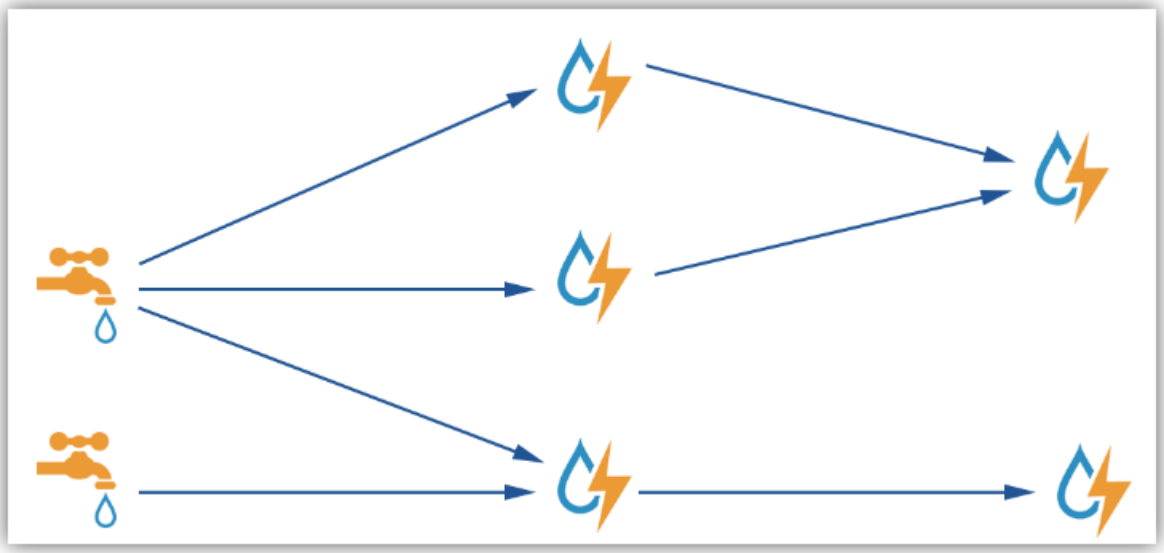
1.1 简介

Storm 是 Apache 旗下免费开源的分布式实时计算框架。Storm可以轻松、可靠地处理无限数据流，对实时分析、在线机器学习、连续计算、分布式RPC，ETL等提供高效、可靠的支持。



1.2 什么是分布式计算

分布式计算，将一个任务分解为多个任务，分发给多台计算机，节约整体计算时间。

注：

> ***集中式计算，与分布式相对，一个任务一台计算机负责，多个任务多个计算机，不节约整体计算时间。***

1.3 Storm 分布式实时计算框架的要点

① 开源，遵循Apache理念，项目创立便是开源

② 流式计算，非常的快，每个节点每秒可以处理百万元组数据

③ 适用于数据实时处理而非批量处理（例如Hadoop）

④ 分布式系统，可以充分的利用计算机集群资源

⑤ 擅长处理海量数据。

1.4 与Hadoop比较

与 Hadoop磁盘级计算不同，storm为流式计算即内存计算，Hadoop在进行计算时数据在磁盘中需要读写磁盘，而读取内存比读写磁盘的速度快n个数量级别。磁盘访问延迟约为内存访问延迟的75000倍，所以storm更快。两者面向的领域不同，Hadoop为批量处理，基于任务调度；storm为实时处理，基于流。

注：

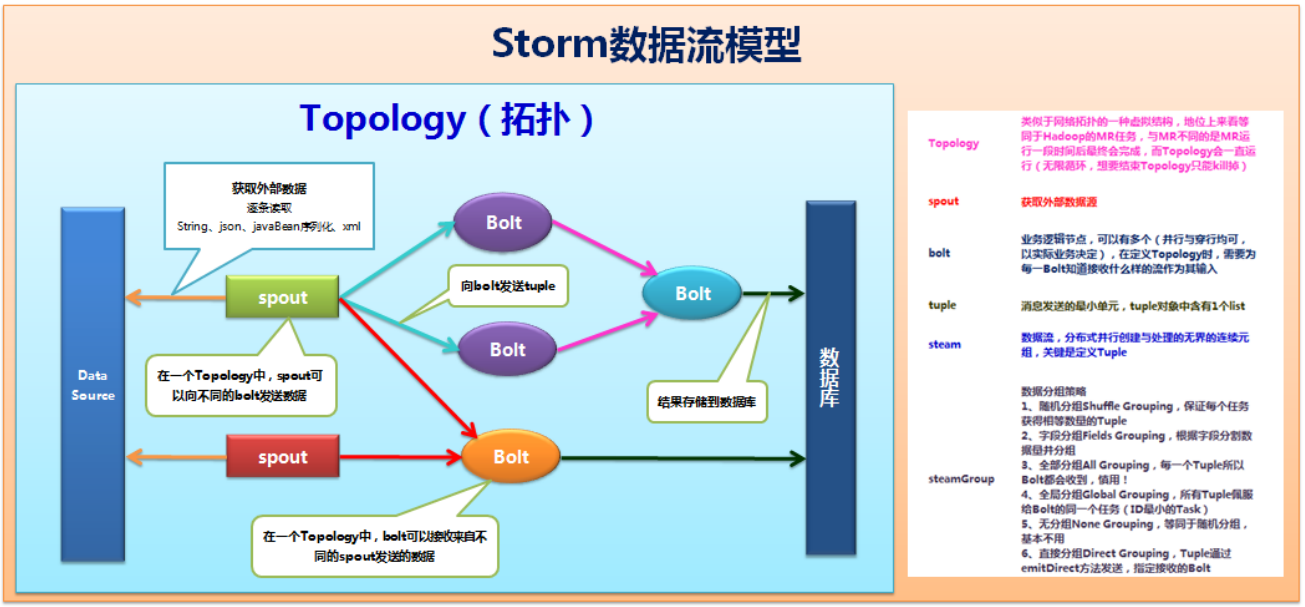
延迟，指数据从产生到运行计算产生结果的时间。

吞吐，指系统单位处理的数据量。

1.5 优势

Storm 拥有低延迟、高性能、分布式、可扩展、容错等特性，可确保消息不丢失且严格有序
以下六点为Storm优势：
① 简单的计算编程，类似于MapReduce降低并行处理的复杂性，Storm降低了进行实时处理的复杂性。
② 实时性，系统不停运转等待任务到达，接受任务后迅速处理。
③ 可伸缩性，随时添加新的节点扩充集群的计算能力。
④ 容错性，管理所有工作进程和节点的故障，并且循环修复故障。
⑤ 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。
⑥ 多语言支持，Storm提供多语言协议支持。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。

1.6 Storm的数据流模型



Storm分布式计算结构称为Topology（拓扑），由stream（数据流）、spout（数据流的生成者）、bolt（运算）组成。

1.6.1 topology

Storm分布式计算结构称为Topology（拓扑），类似于网络拓扑的一种虚拟结构，地位上等同于Hadoop的MR任务，与MR不同的是MR运行一段时间就会完成，而Topology会一直运行知道手动kill掉。

1.6.2 stream

数据流，分布式并行创建与处理的无界的连续元组，关键是定义tuple

1.6.3 tuple

元组，storm的核心数据结构，消息发送的最小单元，tuple是包含一个或多个键值对的列表

1.6.4 spout

spout为一个Storm Topology的主要数据入口，充当采集器的角色，连接到数据源，将数据转化为一个个tuple，并将tuple作为数据流进行发射。

注：

Storm为spout提供了简易的API，开发一个spout的主要工作就是编写代码从数据源或者API消费数据。数据源的种类：

① Web或者移动程序的点击数据

② 应用程序的日志数据

③ 传感器的输出

1.6.5 bolt

业务逻辑运算节点，可以有多个（并行与串行均可，以实际业务决定），选择性地输出一个或者多个数据流。bolt可以订阅多个spout或者其他bolt发射的数据流，这样可以建立多个复杂的数据流转换网络。在定义Topology时，需要为每一个bolt指定接收什么样的流作为输入（通过Storm grouping）。

注：

bolt可以执行各式各样的处理功能，通常我们会将业务逻辑写在bolt，典型的功能：

① 过滤tuple

② 连接（join）和聚合操作（aggregation）

③ 计算

④ 数据库读写

1.6.6 数据流分组

Storm grouping（数据流分组）定义一个Topology中每个bolt接受什么样的流作为输入。

Storm定义了六种内置数据流分组的定义：

① 随机分组（Shuffle grouping）：这种方式下元组会被尽可能随机地分配到 bolt 的不同任务。（tasks）中，使得每个任务所处理元组数量能够能够保持基本一致，以确保集群的负载均衡。

② 按字段分组（Fields grouping）：这种方式下数据流根据定义的“字段”来进行分组。例如，如果某个数据流是基于一个名为“user-id”的字段进行分组的，那么所有包含相同的“user-id”的元组都会被分配到同一个task中，这样就可以确保消息处理的一致性。

③ 完全分组（All grouping）：将所有的tuple复制后分发给所有的bolt task。每个订阅的task都会接收到tuple的拷贝，所有在使用此分组时需小心使用。

④ 全局分组（Global grouping）：这种方式下所有的数据流都会被发送到 Bolt 的同一个任务中，也就是 id 最小的那个任务。

⑤不分组（None grouping）：使用这种方式说明你不关心数据流如何分组。目前这种方式的结果与随机分组完全等效，不过未来 Storm 社区可能会考虑通过非分组方式来让 Bolt 和它所订阅的 Spout 或 Bolt 在同一个线程中执行。

⑥ 指向型分组（Direct grouping）：数据源会调用emitDirect（）方法来判断一个tuple应该由那个Storm组件来接收。只能在声明了是指向型的数据流上使用。

1.7 可靠的数据处理

Storm 可以通过拓扑来确保每个发送的元组都能得到正确处理。通过跟踪由 Spout 发出的每个元组构成的元组树可以确定元组是否已经完成处理。每个拓扑都有一个“消息延时”参数，如果 Storm 在延时时间内没有检测到元组是否处理完成，就会将该元组标记为处理失败，并会在稍后重新发送该元组。

注：

为了充分利用 Storm 的可靠性机制，你必须在元组树创建新结点的时候以及元组处理完成的时候通知 Storm。这个过程可以在 Bolt 发送元组时通过 OutputCollector 实现：在 emit 方法中实现元组的锚定（Anchoring），同时使用 ack 方法表明你已经完成了元组的处理。

Storm的tuple锚定和应答确认机制中，当打开可靠传输的选项，传输到故障节点上的tuples将不会受到应答确认，spout会因为超时重新发射原始的tuple。这样的过程会一直重复直到Topology从故障中恢复开始正常处理数据。

在Storm集群中真正运行Topology的主要有三个实体：工作进程、线程和任务。Storm集群中的每台机器上都可以运行多个工作进程，每个工作进程又可创建多个线程,每个线程可以执行多个任务,任务是真正进行数据处理的实体，我们开发的spout、bolt就是作为一个或者多个任务的方式执行的。因此，计算任务在多个线程，进程和服务器之间并行进行,支持灵活的水平扩展。