flink-scala-study

flink1.10 scala version study

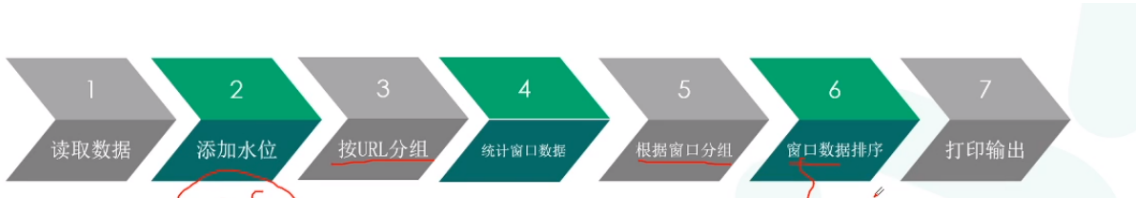
需求一：实时热门页面统计

1. 需求分析：每隔5秒统计最近5分钟热门页面
2. 数据展示

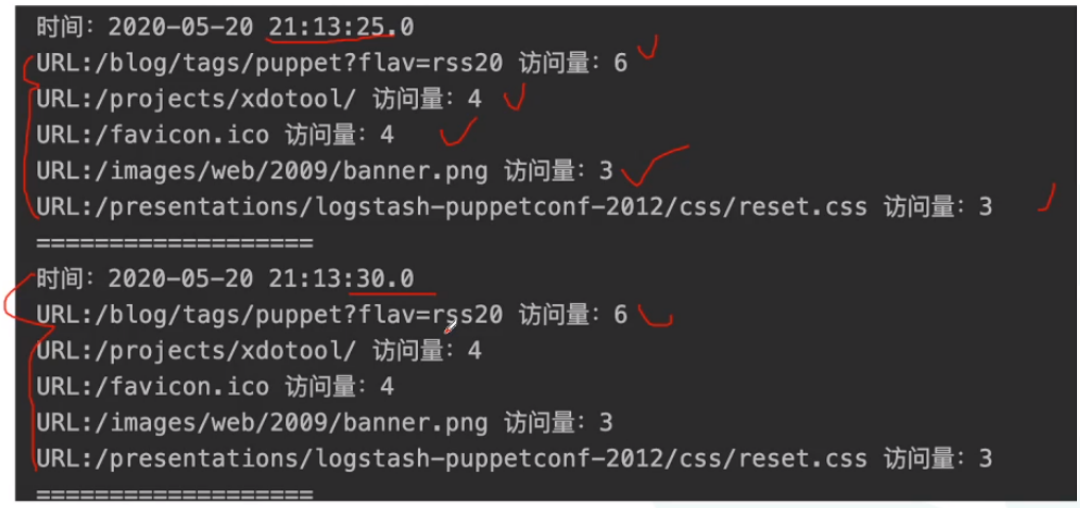
IP地址	用户ID	事件时间	请求方式	URL
24.233.162.179	-	2020-12-10 11:11:11	GET	/images/jordan-80.png

其中：**事件时间**和**URL**对于这个需求是核心

3. 实现思路



4. 展示效果

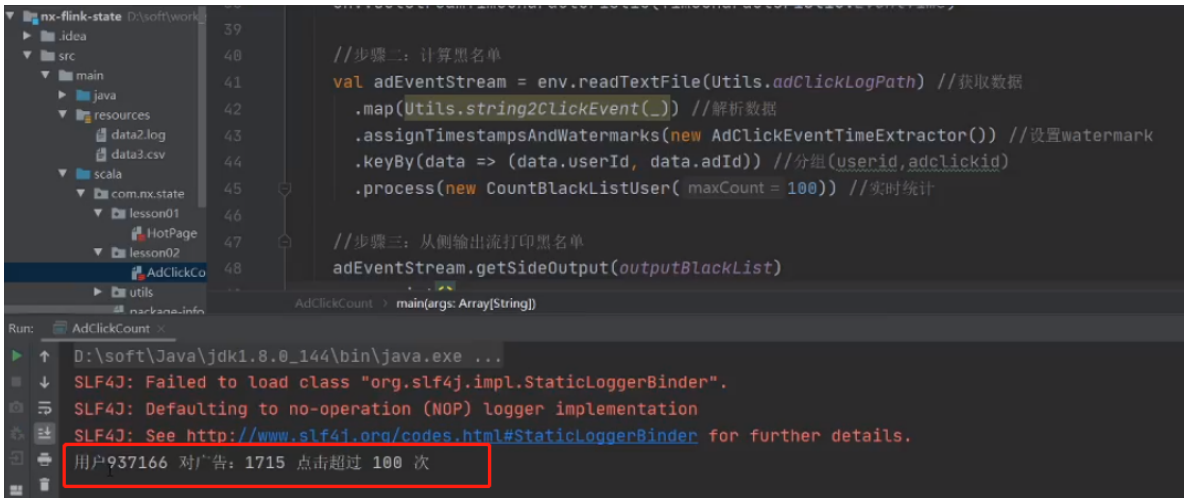


需求二：实时统计广告点击（实时计算黑名单）

1. 需求分析：同一天，同一个用户，同一个广告被点击100次即为黑名单
2. 数据展示

3. 实现思路

4. 展示效果



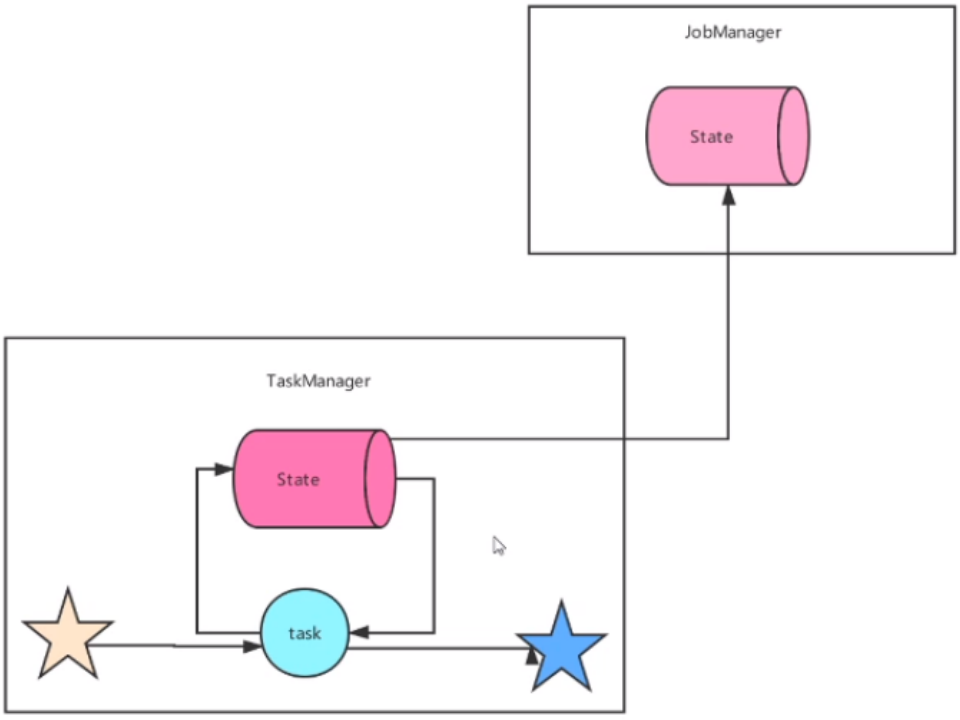
4.1 State Backend

4.1.1 概述

Flink支持的State Backend:

- *MemoryStateBackend* 默认的state的类型就是这种
- *FsStateBackend*
- *RocksDBStateBackend*

MemoryStateBackend



默认情况下，状态信息是存储在 TaskManager 的堆内存中的，checkpoint 的时候将状态保存到 JobManager 的堆内存中。

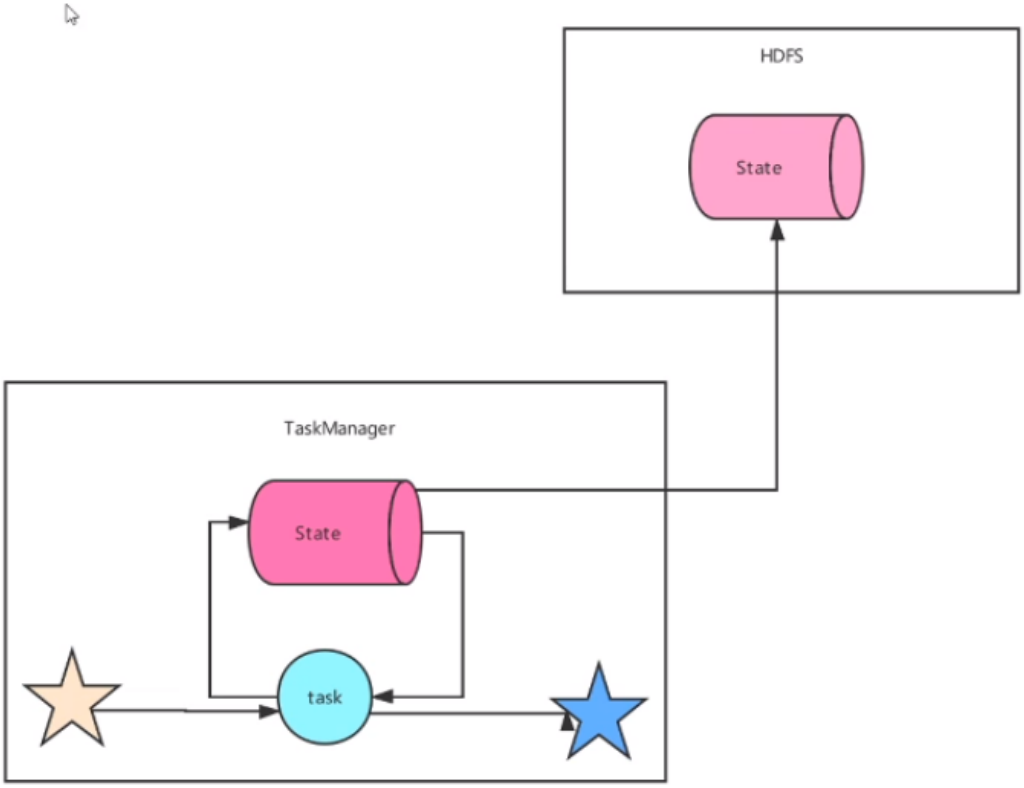
缺点：

- 只能保存数据量小的状态
- 状态数据有可能会丢失

优点：

- 开发测试很方便

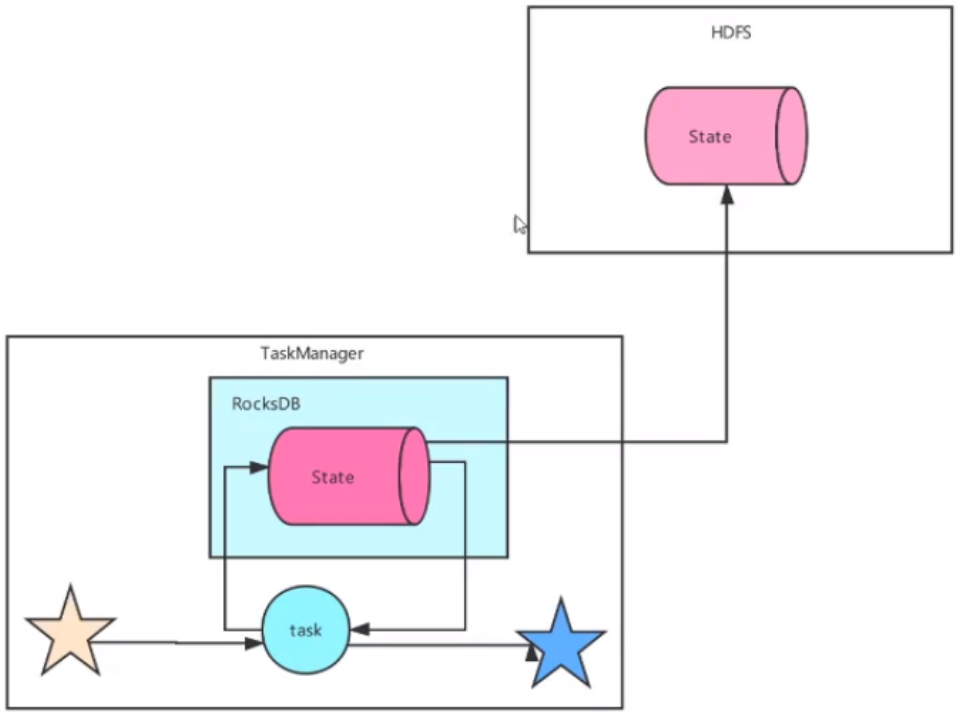
FSStateBackend



状态信息存储在 TaskManager 的堆内存中的，checkpoint 的时候将状态保存到指定的文件中 (HDFS 等文件系统)

```
缺点：
    状态大小受TaskManager内存限制(默认支持5M)
优点：
    状态访问速度很快
    状态信息不会丢失
用于：  生产，也可存储状态数据量大的情况
```

RocksDBStateBackend



状态信息存储在 RocksDB 数据库 (key-value 的数据存储服务)， 最终保存在本地文件中
checkpoint 的时候将状态保存到指定的文件中 (HDFS 等文件系统)

```
缺点：
    状态访问速度有所下降
优点：
    可以存储超大量的状态信息
    状态信息不会丢失
用于：  生产，可以存储超大量的状态信息
```

StateBackend配置方式

（1）单任务调整

```
修改当前任务代码
env.setStateBackend(new FsStateBackend("hdfs://namenode:9000/flink/checkpoints"));
或者new MemoryStateBackend()
或者new RocksDBStateBackend(filebackend, true);【需要添加第三方依赖】
```

```
如果使用RocksDBStateBackend方式，需要在pom.xml文件中，添加如下依赖
 <dependency>
            <groupId>org.apache.flink</groupId>
            <artifactId>flink-statebackend-rocksdb_2.11</artifactId>
            <version>${flink.version}</version>
  </dependency>
```

Java

（2）全局调整(不建议)

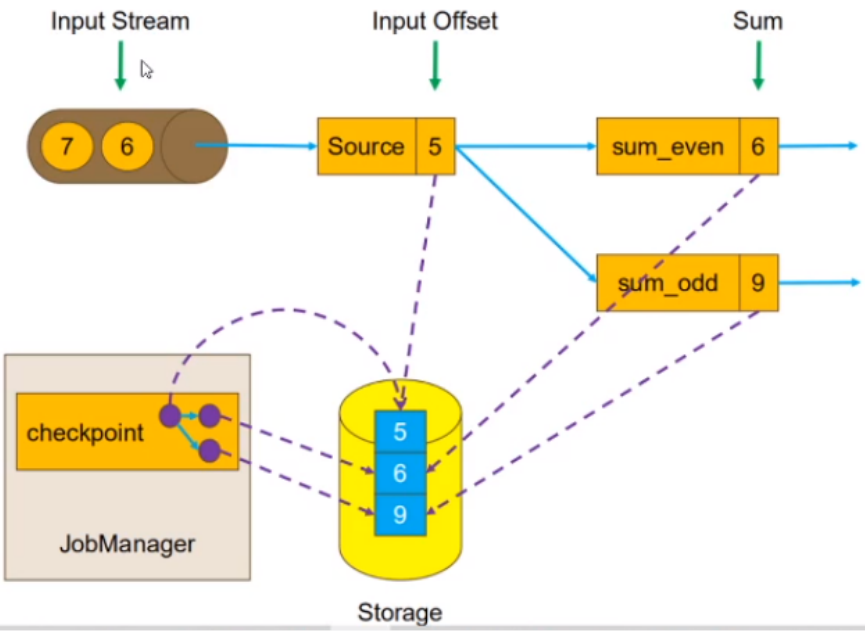
```
修改flink-conf.yaml
state.backend: filesystem
state.checkpoints.dir: hdfs://namenode:9000/flink/checkpoints
注意：state.backend的值可以是下面几种：jobmanager(MemoryStateBackend),
filesystem(FsStateBackend), rocksdb(RocksDBStateBackend)
```

4.2 Checkpoint原理

4.2.1 Checkpoint概述

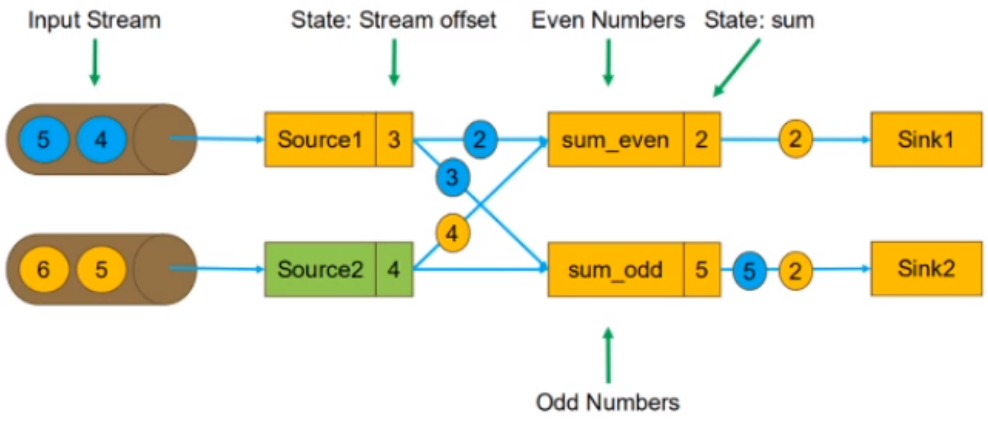
checkpoint机制是Flink可靠性的基石，可以保证Flink集群在某个算子因为某些原因(如 异常退出)出现故障时，能够将整个应用流图的状态恢复到故障之前的某一状态，保 证应用流图状态的一致性。

4.2.2 Checkpoint的简单想法



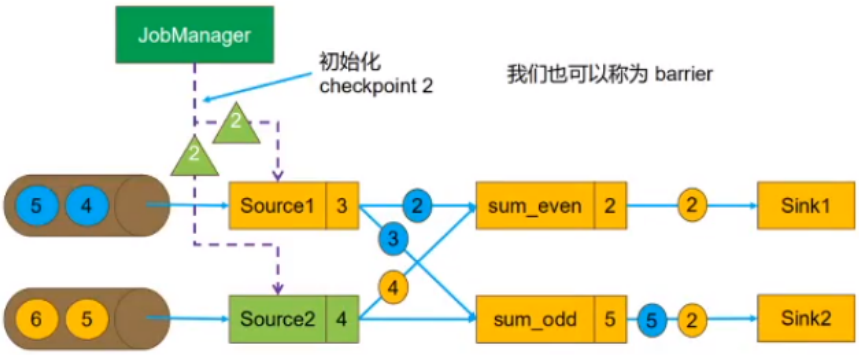
4.2.3 Chandy-Lamport 算法

1. 任务开启

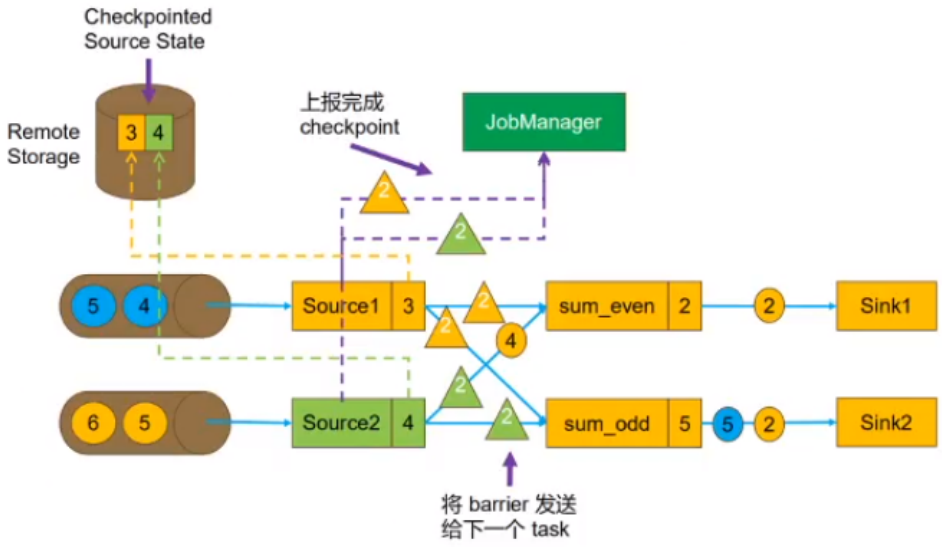


2. JobMananger发起Checkpoint

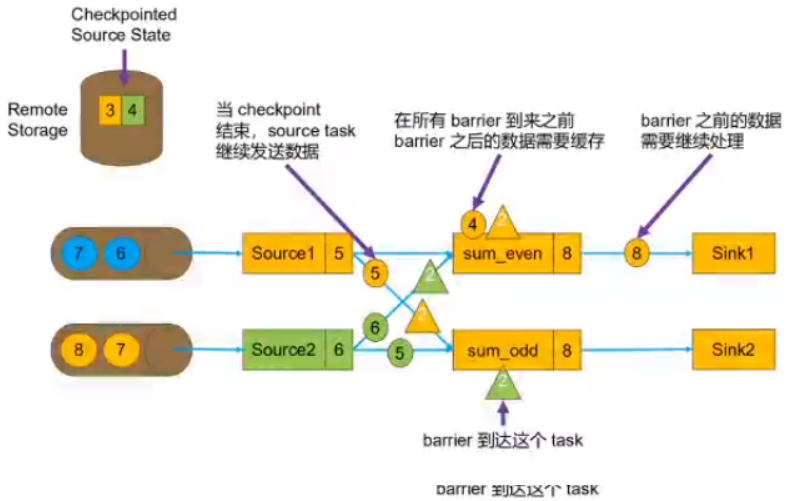
3.



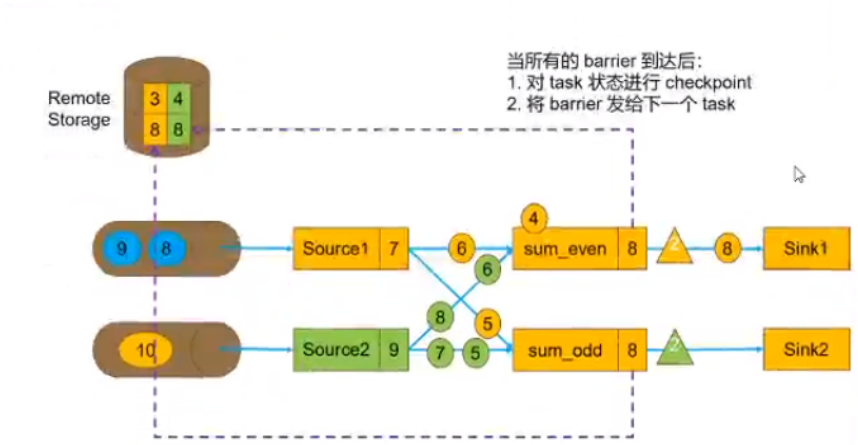
3.source上报checkpoint



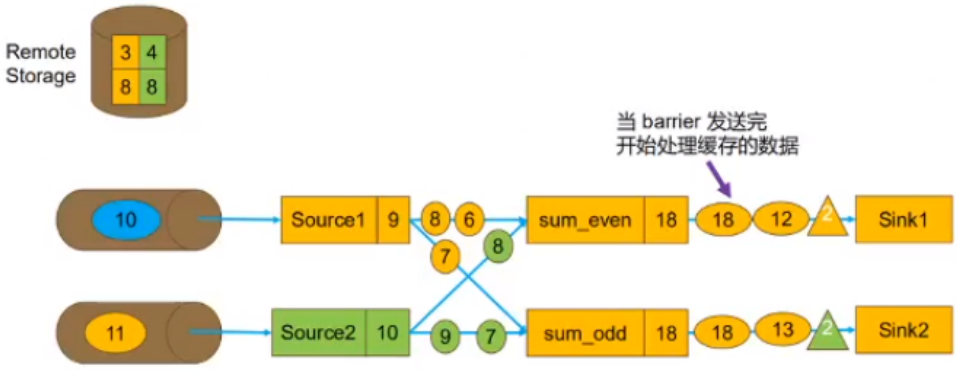
4. 数据处理



5. barrier对齐



6. 缓存数据处理



7. sink上报checkpoint