Spark生态回顾五

1、Spark SQL

1.1 求出各个省份不同用户登录次数的top3

1.1.1 Spark SQL版本

1.1.1.1 源码

```
package com.qf.part3_sparksql.demo03

import org.apache.spark.sql.SparkSession

/**
  * Description: 使用Spark SQL实现→求出各个省份不同用户登录次数的top3<br/>
  * Copyright (c) , 2020 ,  Jansonxu <br/>
  * This program is protected by copyright laws. <br/>
  * Date:  2020年02月20日
  *
  * @author 徐文波
  * @version : 1.0
  */
object TopNDemo {

  def main(args: Array[String]): Unit = {

    //步骤:
    //①SparkSession
    val spark = SparkSession
      .builder
      .appName(this.getClass.getSimpleName)
      .master("local[*]")
      .getOrCreate

    val sqlContext = spark.sqlContext

    //②计算
    spark.read
      .json("file:///C:\\Users\\Administrator\\IdeaProjects\\spark-
restudy\\a_input\\data\\user_acc_logs.json")
      .createOrReplaceTempView("tb_user_info")

    sqlContext.cacheTable("tb_user_info")

    spark.sql(
      """
        |select
        |    id `用户编号`,
        |    province `省份名`,
        |    cnt `访问次数`,
        |    row_number() over( partition by  province order by cnt desc) rank
        |from
        |(
```

```
        |    select
        |        id,
        |        province,
        |        count(id) cnt
        |    from tb_user_info
        |    group by id,province
        |) t
        |having rank<=3
      """.stripMargin)
        .show()



    //⑥资源的释放
    sqlContext.uncacheTable("tb_user_info")
    spark.stop()

  }

}
```

1.1.1.2 结果

```
|用户编号|省份名|访问次数|rank|
+----+---+----+----+
|   7| 江苏|   3|   1|
|   5| 江苏|   3|   2|
|  10| 江苏|   3|   3|
|   9| 辽宁|   4|   1|
|   5| 辽宁|   2|   2|
|   3| 辽宁|   2|   3|
|   4| 山东|   7|   1|
|   8| 山东|   5|   2|
|   1| 山东|   5|   3|
|   4| 河北|   4|   1|
|   8| 河北|   3|   2|
|   1| 河北|   2|   3|
|   1| 浙江|   5|   1|
|   4| 浙江|   4|   2|
|   3| 浙江|   4|   3|
+----+---+----+----+
```

1.1.1.3 注意点

```
HQL的窗口函数的发生时间在having之后，因此它的窗口函数可以统计group聚合后的数据，spark sql
也是如此（测试的版本是spark2.2.3，其他spark版本暂时没有测试，有可能有区别）
```

```
package com.qf.part3_sparksql.demo03
```

```
import org.apache.spark.sql.SparkSession

/**
  * Description: 使用Spark SQL实现→求出各个省份不同用户登录次数的top3<br/>
  * Copyright (c) , 2020 ,  Jansonxu <br/>
  * This program is protected by copyright laws. <br/>
  * Date:  2020年02月20日
  *
  * @author 徐文波
  * @version : 1.0
  */
object TopNDemo {

  def main(args: Array[String]): Unit = {

    //步骤:
    //①SparkSession
    val spark = SparkSession
      .builder
      .appName(this.getClass.getSimpleName)
      .master("local[*]")
      .getOrCreate

    val sqlContext = spark.sqlContext

    //②计算
    spark.read
      .json("file:///C:\\Users\\Administrator\\IdeaProjects\\spark-
restudy\\a_input\\data\\user_acc_logs.json")
      .createOrReplaceTempView("tb_user_info")

    sqlContext.cacheTable("tb_user_info")

    //spark sql的写法:
    //    spark.sql(
    //      """
    //        |select
    //        |    id `用户编号`,
    //        |    province `省份名`,
    //        |    cnt `访问次数`,
    //        |    row_number() over( partition by  province order by cnt desc)
rank
    //        |from
    //        |(
    //        |    select
    //        |        id,
    //        |        province,
    //        |        count(id) cnt
    //        |    from tb_user_info
    //        |    group by id,province
    //        |) t
    //        |having rank<=3
    //      """.stripMargin)
    //        .show()


    //hql的写法
```

```
    spark.sql(
      """
        |select
        |    *
        |from(
        |  select
        |      id `用户编号`,
        |      province `省份名`,
        |      count(id) `→访问次数←`,
        |      row_number() over( partition by  province order by count(id)
desc) rank
        |  from tb_user_info
        |  group by id,province
        |) t where t.rank<=3
      """.stripMargin)
      .show(100)


    //③资源的释放
    sqlContext.uncacheTable("tb_user_info")
    spark.stop()

  }

}
```

1.1.2 Hive SQL版本

1.1.2.1 步骤

```
①给hive服务以及hive所有的客户端安装插件包json-serde-1.3.8-jar-with-dependencies.jar
 将jar包copy到所有hive安装的lib目录下，并重启hive服务
②建hive表tb_user_info
③装载json文件中的数据到hive表中
④验证
⑤编写hql脚本，计算：求出各个省份不同用户登录次数的top3
```

1.1.2.2 实施

topN.sql

```
drop table if exists tb_user_info;

CREATE TABLE tb_user_info (
        id int,
        province string,
        `time` string
)
ROW FORMAT SERDE 'org.openx.data.jsonserde.JsonSerDe'
STORED AS TEXTFILE;

LOAD DATA LOCAL INPATH '/root/hive-study/topN/topN.json' OVERWRITE INTO TABLE
tb_user_info;

select * from tb_user_info;
```

执行

```
[root@NODE03 topN]# hive -f topN.sql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-
2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-
log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/opt/hive/lib/hive-common-
2.3.0.jar!/hive-log4j2.properties Async: true
OK
Time taken: 6.203 seconds
OK
Time taken: 1.224 seconds
Loading data to table default.tb_user_info
OK
Time taken: 2.875 seconds
OK
9       辽宁    2020-02-19T10:22:18Z
4       浙江    2020-02-19T10:22:18Z
8       山东    2020-02-19T10:22:18Z
4       辽宁    2020-02-19T10:22:18Z
7       浙江    2020-02-19T10:22:18Z
4       浙江    2020-02-19T10:22:18Z
2       浙江    2020-02-19T10:22:18Z
3       浙江    2020-02-19T10:22:18Z
5       辽宁    2020-02-19T10:22:18Z
1       山东    2020-02-19T10:22:18Z
3       浙江    2020-02-19T10:22:18Z
8       河北    2020-02-19T10:22:18Z
8       山东    2020-02-19T10:22:18Z
4       河北    2020-02-19T10:22:18Z
6       浙江    2020-02-19T10:22:18Z
10      江苏    2020-02-19T10:22:18Z
5       浙江    2020-02-19T10:22:18Z
1       山东    2020-02-19T10:22:18Z
4       江苏    2020-02-19T10:22:18Z
3       山东    2020-02-19T10:22:18Z
1       浙江    2020-02-19T10:22:18Z
9       辽宁    2020-02-19T10:22:18Z
6       江苏    2020-02-19T10:22:18Z
8       浙江    2020-02-19T10:22:18Z
2       山东    2020-02-19T10:22:18Z
4       山东    2020-02-19T10:22:18Z
1       河北    2020-02-19T10:22:18Z
3       江苏    2020-02-19T10:22:18Z
1       山东    2020-02-19T10:22:18Z
9       河北    2020-02-19T10:22:18Z
4       山东    2020-02-19T10:22:18Z
8       河北    2020-02-19T10:22:18Z
9       江苏    2020-02-19T10:22:18Z
3       江苏    2020-02-19T10:22:18Z
4       山东    2020-02-19T10:22:18Z
7       山东    2020-02-19T10:22:18Z
```

7	河北	2020-02-19T10:22:18Z
7	辽宁	2020-02-19T10:22:18Z
10	浙江	2020-02-19T10:22:18Z
7	山东	2020-02-19T10:22:18Z
5	山东	2020-02-19T10:22:18Z
1	山东	2020-02-19T10:22:18Z
8	山东	2020-02-19T10:22:18Z
5	辽宁	2020-02-19T10:22:18Z
5	江苏	2020-02-19T10:22:18Z
9	河北	2020-02-19T10:22:18Z
1	江苏	2020-02-19T10:22:18Z
3	河北	2020-02-19T10:22:18Z
5	江苏	2020-02-19T10:22:18Z
4	浙江	2020-02-19T10:22:18Z
1	江苏	2020-02-19T10:22:18Z
8	山东	2020-02-19T10:22:18Z
3	辽宁	2020-02-19T10:22:18Z
8	河北	2020-02-19T10:22:18Z
10	山东	2020-02-19T10:22:18Z
9	江苏	2020-02-19T10:22:18Z
3	辽宁	2020-02-19T10:22:18Z
4	山东	2020-02-19T10:22:18Z
4	浙江	2020-02-19T10:22:18Z
4	河北	2020-02-19T10:22:18Z
4	河北	2020-02-19T10:22:18Z
4	山东	2020-02-19T10:22:18Z
2	辽宁	2020-02-19T10:22:18Z
7	江苏	2020-02-19T10:22:18Z
7	山东	2020-02-19T10:22:18Z
10	河北	2020-02-19T10:22:18Z
9	辽宁	2020-02-19T10:22:18Z
7	浙江	2020-02-19T10:22:18Z
10	山东	2020-02-19T10:22:18Z
8	辽宁	2020-02-19T10:22:18Z
1	浙江	2020-02-19T10:22:18Z
3	山东	2020-02-19T10:22:18Z
8	山东	2020-02-19T10:22:18Z
3	浙江	2020-02-19T10:22:18Z
7	浙江	2020-02-19T10:22:18Z
8	浙江	2020-02-19T10:22:18Z
9	浙江	2020-02-19T10:22:18Z
6	辽宁	2020-02-19T10:22:18Z
2	山东	2020-02-19T10:22:18Z
1	山东	2020-02-19T10:22:18Z
4	山东	2020-02-19T10:22:18Z
9	辽宁	2020-02-19T10:22:18Z
1	浙江	2020-02-19T10:22:18Z
5	江苏	2020-02-19T10:22:18Z
8	江苏	2020-02-19T10:22:18Z
1	浙江	2020-02-19T10:22:18Z
10	江苏	2020-02-19T10:22:18Z
10	辽宁	2020-02-19T10:22:18Z
10	浙江	2020-02-19T10:22:18Z
4	山东	2020-02-19T10:22:18Z
2	山东	2020-02-19T10:22:18Z
8	浙江	2020-02-19T10:22:18Z
1	河北	2020-02-19T10:22:18Z
4	河北	2020-02-19T10:22:18Z

```
3        浙江     2020-02-19T10:22:18Z
7        江苏     2020-02-19T10:22:18Z
10       江苏     2020-02-19T10:22:18Z
1        浙江     2020-02-19T10:22:18Z
10       浙江     2020-02-19T10:22:18Z
7        江苏     2020-02-19T10:22:18Z
Time taken: 4.251 seconds, Fetched: 100 row(s)
[root@NODE03 topN]# ll
total 20
-rw-r--r-- 1 root root  242 Feb 20 08:14 cal2.sql
-rw-r--r-- 1 root root  283 Feb 19 20:27 cal.sql
-rw-r--r-- 1 root root 5910 Feb 19 20:07 topN.json
```

求top3对应的hql脚本（求出各个省份不同用户登录次数的top3）

注意点1：必须将rank若在的语句封装到虚拟表中，否则，报错

```
select
    id `用户编号`,
    province `省份名`,
    cnt `访问次数`,
    row_number() over( partition by  province order by cnt desc) rank
from
(
    select
        id,
        province,
        count(id) cnt
    from tb_user_info
    group by id,province
) t
having rank<=3;
```

证明：hql的解析其与sparksql的解析器不一样的

having后的字段rank在同级不能访问的，需要进行封装 ，但是，spark sql不需要。

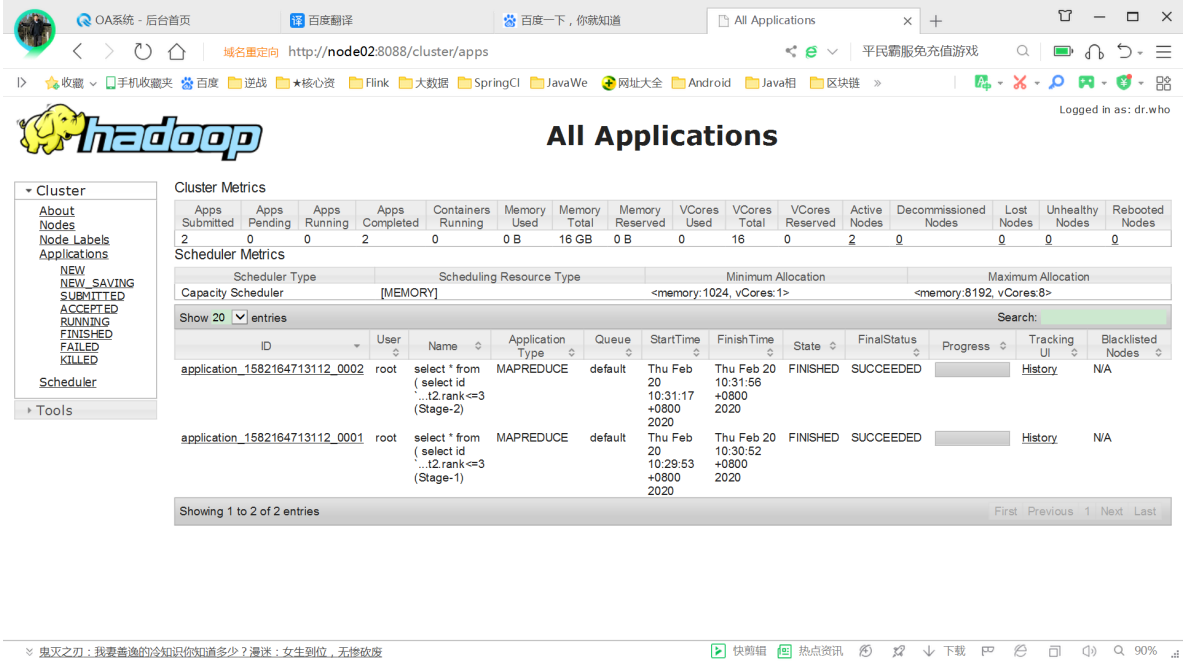
```
[root@NODE03 topN]# hive -f cal.sql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/opt/hive/lib/hive-common-2.3.0.jar!/hive-log4j2.properties Async: true
FAILED: SemanticException HAVING specified without GROUP BY
```

方式1：rank封装到虚拟表中，再进行查询，正确

```
[root@NODE03 topN]# hive -f cal.sql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-
2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-
log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/opt/hive/lib/hive-common-
2.3.0.jar!/hive-log4j2.properties Async: true
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark, tez)
or using Hive 1.X releases.
Query ID = root_20200220102856_6834ff36-181c-4723-b083-16e2144f9d77
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
```

```
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1582164713112_0001, Tracking URL =
http://NODE02:8088/proxy/application_1582164713112_0001/
Kill Command = /opt/hadoop/bin/hadoop job  -kill job_1582164713112_0001
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-02-20 10:30:22,585 Stage-1 map = 0%,  reduce = 0%
2020-02-20 10:30:46,457 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 1.33
sec
2020-02-20 10:30:52,920 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 3.19
sec
MapReduce Total cumulative CPU time: 3 seconds 190 msec
Ended Job = job_1582164713112_0001
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1582164713112_0002, Tracking URL =
http://NODE02:8088/proxy/application_1582164713112_0002/
Kill Command = /opt/hadoop/bin/hadoop job  -kill job_1582164713112_0002
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-02-20 10:31:32,399 Stage-2 map = 0%,  reduce = 0%
2020-02-20 10:31:49,269 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 1.62
sec
2020-02-20 10:31:56,886 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 4.18
sec
MapReduce Total cumulative CPU time: 4 seconds 180 msec
Ended Job = job_1582164713112_0002
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 3.19 sec   HDFS Read: 13578
HDFS Write: 1214 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 4.18 sec   HDFS Read: 10034
HDFS Write: 553 SUCCESS
Total MapReduce CPU Time Spent: 7 seconds 370 msec
OK
4       山东    7       1
8       山东    5       2
1       山东    5       3
10      江苏    3       1
7       江苏    3       2
5       江苏    3       3
4       河北    4       1
8       河北    3       2
1       河北    2       3
1       浙江    5       1
4       浙江    4       2
3       浙江    4       3
9       辽宁    4       1
5       辽宁    2       2
3       辽宁    2       3
Time taken: 181.469 seconds, Fetched: 15 row(s)
```

```
select
        *
from(
        select
                id `用户编号`,
                province `省份名`,
                cnt `访问次数`,
                row_number() over( partition by  province order by cnt desc)
rank
        from
        (
                select
                        id,
                        province,
                        count(id) cnt
                from tb_user_info
                group by id,province
        ) t1
) t2 where t2.rank<=3;
```



方式2：rank封装到虚拟表中，再进行查询，正确

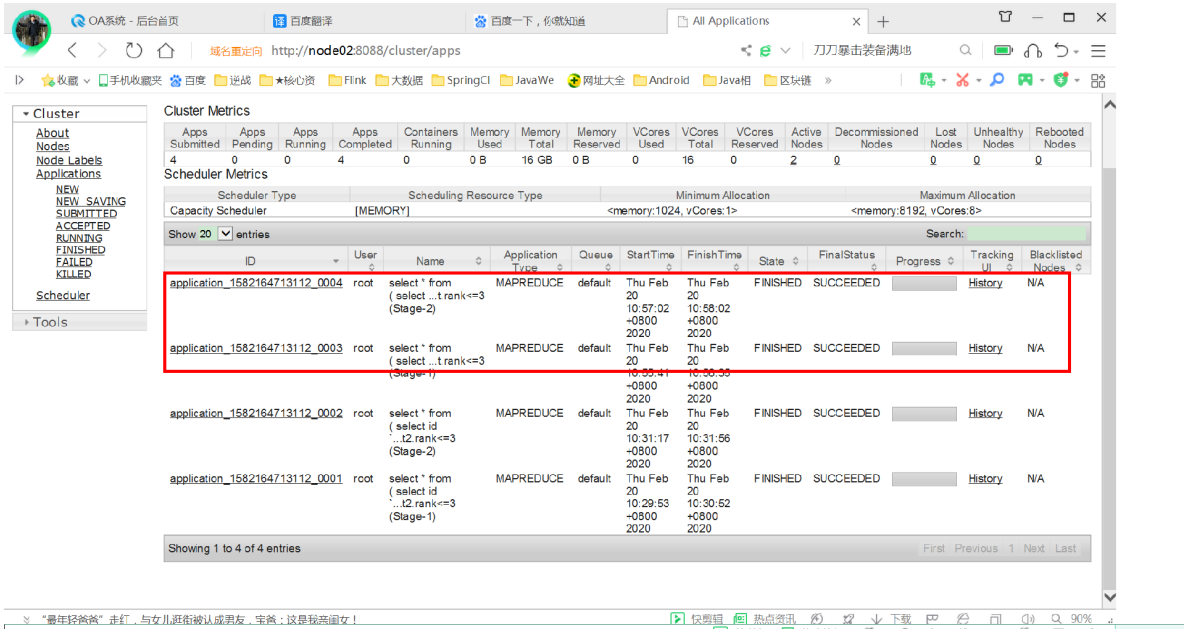
```
[root@NODE03 topN]# hive -f cal2.sql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-
2.6.2.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-
log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]

Logging initialized using configuration in jar:file:/opt/hive/lib/hive-common-
2.3.0.jar!/hive-log4j2.properties Async: true
```

```
WARNING: Hive-on-MR is deprecated in Hive 2 and may not be available in the
future versions. Consider using a different execution engine (i.e. spark, tez)
or using Hive 1.X releases.
Query ID = root_20200220105505_64445009-896a-453b-97ef-d942c0fcb086
Total jobs = 2
Launching Job 1 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1582164713112_0003, Tracking URL =
http://NODE02:8088/proxy/application_1582164713112_0003/
Kill Command = /opt/hadoop/bin/hadoop job  -kill job_1582164713112_0003
Hadoop job information for Stage-1: number of mappers: 1; number of reducers: 1
2020-02-20 10:56:02,601 Stage-1 map = 0%,  reduce = 0%
2020-02-20 10:56:26,360 Stage-1 map = 100%,  reduce = 0%, Cumulative CPU 3.23
sec
2020-02-20 10:56:35,193 Stage-1 map = 100%,  reduce = 100%, Cumulative CPU 5.63
sec
MapReduce Total cumulative CPU time: 5 seconds 630 msec
Ended Job = job_1582164713112_0003
Launching Job 2 out of 2
Number of reduce tasks not specified. Estimated from input data size: 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Starting Job = job_1582164713112_0004, Tracking URL =
http://NODE02:8088/proxy/application_1582164713112_0004/
Kill Command = /opt/hadoop/bin/hadoop job  -kill job_1582164713112_0004
Hadoop job information for Stage-2: number of mappers: 1; number of reducers: 1
2020-02-20 10:57:24,413 Stage-2 map = 0%,  reduce = 0%
2020-02-20 10:57:43,719 Stage-2 map = 100%,  reduce = 0%, Cumulative CPU 3.56
sec
2020-02-20 10:58:03,070 Stage-2 map = 100%,  reduce = 100%, Cumulative CPU 8.16
sec
MapReduce Total cumulative CPU time: 8 seconds 160 msec
Ended Job = job_1582164713112_0004
MapReduce Jobs Launched:
Stage-Stage-1: Map: 1  Reduce: 1   Cumulative CPU: 5.63 sec   HDFS Read: 13577
HDFS Write: 1214 SUCCESS
Stage-Stage-2: Map: 1  Reduce: 1   Cumulative CPU: 8.16 sec   HDFS Read: 10026
HDFS Write: 553 SUCCESS
Total MapReduce CPU Time Spent: 13 seconds 790 msec
OK
4       山东    7       1
8       山东    5       2
1       山东    5       3
10      江苏    3       1
7       江苏    3       2
5       江苏    3       3
4       河北    4       1
8       河北    3       2
```

```
1        河北    2        3
1        浙江    5        1
4        浙江    4        2
3        浙江    4        3
9        辽宁    4        1
5        辽宁    2        2
3        辽宁    2        3
Time taken: 178.52 seconds, Fetched: 15 row(s)
```

```
select
    *
from(
  select
      id `用户编号`,
      province `省份名`,
      count(id) `→访问次数←`,
      row_number() over( partition by  province order by count(id) desc) rank
  from tb_user_info
  group by id,province
) t where t.rank<=3;
```

窗口函数是对goup by后的结果进行的二次分析，根据用户登录的次数进行降序排列



1.1.2 Hive、Hive On Spark 与 SparK On Hive对比分析

1.1.2.1 Spark On Hive 和Spark SQL

```
①Spark SQL包含了Spark On Hive,sql语句中操作的是hive表。（ enableHiveSupport()）
②Spark SQL默认操作的表是内存中的虚拟表，使用完毕之后，在内存中就会释放
```

1.1.2.2 Spark On Hive 和Hive On Spark

```
异:
 ①主导方不同
   Spark On Hive → Spark官方（Databricks）
   Hive On Spark → Coudera,IBM, Intel
 ②对sql语句的解析引擎不同
   Spark On Hive →spark解析器
   hive on spark→ hive解析器
 ③运行的速度稍有不同，相差不是很大
 ④技术所面向的受众不同
   Spark On Hive → spark程序员（特点：需要编写spark程序，在代码中嵌入sql语句，操作hive表）
```

```
  Hive On Spark → 数仓程序员 （特点：之前的数仓项目不用作任何的变更，只需要将hive底层的计算引擎从mr换成spark即可）

同：
 ①计算引擎相同，都是Spark
 ②适用场景相同，都适用于对时效性要求较高的场合
```

1.1.2.3 Hive On Spark

1.1.2.3.1 环境准备

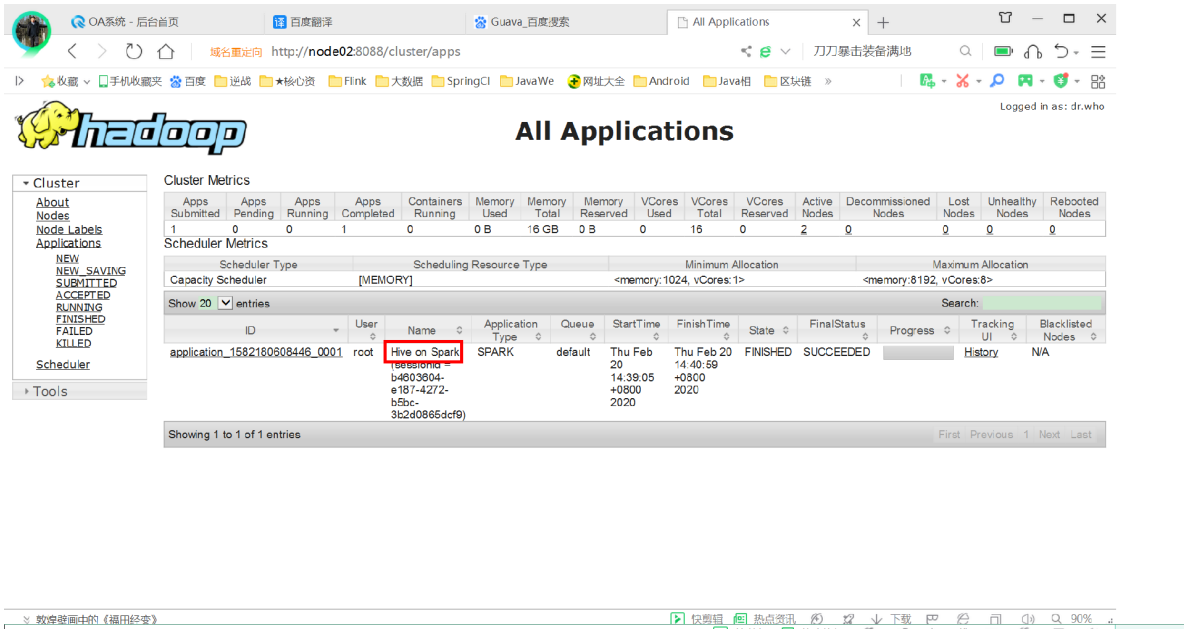
```
请产考：
2020-02-20-【Spark生态复习Ⅴ】\1_资料\Ⅱ-Spark生态\3-SparkSQL\『美团』外卖平台高效运营支撑系统二期Hive On Spark 版.pdf
```

1.1.2.3.2 实操

```
[root@NODE03 topN]# hive -f cal2.sql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hbase/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 9fbfc415-9ed0-458f-b50a-81466a9c27bb

Logging initialized using configuration in jar:file:/opt/hive/lib/hive-common-3.0.0.jar!/hive-log4j2.properties Async: true
Query ID = root_20200220143842_ccb1e082-227a-4ad3-b87b-68d629d51611
Total jobs = 1
Launching Job 1 out of 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Running with YARN Application = application_1582180608446_0001
Kill Command = /opt/hadoop/bin/yarn application -kill application_1582180608446_0001
Hive on Spark Session Web UI URL: http://NODE03:44682

Query Hive on Spark job[0] stages: [0, 1, 2]
Spark job[0] status = RUNNING
--------------------------------------------------------------------------------------
```

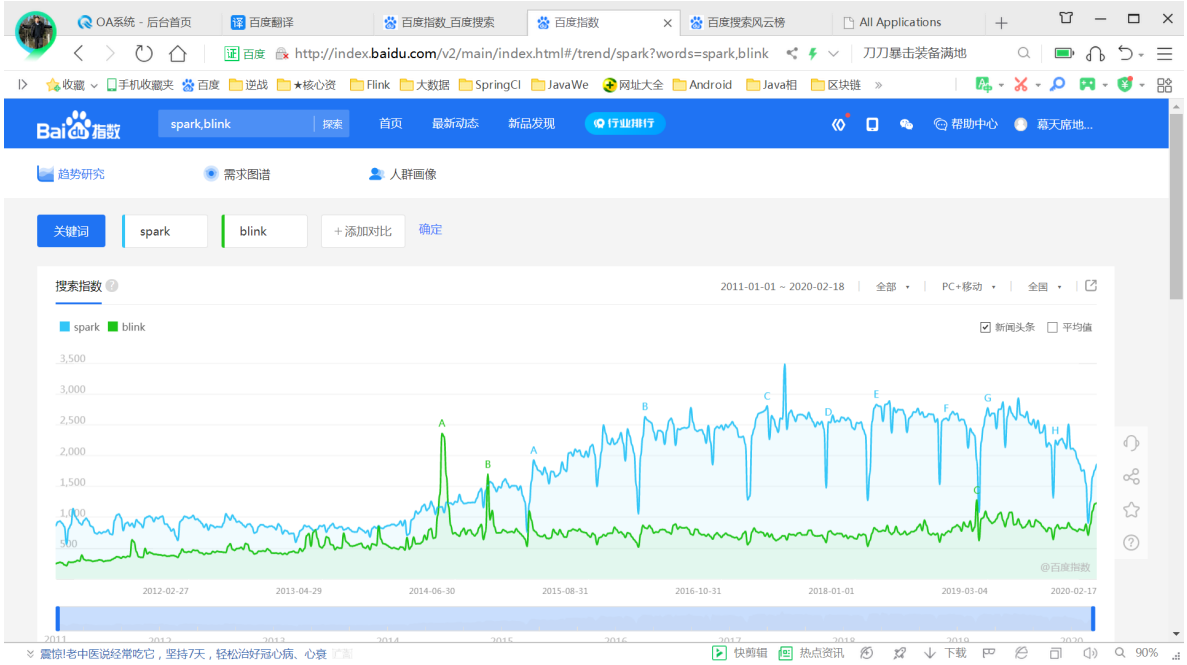
```
          STAGES   ATTEMPT        STATUS  TOTAL  COMPLETED  RUNNING  PENDING
FAILED
--------------------------------------------------------------------------------
------
Stage-0 ........         0      FINISHED      1          1        0        0
  0
Stage-1 ........         0      FINISHED      1          1        0        0
  0
Stage-2 ........         0      FINISHED      1          1        0        0
  0
--------------------------------------------------------------------------------
------
STAGES: 03/03    [==========================>>] 100%  ELAPSED TIME: 39.55 s
--------------------------------------------------------------------------------
------
Spark job[0] finished successfully in 39.55 second(s)
OK
4       山东    7       1
8       山东    5       2
1       山东    5       3
5       江苏    3       1
10      江苏    3       2
7       江苏    3       3
4       河北    4       1
8       河北    3       2
1       河北    2       3
1       浙江    5       1
4       浙江    4       2
3       浙江    4       3
9       辽宁    4       1
5       辽宁    2       2
3       辽宁    2       3
Time taken: 136.455 seconds, Fetched: 15 row(s)
[root@NODE03 topN]#
```

```
[root@NODE03 topN]# hive -f cal2.sql
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hbase/lib/slf4j-log4j12-1.7.25.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.slf4j.impl.Log4jLoggerFactory]
SLF4J: Class path contains multiple SLF4J bindings.
SLF4J: Found binding in [jar:file:/opt/hive/lib/log4j-slf4j-impl-2.10.0.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: Found binding in [jar:file:/opt/hadoop/share/hadoop/common/lib/slf4j-log4j12-1.7.10.jar!/org/slf4j/impl/StaticLoggerBinder.class]
SLF4J: See http://www.slf4j.org/codes.html#multiple_bindings for an explanation.
SLF4J: Actual binding is of type [org.apache.logging.slf4j.Log4jLoggerFactory]
Hive Session ID = 9fbfc415-9ed0-458f-b50a-81466a9c27bb

Logging initialized using configuration in jar:file:/opt/hive/lib/hive-common-3.0.0.jar!/hive-log4j2.properties Async: true
Query ID = root_20200220143842_ccb1e082-227a-4ad3-b87b-68d629d51611
Total jobs = 1
Launching Job 1 out of 1
In order to change the average load for a reducer (in bytes):
  set hive.exec.reducers.bytes.per.reducer=<number>
In order to limit the maximum number of reducers:
  set hive.exec.reducers.max=<number>
In order to set a constant number of reducers:
  set mapreduce.job.reduces=<number>
Running with YARN Application = application_1582180608446_0001
Kill Command = /opt/hadoop/bin/yarn application -kill application_1582180608446_0001
Hive on Spark Session Web UI URL: http://NODE03:44682

Query Hive on Spark job[0] stages: [0, 1, 2]
Spark job[0] status = RUNNING
--------------------------------------------------------------------------------------
          STAGES   ATTEMPT        STATUS  TOTAL  COMPLETED  RUNNING  PENDING  FAILED
--------------------------------------------------------------------------------------
Stage-0 ........         0      FINISHED      1          1        0        0       0
Stage-1 ........         0      FINISHED      1          1        0        0       0
Stage-2 ........         0      FINISHED      1          1        0        0       0
--------------------------------------------------------------------------------------
STAGES: 03/03    [==========================>>] 100%  ELAPSED TIME: 39.55 s
--------------------------------------------------------------------------------------
Spark job[0] finished successfully in 39.55 second(s)
OK
4       山东    7       1
8       山东    5       2
1       山东    5       3
5       江苏    3       1
10      江苏    3       2
7       江苏    3       3
4       河北    4       1
8       河北    3       2
1       河北    2       3
1       浙江    5       1
4       浙江    4       2
3       浙江    4       3
9       辽宁    4       1
5       辽宁    2       2
3       辽宁    2       3
Time taken: 136.455 seconds, Fetched: 15 row(s)
[root@NODE03 topN]#
```



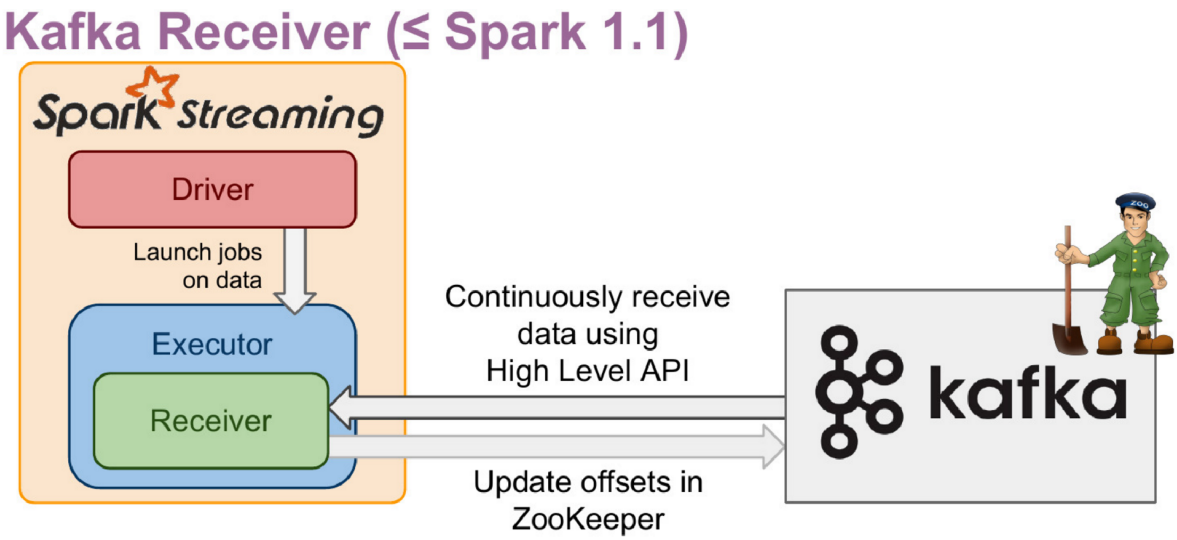
1、Spark Streaming

1.1 spark生态与blink在全球受欢迎调查

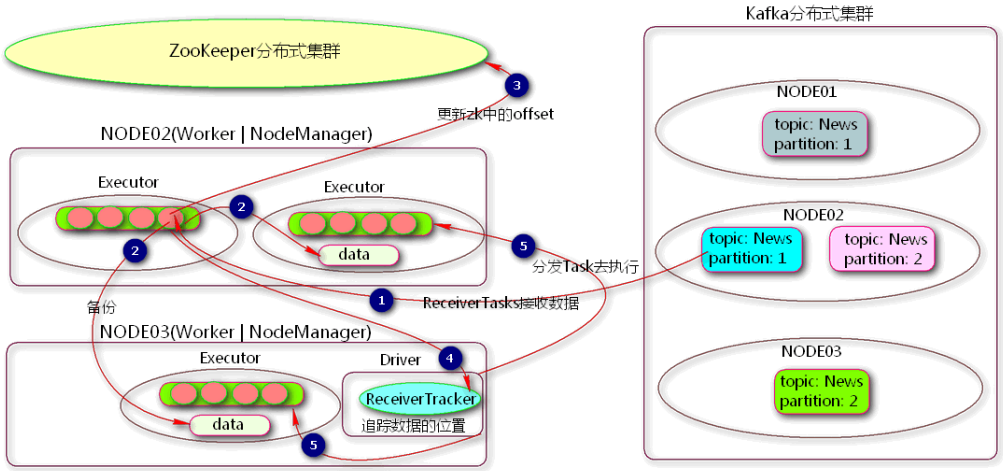


1.2 Spark Streaming与Kafka的整合

1.2.1 receiver方式



Spark Streaming + Kafka之Receiver模式



说明：
步骤2→接收来的数据存储级别为MEMORY_AND_DISK_SER_2
步骤4→将备份数据的位置汇报给Driver中的ReceiverTracker。Driver就知道了data的位置。