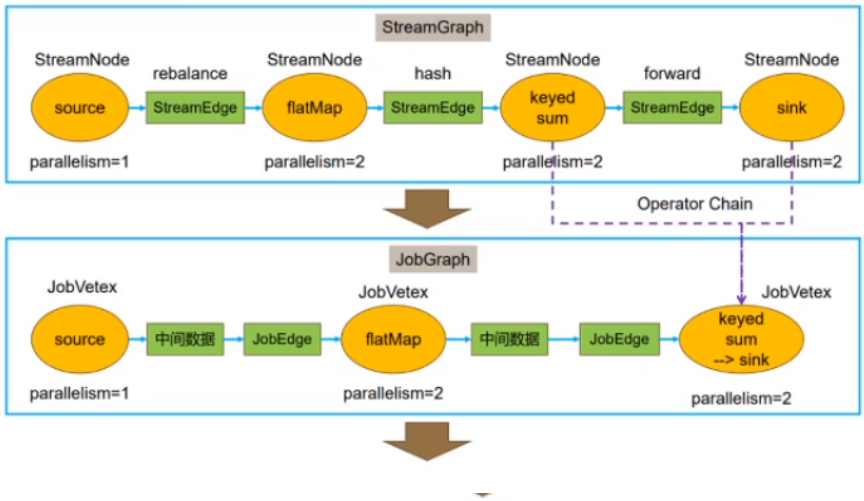
flink-study

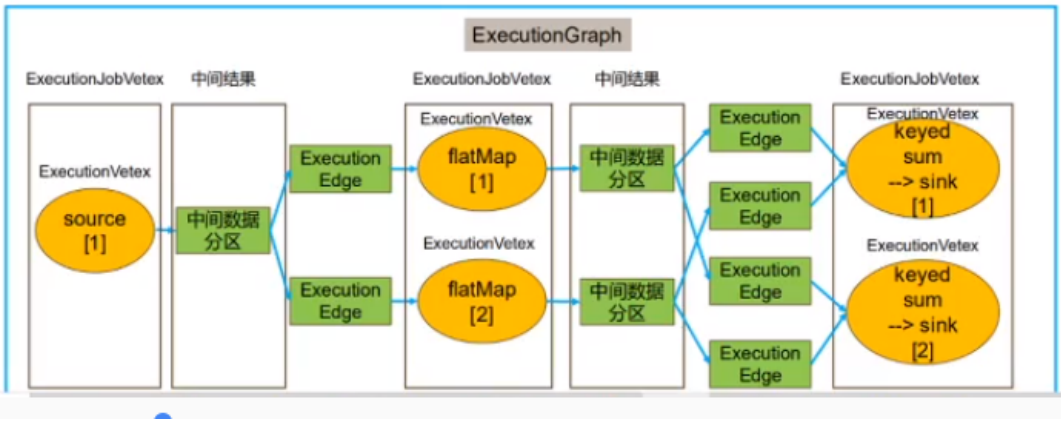
flink1.10 study

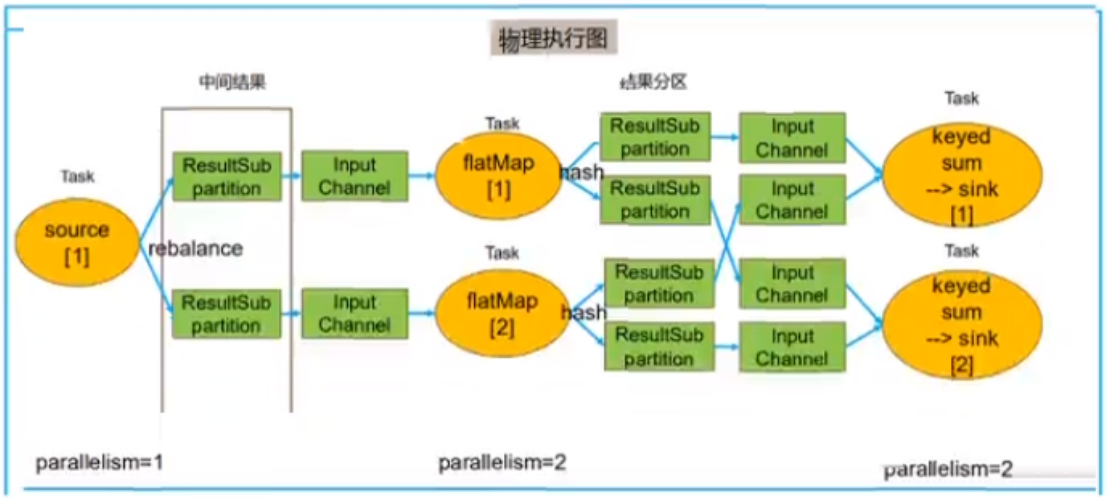
4.1.4 Flink四层模型（三）

Flink 代码开发就是要构建一个 dataflow，这个 dataflow 运行需要经历如下 4 个阶段：

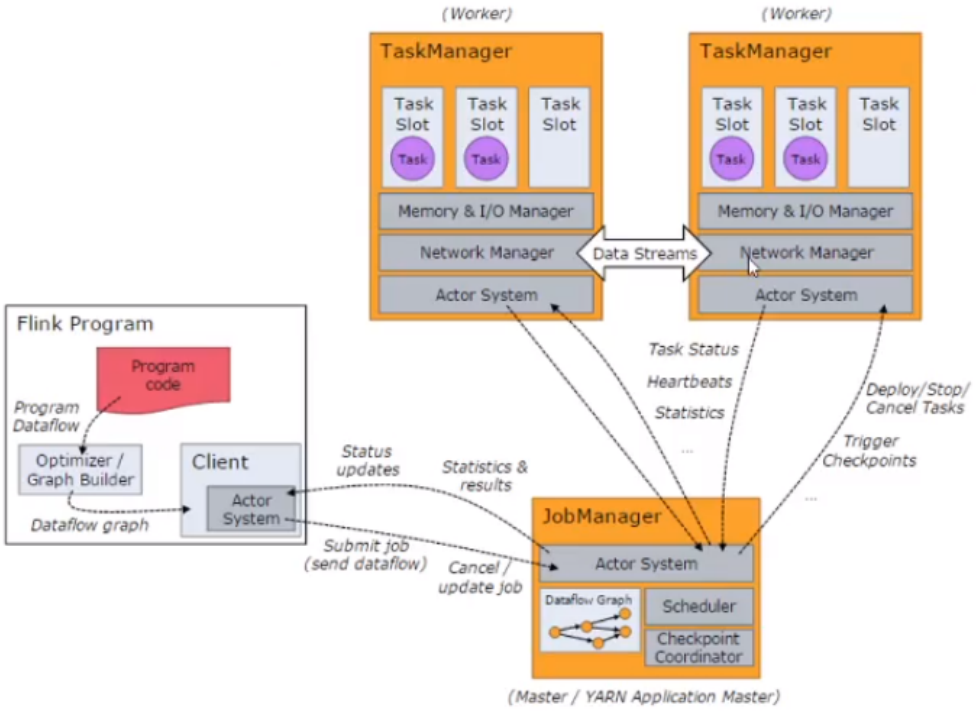
- ▶ Stream Graph
- ▶ Job Graph
- ▶ Execution Graph
- ▶ Physical Execution Graph







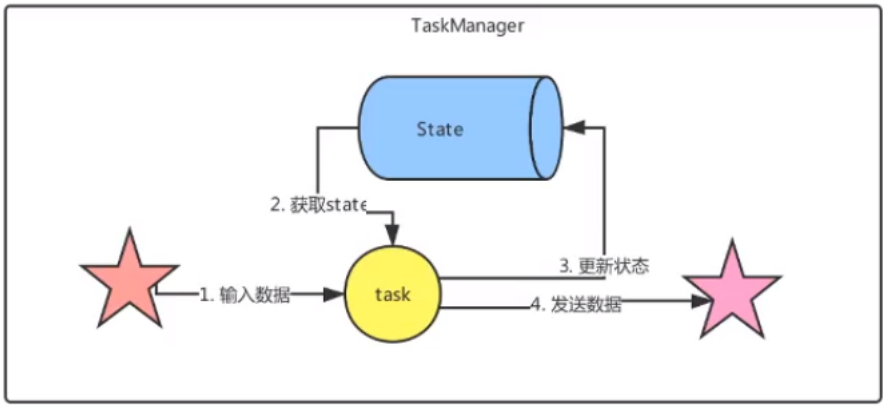
4.1.5 Flink任务运行流程



4.2 Flink State

4.1.1 state概述

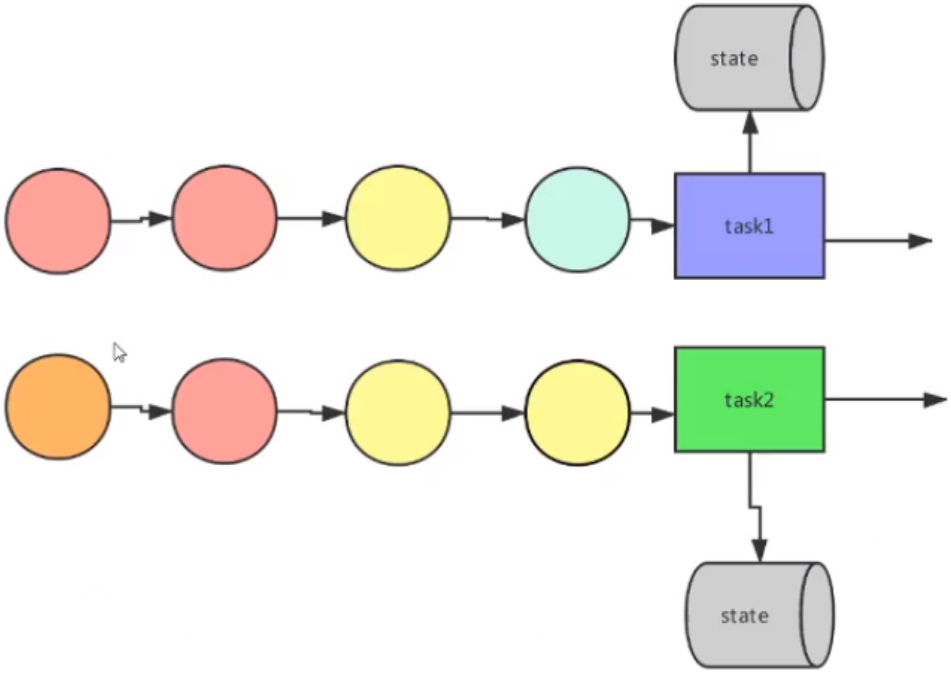
我们会发现，单词出现的次数有累计的效果。如果没有状态的管理，是不会有累计的效果的，所以Flink里面还有state的概念。



state：一般指一个具体的task/operator的状态。State可以被记录，在失败的情况下数据还可以恢复，Flink中有两种基本类型的State：Keyed State，Operator State，他们两种都可以以两种形式存在：原始状态(raw state)和托管状态(managed state).
托管状态：由Flink框架管理的状态，我们通常使用的就是这种。
原始状态：由用户自行管理状态具体的数据结构，框架在做checkpoint的时候，使用byte[]来读写状态内容，对其内部数据结构一无所知。通常在DataStream上的状态推荐使用托管的状态，当实现一个用户自定义的operator时，**会使用到原始状态**。但是我们工作中一般不常用，所以我们不考虑他。

State类型

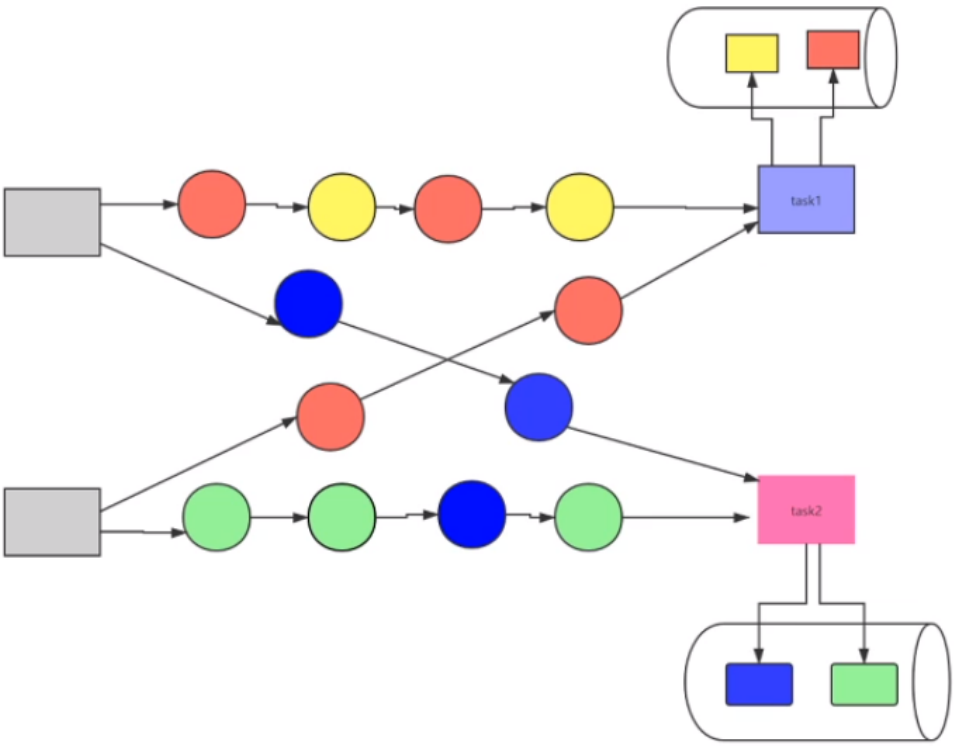
1. Operator State



state是task级别的state，说白了就是每个task对应一个state

Kafka Connector source中的每个分区（task）都需要记录消费的topic的partition和offset等信息。

2.Keyed State



1. keyed state 记录的是每个key的状态
2. Keyed state托管状态有五种类型：
   1. ValueState
   2. ListState
   3. MapState
   4. ReducingState
   5. AggregatingState

4.1.3 Keyed State演示

ValueState

```
/**
 * 需求：当接收到的相同 key 的元素个数等于 3个就计算这些元素的 value 的平均值。
 * 计算keyed stream中每3个元素的 value 的平均值
 */
```

4.1.3 Keyed State演示

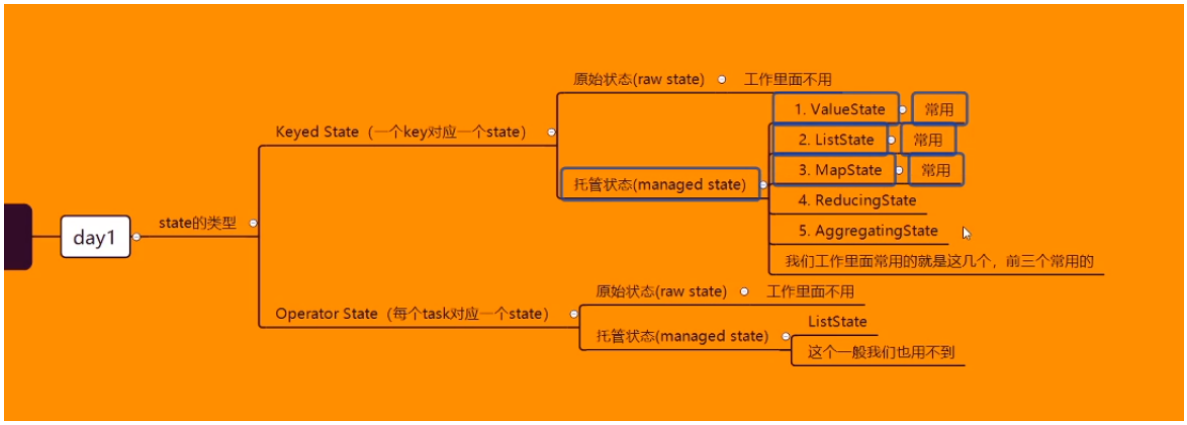
ValueState

```
/**
 * 需求：当接收到的相同 key 的元素个数等于 3个就计算这些元素的 value 的平均值。
 * 计算keyed stream中每3个元素的 value 的平均值
 */
public class TestKeyedStateMain {
    public static void main(String[] args) throws  Exception{
        StreamExecutionEnvironment env =
StreamExecutionEnvironment.getExecutionEnvironment();
        env.setParallelism(16);

        DataStreamSource<Tuple2<Long, Long>> dataStreamSource =
                env.fromElements(
                        Tuple2.of(1L, 3L),
                        Tuple2.of(1L, 7L),
                        Tuple2.of(2L, 4L),
                        Tuple2.of(1L, 5L),
                        Tuple2.of(2L, 2L),
                        Tuple2.of(2L, 6L));

        // 输出：
        //(1,5.0)
        //(2,4.0)
        dataStreamSource
                .keyBy(0)
                .flatMap(new CountAverageWithValueState())
                .print();

        env.execute("TestStatefulApi");
    }
}
```



4.1.5 State总结

1. 是否存在当前处理的 key（current key）：operator state 是没有当前 key 的概念，而 keyed state的数值总是与一个 current key 对应。
2. 存储对象是否 on heap: 目前 operator state backend 仅有一种 on-heap 的实现；而 keyed state backend 有 on-heap 和 off-heap（RocksDB）的多种实现。
3. 是否需要手动声明快照（snapshot）和恢复 (restore) 方法：operator state 需要手动实现 snapshot 和 restore 方法；而 keyed state 则由 backend 自行实现，对用户透明。
4. 数据大小：一般而言，我们认为 operator state 的数据规模是比较小的；认为 keyed state 规模是相对比较大的。需要注意的是，这是一个经验判断，不是一个绝对的判断区分标准。

五 常见面试题

1. Flink任务的运行流程
2. Flink的三层模型
3. Flink的数据传输策略

六 思考题

1. Flink的checkpoint算法原理
2. Flink的State backend的类型和特点
3. Flink相比SparkStreaming的优势

七 扩展

五 常见面试题

1. Flink任务的运行流程
2. Flink的三层模型
3. Flink的数据传输策略

六 思考题

1. Flink的checkpoint算法原理
2. Flink的State backend的类型和特点
3. Flink相比SparkStreaming的优势

七 扩展

1. Flink State TTL 概述 https://cloud.tencent.com/developer/article/1452844

Flink State TTL 概述

https://cloud.tencent.com/developer/article/1452844

🔎思考题：
1、State的数据存到了哪儿?
2、 Flink相比SparkStreaming优势在于哪儿?