Execution happens immediately

Pandas	Spark
Eager evaluation of transformations	Lazy evaluation of transformations
Resides in memory	Requires a SparkSession
Not scalable (multithread operators exist, but manual split is required)	Transparently scalable in the Cloud
Transposable	Non-transposable (problems with too many rows)

D. Petersohn, et al. *Towards Scalable Dataframe Systems. Proc. VLDB Endow. 13(11), 2020*

Spark dataframe operations

a) Input/Output

b) Transformations Modifying data and generating new dataframe

- Lower abstraction: O-O interface (similar to RDDs)
    - Functions over columns
- Higher abstraction: SQL

c) Actions

d) Schema management

Input/Output

- Matrix
- Pandas dataframe
- CSV
- JSON
- RDBMS
- HDFS file formats:
  - ORC
  - Parquet

We can create data frame from each one of these
You cannot modify dataframe, read transform write modified data

Transformations available

- select select columns you want
- filter/where
- sample sample data
- distinct/dropDuplicates
- sort
- replace replace does not modify dataframe
- groupBy+agg
- union/unionAll/unionByName
- subtract difference
- join
- …

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/dataframe.html

Functions over columns

Make transformation in content of column

- Normal (*lit, isNull, …*) add a new column to the df by assigning a literal or constant value
- Math (*sqrt, sin, ceil, log, …*)
- Daytime (*current_date, dayofweek, …*)
- Collection (*array_sort, forall, zip_with, …*)
- Aggregate (*avg, count, first, corr, max, min, …*)
- Sort (*asc, desc, asc_nulls_first, …*)
- String (*length, lower, trim, …*)

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/functions.html

Actions available

triggers execution of dataframe

yo aayepachi exeucte huncha transformation
agadi lazy wala huncha

- count
- first
- collect ☠ collect all rows
- take/head/tail
- show
- write write to file
- toPandas ☠ transform to pandas
- …

dataframe reside in cloud
at the moment we execute collect and toPandas, TB of data is brought
to local machine, for testing okay, not in production

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/dataframe.html

Schema operations

- summary/describe
- printSchema
- columns Print available columns

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/dataframe.html

Optimizations

Checkpointing
1. More expensive
2. Goes to disk
3. Discards all the execution history (lineage is discarded)
    1. Because everything is stored in disk
4. Guarantees even in case of failure, it is possible to recover data (Not the case of caching)

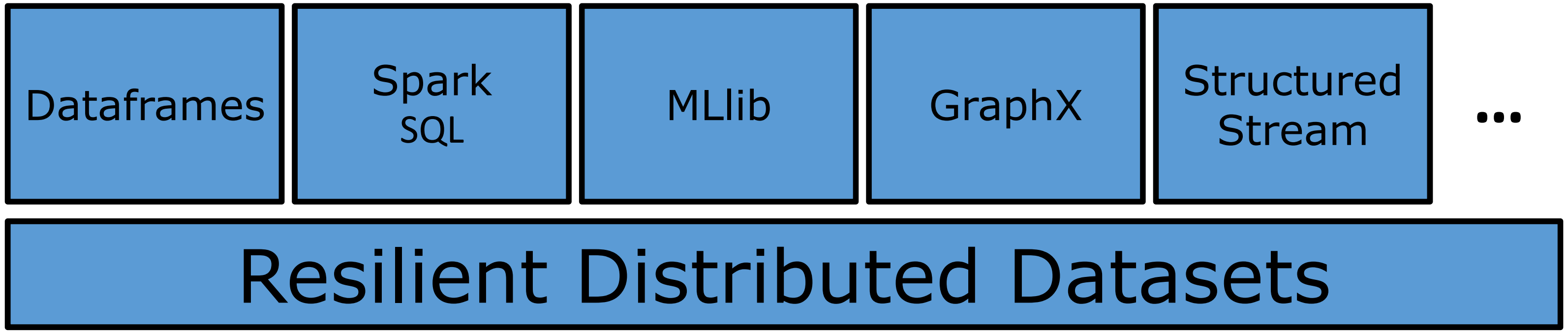
- Lazy evaluation
  - cache/persist Guarantees if everything goes well, we can recover the data without re-executing
    - unpersist
  - checkpoint no lineage
- Parallelism is based on number of partitions we have in the system and more resources are consumed
  - repartition/coalesce

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/dataframe.html

Abstractions

Spark Abstractions

Resilient Distributed Dataset is baseline to build --- on top of it



Spark SQL

- Besides the O-O interface of dataframes, there is a declarative one
- It can be used independently of the kind of source
  - Not only Relational tables
- It is translated into functional programming by an optimizer
  - Based on
    a) Rules
      - Predicate push down
      - Column pruning

    b) Cost model
  - Extensible

https://spark.apache.org/sql

Spark SQL interface

- There is a catalog with all tables available

    SparkSession.catalog

- Dataframes are registered as views in the catalog

    DataFrame.createOrReplaceTempView(<tablename>)

- Queries:

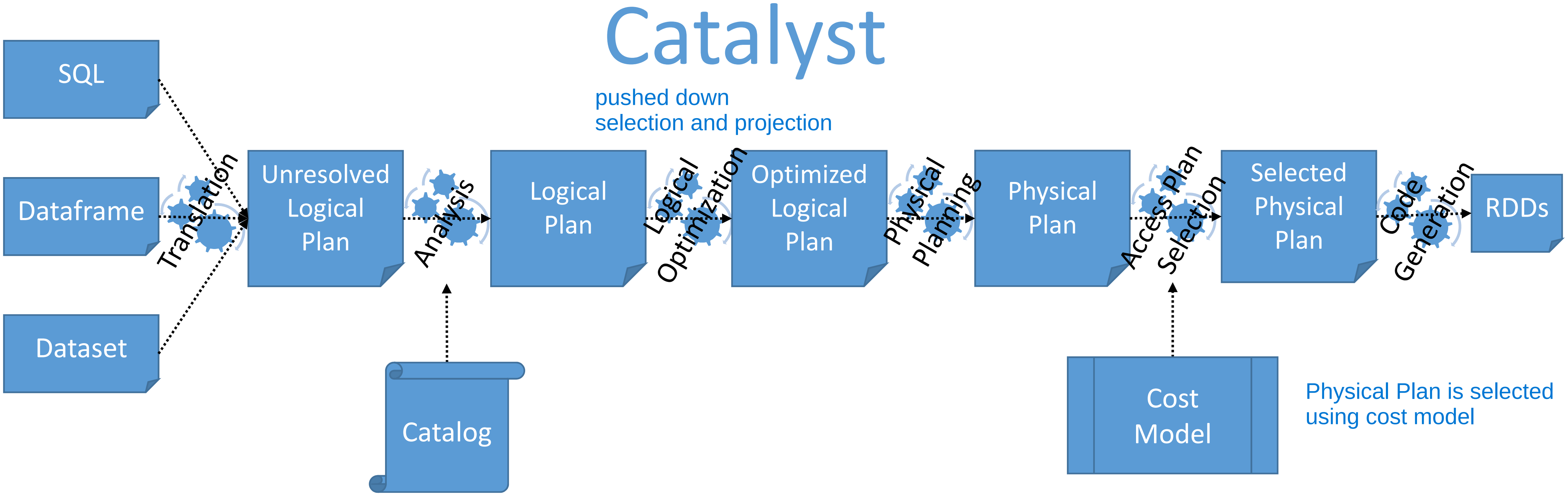
    SparkSession.sql(<query>)

    - Input is simply a string
    - Output is a dataframe

https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/catalog.html

Shared Optimization and Execution

Optimizer is called catalyst



Physical Plan is selected using cost model

Closing

Summary

- Overcoming MapReduce limitations
- Dataframes
  - Comparison
    - Differences with Relations
    - Differences with Matrixes
    - Differences in Pandas and Spark
  - Operations
    - Transformations
    - Actions
  - Optimizations
    - Lazy evaluation
    - Parallelism
- Abstraction

References

- H. Karau et al. *Learning Spark*. O'Really, 2015
- D. Petersohn, W. W. Ma, D. Jung Lin Lee, S. Macke, D. Xin, X. Mo, J. Gonzalez, J. M. Hellerstein, A. D. Joseph, A. G. Parameswaran. *Towards Scalable Dataframe Systems. Proc. VLDB Endow. 13(11), 2020*
- A. Hogan. *Procesado de Datos Masivos* (Universidad de Chile). http://aidanhogan.com/teaching/cc5212-1-2020