1. 自测基本概念

HDFS

create folder: hadoop fs -mkdir /user/bigdata/inputFile-Folder

put file into hadoop: hadoop fs -put /home/input.txt /user/bigdarta/inputFile-Folder

remove file: hadoop fs -rm /user/bigdata/inputFile-Folder/input.txt

list file: hadoop fs -ls

View detail in a file: hadoop fs -cat /user/bigdata/inputFile-Folder/input.txt

———————————————————————

MapReduce

Map

- input: a table with key-value pair or file(a b)

- process: read in user input, generate user input into key-value pair

- output: (a, b)/(a, 1)

Shuffle and sort

- input: the output in Map Stage

- process: sort the output by key, generate sorted key-value pair

- output: sorted key -value pair(a,((a, b),(a, 1)))

Reduce

- input: the output in shuffle and sort stage

- process:using user defined reduce function in the input to sum/count the value by key

- output: generate computed value by key as output

———————————————————————

HIVE, HQL shell command（like sql statement）

create internal table:

create table table\_name(

F\_name VARCHAR(20),

L\_name VARCHAR(20),

age DECIMAL(10))

row format delimited fields terminated by ‘,’ stored as textfile;

create external table:

create external table table\_name(

F\_name VARCHAR(20),

L\_name VARCHAR(20),

age DECIMAL(10))

row format delimited fields terminated by ‘,’ stored as textfile location ’store\_file\_path’;

array:

friend array<string> (row format delimited fields terminated by ‘|’

collection items terminated by ‘,’;)

map:

Item map<sting:int> (row format delimited field terminated by ‘|’

collection items terminated by ‘,’

map key terminated by ‘:’ stored as textfile;),

struct:

person struct<name:string, age:int> (row format delimited fields terminated by ‘|’

collection items terminated by ‘,’ stored as textfile;)

Load data file:

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

Query statement:

Select age, ave(age)

From table\_name

Join ON

Where age >20

Group by f\_name;

———————————————————————

HBase

Create table:

create ‘table\_name’, ‘cf1’, ‘cf2’, ‘cf3’

Load sample data into table:

put ‘table\_name’, ‘row\_id’, ‘cf1:att1’, ‘value’

put ‘table\_name’, ‘row\_id’, ‘cf1:att2’, ‘value’

put ‘table\_name’, ‘row\_id’, ‘cf1:att3’, ‘value’

put ‘table\_name’, ‘row\_id’, ‘cf2:att1 , ‘value’

Query statement:

filter \* 2 (filter => “valueFilter()”)

scan ‘table\_name’, {column => ‘cf1:att1’, filter => “valueFilter(=, ‘binary:1000’)”}

scan ‘table\_name’, {column => ‘cf1:att1’, filter => singleColumnValueFilter(‘cf1’, ‘att1’, =, ‘binary:1000’)”}

add column family: alter ‘table\_name’, {NAME => ‘cf1’}

delete column family: alter ’table\_name’, {NAME => ‘cf1’, METHOD => ‘delete’}

modify table attribute(version): alter ‘table\_name’, {NAME = ‘cf1’, version => 3}

———————————————————————

Pig Latin(container)

Load data file & create table:

data\_file = load ‘file path’ using PigStorage(‘,’) AS (name:chararray, age:int);

Query statement

select single columns: name = foreach data\_file generate name;

filter: age20 = filter name by age > 20;

avg:

ave = foreach data\_filegenerate age, avg(age);

count:

counter = foreach data\_file generate name, count(name);

Group by:

grouped\_name = group t1 by name;

Join:

t1\_t2 = join t1 by id, t2 by id

t1\_t2\_left = join t1 by id left outer, t2 by id;

difference:

dis\_t1 = distinct t1;

limit:

data\_file\_1 = limit dis\_t1 1;

dump container\_name

———————————————————————

Spark

RDDs

create rdds:

val rdds = sc.textFile(“file.txt”)

generate key-value pair:

val pairs = rdds.map(s => s.split(“,”)(0), (s.split(“,”)(1).toInt())

collection action to return result:

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

collect: count.collect()

Print: count.foreach(println)

Dataset

create class:

case class Person(name:String, age:Int)

read in file: file =sc.textFile(“file.txt”)

split file into line or key-value pair:

result = file.map(\_.split(“,”))

.map(s => (s(0), s(1).toInt()))

.toDS()

DataFrame

read in file, read in data, process data:

val df = spark.read.textFile(“file.txt”)

.map(\_.split(“,”))

.map(s => person(s(0), s(1).toInt))

.toDF()

Read in json file

val jsonDF = spark.read.json(“jsonfile.json”)

count：

df.filter(s => s.contains(“spark”)).count()

create logic view:

df.createOrReplaceTempView(“temp\_df”)

sql query(sum query):

val sqldf = spark.sql(“select name, sum(age) from temp\_df where name = ‘shaw’ group by name”)

1. 自测2020，2021试卷