U O W

Data Processing with Apache Spark

– Handling Massive Data Set (Part I)

CSCI316: Big Data Mining Techniques and Implementation

UNIVERSITY
OF WOLLONGONG
AUSTRALIA

Contents

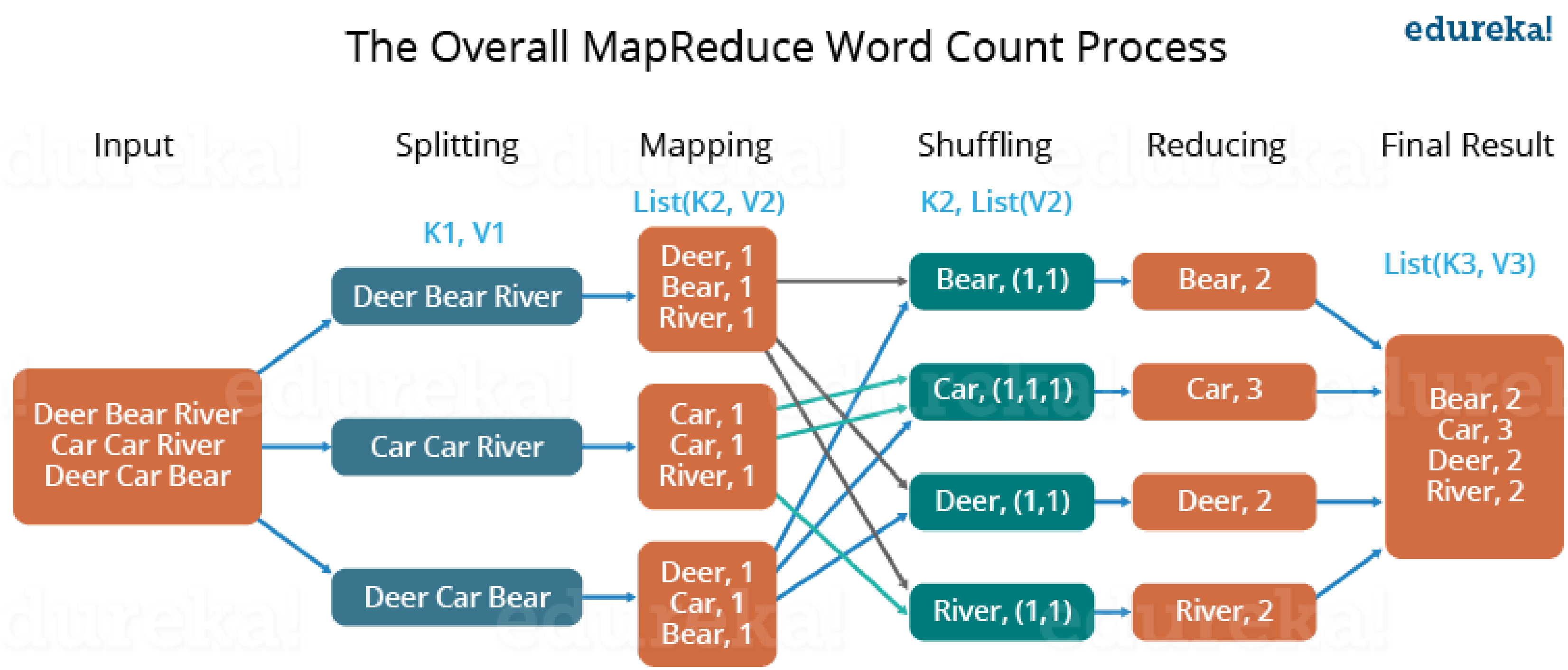
MapReduce and Spark Model

Processing Massive Data

- Processing massive datasets requires long runtime
  - One solution to speed up the computation is *parallelism (or distributed computing)*
- **MapReduce:** One preeminent model of parallel computation
- A very simple model:
  - One *Map* stage, which performs simple mapping-alike operations to produce key-value pairs
  - One intermediate stage, which merges key-value pairs per key
  - One *Reduce* stage, which performs aggregation-alike operations per key
- However...
  - from the algorithm point of view: very powerful!
  - from the implementation point of view: very suitable for a computing cluster

MapReduce Workflow: The WordCount Example

- MapReduce uses (key, value) as the basic data structure.



UNIVERSITY OF WOLLONGONG AUSTRALIA

Relational-Algebra Operations in MapReduce

- Although big data frameworks are not traditionally DB systems, **relational-algebra operations** are useful, especially in preprocessing.
- Recall that a *relation* is a table. We call the column headers as *attributes* and the rows as *tuples*. The bag of attributes of a relation is called its *schema*.
- We use $R[A_1, \ldots, A_n]$ to denote a relation R with schema $A_1, \ldots, A_n$.
- **Selections**: Apply a condition C to each tuple in the relation and produce as output only the tuples that satisfy C.
- <u>Computing selections in MapReduce</u>
  - The Map function: For each tuple t in R, test if it satisfies C. If so the mapper produce the key-value pair (t, t); otherwise, it produces nothing.
  - The Reduce function: The identity function.

Relational-Algebra Operations in MapReduce

- **Natural Join**
  - Given two relations, compare each pair of tuples, one from each relation. If the two tuples agree on all the attributes that are common to the two schemas, then produce a tuple that has components for each of the attributes in either schema or both.

A	B
a_1	b_1
a_2	b_1
a_3	b_2
a_4	b_3

R

$\bowtie$

B	C
b_2	c_1
b_2	c_2
b_3	c_3

S

=

A	C
a_3	c_1
a_3	c_2
a_4	c_3

- Computing Natural Join in MapReduce
  - **Map function**. For each tuple (a, b) in $R[A, B]$, produce the key-value pair $b: (R, a)$. For each tuple (b, c) in $S[B, C]$, produces the key-value pair $b: (S, c)$.
  - **Reduce function**. Each key value b will be associated with a list of pairs that are of the form (R, a) or (S, c). Construct all tuples (a, b, c) if both (R, a) and (S, c) appear in the list that b is associated with.

UNIVERSITY OF WOLLONGONG AUSTRALIA

Example

A	B
a_1	b_1
a_2	b_1
a_3	b_2
a_4	b_3

R

$\bowtie$

B	C
b_2	c_1
b_2	c_2
b_3	c_3

S

=

A	C
a_3	c_1
a_3	c_2
a_4	c_3

- Map function
  - From table R.

$$(a_1, b_1) \longmapsto b_1\colon (R, a_1)$$
$$(a_2, b_1) \longmapsto b_1\colon (R, a_2)$$
$$(a_3, b_2) \longmapsto b_2\colon (R, a_3)$$
$$(a_4, b_3) \longmapsto b_3\colon (R, a_4)$$

  - From table S.

$$(b_2, c_1) \longmapsto b_2\colon (S, c_1)$$
$$(b_2, c_2) \longmapsto b_2\colon (S, c_2)$$
$$(b_3, c_3) \longmapsto b_3\colon (S, c_3)$$

Example

- Reduce function
  - $b_1 : (R, a_1), (S, a_2) \longmapsto \text{None}$
  - $b_2 : (R, a_3), (S, c_1), (S, c_2) \longmapsto (a_3, c_1), (a_3, c_2)$
  - $b_3 : (R, a_4), (S, c_3) \longmapsto (a_4, c_3)$

A	B
a_1	b_1
a_2	b_1
a_3	b_2
a_4	b_3

R

$\bowtie$

B	C
b_2	c_1
b_2	c_2
b_3	c_3

S

=

A	C
a_3	c_1
a_3	c_2
a_4	c_3

Relational-Algebra Operations in MapReduce

- You can verify that other all other usual relational-algebra operations, such as projection, union, intersection, grouping and left/right/outer-joins, can be expressed in the model of MapReduce
- **Conclusion:** ***All common SQL queries can be implemented with MapReduce***

Reference: Sec. 2.3.3 – 2.3.8, J. Leskovec, A. Rajaraman, J. Ullman. Mining of Massive Datasets, 3rd Edition

Matrix-Matrix Multiplication in MapReduce

- If $\boldsymbol{M}$ is a matrix with element m_{ij} in row i and column j, and $\boldsymbol{N}$ is a matrix with element n_{jk} in row j and column k, then the product $\boldsymbol{P} = \boldsymbol{MN}$ is the matrix with element p_{ik} in row i and column k, where $p_{ik} = \sum_j m_{ij} n_{jk}$
  - Note that the number of columns of $\boldsymbol{M}$ must equals to the number of row in $\boldsymbol{N}$.
- We can view as a matrix as a *relation* with three attributes: the row number, the column number, and the value in that row and column.
- Thus, $\boldsymbol{M}[I, J, V]$ with tuples (i, j, m_{ij}) and $\boldsymbol{N}[J, K, W]$ with tuples (j, k, n_{jk})
- Adopting this idea, can develop a *two-stage* MapReduce job for Matrix-Matrix Multiplication.

Matrix-Matrix Multiplication in MapReduce

The first pair of MapReduce functions:

- **Map Function A**: For each matrix element m_{ij}, produce the key value pair $j : \left(\mathrm{M}, i, m_{ij}\right)$. Likewise, for each matrix element n_{jk}, produce the key value pair $j : \left(\mathrm{N}, k, n_{jk}\right)$.
- **Reduce Function A**: For each key j, examine its list of associated values. For each value that comes from M, say $\left(\mathrm{M}, i, m_{ij}\right)$, and each value that comes from N, say $\left(\mathrm{N}, k, n_{jk}\right)$, produce a key-value pair with key equal to (i, k) and value equal to the product of these elements, $m_{ij}n_{jk}$.

Matrix-Matrix Multiplication in MapReduce

The second MapReduce performs a grouping and aggregation applied to the output of the first MapReduce.

- The **Map Function B**: This function is just the identity. That is, for every input element with key (i, k) and value v, produce exactly this key-value pair.
- The **Reduce Function B**: For each key (i, k), produce the sum of the list of values associated with this key. The result is a pair $(i, k) : v$, where v is the value of the element in row i and column k of the matrix $\boldsymbol{P} = \boldsymbol{MN}$.

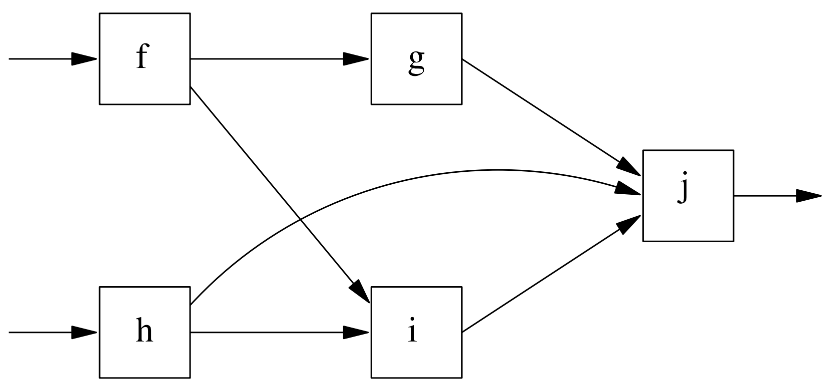
Reference: Sec. 2.3.9, J. Leskovec, A. Rajaraman, J. Ullman. Mining of Massive Datasets, 3rd Edition

The kNN Classifier in MapReduce

- How to write a kNN classifier in MapReduce?
- Recall the movie example in the second lecture.
- The basic idea:
  - Mapper returns **<key1, val1>** where **key1** is a movie name and **val1** is the distance to the unknown movie
  - Reducer returns **<key2, val2>** where **key2** is **null** (not important) and **val2** is a list of k nearest movies (to the unknown movie) and the distances
    - ❖ Can use a combiner to improve the performance (why?)
  - Finally, a voting function is used based on **val2** to determine the class for the unknown movie.

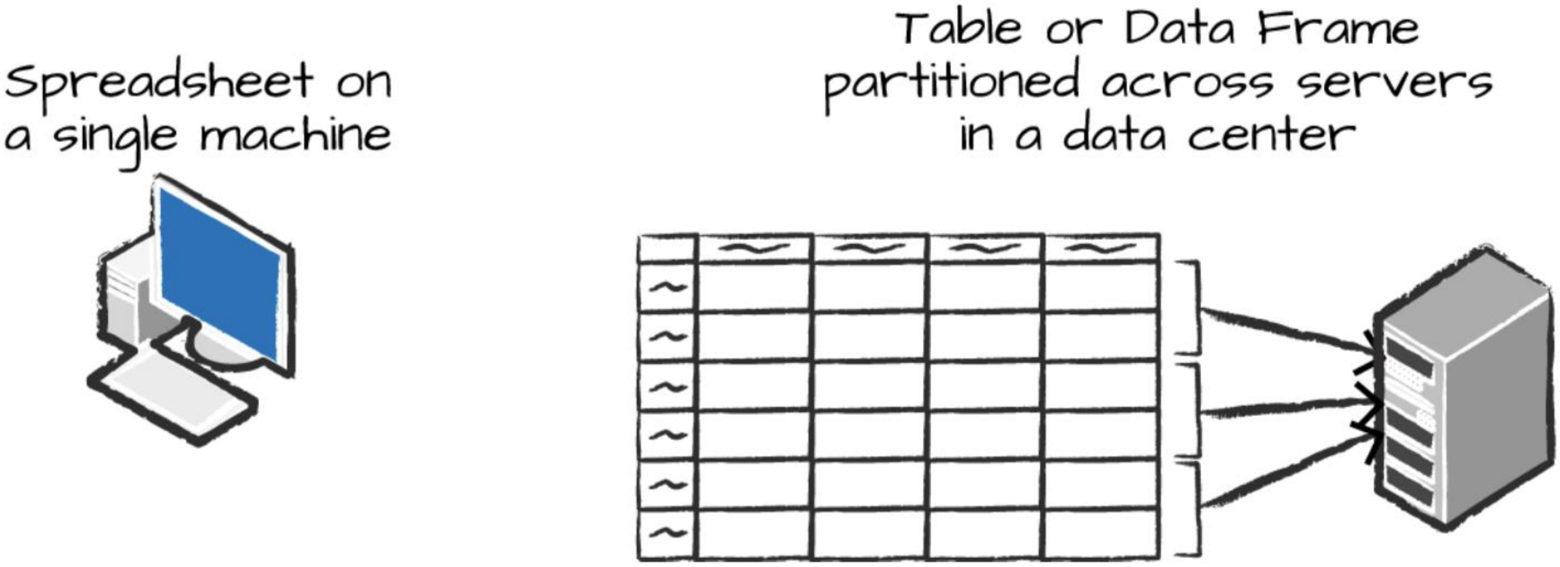
MapReduce to DAG

- A **directed acyclic graph** (DAG) extends MapReduce
  - from a simple two-step model (with a mapper and a reducer) to a orchestration of any steps that form a DAG
  - Although in theory is possible to pipeline MapReduce jobs to form any workflow, however…
  - You need to store the temporal output of intermediate jobs in HDFS (which is a natural idea in MapReduce) rather than keep it in memory
  - Workflow system in Apache Spark:



Using Spark

Distributed vs. single-machine computing



UNIVERSITY
OF WOLLONGONG
AUSTRALIA

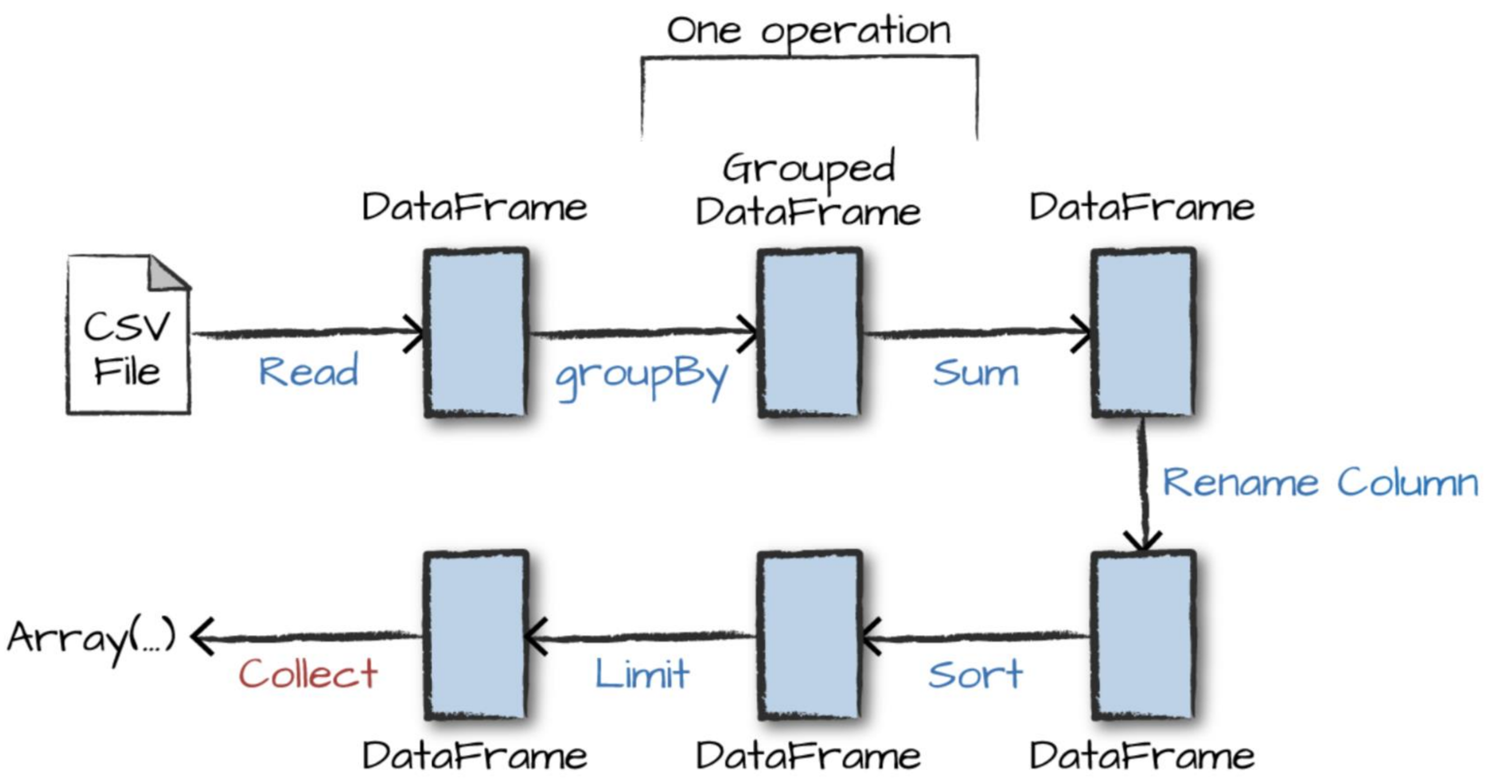
Spark’s Abstractions and APIs

- DataFrame—high-level *structured* abstraction
  - Similar to Pandas Dataframe but support distributed computing
  - Intuitively, a DataFrame is a table of data with rows and columns
    - which may be stored in a *single or multiple* machines
  - There is a scheme that defines the meta information (e.g. data types) for the columns
- *Resilient Distributed Dataset (RDD)—low-level abstraction
  - More control, sometimes more flexible, but less efficient than DataFrame
  - Can convert to a DataFrame.
- Immutabilility: Both DataFrame and RDD are immutable.
  - Instead of altering elements in a DataFrame or RDD, you create a new one

Transformation and Action

- End users operate on DataFrame as if the data is on a single computer
- Two kinds of operations: Transformations and Actions
  - Transformations create another DataFrame/RDD (e.g., create a new row)
  - Actions produce a computational result (e.g., count the rows)
  - A spark application can be viewed as a DAG (direct acyclic graph) of transformations and actions

Spark Data Analytics Pipeline as a DAG



Lazy evaluation: Spark don't evaluate the DataFrame/RDD until it has to, e.g., an action is performed.

Spark's DataFrame API (PySpark)

- See supplementary materials for some examples.
- Comprehensive API reference: https://spark.apache.org/docs/latest/api/python/reference/pyspark.sql/dataframe.html#

Summary

- MapReduce Model
  - A powerful computation model for processing massive data
  - Hadoop's MapReduce Framework
- Spark's DAG Model
  - A DAG model consisting of a series of transformations and an action
  - Spark's rich set of APIs