数据交换插件使用详解

1 配置

配置文件以json格式保存，json的内容主要包括三个部分，job、reader和writer。这三个部分的功能分别是任务的配置，数据交换源端配置以及数据交换目标端配置。

1.1 job

模板

```
"job": {
    "id": "hdfsWriterTest",
    "name": "测试2",
```

```
    "parallelism": 20,
    "rateLimit": "2000M",
    "sparkConf": "set spark.executor.memory=48G;set spark.executor.instance
s=50;set spark.executor.cores=6;set spark.driver.memory=2g;set spark.yarn.q
ueue=default;set spark.yarn.executor.memoryOverhead=1g;set spark.memory.sto
rageFraction=0.1;set spark.memory.fraction=0.8;set spark.merge.table.enable
d=true;",
    "url": "http://cdh1:8181/v1/jobInstance/add",
    "appkey": "张三的appkey",
    "appsecret": "张三的appsecret",
    "owner": "张三#san.zhang",
    "projectCode": "bigdata",
    "jobType": "exchange4",
    "clusterCode": "spark_qy",
    "jobName": "test_exchange_hdfs_to_hive"
  }
```

参数解释
1 id,name 这两个参数作为数据交换内部控制使用，随便填即可。

2 parallelism 数据交换reader使用的spark分区数。

3 rateLimit 限流兆数。超过传输流量将启用降级和熔断机制保障平台的稳定性。

4 sparkConf spark任务的核心配置
可以参考Spark参数配置说明

```
spark.executor.memory spark执行器的使用堆内内存的大小，该内存数量是spark.executor.
cores中设置的内核数共用的内存数量
spark.executor.instances 该参数决定了yarn集群中，最多能够同时启动的executor的实例个
数。
spark.executor.cores 该参数为设置每个executor能够使用的CPU core的数量。
spark.driver.memory 该参数设置的是driver分配的内存的大小
spark.yarn.queue 使用yarn队列名
spark.yarn.executor.memoryOverhead 当执行超出spark.executor.memory时使用的额外
内存。
spark.merge.table.enabled 数据交换完写入的表是否触发合并操作。
spark.job.callback.url 数据交换完成后的回调地址。
```

5 url dc平台提交任务的url地址，不同环境只需要替换ip和端口即可，后面rest地址不用更改。

6 appkey,appsecret,owner 数据交换使用的用户信息，这些信息可以从dc_user_info中查询，决定了什么用户来执行reader和writer的命令，若用户不存在或权限不足数据交换将失

败。

7 projectCode 数据交换执行时所在hive的数据库名。

8 jobType 作业类型，数据交换写死exchange4即可。

9 clusterCode 运行所在的集群名。

10 jobName 用于在dc平台显示的数据交换名。

1.2 reader

reader是数据交换读插件，其功能是读取数据源，重新设置spark分区，并交给writer插件写数据。

1.2.1 hive

模板

```
"reader": {
    "name": "cn.tongdun.pontus.hive.reader.HiveReader",
    "querySql" : "select CONCAT_WS('_',p.id,h.phone) as pk,p.id as zhyid,p.
name as zhyname,p.phone,h.brand,h.provider,p.salary from \nabc.people_msg_d
t p \njoin\nabc.phone_msg_dt h\non p.phone=h.phone;"
}
```

参数解释
1 querySql 表示读取hive表的语句，对于每一列都可以用as来设置别名，作为writer写入的列名。默认情况下列名和hive搜索出来的列名一致。

2 clusterConfig 当读取非本集群的hive时，往往是跨集群的时候，需要配置，在clusterConfig中需要配置以下字段(其中a,b根据情况填写)，以下字段均可以从cdh平台上下载客户端来获取配置信息：

```
"clusterConfig": {
  "hive.metastore.uris": "",
  "dfs.client.failover.proxy.provider.tdhdfs": "",
  "hive.metastore.warehouse.dir": "",
  "dfs.ha.namenodes.tdhdfs": "a,b",
  "dfs.namenode.rpc-address.tdhdfs.a": "",
  "dfs.namenode.rpc-address.tdhdfs.b": "",
  "dfs.nameservices": ""
}
```

1.2.2 hdfs

模板

```
"reader": {
    "name": "cn.tongdun.pontus.hdfs.reader.HdfsReader",
    "extraConfig": {
      "header": true
    },
    "dataType": "csv",
    "path": [
      "/user/datacompute/data_all/data_20190601.csv",
      "/user/datacompute/data_all/data_20190602.csv"
    ],
    "cols": [
      "eventOccurTime",
      "trnsId",
      "certType",
      "status",
      "amount",
      "eventType",
      "userId",
      "certNo",
      "payerAcct",
      "payerName",
      "deviceId",
      "recvAcct",
      "recvName",
      "ipAddress"
    ]
  }
```

参数解释
1 dataType 解析hdfs的文件类型，可以从以下几个中选取:
csv,json,parquet,text,other
这几个类型的意义分别是:
1) csv 解析hdfs上的csv文件，只能有一个字符的分隔符。csv文件格式请参考网上。
2) json 解析hdfs上的json文件，json文件格式请参考网上。
3) parquet 解析hdfs上的parquet文件，parquet文件格式请参考网上。
4) text 解析hdfs上的text文件，以换行符分隔的文件。
5) other 类似于csv的解析，比csv解析多的功能是支持多字符的分隔符。

2 path 解析的文件，当多个时，用逗号分隔，注意所有文件都必须是dataType定义的类型。当定义多个文件时，是通过并行的方式读取。

3 cols 文件中每一列对应的列名，多个列用逗号分隔。

4 clusterConfig 当读取非本集群的hdfs时，往往是跨集群的时候，需要配置，在clusterConfig中需要配置以下字段(其中a,b根据情况填写)，以下字段均可以从cdh平台上下载客户端来获取配置信息：

```
"fs.defaultFS": "",
"ha.zookeeper.quorum": "",
"dfs.client.failover.proxy.provider.tdhdfs": "",
"dfs.ha.namenodes.tdhdfs": "a,b",
"dfs.namenode.rpc-a": "",
"dfs.namenode.rpc-b": "",
"dfs.nameservices": ""
```

5 extraConfig 特殊配置，根据dataType的不同，有不同的配置
1) csv
解析csv文件

```
"extraConfig": {
  "header": true,
  “sep”, “,”
  "comment":""
}
```

header ： 是否有第一行头信息，默认值false
sep ： 分隔符字符，默认值, 注意只能是一个字符，如果是多分隔字符，使用other模式
comment ： 注释行(单字符)，当设置后，遇到设置字符开头的行都默认是注释，不解析。

2) text
默认每一行都是一个列。

```
"extraConfig": {
  "wholetext": false
}
```

wholetext ： 当设置为true，整个文本都当作一列，lineSep参数无效

3) other
解析自定义hdfs文件

```
"extraConfig": {
  "delimiter": '\\|!'
}
```

delimiter ： 分隔符，支持多字符，注意转义符前加上\

parquet和json不再赘述。

1.3 writer

writer是数据交换写插件，其功能是获取reader插件提交的数据并写进目标数据源中。

1.3.1 hive

模板

```
"writer": {
    "name": "cn.tongdun.pontus.hive.writer.HiveWriter",
    "writeMode": "OVERWRITE",
    "cols": [
      "recvName",
      "ipAddress",
      "unix_timestamp(eventOccurTime)*1000",
      "substr(eventOccurTime,1,4)",
      "substr(eventOccurTime,6,2)",
      "substr(eventOccurTime,9,2)",
      "substr(eventOccurTime,12,2)",
      "substr(eventOccurTime,15,2)",
      "substr(eventOccurTime,18,2)",
      "concat(substr(eventOccurTime,1,4),substr(eventOccurTime,6,2),substr(
eventOccurTime,9,2))"
    ],
    "part": "ds",
    "table": [
      {
        "name": "bigdata.nh_poc_detail_18yi_dt"
      }
    ]
  }
```

参数解释

1 clusterConfig 参考hive reader的clusterConfig配置，支持跨数据源同步。

2 writeMode 分为两种，OVERWRITE和INTO。INTO表示追加模式，数据不会清除；OVERWRITE表示覆盖模式，当插入普通表的情况下，之前的数据会全部清除，当插入分区表的情况下，插入的分区会全部清除（不会影响其他分区的数据）。

3 part 分区字段 有两种填写方式：动态分区和静态分区

```
动态分区 多个分区字段用逗号分隔
"part": "ds"

静态分区
"part": "ds='20191010'"

动静结合(不常用)
"part": "ds='20191010',hour"
```

4 cols 表示写入hive的列，有以下注意点：
1) 当part不填写或只填写分区字段名的情况下，cols的列需要和hive表的列一致；当part为静态分区的话，cols里不需要填写静态分区的字段。
2) cols中列的顺序和hive建表列的顺序一致。
3) cols中的列名区分大小写，且需要和reader中定义的列名一致。
4) cols中可以填写函数表达式，只要函数中出现reader中定义的列名即可。

5 table 写入的表，可以写多个。

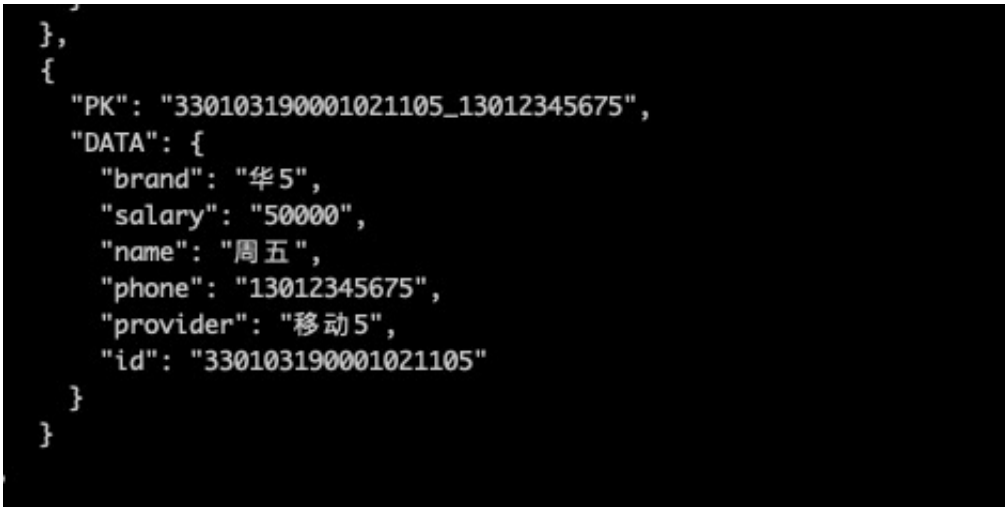
1.3.2 aerospike

模板

```
"writer": {
"name" : "cn.tongdun.pontus.aerospike.writer.AerospikeWriter",
  "isDynamic" : "false",
  "namespace" : "ns1",
  "hosts" : "1.1.1.1:3000",
  "code" : "zhy_11111",
  "threadNum" : 100,
  "timeout" : 86400000,
  "sendKey" : false,
  "soTimeout" : 0,
  "binName": "TEST"
}
```

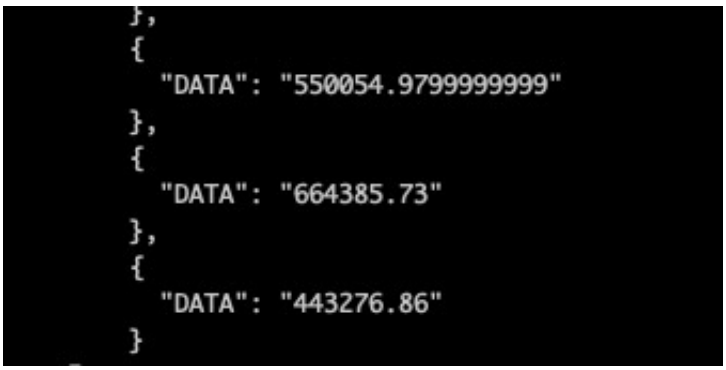
参数解释

1 isDynamic as的写入模式，有三种:false(默认),true,easy。
1) false 多列模式，最常用的模式。 写入as后第一个字段作为主键，后面的字段写在BIN名为DATA下面，写入后的效果如图所示：



2) true 动态列模式，写入as后第一个字段作为主键，第二列是列名，第三列value,第四列是列名，第五列value,依此类推，默认写在DATA这个BIN下面。这种方式作为替代可以在Reader端定义好别名。

3) easy 简单模式，写入as后第一个字段作为主键，第二个字段是值，默认写在DATA这个BIN下面



2 namespace as写入的namespace名。

3 code as写入的SET名。

4 hosts as的地址。ip和端口之间用冒号分隔，集群环境下多个地址之间用逗号分隔。

5 sendKey 是否写入主键。默认值是false不保存主键，true表示再存储一份主键方便调试(as客户端默认不显示主键值)。

6 threadNum as写入的线程数。默认值是100，通过修改线程数可以很大程度上提升as的写入性能。

7 timeout as写入数据的过期时间，单位毫秒，默认值为-1，表示不过期。

8 soTimeout as执行命令的超时时间，单位毫秒，默认值为0，表示不设置超时时间。

9 binName 默认写入的BIN是DATA，如果需要写入其他BIN需要指定，默认值是DATA。

1.3.3 kafka

模板

```
"writer": {
    "name": "cn.tongdun.pontus.kafka.writer.KafkaWriter",
    "topic": "zhy_test",
    "importType": "0",
    "ack": "1",
    "brokerList": "1.1.1.1:9092"
}
```

参数解释
1 brokerList kafka的brokerList地址

2 topic kafka写入的topic名

3 importType kafka的写入方式，分为两种0和1。
1) 0 表示以csv格式，逗号分隔符方式写入数据，不带列名。
2) 1 表示json格式，以json方式写入数据，带列名。

4 partitionerClass 设置Partition类, 对队列进行合理的划分。 默认不设置。

5 serializerClass 消息的序列化类。默认为 org.apache.kafka.common.serialization.StringSerializer。 针对key.serializer和 value.serializer

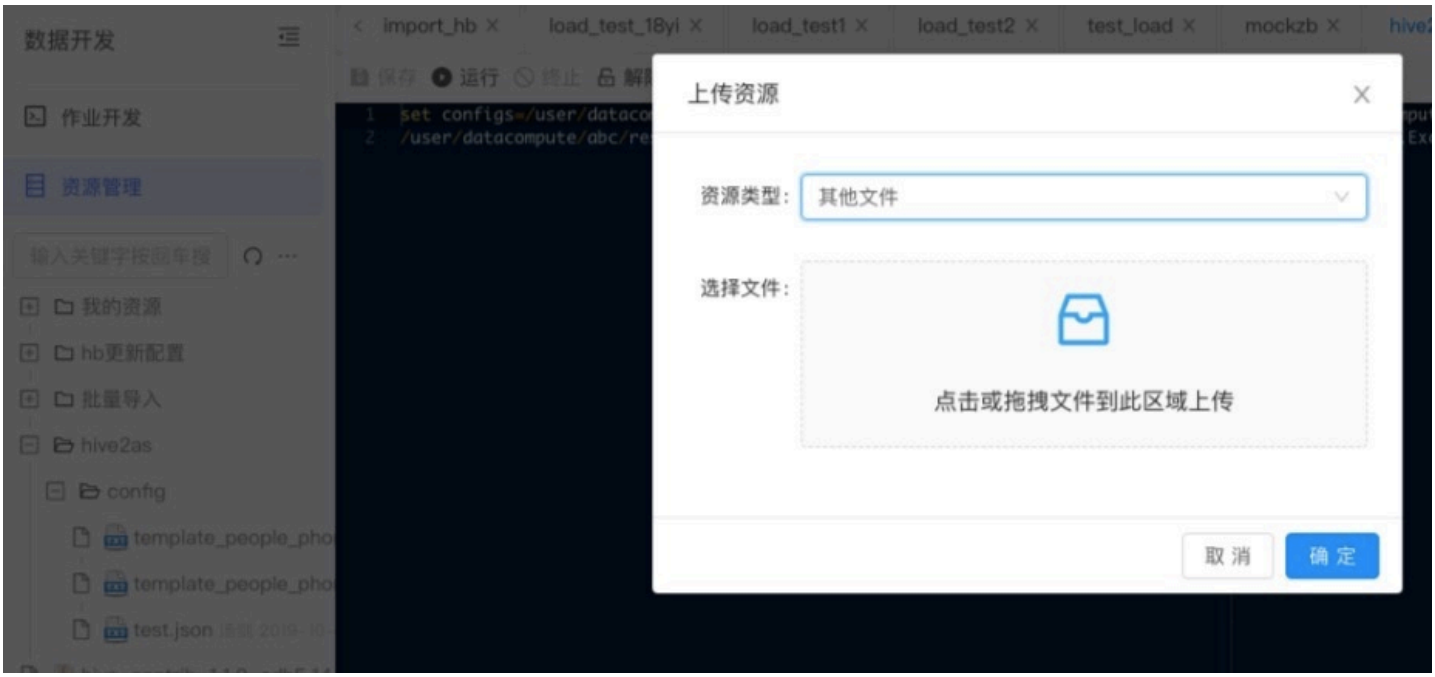
6 ack ACK机制, 消息发送需要kafka服务端确认。 0不等待确认 1等待主确认 -1等待副本确认 默认值为1

7 extraConfig 额外的kafka producer配置，参考kafka文档。

2 在dc上使用

2.1 上传配置文件

在dc平台->数据开发->资源管理上传配置文件


数据开发
作业开发
资源管理
我的资源
hb更新配置
批量导入
hive2as
config
import_hb
load_test_18yi
load_test1
load_test2
test_load
mockzb
上传资源
资源类型：
其他文件
选择文件：
点击或拖拽文件到此区域上传
取 消
确 定


资源管理
输入关键字按回车搜
我的资源
hb更新配置
批量导入
hive2as
config
template_people_phone_card.json
template_people_phone.json
test.json 汤剑 2019-10-12 15:36:42

2.2 编写交换任务

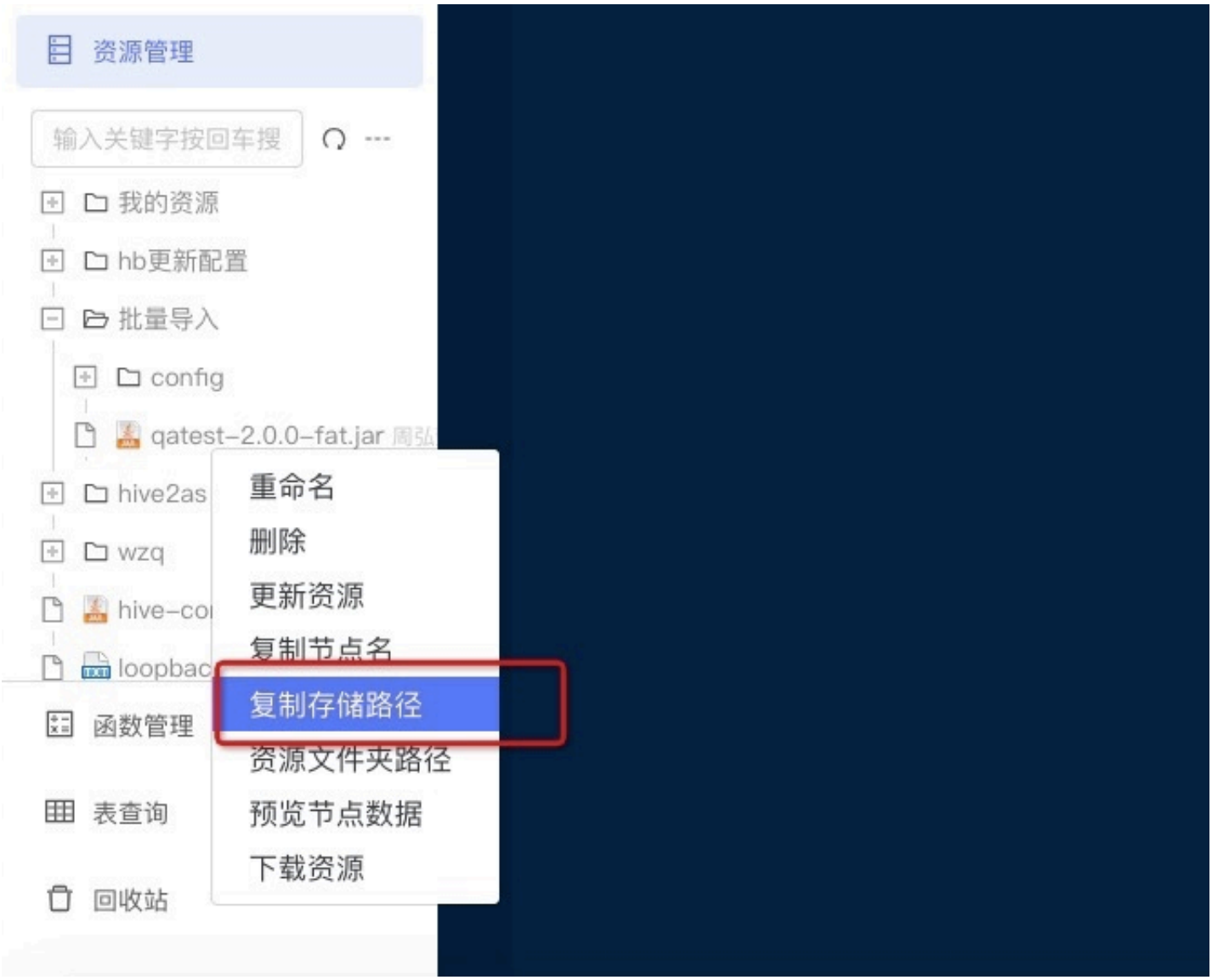

```
set configs=/user/datacompute/abc/resources/10810/latest/template_function1
.json;
/user/datacompute/abc/resources/10796/latest/qatest-2.0.0-fat.jar cn.tongdu
n.datacompute.qa.demo.ExchangeMain
```

参数解释

1 set configs 后表示执行的json配置文件。 多个配置文件之间用逗号分隔。当执行多个配置文件的时候是串行执行，从左往右依次执行，每个任务都依赖于前一个任务的执行成功才会执行。

2 qatest-2.0.0-fat.jar表示数据交换的包 cn.tongdun.datacompute.qa.demo.ExchangeMain是数据交换入口程序。

3 hdfs的路径可从资源管理->复制存储路径获取。



3 案例

介绍几个常用案例，基本覆盖业务面，修改上述介绍的配置即可。

当源和目标的类型不同的话替换reader或writer部分的配置即可，

3.1 hdfs->hive

3.1.1 带动态分区以及函数的导入

此例子包括两个动态分区，ds和hour，需要注意的点：
1 writer的cols中倒数第二列是ds，最后一列是hour。
2 writer的cols中每个列都必须包含reader的cols中定义的列名。
3 reader中path里有5个文件。job中parallelism和spark.executor.instances都设置为5会有比较好的性能。

```
{
  "job": {
    "id": "hdfsWriterTest",
    "name": "测试2",
    "parallelism": 5,
    "rateLimit": "2000M",
    "sparkConf": "set spark.executor.memory=24G;set spark.executor.instance
s=5;set spark.executor.cores=6;set spark.driver.memory=2g;set spark.yarn.qu
eue=default;set spark.yarn.executor.memoryOverhead=1g;set spark.memory.stor
ageFraction=0.1;set spark.memory.fraction=0.8;set spark.merge.table.enabled
=true;",
    "url": "http://cdh:8181/v1/jobInstance/add",
    "appkey": "",
    "appsecret": "",
    "owner": "超管#binsong.li",
    "projectCode": "bigdata",
    "jobType": "exchange4",
    "clusterCode": "spark_qy",
    "jobName": "test_exchange_hdfs_to_hive"
  },
  "reader": {
    "name": "cn.tongdun.pontus.hdfs.reader.HdfsReader",
    "extraConfig": {
      "header": true
    },
    "dataType": "csv",
    "path": [
      "/user/datacompute/data_all/data_20190601.csv",
      "/user/datacompute/data_all/data_20190602.csv",
      "/user/datacompute/data_all/data_20190603.csv",
      "/user/datacompute/data_all/data_20190604.csv",
      "/user/datacompute/data_all/data_20190605.csv"
    ],
    "cols": [
      "eventOccurTime",
      "trnsId",
      "certType",
      "status",
      "amount",
      "eventType",
      "userId",
      "certNo",
      "payerAcct",
      "payerName",
      "deviceId",
      "recvAcct",
```

```
      "recvName",
      "ipAddress"
    ]
  },
  "writer": {
    "name": "cn.tongdun.pontus.hive.writer.HiveWriter",
    "writeMode": "OVERWRITE",
    "cols": [
      "trnsId",
      "certType",
      "status",
      "amount",
      "eventType",
      "userId",
      "certNo",
      "payerAcct",
      "payerName",
      "deviceId",
      "recvAcct",
      "recvName",
      "ipAddress",
      "unix_timestamp(eventOccurTime)*1000",
      "substr(eventOccurTime,1,4)",
      "substr(eventOccurTime,6,2)",
      "substr(eventOccurTime,9,2)",
      "substr(eventOccurTime,15,2)",
      "substr(eventOccurTime,18,2)",
      "concat(substr(eventOccurTime,1,4),substr(eventOccurTime,6,2),substr(
eventOccurTime,9,2))",
      "substr(eventOccurTime,12,2)"
    ],
    "part": "ds,hour",
    "table": [
      {
        "name": "bigdata.nh_poc_detail_alldata2_dt"
      }
    ]
  }
}
```

3.1.2 超大量数据导入

当资源不足以一次性导完的话，在写交换任务的时候set configs后接多个配置文件，可以串行执行。

```
{
  "job": {
    "id": "hdfsWriterTest",
    "name": "测试",
    "parallelism": 20,
    "rateLimit": "2000M",
    "sparkConf": "set spark.executor.memory=48G;set spark.executor.instance
s=20;set spark.executor.cores=6;set spark.driver.memory=2g;set spark.yarn.q
ueue=default;set spark.yarn.executor.memoryOverhead=1g;set spark.memory.sto
rageFraction=0.1;set spark.memory.fraction=0.8;set spark.merge.table.enable
d=true;",
    "url": "http://cdh114:8181/v1/jobInstance/add",
    "appkey": "45jwEKtJ9gnuQAZ4",
    "appsecret": "WGaI446ESPW2Nsg7I7NleC0ZSNM9FNZY",
    "owner": "超管#binsong.li",
    "projectCode": "bigdata",
    "jobType": "exchange4",
    "clusterCode": "spark_qy",
    "jobName": "test_exchange_hdfs_to_hive_18yi_part1"
  },
  "reader": {
    "name": "cn.tongdun.pontus.hdfs.reader.HdfsReader",
    "extraConfig": {
      "header": true
    },
    "dataType": "csv",
    "path": [
      "/user/datacompute/data_all/data_20190601.csv",
      "/user/datacompute/data_all/data_20190602.csv",
      "/user/datacompute/data_all/data_20190603.csv",
      "/user/datacompute/data_all/data_20190604.csv",
      "/user/datacompute/data_all/data_20190605.csv",
      "/user/datacompute/data_all/data_20190606.csv",
      "/user/datacompute/data_all/data_20190607.csv",
      "/user/datacompute/data_all/data_20190608.csv",
      "/user/datacompute/data_all/data_20190609.csv",
      "/user/datacompute/data_all/data_20190610.csv"
    ],
    "cols": [
      "eventOccurTime",
      "trnsId",
      "certType",
      "status",
      "amount",
      "eventType",
      "userId",
```

```
      "certNo",
      "payerAcct",
      "payerName",
      "deviceId",
      "recvAcct",
      "recvName",
      "ipAddress"
    ]
  },
  "writer": {
    "name": "cn.tongdun.pontus.hive.writer.HiveWriter",
    "writeMode": "INTO",
    "cols": [
      "trnsId",
      "certType",
      "status",
      "amount",
      "eventType",
      "userId",
      "certNo",
      "payerAcct",
      "payerName",
      "deviceId",
      "recvAcct",
      "recvName",
      "ipAddress",
      "unix_timestamp(eventOccurTime)*1000",
      "substr(eventOccurTime,1,4)",
      "substr(eventOccurTime,6,2)",
      "substr(eventOccurTime,9,2)",
      "substr(eventOccurTime,12,2)",
      "substr(eventOccurTime,15,2)",
      "substr(eventOccurTime,18,2)",
      "concat(substr(eventOccurTime,1,4),substr(eventOccurTime,6,2),substr(
eventOccurTime,9,2))"
    ],
    "part": "ds",
    "table": [
      {
        "name": "bigdata.nh_poc_detail_18yi_dt"
      }
    ]
  }
}
```

```
set configs=/user/datacompute/abc/resources/10810/latest/template_function1
.json,/user/datacompute/abc/resources/10810/latest/template_function1.json,
/user/datacompute/abc/resources/10810/latest/template_function1.json;
/user/datacompute/abc/resources/10796/latest/qatest-2.0.0-fat.jar cn.tongdu
n.datacompute.qa.demo.ExchangeMain
```

3.2 hive->aerospike

3.2.1 两表关联

```
{
  "job": {
    "id": "hive2as",
    "name": "测试",
    "parallelism": 10,
    "rateLimit": "2000M",
    "sparkConf": "set spark.executor.memory=24G;set spark.executor.instance
s=10;set spark.executor.cores=6;set spark.driver.memory=2g;set spark.yarn.q
ueue=default;set spark.yarn.executor.memoryOverhead=1g;set spark.memory.sto
rageFraction=0.1;set spark.memory.fraction=0.8;set spark.merge.table.enable
d=true;",
    "url": "http://cdh114:8181/v1/jobInstance/add",
    "appkey": "YYFRf2JhXVLrGdN4",
    "appsecret": "you3ShoOrK3duIu1heXSwQPO5xgwqbRD",
    "owner": "周弘懿#hongyi.zhou",
    "projectCode": "abc",
    "jobType": "exchange4",
    "clusterCode": "spark_qy",
    "jobName": "job_hive2as_people_phone_card"
  },
  "reader": {
    "name": "cn.tongdun.pontus.hive.reader.HiveReader",
    "querySql" : "select CONCAT_WS('_',p.id,h.phone,c.cardno) as pk,p.id,p.
name,p.phone,h.brand,h.provider,p.cardno,c.bankname, c.opendate,c.expiredat
e,p.salary from \nabc.people_msg_dt p \njoin\nabc.phone_msg_dt h\njoin\nabc
.card_msg_dt c\non p.phone=h.phone and p.cardno=c.cardno;"
  },
  "writer": {
    "name": "cn.tongdun.pontus.aerospike.writer.AerospikeWriter",
    "namespace" : "ns1",
    "hosts" : "xx:3000",
    "code" : "zhy_people_phone_card",
```

```
    "threadNum" : 10,
    "sendKey" : true
  }
}
```

3.2.2 三表关联

```
{
  "job": {
    "id": "hive2as",
    "name": "测试",
    "parallelism": 10,
    "rateLimit": "2000M",
    "sparkConf": "set spark.executor.memory=24G;set spark.executor.instance
s=10;set spark.executor.cores=6;set spark.driver.memory=2g;set spark.yarn.q
ueue=default;set spark.yarn.executor.memoryOverhead=1g;set spark.memory.sto
rageFraction=0.1;set spark.memory.fraction=0.8;set spark.merge.table.enable
d=true;",
    "url": "http://cdh114:8181/v1/jobInstance/add",
    "appkey": "YYFRf2JhXVLrGdN4",
    "appsecret": "you3ShoOrK3duIu1heXSwQPO5xgwqbRD",
    "owner": "周弘懿#hongyi.zhou",
    "projectCode": "abc",
    "jobType": "exchange4",
    "clusterCode": "spark_qy",
    "jobName": "job_hive2as_people_phone"
  },
  "reader": {
    "name": "cn.tongdun.pontus.hive.reader.HiveReader",
    "querySql" : "select CONCAT_WS('_',p.id,h.phone) as pk,p.id as zhyid,p.
name as zhyname,p.phone,h.brand,h.provider,p.salary from \nabc.people_msg_d
t p \njoin\nabc.phone_msg_dt h\non p.phone=h.phone;"
  },
  "writer": {
    "name": "cn.tongdun.pontus.aerospike.writer.AerospikeWriter",
    "namespace" : "ns1",
    "hosts" : "xxx:3000",
    "code" : "zhy_people_phone",
    "threadNum" : 10,
    "sendKey" : true
  }
}
```

3.2.3 相关建表与插入脚本

```
create table if not exists abc.people_msg_dt (
id string comment '身份证号',
name string comment '姓名',
phone string comment '手机号',
cardno string comment '卡号',
salary int comment '薪水'
)
STORED AS PARQUET TBLPROPERTIES('parquet.compression'='SNAPPY')
comment '123'
lifecycle 100;

create table if not exists abc.phone_msg_dt (
phone string comment '手机号',
brand string comment '手机品牌',
provider string comment '服务供应商'
)
STORED AS PARQUET TBLPROPERTIES('parquet.compression'='SNAPPY')
comment '123'
lifecycle 100;

create table if not exists abc.card_msg_dt (
cardno string comment '卡号',
bankname string comment '银行名',
opendate string comment '开卡时间',
expiredate string comment '过期时间'
)
STORED AS PARQUET TBLPROPERTIES('parquet.compression'='SNAPPY')
comment '123'
lifecycle 100;

insert into table abc.people_msg_dt values('330103190001021101','周一','1301
2345671','no1',10000);
insert into table abc.people_msg_dt values('330103190001021102','周二','1301
2345672','no2',20000);
insert into table abc.people_msg_dt values('330103190001021103','周三','1301
2345673','no3',30000);
insert into table abc.people_msg_dt values('330103190001021104','周四','1301
2345674','no4',40000);
insert into table abc.people_msg_dt values('330103190001021105','周五','1301
2345675','no5',50000);
insert into table abc.people_msg_dt values('330103190001021106','周六','1301
2345676','no6',60000);
```

```
insert into table abc.people_msg_dt values('3301031900010211O7','周日','1301
2345677','no7',70000);

insert into table abc.phone_msg_dt values('13012345671','华1','移动1');
insert into table abc.phone_msg_dt values('13012345672','华2','移动2');
insert into table abc.phone_msg_dt values('13012345673','华3','移动3');
insert into table abc.phone_msg_dt values('13012345674','华4','移动4');
insert into table abc.phone_msg_dt values('13012345675','华5','移动5');
insert into table abc.phone_msg_dt values('13012345676','华6','移动6');
insert into table abc.phone_msg_dt values('13012345677','华7','移动7');

insert into table abc.card_msg_dt values('no1','农行1','20190101','20300101'
);
insert into table abc.card_msg_dt values('no2','农行2','20190102','20300102'
);
insert into table abc.card_msg_dt values('no3','农行3','20190103','20300103'
);
insert into table abc.card_msg_dt values('no4','农行4','20190104','20300104'
);
insert into table abc.card_msg_dt values('no5','农行5','20190105','20300105'
);
insert into table abc.card_msg_dt values('no6','农行6','20190106','20300106'
);
insert into table abc.card_msg_dt values('no7','农行7','20190107','20300107'
);
```

测试spark jar 命令编写

```
set configs=/user/datacompute/abc/resources/10809/latest/template_people_ph
one.json,/user/datacompute/abc/resources/10808/latest/template_people_phone
_card.json;
/user/datacompute/abc/resources/10796/latest/qatest-2.0.0-fat.jar cn.tongdu
n.datacompute.qa.demo.ExchangeMain
```

3.2.4 结果展示

```
aql> select * from ns1.zhy_people_phone;
+---------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------+
| PK                              | DATA                                                                                                                                                 |
+---------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------+
| "330103190001021101_13012345671" | KEY_ORDERED_MAP('{"brand":"华1", "salary":"10000", "name":"周一", "phone":"13012345671", "provider":"移动1", "id":"330103190001021101"}') |
| "330103190001021103_13012345673" | KEY_ORDERED_MAP('{"brand":"华3", "salary":"30000", "name":"周三", "phone":"13012345673", "provider":"移动3", "id":"330103190001021103"}') |
| "330103190001021107_13012345677" | KEY_ORDERED_MAP('{"brand":"华7", "salary":"70000", "name":"周日", "phone":"13012345677", "provider":"移动7", "id":"330103190001021107"}') |
| "330103190001021102_13012345672" | KEY_ORDERED_MAP('{"brand":"华2", "salary":"20000", "name":"周二", "phone":"13012345672", "provider":"移动2", "id":"330103190001021102"}') |
| "330103190001021106_13012345676" | KEY_ORDERED_MAP('{"brand":"华6", "salary":"60000", "name":"周六", "phone":"13012345676", "provider":"移动6", "id":"330103190001021106"}') |
| "330103190001021104_13012345674" | KEY_ORDERED_MAP('{"brand":"华4", "salary":"40000", "name":"周四", "phone":"13012345674", "provider":"移动4", "id":"330103190001021104"}') |
| "330103190001021105_13012345675" | KEY_ORDERED_MAP('{"brand":"华5", "salary":"50000", "name":"周五", "phone":"13012345675", "provider":"移动5", "id":"330103190001021105"}') |
+---------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------+
7 rows in set (0.260 secs)

OK
```

```
aql> select * from ns1.zhy_people_phone_card;
+-------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| PK                                  | DATA
                                                                               |
+-------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
| "330103190001021104_13012345674_no4" | KEY_ORDERED_MAP('{"bankname":"农行4", "salary":"40000", "name":"周四", "expiredate":"20300104", "phone":"13012345674", "cardno":"no4", "opend
ate":"20190104", "id":"330103190001021104", "brand":"华4", "provider":"移动4"}') |
| "330103190001021101_13012345671_no1" | KEY_ORDERED_MAP('{"bankname":"农行1", "salary":"10000", "name":"周一", "expiredate":"20300101", "phone":"13012345671", "cardno":"no1", "opend
ate":"20190101", "id":"330103190001021101", "brand":"华1", "provider":"移动1"}') |
| "330103190001021103_13012345673_no3" | KEY_ORDERED_MAP('{"bankname":"农行3", "salary":"30000", "name":"周三", "expiredate":"20300103", "phone":"13012345673", "cardno":"no3", "opend
ate":"20190103", "id":"330103190001021103", "brand":"华3", "provider":"移动3"}') |
| "330103190001021106_13012345676_no6" | KEY_ORDERED_MAP('{"bankname":"农行6", "salary":"60000", "name":"周六", "expiredate":"20300106", "phone":"13012345676", "cardno":"no6", "opend
ate":"20190106", "id":"330103190001021106", "brand":"华6", "provider":"移动6"}') |
| "330103190001021105_13012345675_no5" | KEY_ORDERED_MAP('{"bankname":"农行5", "salary":"50000", "name":"周五", "expiredate":"20300105", "phone":"13012345675", "cardno":"no5", "opend
ate":"20190105", "id":"330103190001021105", "brand":"华5", "provider":"移动5"}') |
| "330103190001021102_13012345672_no2" | KEY_ORDERED_MAP('{"bankname":"农行2", "salary":"20000", "name":"周二", "expiredate":"20300102", "phone":"13012345672", "cardno":"no2", "opend
ate":"20190102", "id":"330103190001021102", "brand":"华2", "provider":"移动2"}') |
| "330103190001021107_13012345677_no7" | KEY_ORDERED_MAP('{"bankname":"农行7", "salary":"70000", "name":"周日", "expiredate":"20300107", "phone":"13012345677", "cardno":"no7", "opend
ate":"20190107", "id":"330103190001021107", "brand":"华7", "provider":"移动7"}') |
+-------------------------------------+------------------------------------------------------------------------------------------------------------------------------------------------------
-------------------------------------------------------------------------------+
7 rows in set (0.282 secs)

OK
```