伴随着信息科技日新月异的发展，信息呈现出爆发式的膨胀，人们获取信息的途径也更加多样、更加便捷，同时对于信息的时效性要求也越来越高。举个搜索场景中的例子，当一个卖家发布了一条宝贝信息时，他希望的当然是这个宝贝马上就可以被卖家搜索出来、点击、购买啦，相反，如果这个宝贝要等到第二天或者更久才可以被搜出来，估计这个大哥就要骂娘了。再举一个推荐的例子，如果用户昨天在淘宝上买了一双袜子，今天想买一副泳镜去游泳，但是却发现系统在不遗余力地给他推荐袜子、鞋子，根本对他今天寻找泳镜的行为视而不见，估计这哥们心里就会想推荐你妹呀。其实稍微了解点背景知识的码农们都知道，这是因为后台系统做的是每天一次的全量处理，而且大多是在夜深人静之时做的，那么你今天白天做的事情当然要明天才能反映出来啦。

* **实现一个实时计算系统**

全量数据处理使用的大多是鼎鼎大名的hadoop或者hive，作为一个批处理系统，hadoop以其吞吐量大、自动容错等优点，在海量数据处理上得到了广泛的使用。但是，hadoop不擅长实时计算，因为它天然就是为批处理而生的，这也是业界一致的共识。否则最近这两年也不会有s4,storm,puma这些实时计算系统如雨后春笋般冒出来啦。先抛开s4,storm,puma这些系统不谈，我们首先来看一下，如果让我们自己设计一个实时计算系统，我们要解决哪些问题。

1. 低延迟。都说了是实时计算系统了，延迟是一定要低的。
2. 高性能。性能不高就是浪费机器，浪费机器是要受批评的哦。
3. 分布式。系统都是为应用场景而生的，如果你的应用场景、你的数据和计算单机就能搞定，那么不用考虑这些复杂的问题了。我们所说的是单机搞不定的情况。
4. 可扩展。伴随着业务的发展，我们的数据量、计算量可能会越来越大，所以希望这个系统是可扩展的。
5. 容错。这是分布式系统中通用问题。一个节点挂了不能影响我的应用。

好，如果仅仅需要解决这5个问题，可能会有无数种方案，而且各有千秋，随便举一种方案，使用消息队列+分布在各个机器上的工作进程就ok啦。我们再继续往下看。

1. 容易在上面开发应用程序。亲，你设计的系统需要应用程序开发人员考虑各个处理组件的分布、消息的传递吗？如果是，那有点麻烦啊，开发人员可能会用不好，也不会想去用。
2. 消息不丢失。用户发布的一个宝贝消息不能在实时处理的时候给丢了，对吧？更严格一点，如果是一个精确数据统计的应用，那么它处理的消息要不多不少才行。这个要求有点高哦。

**诞 生**

 在2011年Storm开源之前，由于Hadoop的火红，整个业界都在喋喋不休地谈论大数据。Hadoop的高吞吐，海量数据处理的能力使得人们可以方便地处理海量数据。但是，Hadoop的缺点也和它的优点同样鲜明——延迟大，响应缓慢，运维复杂。

有需求也就有创造，在Hadoop基本奠定了大数据霸主地位的时候，很多的开源项目都是以弥补Hadoop的实时性为目标而被创造出来。而在这个节骨眼上Storm横空出世了。

Storm带着流式计算的标签华丽丽的出场了，看看它的一些卖点：

* + **分布式系统**：可横向拓展,现在的项目不带个分布式特性都不好意思开源。
  + **运维简单**：Storm的部署的确简单。虽然没有Mongodb的解压即用那么简单，但是它也就是多安装两个依赖库而已。
  + **高度容错**：模块都是无状态的，随时宕机重启。
  + **无数据丢失**：Storm创新性提出的ack消息追踪框架和复杂的事务性处理,能够满足很多级别的数据处理需求。不过，越高的数据处理需求，性能下降越严重。
  + **多语言**：实际上，Storm的多语言更像是临时添加上去似的。因为，你的提交部分还是要使用Java实现。

下面，我们简单地认识一下Storm这个产品。

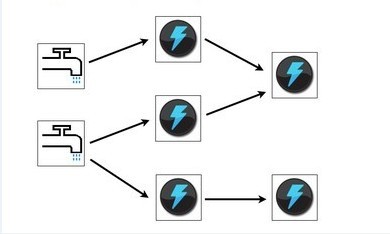
**认 识**

    Storm是一个免费开源、分布式、高容错的实时计算系统。Storm令持续不断的流计算变得容易，弥补了Hadoop批处理所不能满足的实时要求。Storm经常用于在实时分析、在线机器学习、持续计算、分布式远程调用和ETL等领域。Storm的部署管理非常简单，而且，在同类的流式计算工具，Storm的性能也是非常出众的。

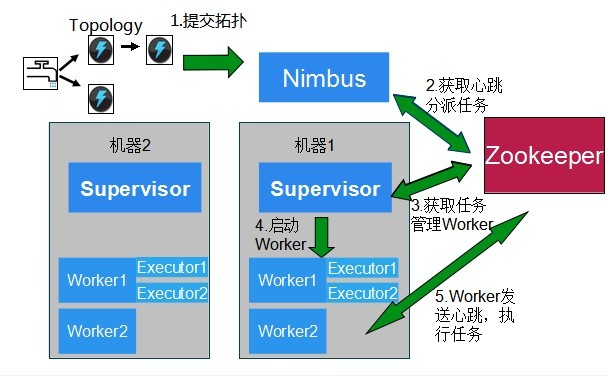
    Storm主要分为两种组件Nimbus和Supervisor。这两种组件都是快速失败的，没有状态。任务状态和心跳信息等都保存在Zookeeper上的，提交的代码资源都在本地机器的硬盘上。

* + Nimbus负责在集群里面发送代码，分配工作给机器，并且监控状态。全局只有一个。
  + Supervisor会监听分配给它那台机器的工作，根据需要启动/关闭工作进程Worker。每一个要运行Storm的机器上都要部署一个，并且，按照机器的配置设定上面分配的槽位数。
  + Zookeeper是Storm重点依赖的外部资源。Nimbus和Supervisor甚至实际运行的Worker都是把心跳保存在Zookeeper上的。Nimbus也是根据Zookeerper上的心跳和任务运行状况，进行调度和任务分配的。
  + Storm提交运行的程序称为Topology。
  + Topology处理的最小的消息单位是一个Tuple，也就是一个任意对象的数组。
  + Topology由Spout和Bolt构成。Spout是发出Tuple的结点。Bolt可以随意订阅某个Spout或者Bolt发出的Tuple。Spout和Bolt都统称为component。

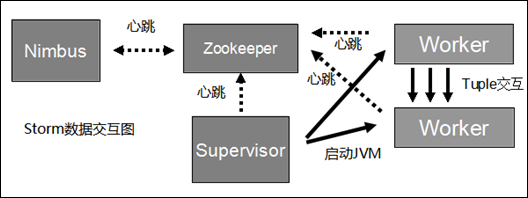
下图是一个Topology设计的逻辑图的例子。



下图是Topology的提交流程图。



下图是Storm的数据交互图。可以看出两个模块Nimbus和Supervisor之间没有直接交互。状态都是保存在Zookeeper上。Worker之间通过ZeroMQ传送数据。



虽然，有些地方做得还是不太好，例如，底层使用的ZeroMQ不能控制内存使用(下个release版本，引入了新的消息机制使用netty代替ZeroMQ），多语言支持更多是噱头，Nimbus还不支持HA。但是，就像当年的Hadoop那样，很多公司选择它是因为它是唯一的选择。而这些先期使用者，反过来促进了Storm的发展。

**发 展**

Storm已经发展到0.8.2版本了，看一下两年多来，它取得的成就：

* + 有50个大大小小的公司在使用Storm，相信更多的不留名的公司也在使用。这些公司中不乏淘宝，百度，Twitter，Groupon，雅虎等重量级公司。
  + 从开源时候的0.5.0版本，到现在的0.8.0+，和即将到来的0.9.0+。先后添加了以下重大的新特性：
    - 使用kryo作为Tuple序列化的框架（0.6.0）
    - 添加了Transactional topologies（事务性拓扑）的支持（0.7.0）
    - 添加了Trident的支持（0.8.0）
    - 引入netty作为底层消息机制（0.9.0）

Transactional topologies和Trident都是针对实际应用中遇到的重复计数问题和应用性问题的解决方案。可以看出，实际的商用给予了Storm很多良好的反馈。

* + 在GitHub上超过4000个项目负责人。Storm集成了许多库，支持包括Kestrel、Kafka、JMS、Cassandra、Memcached以及更多系统。随着支持的库越来越多，Storm更容易与现有的系统协作。Storm的拥有一个活跃的社区和一群热心的贡献者。过去两年，Storm的发展是成功的。

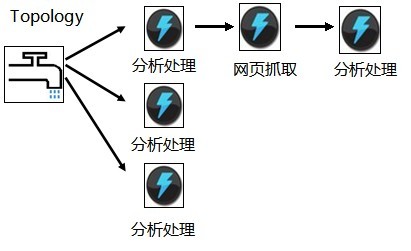
**当 前**

Storm被广泛应用于实时分析，在线机器学习，持续计算、分布式远程调用等领域。来看一些实际的应用:

* + 一淘-[实时分析系统pora](http://www.searchtb.com/2012/11/pora.html)：实时分析用户的属性，并反馈给搜索引擎。最初，用户属性分析是通过每天在云梯上定时运行的MR job来完成的。为了满足实时性的要求，希望能够实时分析用户的行为日志，将最新的用户属性反馈给搜索引擎，能够为用户展现最贴近其当前需求的结果。
  + 携程-[网站性能监控](http://www.programmer.com.cn/14601/)：实时分析系统监控携程网的网站性能。利用HTML5提供的performance标准获得可用的指标，并记录日志。Storm集群实时分析日志和入库。使用DRPC聚合成报表，通过历史数据对比等判断规则，触发预警事件。

如果，业务场景中需要低延迟的响应，希望在秒级或者毫秒级完成分析、并得到响应，而且希望能够随着数据量的增大而拓展。那就可以考虑下，使用Storm了。

* + 试想下，如果，一个游戏新版本上线，有一个实时分析系统，收集游戏中的数据，运营或者开发者可以在上线后几秒钟得到持续不断更新的游戏监控报告和分析结果，然后马上针对游戏的参数和平衡性进行调整。这样就能够大大缩短游戏迭代周期，加强游戏的生命力（实际上，zynga就是这么干的！虽然使用的不是Storm……[Zynga研发之道探秘：用数据说话](http://www.csdn.net/article/2011-08-26/303631)）。
  + 除了低延迟，Storm的Topology灵活的**编程方式**和**分布式协调**也会给我们带来方便。用户属性分析的项目，需要处理大量的数据。使用传统的MapReduce处理是个不错的选择。但是，处理过程中有个步骤需要根据分析结果，采集网页上的数据进行下一步的处理。这对于MapReduce来说就不太适用了。但是，Storm的Topology就能完美解决这个问题。基于这个问题，我们可以画出这样一个Storm的Topology的处理图。



我们只需要实现每个分析的过程，而Storm帮我们把消息的传送和接受都完成了。更加激动人心的是，你只需要增加某个Bolt的并行度就能够解决掉某个结点上的性能瓶颈。

**未 来**

在流式处理领域里，Storm的直接对手是S4。不过，S4冷淡的社区、半成品的代码，在实际商用方面输给Storm不止一条街。

如果把范围扩大到实时处理，Storm就一点都不寂寞了。

* + **Puma**：Facebook使用puma和Hbase相结合来处理实时数据,使批处理计算平台具备一定实时能力。 不过这不算是一个开源的产品。只是内部使用。
  + **HStreaming**：尝试为Hadoop环境添加一个实时的组件HStreaming能让一个Hadoop平台在几天内转为一个实时系统。分商业版和免费版。也许HStreaming可以借Hadoop的东风，撼动Storm。
  + **Spark Streaming**：作为UC Berkeley云计算software stack的一部分，Spark Streaming是建立在Spark上的应用框架，利用Spark的底层框架作为其执行基础，并在其上构建了DStream的行为抽象。利用DStream所提供的api，用户可以在数据流上实时进行count，join，aggregate等操作。

当然，Storm也有**Yarn-Storm**项目，能让Storm运行在Hadoop2.0的Yarn框架上，可以让Hadoop的MapReduce和Storm共享资源。

**总 结**

知乎上有一个挺好的问答：

问：实时处理系统（类似s4, storm）对比直接用MQ来做好处在哪里？

答：好处是它帮你做了： 1) 集群控制。2) 任务分配。3) 任务分发 4) 监控 等等。

需要知道Storm不是一个完整的解决方案。使用Storm你需要加入消息队列做数据入口，考虑如何在流中保存状态，考虑怎样将大问题用分布式去解决。解决这些问题的成本可能比增加一个服务器的成本还高。但是，一旦下定决定使用了Storm并解决了那些恼人的细节，你就能享受到Storm给你带来的简单，可拓展等优势了。

技术的发展日新月异，数据处理领域越来越多优秀的开源产品。Storm的过去是成功的，将来会如何发展，我们拭目以待吧。

**后记**

本文的重点是描述Storm的应用场景和未来的发展前景，让大家对Storm有一个初步的印象。如果，要落地使用的朋友，在网上可以找到很多优秀的Storm的技术文章。例如：Storm的核心贡献者[徐明明](http://xumingming.sinaapp.com/)的博客和淘宝关于storm的文章。