<https://www.bilibili.com/video/av25200735?from=search&seid=10393394128202278433>

实时流计算，看特性，以及清楚特性是怎么实现的（看问题也是一样，看问题是怎么发生的）

Strom是无状态的

1. 数据单元
2. Stream 数据单元集合
3. Spout源源不断产生数据的地方（多实例）
4. Bolt转接点（多实例）
5. Task真正执行的（多线程）
6. Group---流的bolt之间传递的方式（广播，shuffle，filed等等）
7. Nimbus，把task进行分发，supervisor是工作节点
8. Worker process创建executor

Q:spout每个方法的解析？spout可以有多个么？这些方法都是怎么执行的？

Q:对于集群的麻烦之处就是你不清楚到底是怎么运行的，分发的到底是什么？

<https://blog.csdn.net/xlgen157387/article/details/77839107> 这一篇是专门解释并行度及其有关的并发流程

Q:storm 是如何进行任务的分发的？或者问storm集群环境下任务的发布

已看完3

初步理解：任务的分发应该是计算好并行度，需要多少executor，多少task，多少jvm，然后把每个bolt（spout也看出）分发到相应节点上，然后执行。

比如：先创建tp，类似于spark的dag，有了详细的状态，

<https://blog.csdn.net/xlgen157387/article/details/77839107> 这个可以很好的理解并行度，但是还是无法解释node层面的问题

<https://blog.csdn.net/ll666634/article/details/79211178> nimbus在运行之前会做好映射，映射的规则是怎样的呢？这里实质是指负载均衡。可以。任务分配信息是保存在zookeeper上的，supervisor从zookeeper读取一些。Executor跟node有一定的key-value对应。

Bolt中各个方法的声明周期：

在storm-user中看到的一个关于Storm Bolt内部实现的执行顺序的[提问](https://groups.google.com/forum/?fromgroups#!topic/storm-user/FuRHdHt71mk), 觉得对理解Bolt内部实现有帮助, 记录一下.   
  
prepare方法在worker初始化task的时候调用.   
  
execute方法在每次有tuple进来的时候被调用   
  
cleanup实际上仅用于本地模式(local mode), 在集群模式(cluster mode)下该方法很难保证每次被正确执行.   
  
declearOutputFields方法仅在有新的topology提交到服务器, 用来决定输出内容流的格式(相当于定义spout/bolt之间传输stream的name:value格式), 在topology执行的过程中并不会被调用.

Q:现在最关注的的就是，任务分发的是什么代码？

A:<http://www.tianshouzhi.com/api/tutorials/storm/104> 从这个里面可以知道，每个节点上都有一份拓扑jar包

Q：多个拓扑如何在有限的slot上运行（slot其实就是节点的本质，在一个节点上配置不同的端口就可以实现伪集群）

A:其实slot可以理解为一个server，nimbus通过这个server进行通信，并不是具体执行现场的

Q:storm多个topology是如何运行的?因为很多时候slot都是有限的

A:直接运行就好，slot不等于拓扑的数量，之前肯定以为slot就是拓扑了。

Q:tuple是通过什么传递的，socket？rpc？还是什么

A:socket

Bolt各个方法的声明周期：

构造：对象的创建，nimbus上，什么时候不用构造，就是在串行化的时候，通过nimbus创建一次，然后序列化发送到supervisor

Prepare：在工作节点上执行，所以有些对象要在这里创建，比如一些连接，如jdbc

Execute：流来的时候或者数据来的时候

Cleanup：结束的时候

Spout的各个方法的生命周期

构造方法：对象的创建在nimbus上创建整个拓扑值运行一次偶，跟bolt一样

Open：跟bolt的prepare很像，在

Nexttuple：也是数据来的时候

Ack

理解分组策略很容易：

分组策略是面向bolt的，决定的流以什么样的方式到组里面（随机，广播，按照filed，甚至自定义，有些要注意，对于有些数据是不可分割的话，要特别注意分组的方式）

Q:storm是移动计算还是移动数据

并行度生效是在bolt（spout）分发的时候才会，构造方法永远只执行一次。

K:并行度必须跟跟分组策略一起才有意义

Q:为什么使用静态变量不可以

A:其实是可以的，在可以序列化的情况下。这里不行是作者想要共性状态，也就是说这个值会变，但又想保持统一。这个时候只能选择存在第三方的方式。

K：原来消息的可靠性是需要自己实现的！包括ack和fail的处理，虽然现在用的是别人写好的。还有就是在写可靠性程序的时候，可以不考虑线程安全问题，因为，一个消息线是唯一的，不会共享。但是多个task就会出问题，还是需要线程安全的类去做

Spout1 <>bolt1<>bolt2是固定的

Q:Storm这种消息可靠性是如何实现的？（分布式原理里面有没有可靠性的说法）

一般来说一个work一个node或者说一个work一个slot，一个工作进程仅代表一个拓扑

如果work大于slot怎么办，当不能平均分配的时候strom会怎样做，是放弃分配还是，如果设置了task没设置excutor呢。按照work》executor》task创建优先级，什么意思呢？也就说如果就给了俩executor，，那你设置100个task也只会在一个exe上执行。

Node

Work

Execuor

Setack是什用？

虽然可以设置多个任务，但是有可能不会用到，比如字段分组，字段数少于分组数，能用的bolt也只有字段数个是工作，其他会闲置

K：reblance，可以调节work，调节exe



Trident对于原始的storm开发方式可以说是一种新的开发方式，不过有些概念是可以相互映射的。可以理解为spark里面的spark sql

特点：

保证消息被处理一次

有状态

支持事物：事物ID

函数性操作：把Bolt看成function，因为bolt本来就定义为

Trident不想原生storm那样是一条一条的，通过他给的那几个样例spout你就会发现，都有batch这个关键词，

相对来说，难理解的就是group概念在trident出现了变化，使用额其他几种方式来实现

在trident中你可以这样理解，一个调用链就是一个流。流的重用就好比strom中流的多路分发

Batchid来实现批次的

关于两个id

Batchid

Txid

其实如果是存业务的话确实基本bolt就好了，没有什么数据的处理。。。

要锻炼出脑海中形成topo的技能

可以按批次聚合，关键还是看具体的需求吧

聚合：

分区聚合

批次聚合

流聚合

要对分组，聚合等数据的变化要非常清晰。

Learning storm

随机访问和实时读取

**Hbase的缺点：**

* 单一RowKey固有的局限性决定了它不可能有效地支持多条件查询[2]
* 不适合于大范围扫描查询
* 不直接支持 SQL 的语句查询

关键要总结一下Trident的操作以及其聚合，感觉很重要