# Flink 运行架构

左元

2021 年 4 月 25 日

尚硅谷大数据组

# 主要内容

- Flink 运行时的组件
- 任务提交流程
- 任务调度原理

# Flink 架构

- Flink 运行时由两种类型的进程组成：一个 JobManager 和一个或者多个 TaskManager。
- 典型的 Master-Slave 架构。

# Flink 架构


**图 1:** Flink 架构

# 作业管理器 (JobManager)

- 控制一个应用程序执行的主进程，也就是说，每个应用程序都会被一个不同的 JobManager 所控制执行。
- JobManager 会先接收到要执行的应用程序，这个应用程序会包括：作业图 (JobGraph)、逻辑数据流图和打包了所有的类、库和其它资源的 JAR 包。
- JobManager 会把 JobGraph 转换成一个物理层面的数据流图，这个图被叫做“执行图”(ExecutionGraph)，包含了所有可以并发执行的任务。
- JobManager 会向资源管理器 (Flink 的资源管理器) 请求执行任务必要的资源，也就是任务管理器 (TaskManager) 上的任务插槽（slot）。一旦它获取到了足够的资源，就会将执行图 (DAG) 分发到真正运行它们的 TaskManager 上。而在运行过程中，JobManager 会负责所有需要中央协调的操作，比如说检查点 (checkpoints) 的协调。

# 作业管理器 (JobManager)

- 资源管理器 (ResourceManager)：ResourceManager 负责 Flink 集群中的资源提供、回收、分配 - 它管理 task slots，这是 Flink 集群中资源调度的单位。Flink 为不同的环境和资源提供者（例如 YARN、Mesos、Kubernetes 和 standalone 部署）实现了对应的 ResourceManager。在 standalone 设置中，ResourceManager 只能分配可用 TaskManager 的 slots，而不能自行启动新的 TaskManager。
- 分发器 (Dispatcher)：Dispatcher 提供了一个 REST 接口，用来提交 Flink 应用程序执行，并为每个提交的作业启动一个新的 JobMaster。它还运行 Flink WebUI 用来提供作业执行信息。
- JobMaster：JobMaster 负责管理单个 JobGraph 的执行。Flink 集群中可以同时运行多个作业，每个作业都有自己的 JobMaster。

# 资源管理器

- 主要负责管理任务管理器（TaskManager）的插槽（slot），TaskManager 插槽是 Flink 中定义的处理资源单元。
- Flink 为不同的环境和资源管理工具提供了不同资源管理器，比如 YARN、Mesos、Kubernetes（管理 docker 容器组成的集群），以及 Standalone（独立集群）部署。
- 当 JobManager 申请插槽资源时，Flink 的资源管理器会将有空闲插槽的 TaskManager 分配给 JobManager。如果 Flink 的资源管理器没有足够的插槽来满足 JobManager 的请求，它还可以向 Yarn 的资源管理器发起会话，以提供启动 TaskManager 进程的容器。

# 分发器

- 可以跨作业运行，它为应用提交提供了 RESTful 接口 (GET/PUT/DELETE/POST)。
- 当一个应用被提交执行时，分发器就会启动并将应用移交给一个 JobManager。
- Dispatcher 也会启动一个 Web UI(localhost:8081)，用来方便地展示和监控作业执行的信息。
- Dispatcher 在架构中可能并不是必需的，这取决于应用提交运行的方式。

# 任务提交流程

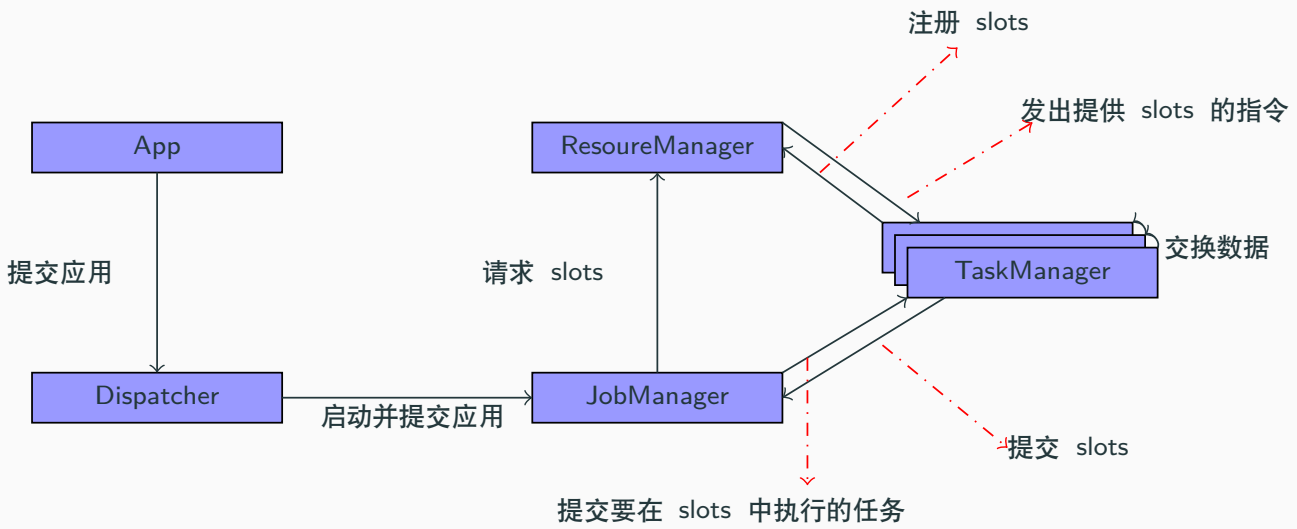



图 2: Flink 任务提交流程

# 任务调度原理

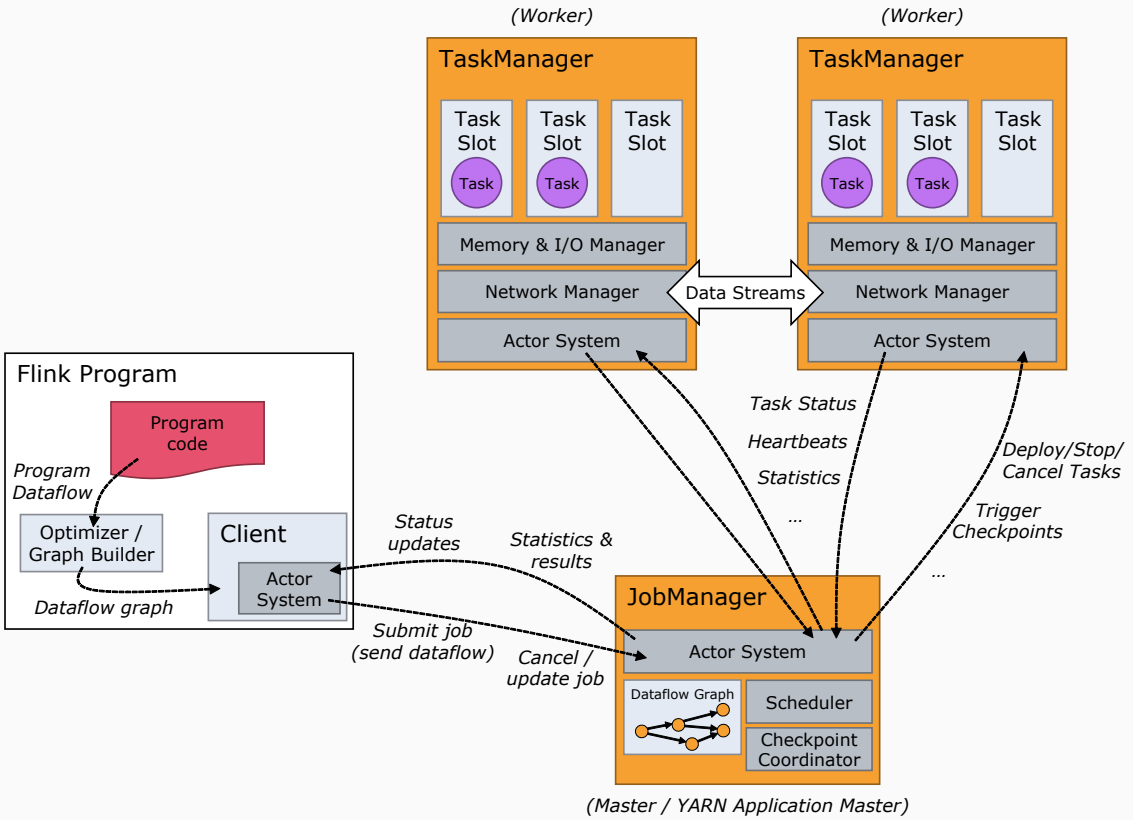



图 3: Flink 运行时的组件

# TaskManager 和 Slots

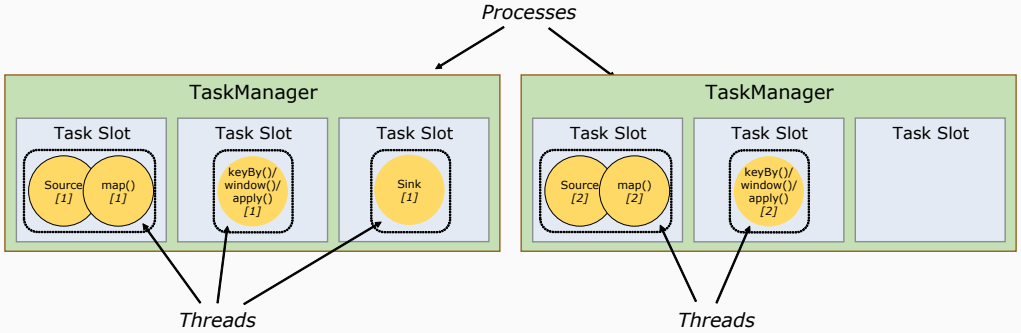



**图 4: 任务管理器和插槽**

- Flink 中每一个 TaskManager 都是一个 JVM 进程，每一个任务插槽都会启动一个线程，它可能会在独立的线程上执行一个或多个 subtask，每一个子任务占用一个任务插槽（Task Slot）
- 为了控制一个 TaskManager 能接收多少个 task，TaskManager 通过 task slot 来进行控制（一个 TaskManager 至少有一个 slot）

# TaskManager 和 Slots

- 默认情况下，Flink 允许子任务共享 slot。这样的结果是，一个 slot 可以保存作业的整个管道。
- Task Slot 是静态的概念，是指 TaskManager 具有的并发执行能力。

# TaskManager 和 Slots

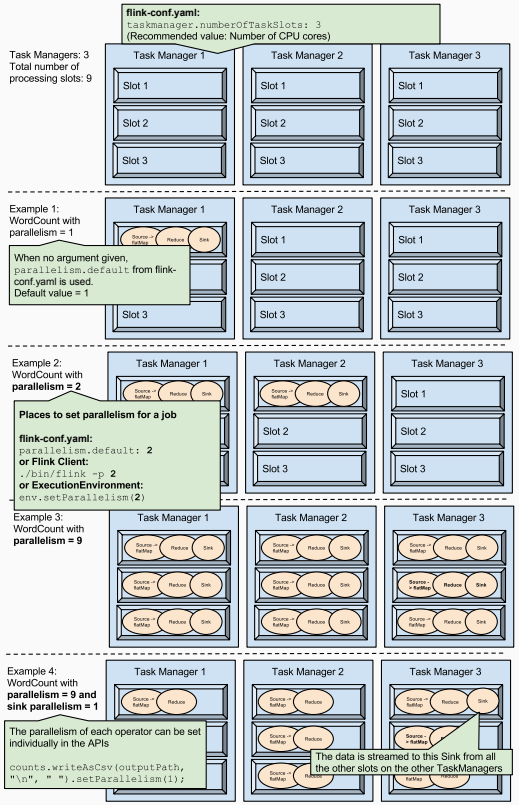



图 5: 并行度

# 程序与数据流 (DataFlow)

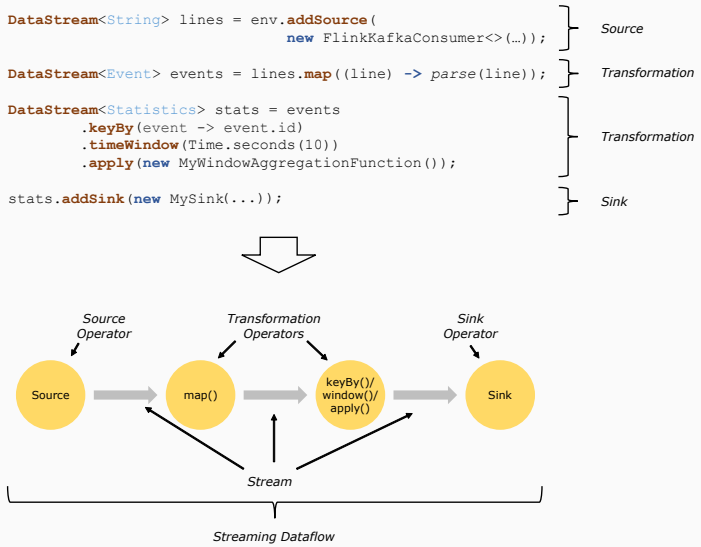



**图 6: 数据流**

- 所有的 Flink 程序都是由三部分组成的：Source、Transformation 和 Sink。
- Source 负责读取数据源，Transformation 利用各种算子进行处理加工，Sink 负责输出。

# 程序与数据流（DataFlow）

- 在运行时，Flink 上运行的程序会被映射成“逻辑数据流”（dataflows），它包含了这三部分
- 每一个 dataflow 以一个或多个 sources 开始以一个或多个 sinks 结束。dataflow 类似于任意的有向无环图（DAG）
- 在大部分情况下，程序中的转换运算（transformations）跟 dataflow 中的算子（operator）是一一对应的关系

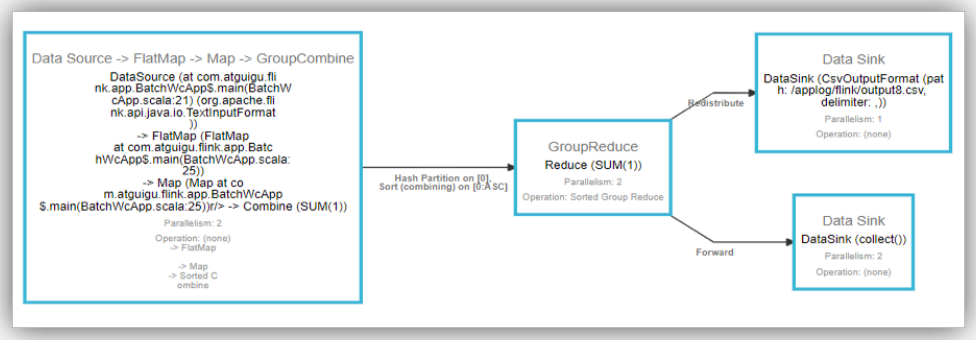
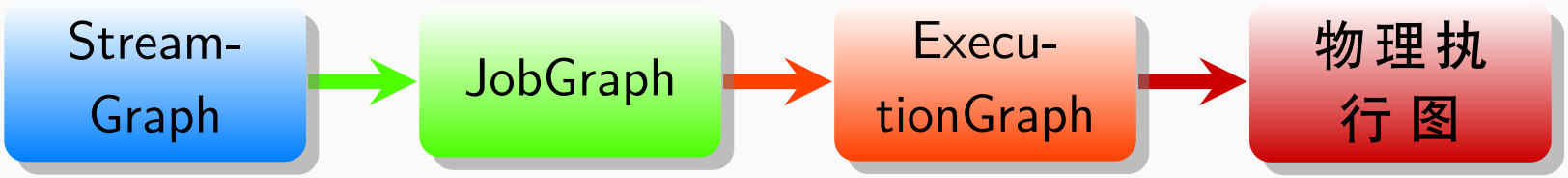



**图 7: 数据流**

# 图数据结构的转化




- StreamGraph：是根据用户通过 Stream API 编写的代码生成的最初的图。用来表示程序的拓扑结构。
- JobGraph：StreamGraph 在编译的阶段经过优化后生成了 JobGraph，提交给 JobManager 的数据结构。主要的优化为，将多个符合条件（窄依赖，没有 shuffle）的算子 chain 在一起作为一个节点。
- ExecutionGraph：JobManager 根据 JobGraph 生成 ExecutionGraph。ExecutionGraph 是 JobGraph 的并行化版本，是调度层最核心的数据结构。
- 物理执行图：JobManager 根据 ExecutionGraph 对 Job 进行调度后，在各个 TaskManager 上部署 Task 后形成的“图”，并不是一个具体的数据结构。

# 图数据结构的转化

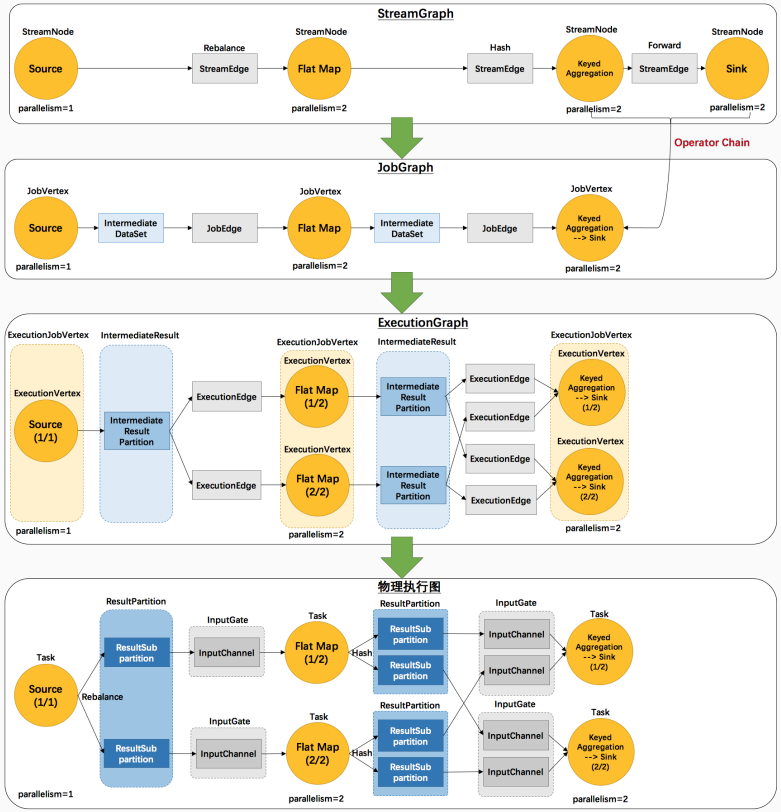



图 8: 图的转化

# 并行度 (Parallelism)

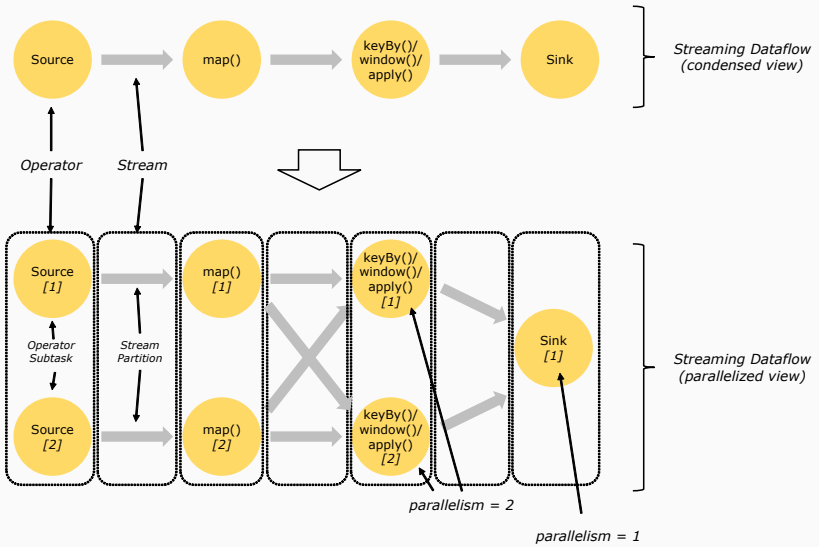



**图 9: 图的转化**

- 一个特定算子的子任务（subtask）的个数被称之为其并行度（parallelism）。一般情况下，一个 stream 的并行度，可以认为就是其所有算子中最大的并行度。

# 并行度 (Parallelism)

- 一个程序中，不同的算子可能具有不同的并行度
- 算子之间传输数据的形式可以是 one-to-one (forwarding) 的模式也可以是 redistributing 的模式，具体是哪一种形式，取决于算子的种类
  - One-to-one：stream 维护着分区以及元素的顺序（比如 source 和 map 之间）。这意味着 map 算子的子任务看到的元素的个数以及顺序跟 source 算子的子任务生产的元素的个数、顺序相同。map、filter、flatMap 等算子都是 one-to-one 的对应关系。
  - Redistributing：stream 的分区会发生改变。每一个算子的子任务依据所选择的 transformation 发送数据到不同的目标任务。例如，keyBy 基于 hashCode 重分区、而 broadcast 和 rebalance 会随机重新分区，这些算子都会引起 redistribute 过程，而 redistribute 过程就类似于 Spark 中的 shuffle 过程。

# 任务链 (Operator Chains)

- Flink 采用了一种称为任务链的优化技术，可以在特定条件下减少本地通信的开销。为了满足任务链的要求，必须将两个或多个算子设为相同的并行度，并通过本地转发（local forward）的方式进行连接
- 相同并行度的 one-to-one 操作，Flink 这样相连的算子链接在一起形成一个 task，原来的算子成为里面的 subtask
- 并行度相同、并且是 one-to-one 操作，两个条件缺一不可

# 并行度 (Parallelism)

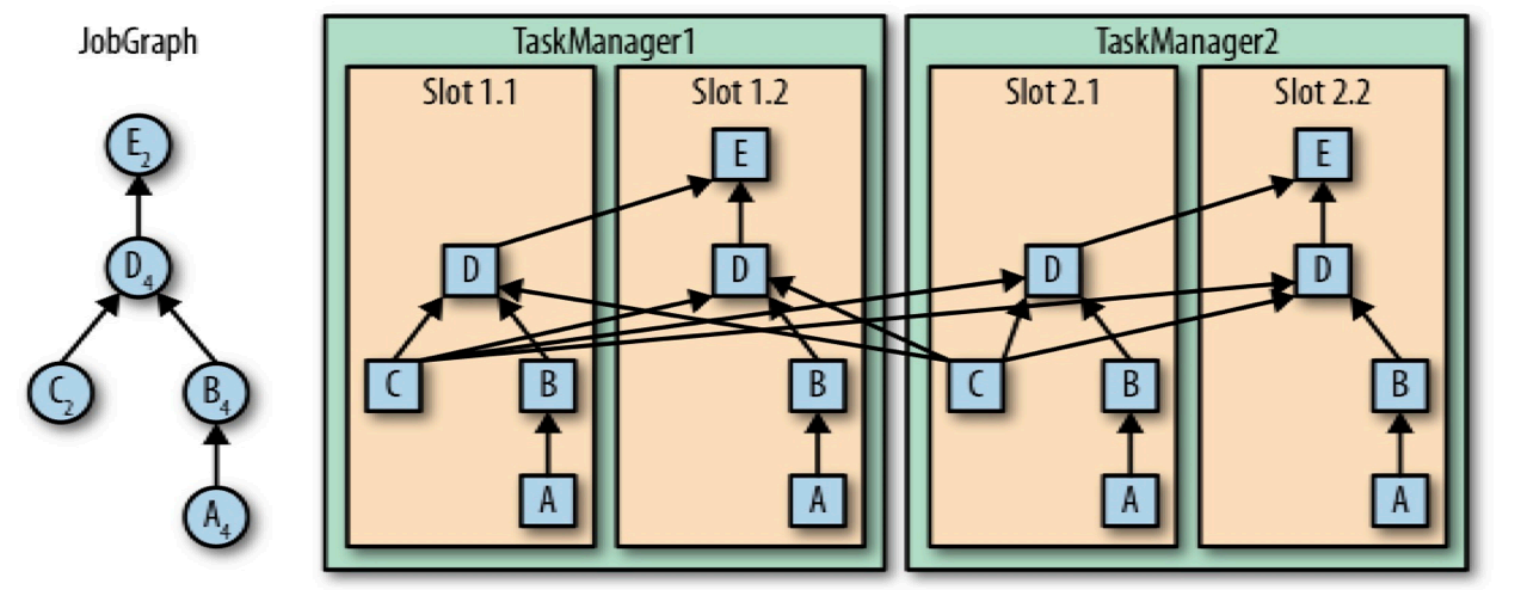



图 10: 并行度

# 任务链

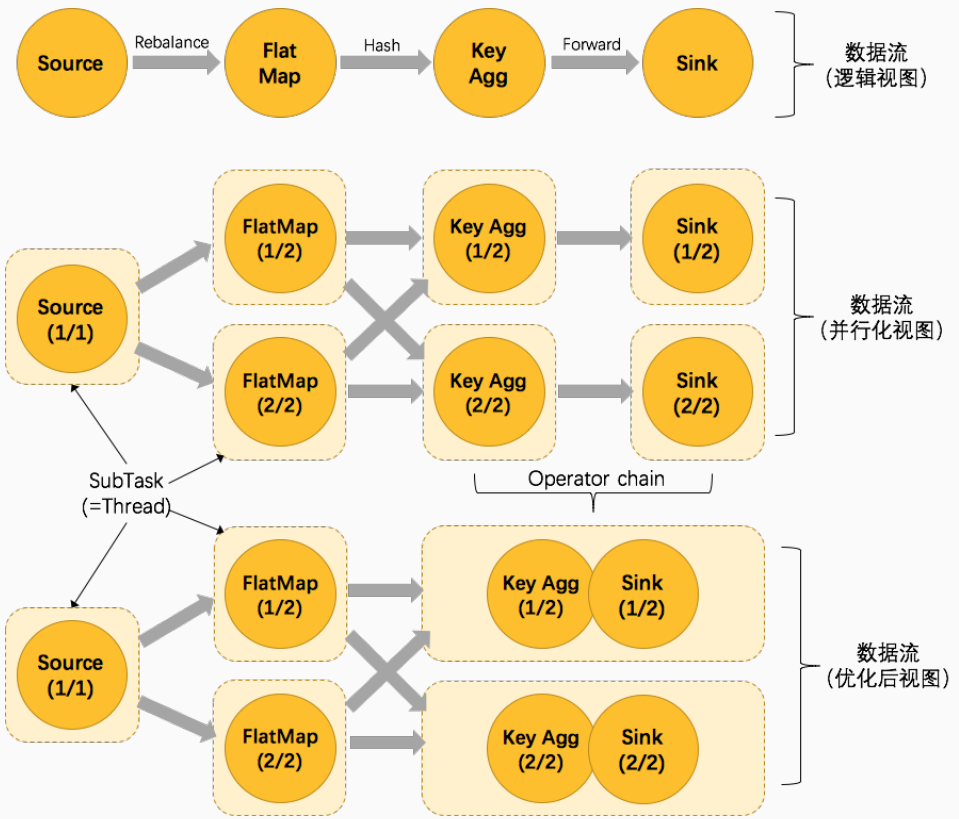

Source
Rebalance
Flat Map
Hash
Key Agg
Forward
Sink
数据流
（逻辑视图）
Source (1/1)
FlatMap (1/2)
FlatMap (2/2)
Key Agg (1/2)
Key Agg (2/2)
Sink (1/2)
Sink (2/2)
数据流
（并行化视图）
SubTask
(=Thread)
Operator chain
Source (1/1)
FlatMap (1/2)
FlatMap (2/2)
Key Agg (1/2)
Sink (1/2)
Key Agg (2/2)
Sink (2/2)
数据流
（优化后视图）

图 11: 任务链

# Q & A