目前采用的流处理系统主要有以下三种：

kafka,storm,spark,flume

其中flume和kafka主要是数据源收集系统，是storm和spark steaming的前端

主要牵涉到三个环节：

1. 数据的接入(接口问题)
2. 集群平台的搭建和数据的处理【核心处理】
3. 处理结果的导出（输出方式问题）

## 一、Kafka

kafka是高吞吐量的分布式发布订阅消息系统，可以处理消费者规模网站中的动作数据流，例如网页的浏览，搜索和其它用户还行为，hadoop是日志数量和离线分析系统，若要求实时处理，kafka的目的是通过hadoop的并行加载机制来统一线上和离线的消息处理，也是通过集群机来实现实时的消费。

kafka集群的配置：

kafka数据的接入：

kafka数据处理结果的导出：

## 二、Storm

storm的原理：实时流计算,分布式系统，运维多简单，高度容错，无数据丢失，Storm的多语言接口不完善，提交部分任然是Java实现。

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。

参考文章：[浅谈Storm流式处理框架](http://mp.weixin.qq.com/s?__biz=MzA5MTc0NTMwNQ==&mid=402831112&idx=1&sn=b80926eb33902e8a62d930fb4dd5bf7d&scene=0#rd)

## 三、Spark

spark streaming的python接口一直配置不起来,在pyspark进而spark-submit中运行起来了。

Spark可以单独运行吗，不需要hadoop的平台。一个hadoop作业指的是mr作业。

spark是基于内存计算的，计算的中间结果也会存在内存中，可以进行反复的迭代计算，而MR计算的中间结果是存磁盘的。MR的task对应一个container,而spark是基于线程池来实现的，资源分配的快。如果spark不把hdfs作为数据源，是不是就需要用hadoop了，spark也是一个单独的生态。spark是低延迟小文件处理，而hadoop是高延迟批处理大吞吐。spark一般部署在yarn或者mesos上。spark on yarn和spark的区别之处在于，让spark运行在一个通用的资源管理系统上，最大的的好处是降低运维成本和提高资源的利用率（资源按需分配），部分容错性和资源管理交由统一的资源管理系统完成，而单独的spark是无法有效提高资源利用率的。

spark提供了集群的分布式内存抽象，也就是所说的RDD，spark提供了RDD的两类操作，转换和动作，转换包括map,flatMap,filter, union,sample,join,groupByKey,reduceByKey, sortByKey等等，动作是返回结果，包括collect,reduce,count等，抽象层次更高，功能更多，调用更灵活。所处理的数据都是放在内存中，速度更快。mapreduce则抽象层次比较低，只有map,reduce两个基本功能。