1. 随机释概念
2. MapReduce流程

- Map

- input: user defined dataset

- processing: receive user input and generate input into key-value pair

- key-value pairs as output

- Shuffle and sort

- group/sort output from Map stage by key, then pass to Reduce stage

- Reduce

- input: key-value pairs that in shuffle and sort stage

- user defined a reduce function(sum(value)/avg(value)/count(value)) that process key-value pair into output

- generate output

1. Data warehouse(OLAP query, lecture 11)

- fact table

- entities(include attribute, primary key, foreign key). For foreign key need to specify the reference(FK: subCode reference SUBJECT(subCode))

- measurement

- dimension(refer to entities)

- hierarchy

1. Datacube, conceptual schema, logic schema(lab4)
2. HDFS(load file into folder in hdfs)

# 312\_2020\_Q3\_(1)

# create a folder in hdfs

hadoop fs -mkdir /user/jrg/ACCIDENT

# load file into folder in hdfs

hadoop fs -put /home/accidents/car.tbl /user/jrg/ACCIDENT/

hadoop fs -put /home/accidents/car.tbl /user/jrg/ACCIDENT/

1. HQL(retrieve(count(\*), avg(\*)), join, left outer join)

# HIVE, HQL

# create internal table

create table table\_names(

first\_name VARCHAR(30),

last\_name VARCHAR(30),

age DECIMAL(3) )

row format delimited fields terminated by ',' stored as textfile;

# load data into table

load data local inpath ‘file\_path/file.txt’ into table table\_names;

# create external table names

create external table enames(

first\_name VARCHAR(30),

last\_name VARCHAR(30),

age DECIMAL(3) )

row format delimited fields terminated by ','

stored as textfile location '/user/bigdata/a-new-hdfs-folder';

# data type

# array

friends array<string> (row format delimited fields terminated by ‘|’ collection items terminated by ‘,’ stored as textfile;)

#map

tools map<string,int> (row format delimited fields terminated by ‘|’ collection items terminated by ',' map keys terminated by ':' stored as textfile; )

#struct

address struct<city:string,street:string,house:int,flat:int>

(row format delimited fields terminated by ‘|’ collection items terminated by ',' stored as textfile; )

1. HBase(create table, load sample data, describe schema, query: columns&filter, alter)

# create table

create 'table\_name', 'column\_family1', 'column\_family2', ...

# load sample data into table

put 'table\_name', 'row\_key', 'column\_family:column\_qualifier', 'value'

-- example

put 'accidents', '1', 'details:accident\_date', '2022-01-01'

put 'accidents', '1', 'details:car\_id', 'PKR856'

put 'accidents', '1', 'details:damages', '1256.67'

# scan query

scan 'table\_name', {OPTIONS}

scan 'table', {COLUMNS => [‘column\_family’:attribute', 'column\_family:attribute']}

-- example:

scan 'accidents', {COLUMNS => ['details:accident\_date', 'details:car\_id']}

# scan query with filter(FILTER=>”CONDITION”)

#ValueFilter()

scan 'table\_name', { FILTER => "ValueFilter(>=, 'binary:value')" }

-- example

scan ‘ACCIDENT’, {FILTER => “ValueFilter(>, ‘binary: 1000’)”}

#SingleColumnValueFilter()

scan 'table\_name', { COLUMNS => ['column\_family:column\_qualifier'], FILTER => "SingleColumnValueFilter('column\_family', 'column\_qualifier', =, 'binary:value')" }

# alter - modify the table structure

# add column family

alter 'table\_name', { NAME => 'new\_column\_family' }

# delete column family

alter 'table\_name', { NAME => 'column\_family', METHOD => 'delete' }

# modify table properties

alter 'table\_name', { METHOD => 'table\_att', 'property\_name' => 'property\_value' }

-- examples

alter 'accidents', { METHOD => 'table\_att', VERSION > 3 }

1. pig语言（load file，select，innner outerjoin等）

Create table, load file,

# load data, need to create a container, and assign the names and types to columns

data = LOAD 'local\_file\_path' USING PigStorage(',') AS (name:data\_type);

orders = load '/user/bigdata/orders.txt' using PigStorage(',') as

(item:chararray,customer:chararray,quantity:int,year:int,month:int,day:int);

# aggregated query(group by & aggregate)

-- Group by field and calculate the average

grouped\_data = GROUP data BY field;

result = FOREACH grouped\_data GENERATE group, AVG(data.value);

# distinct

dis\_items = distinct item;

# filter data(selection)

filtered\_data = FILTER data BY condition;

//biggerorders = filter orders by quantity > 100;

//nulls = filter orders by quantity is null;

# sort data

sorted\_data = ORDER data BY field [ASC | DESC];

# join data

# inner join(return the value that both table have)

joined\_data = JOIN data1 BY field, data2 BY field;

# outer join(return all values include null)

joined\_data = JOIN data1 BY field LEFT OUTER, data2 BY field;

# flatten(ungroup)

orderunnest = foreach ordergrp generate flatten (orders);

# limit

orderunnest4 = limit orderunnest 4;

1. spark（RDDs, dataset, dataframe, query）

# Read text file into RDDs from HDFS(.collect(); foreach(println))

- val rdd = sc.textFile(“file.txt”)

# Transform the dataset items RDD into a new RDD data pairs

-- for each value \*2

-val transformedRDD = rdd.map(x => x \* 2)

-- generate key-value pair according to file

- val pairs = rdd.map(s => s.split(“,”)(0), (s.split(“,”)(1).toFloat())))

# Perform a collection action ‘collect’ to retrieve all the elements of the itemsRDD

- pairs.collect()

# Perform a collection action ‘reduceByKey’ to return the result

-- get sum of value

- val result = pairs.reduceByKey((a, b) => a + b)

# Collect the result

- result.collect()

- result.foreach(println)

# more example in count number

val lines = sc.textFile("data.txt")

val pairs = lines.map(s => (s, 1))

val counts = pairs.reduceByKey((a, b) => a + b)

DATASET(.toDS(); .show())

case class Item(part:String, unitPrice:Float)

val lines = sc.textfile(‘/user/bigdata/items.txt’)

val itemsDS = lines.map(\_.split(“,”)).map(attributes => Item(attribute(0), attribute(1)

trim().toFloat))

.toDS()

DATAFRAME(.toDF(); .show())

case class Item(part:String, unitPrice:Float)

val itemsDF = spark.read.textfile(‘/user/bigdata/items.txt’)

.map(\_.split(“,”))

.map(attributes => Item(attribute(0), attribute(1).trim().toFloat))

.toDF()

# create logic view

itemsDF.createOrReplaceTempValue(‘ItemView’)

val sqlDF = spark.sql(“select part, sum(unitPrice) from ItemView group by part”)

# count words

text.filter(line => line.contains(“Spark”)).count()

# read json file into a dataframe

val df = spark.read.json(“/user/bigdata/people.json”)

-- operation in json

df.slect($“name”, $”age” + 1).show()

df.filter($”age” > 21).show()

df.groupBy(“age”).count().show()

df.createOr replaceTempView(“people”)

val sqlDF = spark.sql(“select \* from people”)