

## 15 | 流式计算的代表：Storm、Flink、Spark Streaming

2018-12-01 李智慧

从0开始学大数据

### Spark [spark]

- n. 火花；电火花；闪现；一点儿
- vi. 飞火星；冒火花
- vt. 引发

[查看详细释义 >](#)

[进入课程 >](#)



讲述：李智慧

时长 12:38 大小 11.58M



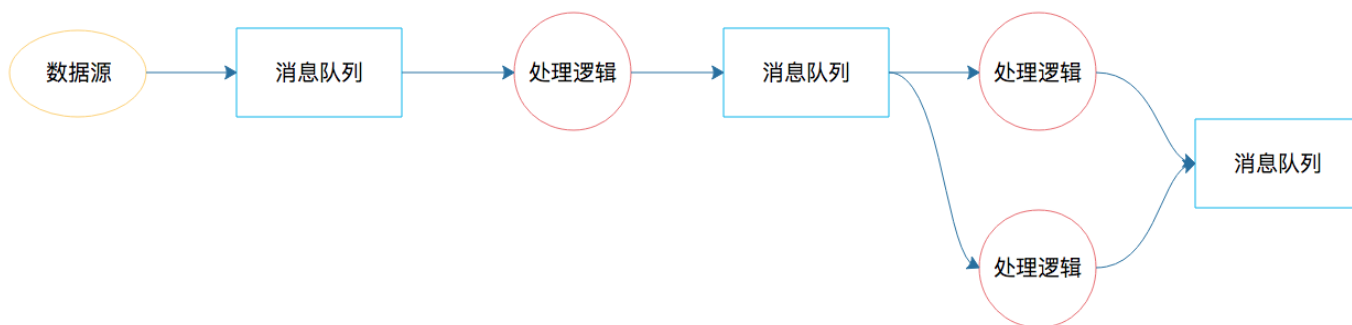
我前面介绍的大数据技术主要是处理、计算存储介质上的大规模数据，这类计算也叫大数据批处理计算。顾名思义，数据是以批为单位进行计算，比如一天的访问日志、历史上所有的订单数据等。这些数据通常通过 HDFS 存储在磁盘上，使用 MapReduce 或者 Spark 这样的批处理大数据计算框架进行计算，一般完成一次计算需要花费几分钟到几小时的时间。

此外，还有一种大数据技术，针对实时产生的大规模数据进行即时计算处理，我们比较熟悉的有摄像头采集的实时视频数据、淘宝实时产生的订单数据等。像上海这样的一线城市，公共场所的摄像头规模在数百万级，即使只有重要场所的视频数据需要即时处理，可能也会涉及几十万个摄像头，如果想实时发现视频中出现的通缉犯或者违章车辆，就需要对这些摄像头产生的数据进行实时处理。实时处理最大的不同就是这类数据跟存储在 HDFS 上的数据不同，是实时传输过来的，或者形象地说是流过来的，所以针对这类大数据的实时处理系统也叫大数据流计算系统。

目前业内比较知名的大数据流计算框架有 Storm、Spark Streaming、Flink，接下来，我们逐一看看它们的架构原理与使用方法。

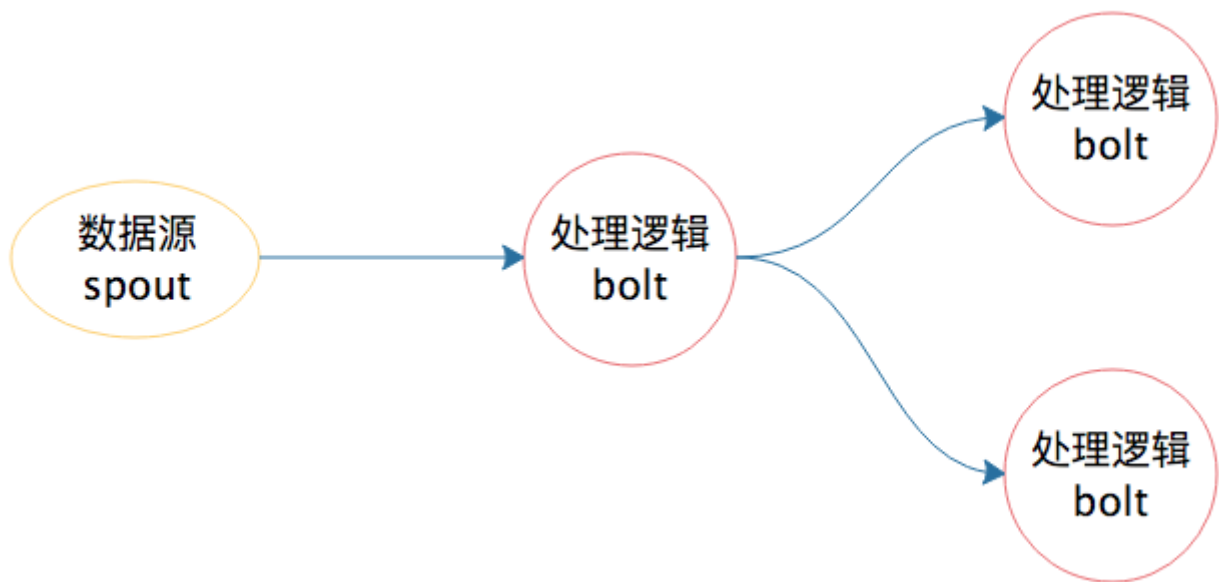
## Storm

其实大数据实时处理的需求早已有之，最早的时候，我们用消息队列实现大数据实时处理，如果处理起来比较复杂，那么就需要很多个消息队列，将实现不同业务逻辑的生产者和消费者串起来。这个处理过程类似下面图里的样子。



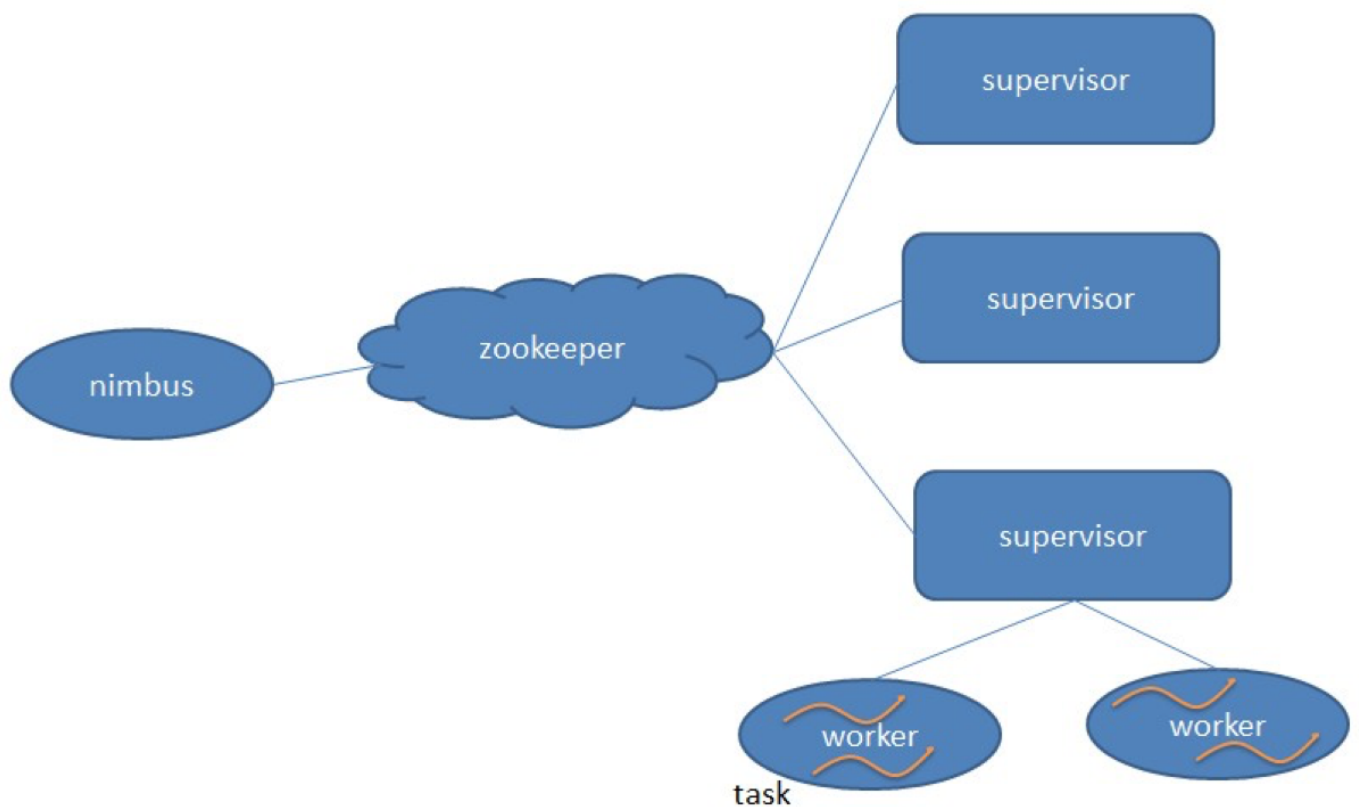
图中的消息队列负责完成数据的流转；处理逻辑既是消费者也是生产者，也就是既消费前面消息队列的数据，也为下个消息队列产生数据。这样的系统只能是根据不同需求开发出来，并且每次新的需求都需要重新开发类似的系统。因为不同应用的生产者、消费者的处理逻辑不同，所以处理流程也不同，因此这个系统也就无法复用。

之后我们很自然地就会想到，能不能开发一个流处理计算系统，我们只要定义好处理流程和每一个节点的处理逻辑，代码部署到流处理系统后，就能按照预定义的处理流程和处理逻辑执行呢？Storm 就是在这种背景下产生的，它也算是一个比较早期的大数据流计算框架。上面的例子如果用 Storm 来实现，过程就变得简单一些了。



有了 Storm 后，开发者无需再关注数据的流转、消息的处理和消费，只要编程开发好数据处理的逻辑 bolt 和数据源的逻辑 spout，以及它们之间的拓扑逻辑关系 topology，提交到 Storm 上运行就可以了。

在了解了 Storm 的运行机制后，我们来看一下它的架构。Storm 跟 Hadoop 一样，也是主从架构。



nimbus 是集群的 Master，负责集群管理、任务分配等。supervisor 是 Slave，是真正完成计算的地方，每个 supervisor 启动多个 worker 进程，每个 worker 上运行多个 task，而 task 就是 spout 或者 bolt。supervisor 和 nimbus 通过 ZooKeeper 完成任务分配、心跳检测等操作。

Hadoop、Storm 的设计理念，其实是一样的，就是把和具体业务逻辑无关的东西抽离出来，形成一个框架，比如大数据的分片处理、数据的流转、任务的部署与执行等，开发者只需要按照框架的约束，开发业务逻辑代码，提交给框架执行就可以了。

而这也正是所有框架的开发理念，就是将业务逻辑和处理过程分离开来，使开发者只需关注业务开发即可，比如 Java 开发者都很熟悉的 Tomcat、Spring 等框架，全部都是基于这种理念开发出来的。

## Spark Streaming

我们知道 Spark 是一个批处理大数据计算引擎，主要针对大批量历史数据进行计算。前面我在讲 Spark 架构原理时介绍过，Spark 是一个快速计算的大数据引擎，它将原始数据分片后装载到集群中计算，对于数据量不是很大、过程不是很复杂的计算，可以在秒级甚至毫秒级完成处理。

Spark Streaming 巧妙地利用了 Spark 的**分片**和**快速计算**的特性，将实时传输进来的数据按照时间进行分段，把一段时间传输进来的数据合并在一起，当作一批数据，再去交给 Spark 去处理。下图这张图描述了 Spark Streaming 将数据分段、分批的过程。



如果时间段分得足够小，每一段的数据量就会比较小，再加上 Spark 引擎的处理速度又足够快，这样看起来好像数据是被实时处理的一样，这就是 Spark Streaming 实时流计算的奥妙。

这里要注意的是，在初始化 Spark Streaming 实例的时候，需要指定分段的时间间隔。下面代码示例中间隔是 1 秒。

```
1 val ssc = new StreamingContext(conf, Seconds(1))
```

当然你也可以指定更小的时间间隔，比如 500ms，这样处理的速度就会更快。时间间隔的设定通常要考虑业务场景，比如你希望统计每分钟高速公路的车流量，那么时间间隔可以设为 1 分钟。

Spark Streaming 主要负责将流数据转换成小的批数据，剩下的就可以交给 Spark 去做了。

## Flink

前面说 Spark Streaming 是将实时数据流按时间分段后，当作小的批处理数据去计算。那么 Flink 则相反，一开始就是按照流处理计算去设计的。当把从文件系统（HDFS）中读入的数据也当做数据流看待，他就变成批处理系统了。

为什么 Flink 既可以流处理又可以批处理呢？

如果要进行流计算，Flink 会初始化一个流执行环境 `StreamExecutionEnvironment`，然后利用这个执行环境构建数据流 `DataStream`。

```
1 StreamExecutionEnvironment see = StreamExecutionEnvironment.getExecutionEnvironment();  
2  
3 DataStream<WikipediaEditEvent> edits = see.addSource(new WikipediaEditsSource());
```

如果要进行批处理计算，Flink 会初始化一个批处理执行环境 `ExecutionEnvironment`，然后利用这个环境构建数据集 `DataSet`。


```
1 ExecutionEnvironment env = ExecutionEnvironment.getExecutionEnvironment();  
2  
3 DataSet<String> text = env.readTextFile("/path/to/file");
```



然后在 `DataStream` 或者 `DataSet` 上执行各种数据转换操作 (transformation)，这点很像 Spark。不管是流处理还是批处理，Flink 运行时的执行引擎是相同的，只是数据源不同而已。

Flink 处理实时数据流的方式跟 Spark Streaming 也很相似，也是将流数据分段后，一小批一小批地处理。流处理算是 Flink 里的“一等公民”，Flink 对流处理的支持也更加完善，它可以对数据流执行 window 操作，将数据流切分到一个一个的 window 里，进而进行计算。

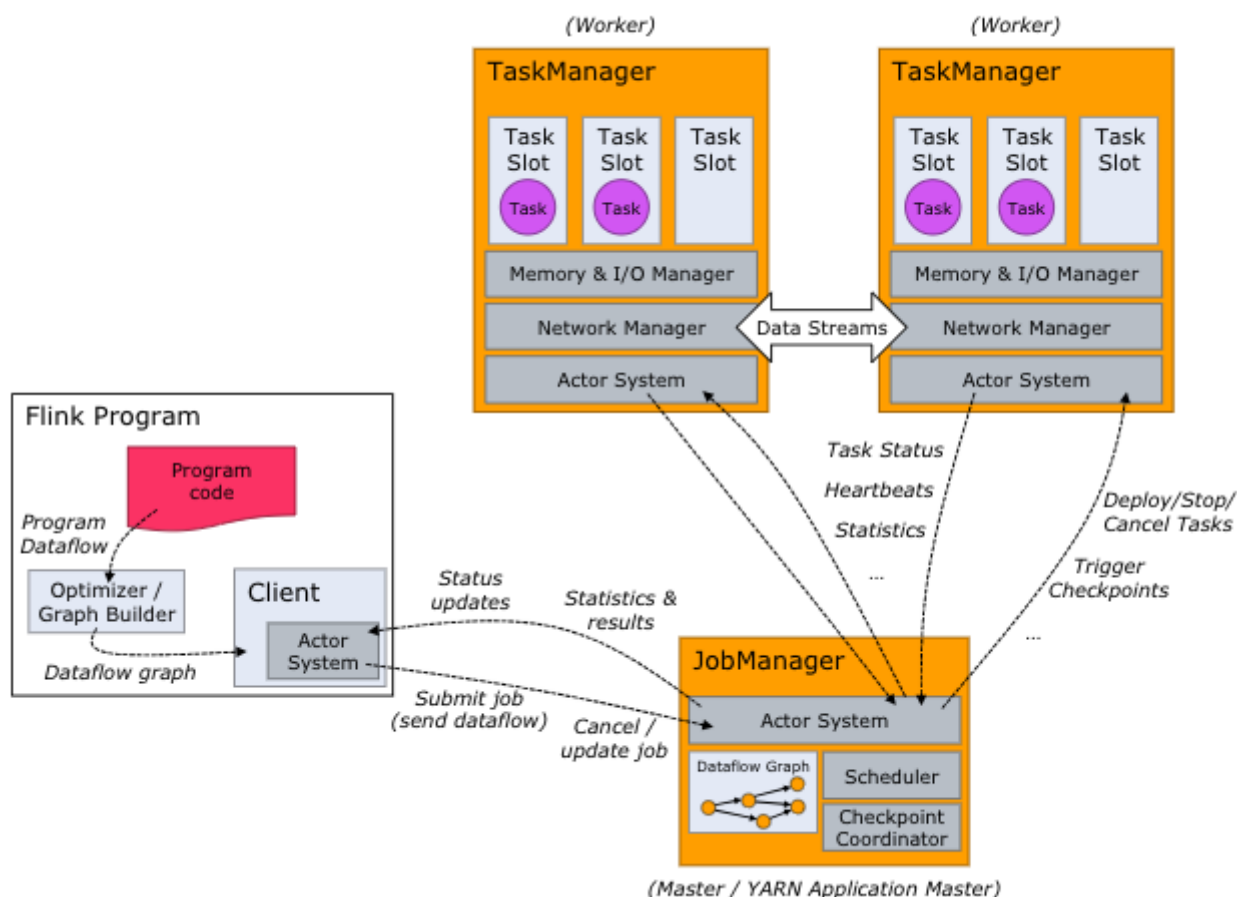
## 在数据流上执行

 复制代码

```
1 .timeWindow(Time.seconds(10))
```

可以将数据切分到一个 10 秒的时间窗口，进一步对这个窗口里的一批数据进行统计汇总。

Flink 的架构和 Hadoop 1 或者 Yarn 看起来也很像，JobManager 是 Flink 集群的管理者，Flink 程序提交给 JobManager 后，JobManager 检查集群中所有 TaskManager 的资源利用状况，如果有空闲 TaskSlot（任务槽），就将计算任务分配给它执行。



## 小结

大数据技术最开始出现的时候，仅仅针对批处理计算，也就是离线计算。相对说来，大数据实时计算可以复用互联网实时在线业务的处理技术方案，毕竟对于 Google 而言，每天几十亿的用户搜索访问请求也是大数据，而互联网应用处理实时高并发请求已经有一套完整的解决方案了（详见我写的《大型网站技术架构：核心原理与案例分析》一书），大数据流计算的需求当时并不强烈。

但是我们纵观计算机软件发展史，发现这部历史堪称一部**技术和业务不断分离**的历史。人们不断将业务逻辑从技术实现上分离出来，各种技术和架构方案的出现，也基本都是为这一目标服务。

最早的时候我们用机器语言和汇编语言编程，直接将业务逻辑用 CPU 指令实现，计算机软件就是 CPU 指令的集合，此时技术和业务完全耦合，软件编程就是面向机器编程，用机器指令完成业务逻辑，当时我们在编程的时候思维方式是面向机器的，需要熟记机器指令。

后来我们有了操作系统和高级编程语言，将软件和 CPU 指令分离开来，我们使用 Pascal、Cobal 这样的高级编程语言进行编程，并将程序运行在操作系统上。这时我们不再面向机

器编程，而是面向业务逻辑和过程编程，这是业务逻辑与计算机技术的一次重要分离。

再后来出现了面向对象的编程语言，这是人类编程史上的里程碑。我们编程的时候关注的重心，从机器、业务过程转移到业务对象本身，分析客观世界业务对象的关系和协作是怎样的，如何通过编程映射到软件上，这是编程思维的一次革命，业务和技术实现从思想上分离了。

再后来出现各种编程框架，一方面使业务和技术分离的更加彻底，想象一下，如果不用这些框架，你自己编程监听 80 通信端口，从获取 HTTP 二进制流开始，到开发一个 Web 应用会是什么感觉。另一方面，这些框架也把复杂的业务流程本身解耦合，视图、业务、服务、存储各个层次模块独立开发、部署，通过框架整合成一个系统。

回到流计算，固然我们可以用各种分布式技术实现大规模数据的实时流处理，但是我们更希望只要针对小数据量进行业务开发，然后丢到一个大规模服务器集群上，就可以对大规模实时数据进行流计算处理。也就是业务实现和大数据流处理技术分离，业务不需要关注技术，于是各种大数据流计算技术应运而生。

其实，我们再看看互联网应用开发，也是逐渐向业务和技术分离的方向发展。比如，云计算以云服务的方式将各种分布式解决方案提供给开发者，使开发者无需关注分布式基础设施的部署和维护。目前比较热门的微服务、容器、服务编排、Serverless 等技术方案，它们则更进一步，使开发者只关注业务开发，将业务流程、资源调度和服务管理等技术方案分离开来。而物联网领域时髦的 FaaS，意思是函数即服务，就是开发者只要开发好函数，提交后就可以自动部署到整个物联网集群运行起来。

总之，流计算就是将大规模实时计算的资源管理和数据流转都统一管理起来，开发者只要开发针对小数据量的数据处理逻辑，然后部署到流计算平台上，就可以对大规模数据进行流式计算了。

## 思考题

流计算架构方案也逐渐对互联网在线业务开发产生影响，目前流行的微服务架构虽然将业务逻辑拆分得很细，但是服务之间的调用还是依赖接口，这依然是一种比较强的耦合关系。淘宝等互联网企业已经在尝试一种类似流计算的、异步的、基于消息的服务调用与依赖架构，据说淘宝的部分核心业务功能已经使用这种架构方案进行系统重构，并应用到今年的“双十一”大促，利用这种架构特有的回压设计对高并发系统进行自动限流与降级，取得很好的效果。



我也邀请了淘宝负责这次架构重构的高级技术专家李鼎，请他在留言区分享一下这次架构重构的心得体会。

你对这种架构方案有什么想法，是否认为这样的架构方案代表了未来的互联网应用开发的方向？

欢迎你写下自己的思考或疑问，与我和其他同学一起讨论。



# 从 0 开始学大数据

## 智能时代你的大数据第一课

**李智慧**  
同程艺龙交通首席架构师  
前 Intel 大数据架构师



新版升级：点击「 请朋友读」，10位好友免费读，邀请订阅更有**现金**奖励。

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 14 | BigTable的开源实现：HBase

下一篇 16 | ZooKeeper是如何保证数据一致性的？

## 精选留言 (18)

 写留言



李鼎(哲良...) 置顶

2018-12-06

 73

数据流是久经考验的典型思路，在网络协议（如TCP）、数据平台这样场景，早就应用多年习以为常了。淘宝业务的应用架构升级可以认为是把这样思路应用到了业务系统开发

中，把『流』作为业务表达上的一等概念和手段，并在业务架构/系统能力优化提升。

简单地说，因为业务面向数据流来编写，一方面业务逻辑表达可以自然接近业务流程； ...  
展开 ▾

作者回复: 📬 欢迎有兴趣的同学进一步探讨。

李鼎的流式架构文档地址: <https://github.com/oldratlee/rp-practice>



万~~

2018-12-01

👍 11

你好 storm spark flink 都是优秀的框架 那我们应该学习哪个呢？ 都学肯定精力不够 而且难以精通



纯洁的憎恶

2018-12-01

👍 5

批计算是对历史数据的一次性处理，流计算是对实时流入的数据实时响应。

storm模仿消息队列（什么是消息队列？卡夫卡？），把消息队列中与业务逻辑无关的过程部分抽象出来形成标准框架，实现复用。开发者不用纠结四面八方涌入的实时数据如何流转，消息如何处理和消费，只用考虑业务流程、数据源、处理逻辑。 ...

展开 ▾



laurencez...

2018-12-04

👍 4

智慧老师你好，这节对storm.spark.flink的介绍感觉过于概述了！后面是否会有详细的文章介绍，比如分析对比一下他们三者各自优缺点在哪里？各自试用与不适用的业务场景有哪些之类的呢？

展开 ▾



Jowin

2018-12-01

👍 3

智慧老师，我是从事金融实时数据处理的，有一类典型需求是从原始实时数据计算出各种衍生数据，但是有状态积累。比如，当前状态是S0，收到数据A0，此时要根据(S0,A0)生

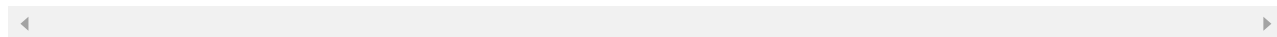
成数据A1，同时要更新当前状态S1，后续的新数据再基于S1处理。团队考虑过使用Stream作为计算平台，有两个问题没想清楚怎么处理：

1) 如果计算任务故障挂掉，会不会导致这期间的数据丢失？ ...

展开 ▾

作者回复: 1 spark streaming有容错机制，不会丢失。当然，服务器数量要充足。

2 试下Redis存储状态。



杰之7

2018-12-01

👍 3

通过本小节的学习，了解了常用的流计算及它们的计算框架，其中Spark Streaming巧妙了运用Spark计算速度的优势，将Spark批计算通过时间间隔装置成流计算。在我们的生活中，股票交易的价格传输应该就是运用了流计算，要求在极短的时间内完成对价格的改变。当然，淘宝，一线城市的摄像头在后台处理上也应用了流计算技术。相比批计算技术，流计算在重要的数据上会用的越来越广。

展开 ▾



常平

2018-12-11

👍 2

流式架构本质上是事件驱动（event-Driven）架构，流由段（segment）组成，段由事件（event）组成，事件由字节（bytes）组成，事件大小有限，而字节流大小无限

展开 ▾



尼糯米

2019-01-10

👍 1

问题一

Strom算是比较早期的大数据流计算框架

》》定义处理流程

》》流程的每个环节上的处理逻辑

数据流转是计算框架按处理流程进行流转吗？ ...

展开 ▾



zhj

2019-01-08

👍 1

1面向数据流的编程在java里逐渐展露头角，之前rxjava更多的是用于android，直到hystrix才算是后端一个大规模应用的案例(也和场景有关吧，后端的大多业务都是短事物处理去构建一条数据流水线反倒显得累赘)，reactor是响应式编程的另一个实现，直到spring5全面拥抱以后，才完全进入人们视野(所以技术落地离不开大厂的支持)，单纯的业务层处理构造一条很短的数据流意义不大(因为数据源可能还是需要返回所有数据),spring...  
展开 ▾

作者回复: ㊦



您的好...

2018-12-04

1

主攻人工智能，机器学习和算法实现的应该学习哪种呢？Storm，Spark还是Flink呢？

作者回复: spark



maomaosty...

2019-02-20

1

想了解storm与flink以及spark streaming 相比有什么区别？感觉后两者都是将数据源做很细粒度的切分，最终看上去“像”是连续的流，那么storm的处理方式呢？

展开 ▾



香港记者

2018-12-25

1

好

展开 ▾



洪喜

2018-12-17

1

感谢老师的讲解，不过也印证了师傅领进门修行在个人这句话，个人需要针对没深入讲解的知识点再次深入自学了。



修行者

2018-12-07



想要了解这三种流式计算框架 Strom、Flink、Spark Streaming，各种的优缺点及适用的业务场景，智慧哥在下面的专栏会详细介绍吗？



提米XXVII...

2018-12-07



老师好，我现在场景是针对一次请求单纯的并行计算后结果合并（单机内存限制和速度考虑），无存hdfs要求，spark合适吗？



WesleyWon...

2018-12-05



使用spark 进行计算，用Java 还是 scala 呢？现在项目组用的JAVA，后面想用scala. 如何取舍呢



Ittzzlll

2018-12-03



锤子和钉子

展开 ∨



三木子

2018-12-01



和老师探讨一个问题，对于架构设计，目前流行的是微服务，我个人觉得微服务还是有些缺点。架构从开始的合走到现在的分，未来会不会从分又到和呢？

展开 ∨

作者回复: 不会

