

# Flink on zeppelin第五弹设置checkpoint

## 概述

Flink的exactly-once语义实现是需要依赖checkpoint的,对于一个有状态的Flink任务来说如果想要在任务发生failover,或者手动重启任务的时候任务的状态不丢失是必须要开启checkpoint的,今天这篇文章主要分享一下Flink on zeppelin里面怎么设置checkpoint以及如何从指定的checkpoint恢复任务.

## checkpoint配置

```
%flink.conf
// 设置任务使用的时间属性是eventtime
pipeline.time-characteristic EventTime
// 设置checkpoint的时间间隔
execution.checkpointing.interval 10000
// 确保检查点之间的间隔
execution.checkpointing.min-pause 60000
// 设置checkpoint的超时时间
execution.checkpointing.timeout 60000
// 设置任务取消后保留hdfs上的checkpoint文件
```

```
execution.checkpointing.externalized-checkpoint-retention RETAIN_ON_CANCELLATION
```

只需要像上面这样就可以配置任务的checkpoint了,当然除了配置这些参数外也可以设置任务的jm,tm等参数,Interpreters 设置里面的很多参数(比如下图的这些)都是可以在note里面直接执行的,然后在这个note里面的任务都会使用刚才的配置.

flink.jm.memory	1024	Memory for JobManager (mb)
flink.tm.memory	1024	Memory for TaskManager (mb)
flink.tm.slot	4	Number of slot per TaskManager
local.number-taskmanager	2	Number of TaskManager in local mode
flink.yarn.appName	Zeppelin Flink Session	Yarn app name
flink.yarn.queue	flink	Yarn queue name
flink.webui.yarn.useProxy	false	Whether use yarn proxy url as flink weburl, e.g. http://localhost:8088/proxy/application_1583396598068_0004

在执行这些配置的时候需要先把Interpreters 重启一下,因为当Interpreters 进程已启动时就无法更改Interpreters 的属性了,否则会遇到下面的报错

```
java.io.IOException: Can not change interpreter properties when interpreter pi
```

```

at org.apache.zepplin.interpreter.InterpreterSetting.setInterpreterGroupPro
at org.apache.zepplin.interpreter.ConfInterpreter.interpret(ConfInterpreter.j
at org.apache.zepplin.notebook.Paragraph.jobRun(Paragraph.java:458)
at org.apache.zepplin.notebook.Paragraph.jobRun(Paragraph.java:72)
at org.apache.zepplin.scheduler.Job.run(Job.java:172)
at org.apache.zepplin.scheduler.AbstractScheduler.runJob(AbstractScheduler.ja
at org.apache.zepplin.scheduler.FIFOScheduler.lambda$runJobInScheduler$0(FIFO
at java.util.concurrent.ThreadPoolExecutor.runWorker(ThreadPoolExecutor.java:1
at java.util.concurrent.ThreadPoolExecutor$Worker.run(ThreadPoolExecutor.java:
at java.lang.Thread.run(Thread.java:748)

```

## 创建kafak流表



```

%flink.sql
DROP TABLE IF EXISTS kafka_table;

CREATE TABLE kafka_table (
  name VARCHAR COMMENT '姓名',
  age int COMMENT '年龄',
  city VARCHAR,
  borth VARCHAR,
  ts BIGINT COMMENT '时间戳',
  t as TO_TIMESTAMP(FROM_UNIXTIME(ts/1000,'yyyy-MM-dd HH:mm:ss')),
  proctime as PROCTIME(),
  WATERMARK FOR t AS t - INTERVAL '5' SECOND
)
WITH (
  'connector' = 'kafka', -- 使用 kafka connector
  'topic' = 'jason_flink', -- kafka topic
  'properties.bootstrap.servers' = 'master:9092,storm1:9092,storm2:9092', -- broker连接信息
  'properties.group.id' = 'jason_flink_test',
  'scan.startup.mode' = 'latest-offset', -- 读取数据的位置
  'format' = 'json' -- 数据源格式为 json
);

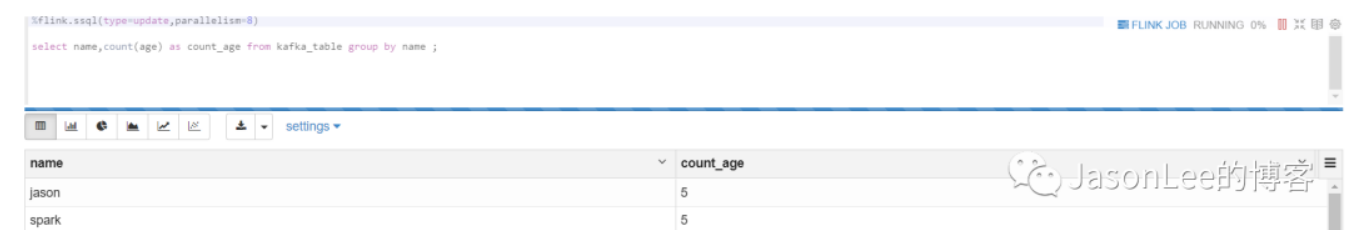
```

Finished

JasonLee的博客

Took 55 sec. Last updated by anonymous at July 27 2020, 10:31:10 PM.

然后来一个非常简单的聚合查询,为了体现任务是从状态中恢复的,所以我们这里根据name分组计算了一个count(age)的操作



```

%flink.sql(type:update,parallelism=8)
select name,count(age) as count_age from kafka_table group by name ;

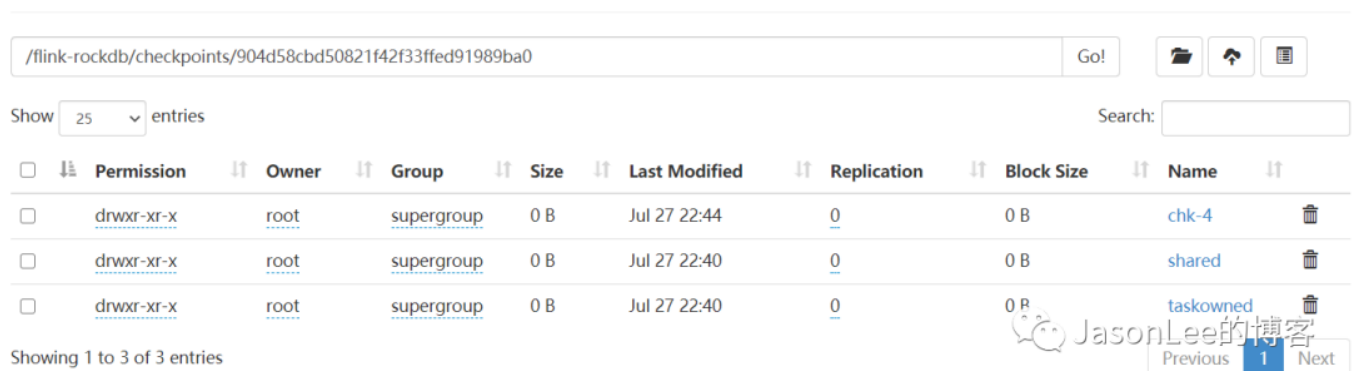
```

name	count_age
jason	5
spark	5

FLINK JOB RUNNING 0%

JasonLee的博客

这里我先写入了10条数据,jason和spark都是5条,然后先来看一下HDFS上的checkpoint文件是否生成了.



/flink-rockdb/checkpoints/904d58cbd50821f42f33ffed91989ba0

Go!

Show 25 entries

Search:

Permission	Owner	Group	Size	Last Modified	Replication	Block Size	Name
drwxr-xr-x	root	supergroup	0 B	Jul 27 22:44	0	0 B	chk-4
drwxr-xr-x	root	supergroup	0 B	Jul 27 22:40	0	0 B	shared
drwxr-xr-x	root	supergroup	0 B	Jul 27 22:40	0	0 B	taskowned

Showing 1 to 3 of 3 entries

Previous 1 Next

JasonLee的博客

可以看到此时任务已经完成了4次checkpoint了,这时候把任务cancel掉,然后从上一次成功的checkpoint处恢复任务,也就是第4个checkpoint的地方.

```
%flink.conf
```

```
execution.savepoint.path hdfs://master:9000//flink-rockdb/checkpoints/904d58c
```

需要指定刚才任务的checkpoint路径,先执行上面的语句,然后再接着执行刚才的那条SQL就可以了.

The screenshot shows the Flink UI with tabs for Overview, History, Summary, and Configuration. The Overview tab is active, displaying checkpoint counts: Triggered: 21, In Progress: 0, Completed: 21, Failed: 0, Restored: 1. A red arrow points to the 'Restored: 1' status. Below this, the 'Latest Completed Checkpoint' section shows ID: 25, Completion Time: 22:50:04, End to End Duration: 86ms, and Checkpointed Data Size: -. The 'Checkpoint Detail' section shows the Path: hdfs://flink-rockdb/checkpoints/11184c5ae8d64dd7440428ecb67a08b7/chk-25 and Discarded: -. The 'Operators' section is a table with columns: Name, Acknowledged, Latest Acknowledgment, End to End Duration, Checkpointed Data Size, and Buffered During Alignment. It lists three operators: Source (TableSourceScan), GroupAggregate, and Sink (Zeppelin Flink Sql Stream Collect Sink). The 'Latest Failed Checkpoint' is None, and the 'Latest Savepoint' is None. The 'Latest Restore' section shows ID: 4, Restore Time: 22:48:05, Type: Savepoint, and Path: hdfs://master:9000/flink-rockdb/checkpoints/904d58cbd50821f42f... JasonLee的博客

从上面的图可以很清楚的看到,任务是从第4个checkpoint处恢复计算的,然后再来往kafka中写入10条数据看下结果是否正确.

The screenshot shows the Flink UI with a table of results. The table has two columns: name and count\_age. The data rows are: jason (10) and spark (10). JasonLee的博客

可以看到结果是正确的,是从刚才的状态里面的值接着计算的,说明任务从checkpoint里面恢复成功了.整个过程是不需要写任何代码的,只需要在note里面执行几行配置就可以了,使用起来还是非常方便的.

这篇文章主要是介绍Flink on zeppelin如何设置任务的checkpoint,以及任务重启的时候怎么从指定的checkpoint位置恢复任务.