16 | 分布式计算模式之Stream：一门背锅的艺术

2019-10-28 聂鹏程 来自北京

《分布式技术原理与算法解析》

你好，我是聂鹏程。今天，我来继续带你打卡分布式核心技术。

在上一篇文章中，我与你介绍了分布式计算模式中的 MapReduce 模式。这种模式的核心思想是，将大任务拆分成多个小任务，针对这些小任务分别计算后，再合并各小任务的结果以得到大任务的计算结果。

这种模式下任务运行完成之后，整个任务进程就结束了，属于短任务模式。但，任务进程的启动和停止是一件很耗时的事儿，因此 MapReduce 对处理实时性的任务就不太合适了。

实时性任务主要是针对流数据的处理，对处理时延要求很高，通常需要有常驻服务进程，等待数据的随时到来随时处理，以保证低时延。处理流数据任务的计算模式，在分布式领域中叫作 Stream。

今天，我将针对流数据的处理展开分享，和你一起打卡 Stream 这种计算模式。

什么是 Stream?

近年来，由于网络监控、传感监测、AR/VR 等实时性应用的兴起，一类需要处理流数据的业务发展了起来。比如各种直播平台中，我们需要处理直播产生的音视频数据流等。这种如流水般持续涌现，且需要实时处理的数据，我们称之为**流数据**。

总结来讲，流数据的特征主要包括以下 4 点：

数据如流水般持续、快速地到达；

海量数据规模，数据量可达到 TB 级甚至 PB 级；

对实时性要求高，随着时间流逝，数据的价值会大幅降低；

数据顺序无法保证，也就是说系统无法控制将要处理的数据元素的顺序。

在分布式领域中，处理流数据的计算模式，就是**流计算，也叫作 Stream**。这个名字是不是非常形象呢?

流计算的职责是实时获取来自不同数据源的海量数据，进行实时分析处理，获得有价值的信息。

它是一个对实时性要求非常高的计算形式，如果数据处理不及时，很容易导致过时、没用的结果，这时就需要对造成的后果进行“背锅”。从这个角度来说，Stream 可谓“一门背锅的艺术”。

类比于水流的持续不断且变幻莫测，流数据也是以大量、快速、时变的流形式持续在应用中产生，因此**流计算一般用于处理数据密集型应用**。

比如，百度、淘宝等大型网站中，每天都会产生大量的流数据，这些数据包括用户的搜索内容、用户的浏览记录等。实时采集用户数据，并通过流计算进行实时数据分析，可以了解每个时刻数据流的变化情况，甚至可以分析用户的实时浏览轨迹，从而进行个性化内容实时推荐，提高用户体验。

此外，我们常用的爱奇艺、腾讯等音视频平台，对电影、电视剧等数据的处理，也是采用了流计算模式。

那么，这种实时的流计算到底是如何运行的呢？接下来，我们就一起看看流计算的工作原理吧。

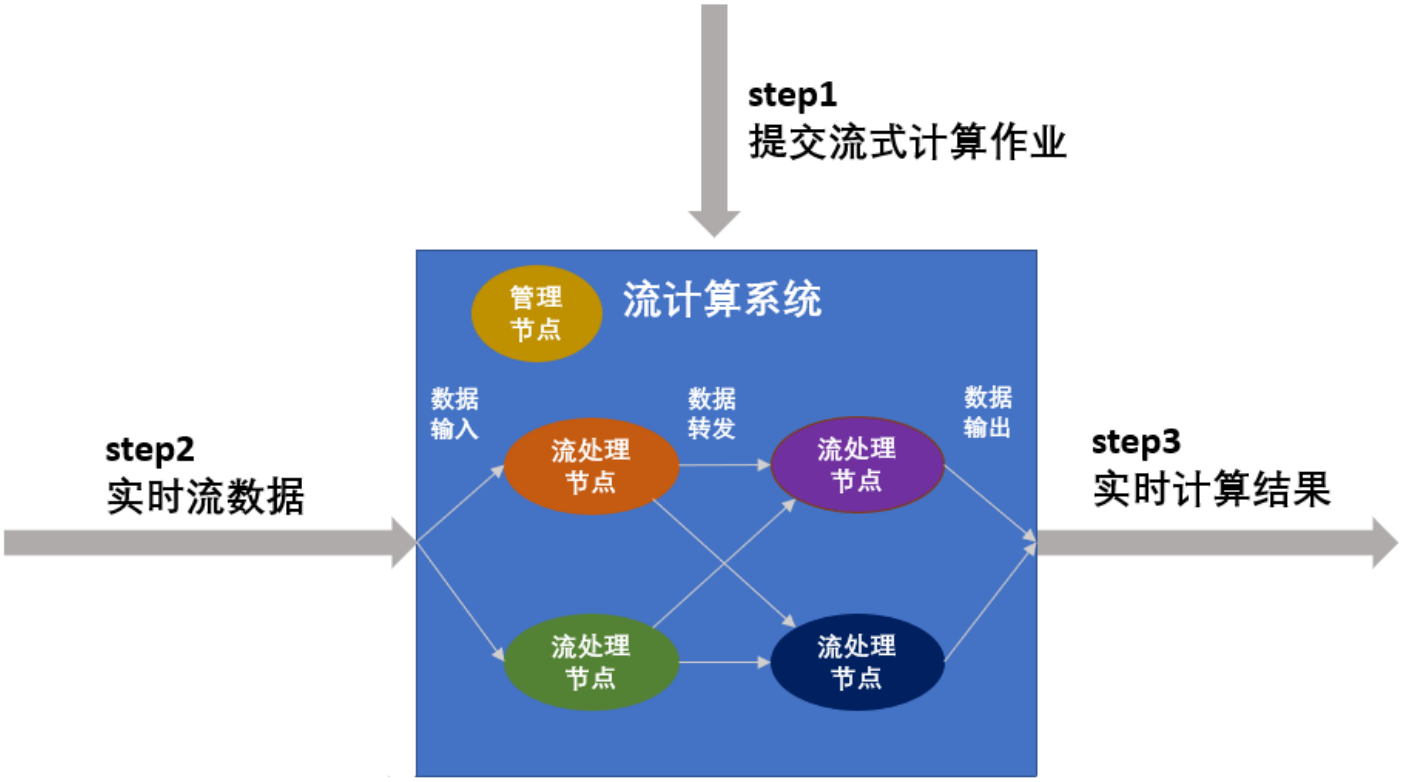
Stream 工作原理

我在上一篇文章中与你介绍的 MapReduce，是一种批量计算的形式。这种模式下，会先收集数据并将其缓存起来，等到缓存写满时才开始处理数据。因此，批量计算的一个缺点就是，从数据采集到得到计算结果之间经历的时间很长。

而流计算强调的是实时性，数据一旦产生就会被立即处理，当一条数据被处理完成后，会序列化存储到缓存中，然后立刻通过网络传输到下一个节点，由下一个节点继续处理，而不是像 MapReduce 那样，等到缓存写满才开始处理、传输。为了保证数据的实时性，在流计算中，不会存储任何数据，就像水流一样滚滚向前。

所以说，流计算属于持续性、低时延、事件驱动型的计算作业。

从这些分析中可以看出，**使用流计算进行数据处理，一般包括 3 个步骤**，如下图所示。



第一步，提交流式计算作业。流式计算作业是一种常驻计算服务，比如实时交通监测服务、实时天气预报服务等。对于流式计算作业，首先必须预先定义计算逻辑，并提交到流计算系统中，使得流计算系统知道自己该如何处理数据。

系统在整个运行期间，由于收集的是同一类型的数据、执行的是同一种服务，因此流式计算作业的处理逻辑不可更改。如果用户停止当前作业运行后再次提交作业，由于流计算不提供数据存储服务，因此之前已经计算完成的数据无法重新再次计算。

第二步，加载流式数据进行流计算。流式计算作业一旦启动将一直处于等待事件触发的状态，一旦有小批量数据进入流式数据存储，系统会立刻执行计算逻辑并迅速得到结果。

从上图中我们可以看出，在流计算系统中，有多个流处理节点，流处理节点会对数据进行预定义的处理操作，并在处理完后按照某种规则转发给后续节点继续处理。此外，流计算系统中还存在管理节点，主要负责管理处理节点以及数据的流动规则。其中，处理节点的个数以及数据转发的规则，都在第一步作业提交时定义。

第三步，持续输出计算结果。流式计算作业在得到小批量数据的计算结果后，可以立刻将结果数据写入在线 / 批量系统，无需等待整体数据的计算结果，以进一步做到实时计算结果的实时

展现。

到这里，我们小结一下吧。流计算不提供流式数据的存储服务，数据是持续流动的，在计算完成后就会立刻丢弃。流计算适用于需要处理持续到达的流数据、对数据处理有较高实时性要求的场景。为了及时处理流数据，流计算框架必须是低延迟、可扩展、高可靠的。

流计算的应用场景有很多，比如它是网络监控、传感监测、AR/VR、音视频流等实时应用的发展的基础。所以，目前流计算相关的框架和平台也有很多了，主流的划分方式是将其分为如下 3 类：

商业级的流计算平台，比如 IBM 的 InfoSphere Streams 和 TIBCO 的 StreamBase。InfoSphere Streams 支持同时分析多种数据类型并实时执行复杂计算。StreamBase 是一个用于实时分析的软件，可以快速构建分析系统，即时做出决策。StreamBase 可以为投资银行、对冲基金、政府机构等提供实时数据分析服务。

开源流计算框架，典型代表是 Apache Storm（由 Twitter 开源）和 S4（由 Yahoo 开源）。Storm 是一个分布式的、容错的实时计算系统，可以持续进行实时数据流处理，也可以用于分布式 RPC。S4 是一个通用的、分区容错的、可扩展的、可插拔的分布式流式系统。这些开源的分布式流计算系统由于具备开源代码，因此比较适合开发人员将其搭建在自身业务系统中。

各大公司根据自身业务特点而开发的流计算框架，比如 Facebook 的 Puma、百度的 Dstream（旨在处理有向无环的数据流）、淘宝的银河流数据处理平台（一个通用的、低延迟、高吞吐、可复用的流数据实时计算系统）。

除了这些框架外，我们还会经常听到 Spark、Flink 等。Spark 和 Flink 与 Storm 框架的不同之处在于，Spark 和 Flink 除了支持流计算，还支持批量计算，因此我没有直接将它们列入上述的流计算框架中。如果你的业务中需要用到或者需要参考某种计算框架或者平台的话，可以再参考其官方文档或者相关的技术文章。

接下来，我就以 Storm 这个开源的流计算框架为例，通过介绍 Storm 的工作原理，以加深你对流计算模式的进一步理解，进而帮助你将其运用到实际业务中。

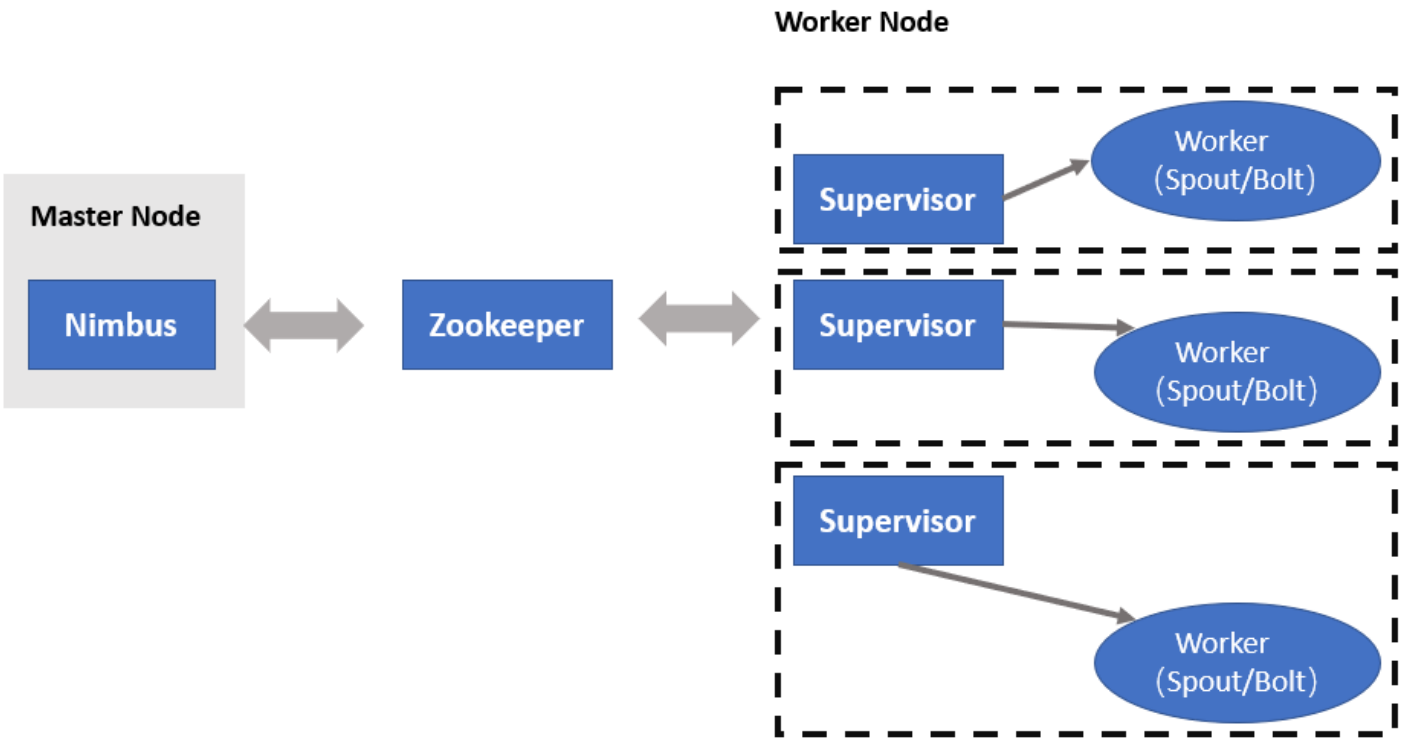
Storm 的工作原理

说到 Storm 的工作原理，我们先来对比下 Storm 与 MapReduce 的区别吧。Hadoop 上运行的是“MapReduce 作业”，而 Storm 上运行的是“计算拓扑（Topologies）”。“作业”和“拓扑”的一个关键区别是：MapReduce 的一个作业在得到结果之后总会结束；而拓扑描述的是计算逻辑，该计算逻辑会永远在集群中运行（除非你杀死该进程）。

如下图所示，Storm 集群上有两种节点，即主节点（Master Node）和工作节点（Worker Nodes）。

Nimbus 是整个 Storm 集群的主守护进程，以唯一实例的方式运行在主节点上。它负责把任务分配和分发给集群的工作节点，并监控这些任务的执行情况。当某个节点故障时，它会重新分配该故障节点上的任务到其它节点。

Supervisor 是 Storm 集群里工作守护进程，每个工作节点都存在一个这样的实例。它通过 Zookeeper 与 Nimbus 守护进程进行通信。在接受到 Nimbus 分配的任务后，它会为每个任务启动单独的工作进程。



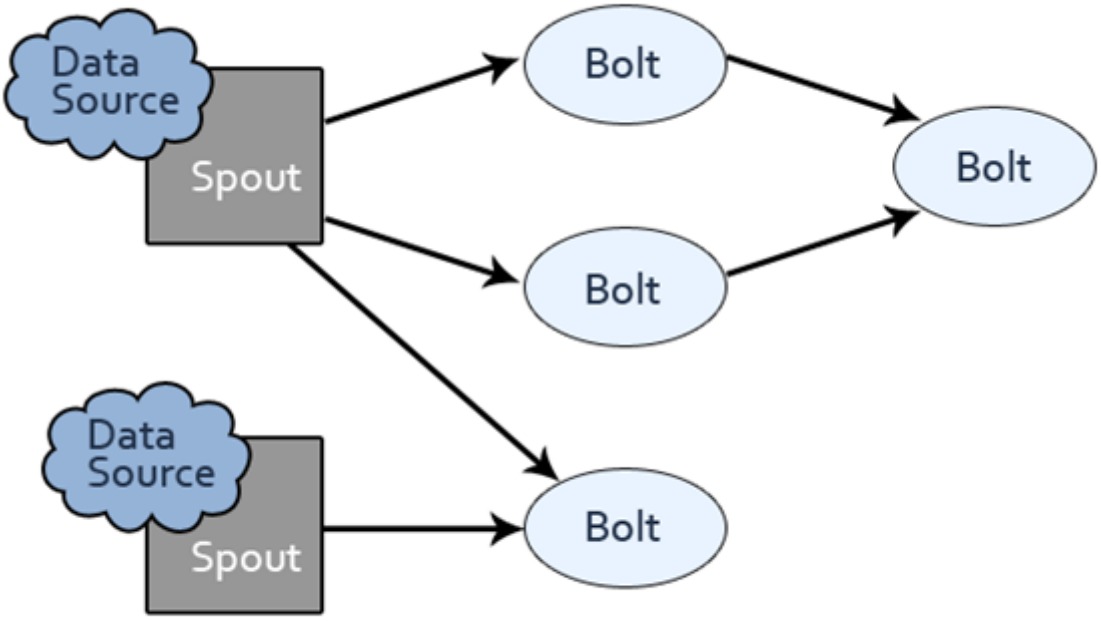
前面我介绍了 Nimbus 是负责分发任务或代码的，Supervisor 是负责接收任务，并启动和停止工作进程以执行任务的。那么 Nimbus 和 Supervisors 之间，具体是怎么协同的呢？下面我们一起看一下。

如果所有数据和信息均存储在 Master Node 上，Master Node 故障后，会导致整个集群信息丢失，因此引入了 ZooKeeper 集群来加强可靠性。为此 Master Node 与 Worker Node 之间的交互通过 ZooKeeper 完成，由于 Nimbus 和 Supervisors 是 Master Node 和 Worker Node 之间负责交互的进程，因此 Nimbus 和 Supervisors 之间的所有协调都是通过 ZooKeeper 集群完成的，比如 Nimbus 会将任务的分配情况或信息发送给 ZooKeeper 集群，然后 Supervisors 向 ZooKeeper 集群获取任务，并启动工作进程以执行任务。

当 Supervisor 接收到分配的任务后，会启动工作节点的工作进程 (Worker) 去执行任务。我们知道，一个计算任务可以分成任务数据的读取以及任务执行两部分。Worker 提供了两个组件 Spout 和 Bolt，分别进行数据读取和任务执行。

在详细介绍 Worker 组件之前，我首先介绍一下 Storm 的核心抽象：数据流。数据流是一个无界序列，是在分布式环境中并行创建、处理的一组元组（tuple）。数据流可以由一种能够表述数据流中元组的域（fields）的模式来定义。

Storm 为进行数据流转换提供了基本组件 Spout 和 Bolt。 Spout 和 Bolt 有用户自定义的接口，用于运行特定应用程序的逻辑。如下图所示，Storm 上运行的计算拓扑其实是由一系列 Spout 和 Bolt 组成的有向无环图，这个有向无环图代表了计算逻辑。



A Storm Topology

备注：

1. 图中箭头，表示数据元组的传递方向。
2. 此图引自“Twitter Analysis with Apache Storm”。

接下来，我们看看 **Spout 和 Bolt 的含义**吧。

> Spout 用于接收源数据。通常情况下，Spout 会从一个外部的数据源读取数据元组，然后将它们发送到拓扑中。例如，Spout 从 Twitter API 读取推文并将其发布到拓扑中。
>
> Bolt 负责处理输入的数据流，比如数据过滤（filtering）、函数处理（functions）、聚合（aggregations）、联结（joins）、数据库交互等。数据处理后可能输出新的流作为下一个 Bolt 的输入。每个 Bolt 往往只具备单一的计算逻辑。当我们执行简单的数据流转换时，比如仅进行数据过滤，则通常一个 Bolt 可以实现；而复杂的数据流转换通常需要使用多个 Bolt 并通过多个步骤完成，比如在神经网络中，对原始数据进行特征转换，需要经过数据过滤、清洗、聚类、正则化等操作。

知识扩展：流计算和批量计算的区别是什么？

MapReduce 可以说是一种批量计算，与我们今天介绍的用于实时数据处理的流计算，是什么关系呢？

虽然流计算和批量计算属于两种不同的计算模式，但并不是非此即彼的关系，只是适用于不同的计算场景。

在流计算中，数据具有时效性，因此在 5G 以及人工智能应用的驱动下，专注于实时处理的流计算越来越得到广泛的关注。流计算的低延时、易扩展等性能非常适用于对时延要求高的终端应用（比如直播中音视频的处理等），从而极大提高用户的服务体验。而批量计算适用于对时延要求低的任务。

在实际运用中，可以根据计算要求，选择不同的计算模式。我将这两种计算模式的特点，总结为了一张表格，以帮助你理解、记忆，以及选择适合自己业务场景的计算模式。

对比指标	批量计算	流式计算
数据特点	大规模非实时数据	持续产生的、具有易逝性的实时数据
数据集成方式	预先加载并存储批量的数据	实时加载数据，不存储数据
数据处理方式	对集中的所有数据或大批量数据一起进行处理	对最近输入的数据进行处理
计算规则	1. 任务处理过程中计算逻辑可以修改 2. 计算逻辑修改后，数据可重新计算	1. 任务处理过程中计算逻辑不可以修改 2. 计算逻辑一旦修改，之前的数据不可重新计算
实时性分析	处理需要几分钟甚至是几小时	处理时间仅为几秒或者几毫秒
适用场景	用于对时延不敏感的批处理任务，比如大规模信息统计等	用于对时延敏感的数据密集型任务，该任务可拆分为小批量数据的任务，如实时天气预报、直播中音视频流处理等
典型计算框架	MapReduce等	Storm、InfoSphere Streams、Dstream等

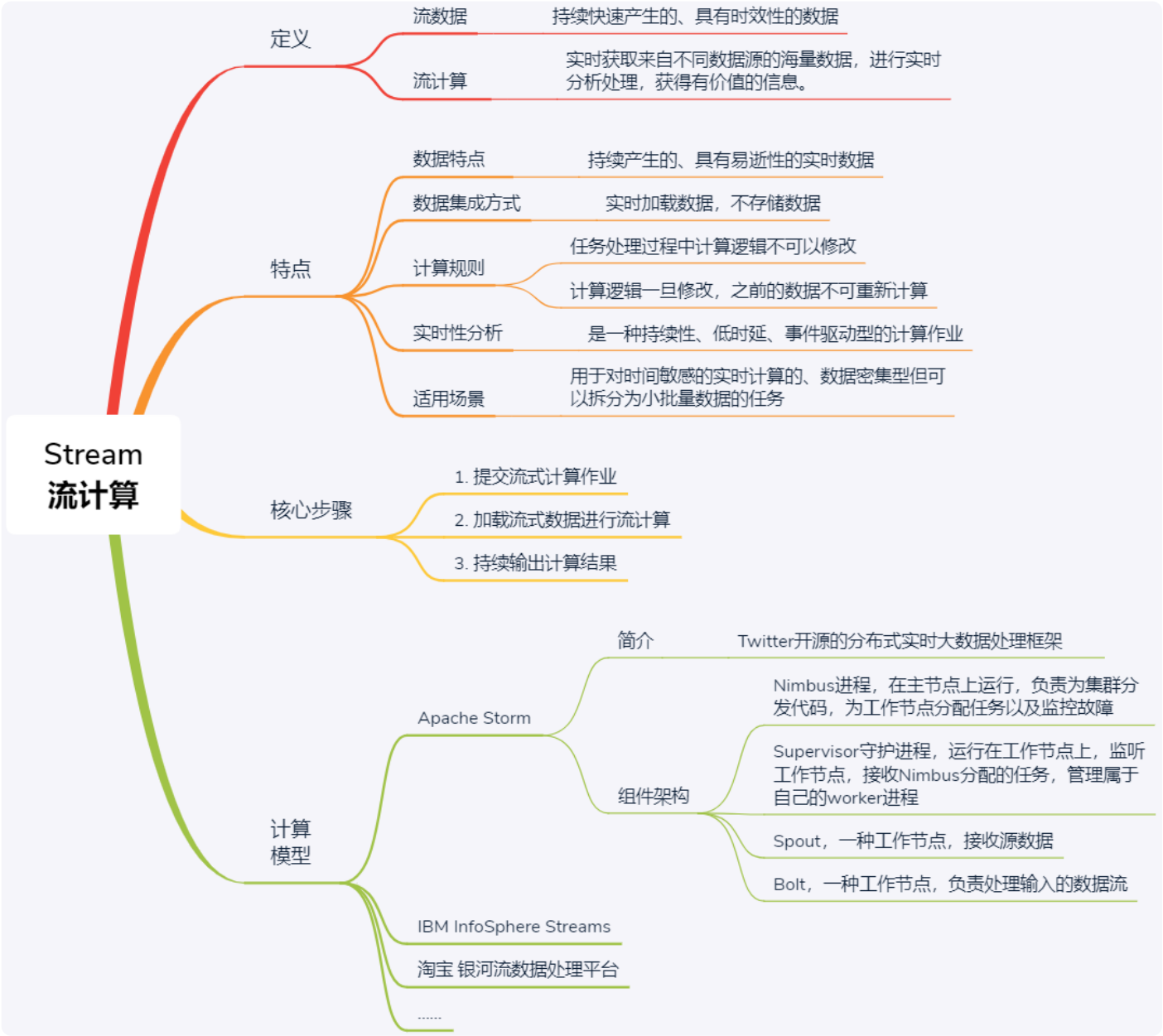
总结

今天，我与你介绍了分布式计算模式中的流计算。流数据的价值会随时间的流逝而降低，“时间就是金钱”在流计算中体现得淋漓尽致。这就要求流计算框架必须是低延迟、可扩展、高可靠的。

在介绍流计算的工作原理时，我首先通过一个流程图，与你介绍了它的 3 个步骤，即提交流式计算作业、加载流式数据进行流计算和持续输出计算结果。

然后，我以流计算开源框架中的 Storm 为例，与你讲述了 Storm 的核心组件以及通过 Spout 和 Bolt 构建有向无环图代表流计算逻辑，以实现流计算，以加深你对流计算原理的理解。

最后，我再通过一张思维导图来归纳一下今天的核心知识点吧。



思考题

离线计算和批量计算，实时计算和流式计算是等价的吗？你能和我说说你做出判断的原因吗？

我是聂鹏程，感谢你的收听，欢迎你在评论区给我留言分享你的观点，也欢迎你把这篇文章分享给更多的朋友一起阅读。我们下期再会！

精选留言 (22)

Eternal
2019-11-02

老师课后思考题：“离线计算和批量计算，实时计算和流式计算是等价的吗？”

其实留言去已经有小伙伴总结得很好（自认为，如果有错望指正）：
离线计算、批量计算、实时计算、流式计算都是要计算的，这是它们的共同特点；

批量和流式是描述计算的时候计算资源的特点的，是描述计算数据的维度。
批量计算
是等到计算数据积累到一定数量才开始计算
流式计算
是数据量很小时候就可以开始计算，有了资源就可以开始，源源不断

离线和实时是描述计算的时效性，是描述计算的维度。
离线计算
因为计算数据不能来了就计算，需要累计到一定阈值，所以按照时效性来说，计算是离线的
实时计算
因为计算数据来了就开始计算，计算没得延迟，，所以按照时效性来说，计算是实时的

批量和流式是一个维度，离线和实时是另外一个维度，但是两个维度之间又有联系：
因为目前现有的技术发展，不能大批量计算做到实时的效果，所以只能少量资源做到实时计算，且通过流式计算来达到实时的效果；
但是如果一旦当前的硬件指标和技术能力突破后，能大批量计算做到实时的效果，这也是可能的。因此我认为，我们的演进目标是想能
做到批量实时的。两个维度之间由相关联的因素来驱动它们相互变化。

共 1 条评论 >　　13

Jackey
2019-10-28

不太好说是否等价，因为离线和实时是针对时延判断的，批量和流式是针对数据处理方式判断的。只能目前说离线和批量使用的框架、处理方法相同（实时和流式相同）。但如果以后发展出能批量进行实时数据计算的计算机就不能说批量=离线了吧

4

随心而至
2019-10-28

In computer science, an online algorithm[1] is one that can process its input piece-by-piece in a serial fashion, i.e., in the order that the input is fed to the algorithm, without having the entire input available from the start.

In contrast, an offline algorithm is given the whole problem data from the beginning and is required to output an answer which solves the problem at hand.

In computer science, streaming algorithms are algorithms for processing data streams in which the input is presented as a sequence of items and can be examined in only a few passes (typically just one). In most models, these algorithms have access to limited memory (generally logarithmic in the size of and/or the maximum value in the stream). They may also have limited processing time per item.

In computer science, real-time computing (RTC), or reactive computing describes hardware and software systems subject to a "real-time constraint", for example from event to system response.[1] Real-time programs must guarantee response within specified time constraints, often referred to as "deadlines".

从维基百科的定义可以，他们侧重的点不同，个人总结如下：
1.online algorithm 只能看到数据的一部分，必须基于此做决定（可能不是最优的）
2.offline algorithm 可以看到解决问题的所有数据
3.streaming algorithms 处理序列流（内存有限，流大小有限，单个序列项处理时间有限）
4.real-time computing 强调响应时间,有时间限制。

参考自https://en.wikipedia.org/wiki/Online_algorithm 及其See also:

3

随心而至
2019-10-28

聂老师从一般到具体，讲的真好。
我认为批量计算是离线计算的一种实现方式；流式计算是实习计算的一种实现方式。待会我查

下他们的定义看看自己理解的对不对

2

旭东(Frank)
2019-10-28

如果历史数据回放进行就处理也算流量处理吧；个人认为注意区别点还是处理数据的时效性，如果对处理结果不要求那么及时，新鲜。

流处理就像喝豆浆，批量处理像吃豆腐。

共 2 条评论 >

2

林通
2020-11-12

storm都快被淘汰了吧，为啥不讲讲最流行的框架呢

2

信xin_n
2019-11-16

终于明白了 Nimbus 和 Supervisor 的关系了

1

lcken
2020-04-03

flink也应该纳入流式计算框架。它是以流计算批数据的

王涛
2020-03-29

MR 针对静态数据，Stream 针对动态数据。并实时性较高。

王涛
2020-03-29

继续打卡

钱

2020-02-17

阅过留痕
知识盲区地带，流式计算没接触过，概念大概能明白。
流数据的价值会随时间的流逝而降低，“时间就是金钱”在流计算中体现得淋漓尽致。这就要求流计算框架必须是低延迟、可扩展、高可靠的。
不过流计算框架具体怎么实现低延迟、可扩展、高可靠的没领会到。


陈
2019-11-21

老师请问流计算怎么做增量计算?


simon
2019-11-08

流计算看上去，这些框架跟普通的分布式微服务框架是不是一样，都是可以并发处理实时数据，并且可以横向扩展?


张先生
2019-11-04

实习的时候做过写过mapreduce和storm的代码，一直不是特别清楚两者的区别，看了这篇文章豁然开朗


Eternal
2019-11-02

继续打卡

分布式计算了解的很少，这几节引起我的兴趣和思考，很受益。
老师将流式计算的一个节点定义成一个拓扑节点，这个让我想到了一个点:
计算机CPU由计算单元，控制单元，内存组成。CPU的计算由很多与门、非门等控制电路组成。
计算机的发展从单机单核，到单机多核，到分布式多核，这是不断的最求性能提升的发展趋势。
如果我们想要用分布式集群来模拟单机单核的计算怎么做呢？也就是通过一个进程来模拟一个CPU最底层的门电路计算。

由上面的推演让我想到了，流式计算将一个CPU的计算能力抽象成了一个进程，也就是老师讲的一个计算拓扑节点，这样我们
就可以获取更加超大规模的分布式计算能力了。

老师总结的流式计算特点：持续产生，易逝，实时。CUP的计算也有类似的特点。
流式计算的节点计算规则可以自定义，CPU的计算逻辑不能很轻易的更新。

再想到一个点：华为将原来通过软件处理的很多能力通过Soc集成到芯片中，这样来提高芯片的处理能力，但是Soc制作成本（俗称流片）很昂贵。
而分布式计算将本来硬件处理的能力抽象到软件层面来实现，这样可以更加灵活，也突破了传统计算机计算的边界。
两者走的是不同的方向，虽然它们的场景可能不经相同，但是都是为了提升计算性能而做的努力。

以上纯属自己的相互思考和联系，望大牛们看到到错误后能给与指正，谢谢！

波波安
2019-11-01

spark streaming就是使用的批处理方式实现的实时计算

波波安
2019-11-01

离线和实时是指数据处理的时延。批量和流式是指数据处理的方式。

天天向善
2019-10-31

实时天气预报，直播音视频处理，在流式计算中是计算什么，没有概念，能介绍下吗

leslie
2019-10-28

离线计算和批量计算不是等价的：批量计算不一定离线，需要使用在线资源；离线计算则是使用闲置资源，时间和线上压力还是不同的。离线计算是批量计算的一种形式，但是离线计

算不一定是批量计算。

实时计算和流式计算不是等价的：实时计算是当下-个体，流式计算可能是处理一组实时计算。流式计算是有1个或者多个实时计算通过分布式处理之后处理完再发送出去的结果；实时计算则是拿到就处理，处理完了就把结果发送出去。

这是我个人对于老师课程的理解：期待老师答案的公布以及后面的继续分享。通过老师对分布式计算的讲解，解释明白了分布式技术的核心。


有人@我

2019-10-28

分布式跟流水计算好像没多大的关系