Flink1.13.2 运行架构介绍——不入坑血亏！

本文作者：在 IT 中穿梭旅行

本文档来自公众号：3 分钟秒懂大数据

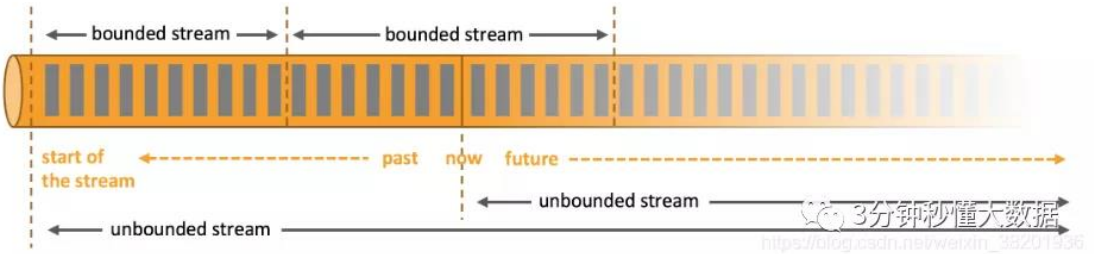
前言

大家好，我是土哥。

今天给大家讲一下 Flink 运行架构

1、Flink 程序结构

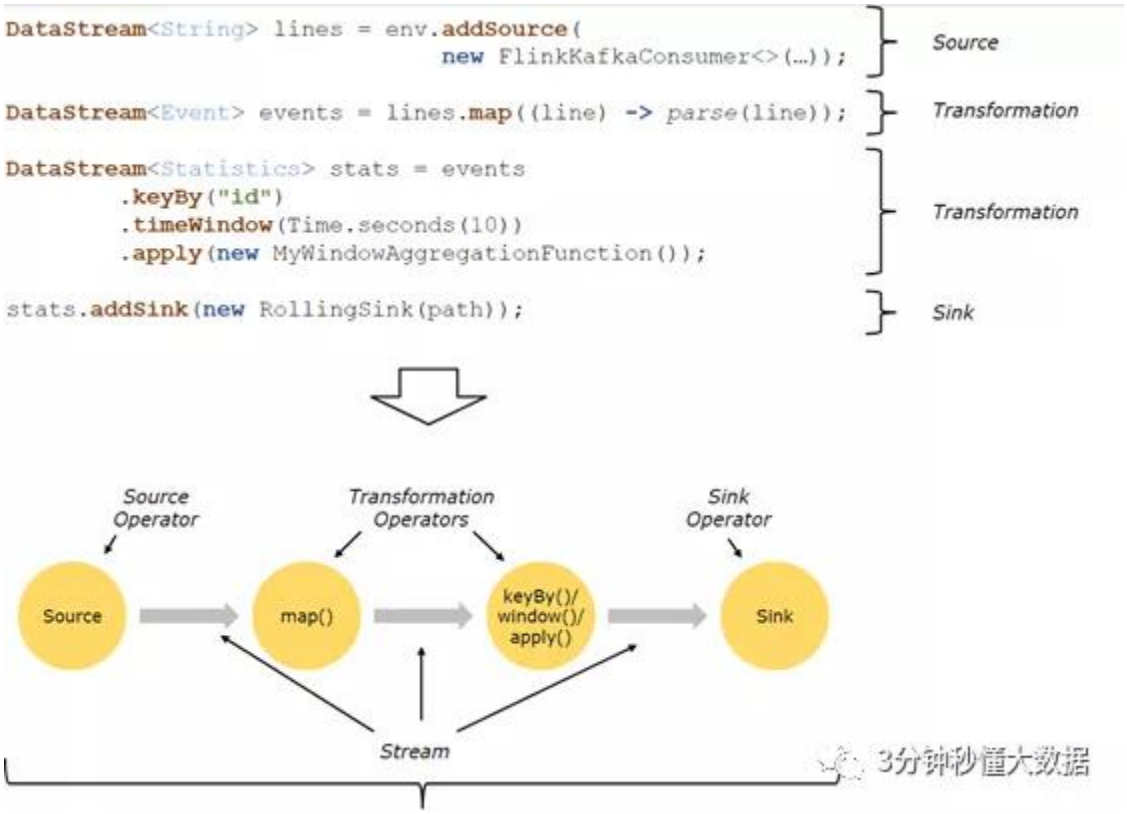
在自然环境中，数据的产生原本就是流式的。无论是来自 Web 服务器的事件数据，证券交易所的交易数据，还是来自工厂车间机器上的传感器数据，其数据都是流式的。但是当你分析数据时，可以围绕 有界流（bounded）或 无界流（unbounded）两种模型来组织处理数据，当然，选择不同的模型，程序的执行和处理方式也都会不同。



批处理 是有界数据流处理的范例。在这种模式下，你可以选择在计算结果输出之前输入整个数据集，这也就意味着你可以对整个数据集的数据进行排序、统计或汇总计算后再输出结果。

流处理 是无界数据流处理的范例。至少理论上来说，它的数据输入永远不会结束，因此程序必须持续不断地对到达的数据进行处理。

Flink 程序的基本构建块是流和转换（请注意，Flink 的 DataSet API 中使用的 DataSet 也是内部流 ）。从概念上讲，流是（可能永无止境的）数据记录流，而转换是将一个 或多个流作为一个或多个流的操作。输入，并产生一个或多个输出流。



Flink 应用程序结构就是如上图所示：

Source: 数据源。Flink 在流处理和批处理上的 source 大概有 4 类：

- 基于本地集合的 source
- 基于文件的 source
- 基于网络套接字的 source
- 自定义的 source

自定义的 source 常见的有 Apache kafka、RabbitMQ 等，当然你也可以定义自己的 source。

Transformation：数据转换的各种操作。包含：

Map / FlatMap / Filter / KeyBy / Reduce / Fold / Aggregations / Window / WindowAll / Union / Window join / Split / Select 等操作，操作很多，可以将数据转换计算成你想要的数据。

Sink：接收器。Flink 将转换计算后的数据发送的地点 ， 你可能需要存储下来，Flink 常见的 Sink 大概有如下几类：

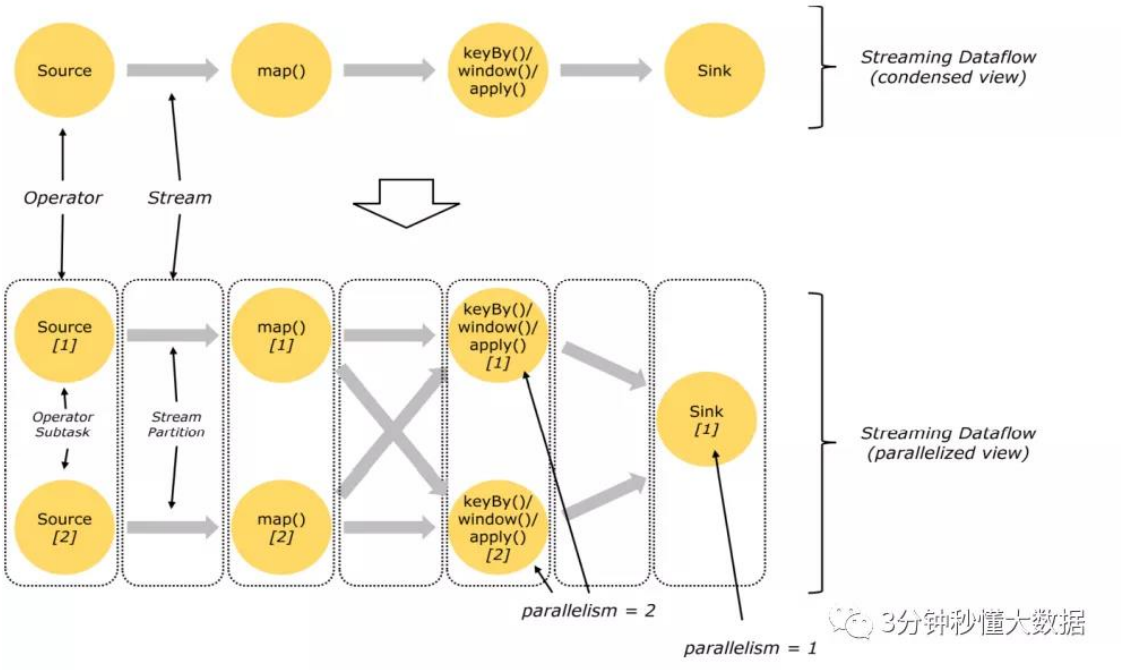
- 写入文件、打印出来
- 写入 socket
- 自定义的 sink 。

自定义的 sink 常见的有 Apache kafka、RabbitMQ、MySQL、ElasticSearch、Apache Cassandra、Hadoop FileSystem 等，同理你也可以定义自己的 sink。

2、Flink 并行数据流

Flink 程序在执行的时候，会被映射成一个 Streaming Dataflow。一个 Streaming Dataflow 是由一组 Stream 和 Transformation Operator 组成的。在启动时从一个或多个 Source Operator 开始，结束于一个或多个 Sink Operator。

Flink 程序本质上是并行的和分布式的，在执行过程中，一个流(stream)包含一个或多个流分区，而每一个 operator 包含一个或多个 operator 子任务。操作子任务间彼此独立，在不同的线程中执行，甚至是在不同的机器或不同的容器上。operator 子任务的数量是这一特定 operator 的并行度。相同程序中的不同 operator 有不同级别的并行度。



一个 Stream 可以被分成多个 Stream 的分区，也就是 Stream Partition。一个 Operator 也可以被分为多个 Operator Subtask。如上图中，Source 被分成 Source1 和 Source2 ，它们分别为 Source 的 Operator Subtask。每一个 Operator Subtask 都是在不同的线程当中独立执行的。一个 Operator 的并行度，就等于 Operator Subtask 的个数。

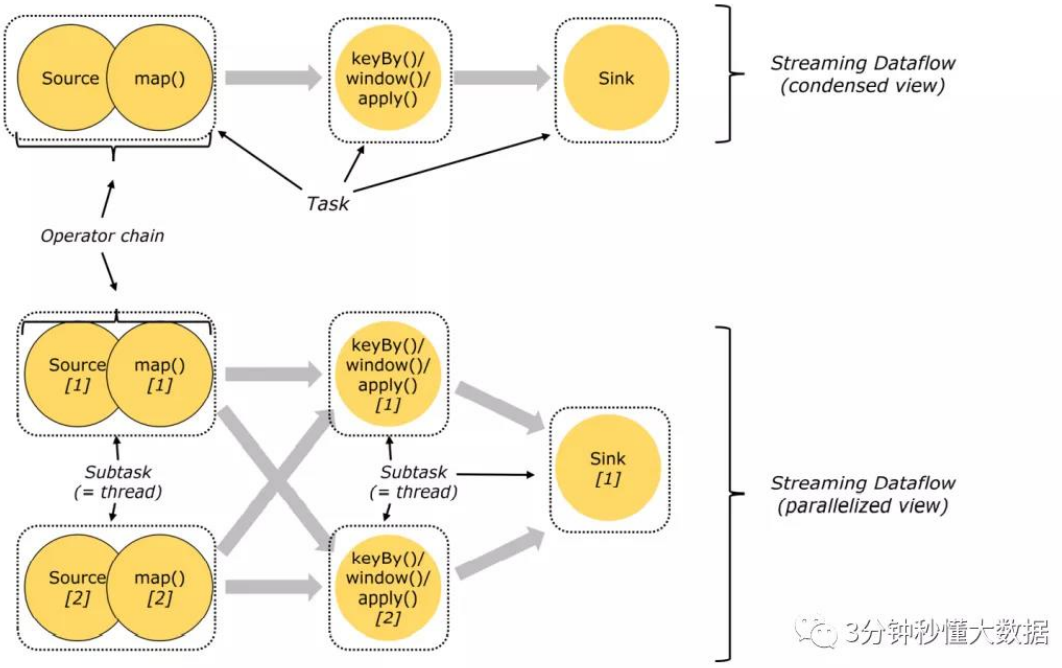
上图 Source 的并行度为 2。而一个 Stream 的并行度就等于它生成的 Operator 的并行度。数据在两个 operator 之间传递的时候有两种模式：

（1）One to One 模式：两个 operator 用此模式传递的时候，会保持数据的分区数和数据的 排序；如上图中的 Source1 到 Map1，它就保留的 Source 的分区特性，以及分区元素处理的有序性。

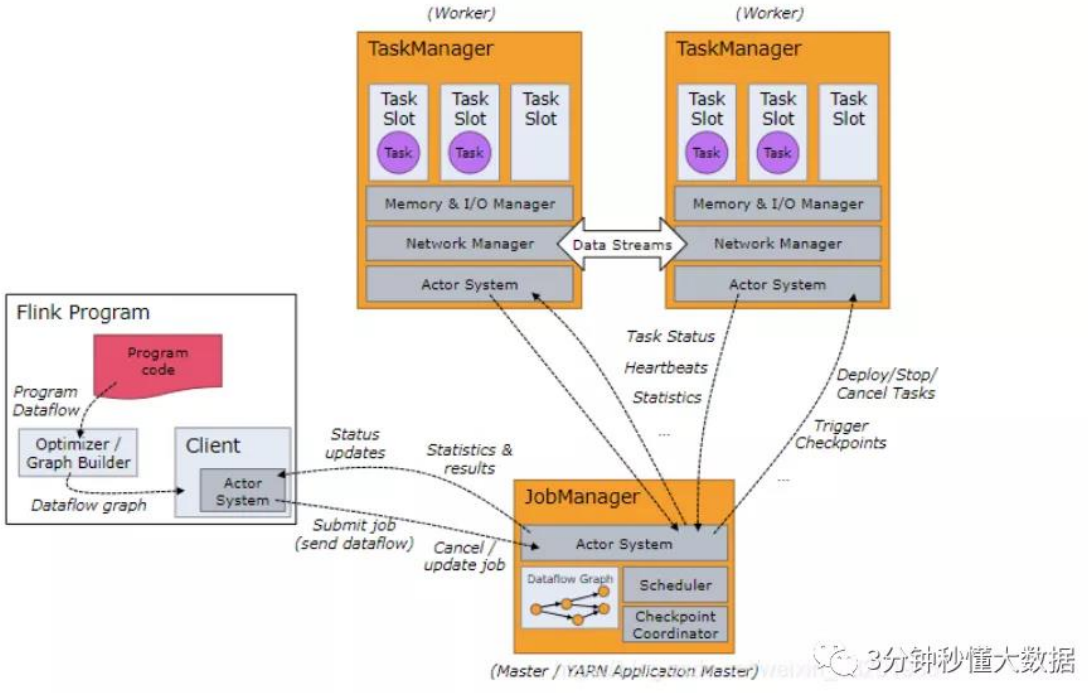
（2）Redistributing （重新分配）模式：这种模式会改变数据的分区数；每个 operator subtask 会根据选择 transformation 把数据发送到不同的目标 subtasks，比如 keyBy()会通过 hashcode 重新分区，broadcast()和 rebalance()方法会随机重新分区；

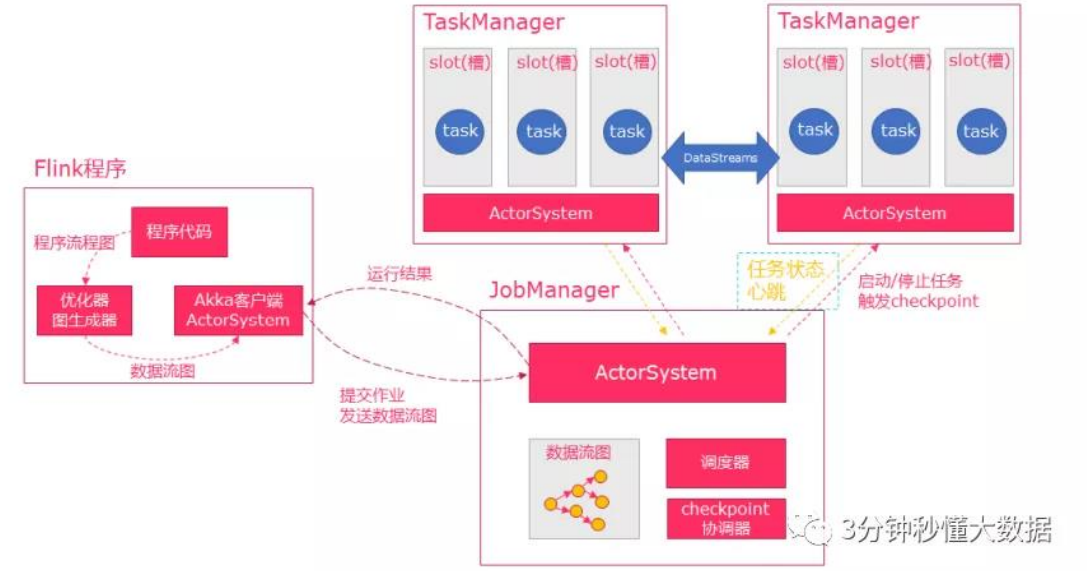
3、Task 和 Operator Chain

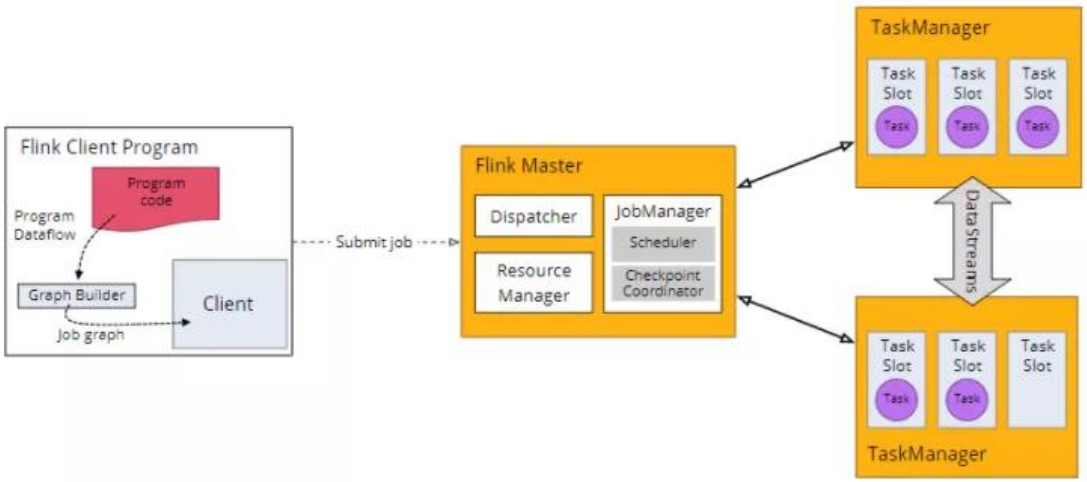
Flink 的所有操作都称之为 Operator，客户端在提交任务的时候会对 Operator 进行优化操作，能进行合并的 Operator 会被合并为一个 Operator，合并后的 Operator 称为 Operator chain，实际上就是一个执行链，每个执行链会在 TaskManager 上一个独立的线程中执行。


Source
map()
keyBy()/
window()/
apply()
Sink
Streaming Dataflow
(condensed view)
Task
Operator chain
Source
[1]
map()
[1]
keyBy()/
window()/
apply()
[1]
Sink
[1]
Subtask
(= thread)
Subtask
(= thread)
Source
[2]
map()
[2]
keyBy()/
window()/
apply()
[2]
Streaming Dataflow
(parallelized view)
3分钟秒懂大数据

4、任务调度与执行


(Worker)
(Worker)
TaskManager
TaskManager
Task Slot
Task
Memory & I/O Manager
Network Manager
Actor System
Data Streams
Flink Program
Program code
Program Dataflow
Optimizer / Graph Builder
Client
Actor System
Dataflow graph
Status updates
Statistics & results
Submit job (send dataflow)
Cancel / update job
Task Status
Heartbeats
Statistics
Deploy/Stop/ Cancel Tasks
Trigger Checkpoints
JobManager
Actor System
Dataflow Graph
Scheduler
Checkpoint Coordinator
(Master / YARN Application Master)
3分钟秒懂大数据





以上三个图是一个图，Flink 执行流程如下：

1. 当 Flink 执行 executor 会自动根据程序代码生成 DAG 数据流图，即 Jobgraph；
2. ActorSystem 创建 Actor 将数据流图发送给 JobManager 中的 Actor；
3. JobManager 会不断接收 TaskManager 的心跳消息，从而可以获取到有效的 TaskManager；
4. JobManager 通过调度器在 TaskManager 中调度执行 Task（在 Flink 中，最小的调度单元就是 task，对应就是一个线程）；

5. 在程序运行过程中，task 与 task 之间是可以进行数据传输的 。

Job Client

- 主要职责是提交任务, 提交后可以结束进程, 也可以等待结果返回 ；
- Job Client 不是 Flink 程序执行的内部部分，但它是任务执行的起点；
- Job Client 负责接受用户的程序代码，然后创建数据流，将数据流提交给 JobManager 以便进一步执行。执行完成后，Job Client 将结果返回给用户。

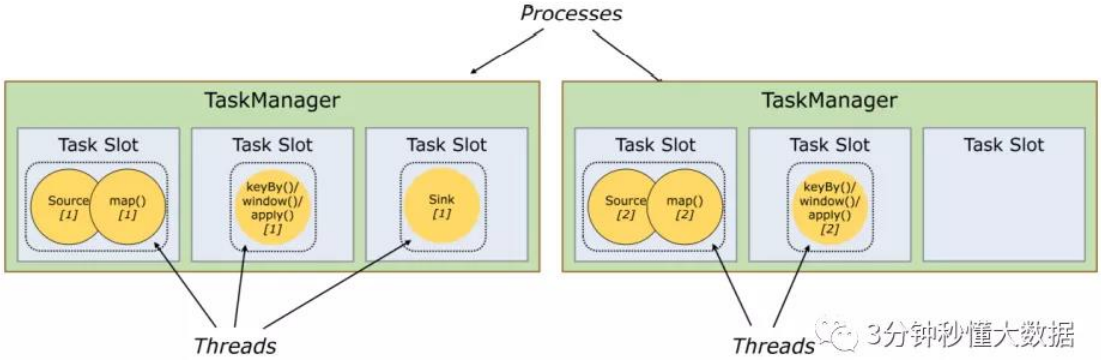
JobManager

- 主要职责是调度工作并协调任务做检查点；
- 集群中至少要有一个 master，master 负责调度 task，协调 checkpoints 和 容错；
- 高可用设置的话可以有多个 master，但要保证一个是 leader, 其他是 stand by；
- Job Manager 包含 Actor System、Scheduler、CheckPoint 三个重要的组件 ；
- JobManager 从客户端接收到任务以后, 首先生成优化过的执行计划, 再调度到 TaskManager 中执行。

TaskManager

- 主要职责是从 JobManager 处接收任务, 并部署和启动任务, 接收上游的数 据并处理
- Task Manager 是在 JVM 中的一个或多个线程中执行任务的工作节点。
- TaskManager 在创建之初就设置好了 Slot, 每个 Slot 可以执行一个任务。

5、任务槽（task slot ）和槽共享（Slot sharing）



每个 TaskManager 是一个 JVM 的进程, 可以在不同的线程中执行一个或多个子任务。为了控制一个 worker 能接收多少个 task。worker 通过 task slot 来进行控制（一个 worker 至少有一个 task slot）。

1、任务槽

每个 task slot 表示 TaskManager 拥有资源的一个固定大小的子集。

一般来说：我们分配槽的个数都是和 CPU 的核数相等，比如 8 核，那么就分配 8 个槽。

Flink 将进程的内存划分到多个 slot 中。

图中有 2 个 TaskManager，每个 TaskManager 有 3 个 slot，每个 slot 占有 1/3 的内存。

内存被划分到不同的 slot 之后可以获得如下好处：

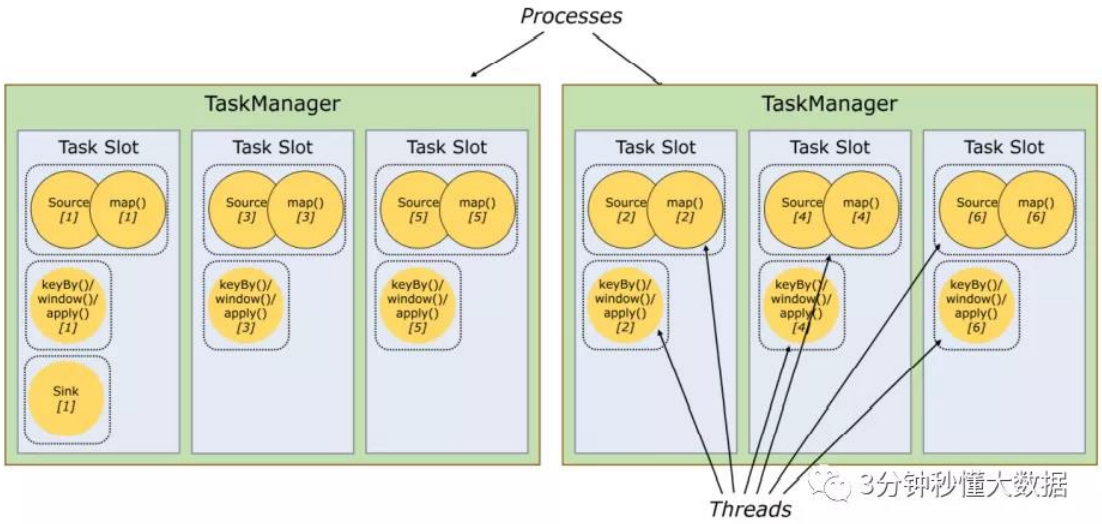
- TaskManager 最多能同时并发执行的任务是可以控制的，那就是 3 个，因为不能超过 slot 的数量。 任务槽的作用就是分离任务的托管内存，不会发生 cpu 隔离。
- slot 有独占的内存空间，这样在一个 TaskManager 中可以运行多个不同的作业，作业之间不受影响。

总结：task slot 的个数代表 TaskManager 可以并行执行的 task 数。

2、槽共享

默认情况下，Flink 允许子任务共享插槽，即使它们是不同任务的子任务，只要它们来自同一个作业。结果是一个槽可以保存作业的整个管道。允许插槽共享有两个主要好处：

- 只需计算 Job 中最高并行度（parallelism）的 task slot。只要这个满足，其他的 job 也都能满足。
- 资源分配更加公平。如果有比较空闲的 slot 可以将更多的任务分配给它。图中若没有任务槽共享，负载不高的 Source/Map 等 subtask 将会占据许多资源，而负载较高的窗口 subtask 则会缺乏资源。
- 有了任务槽共享，可以将基本并行度（base parallelism）从 2 提升到 6。提高了分槽资源的利用率。同时它还可以保障 TaskManager 给 subtask 的分配的 slot 方案更加公平。



以上就是 Flink 运行架构的讲解内容！觉得好的，点赞，在看，分享三连击，谢谢！！！

最近整理了一份计算机类的书籍，包含 python、java、大数据、人工智能、算法等，种类特别齐全。

获取方式：关注公众号：**3 分钟秒懂大数据**，回复：**福利**，就可以获得这份超级大礼！