# Storm笔记

## 1背景

### 1.1应用场景举例[1]

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骂娘了。

### 1.2假如自己实现一个实时计算系统[1]

抛开s4、storm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。实时计算系统，延迟是一定要低的；
2. 高性能；
3. 分布式。系统都是为应用场景而生的，如果应用场景、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。

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，举个例子，使用消息队列+分布在各个机器上的工作进程就可以。除了这几个需求，还要考虑：

1. 容易在上面开发应用程序。系统不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；
2. 消息不丢失。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；
3. 消息严格有序。有些消息之间是有强相关性的，比如同一个淘宝宝贝的更新和删除操作消息，如果处理时搞乱顺序完全是不一样的效果了。

### 1.3 storm特性[2]

对应上面几点，storm提供了如下特性：

1. 简单的编程模型。类似于MapReduce降低了并行批处理复杂性，Storm降低了进行实时处理的复杂性；
2. 可以使用各种编程语言。你可以在Storm之上使用各种编程语言。默认支持Clojure、Java、Ruby和Python。要增加对其他语言的支持，只需实现一个简单的Storm通信协议即可；
3. 容错性。Storm会管理工作进程和节点的故障；
4. 分布式和水平扩展。计算是在多个线程、进程和服务器之间并行进行的；
5. 可靠的消息处理。Storm保证每个消息至少能得到一次完整处理。任务失败时，它会负责从消息源重试消息；
6. 快速。系统的设计保证了消息能得到快速的处理，使用MQ/netty作为其底层消息队列；
7. 本地模式。Storm有一个“本地模式”，可以在处理过程中完全模拟Storm集群。这让你可以快速进行开发和单元测试。

## 2使用情况和社区[3]

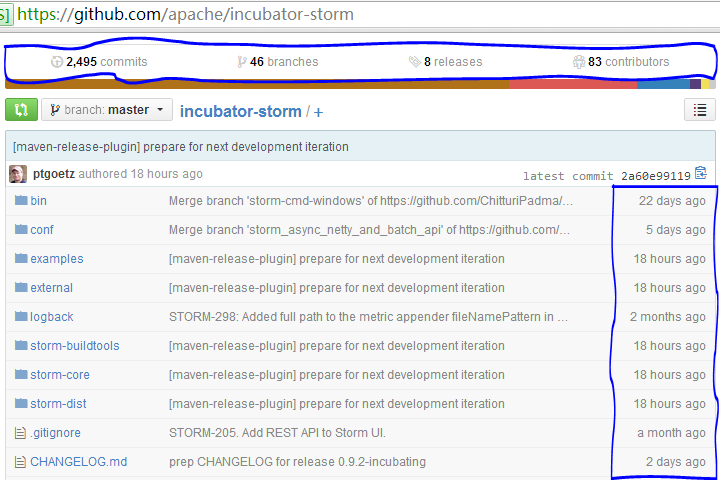
### 2.1使用情况



### 2.2 邮件列表和github

看起来邮件列表比较活跃，源码更新也比较频繁，整体还算活跃。





## 3架构和原理

## 4安装入门

## 5源码

## 6关键技术

## 7参考资料