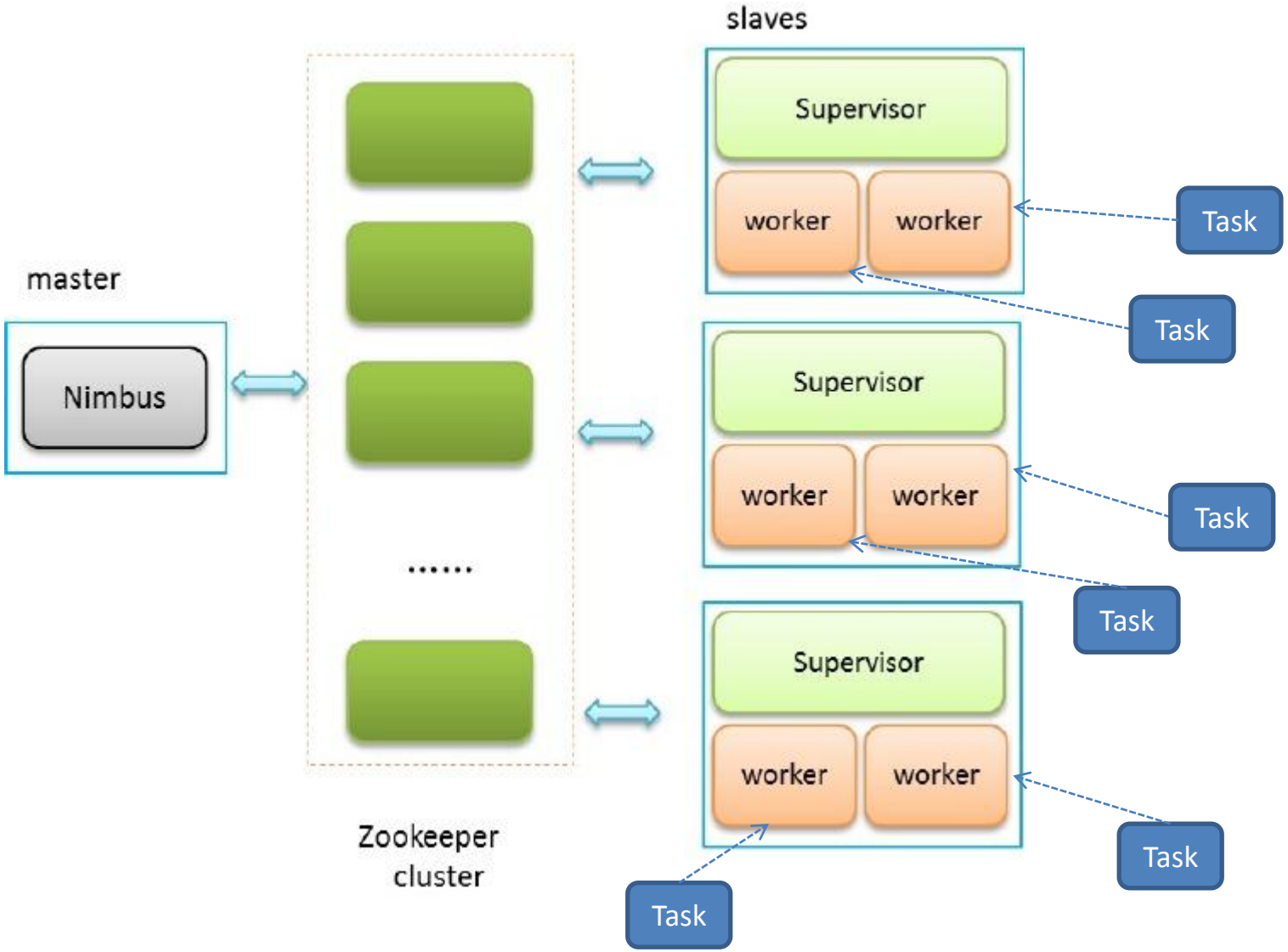
Storm特性

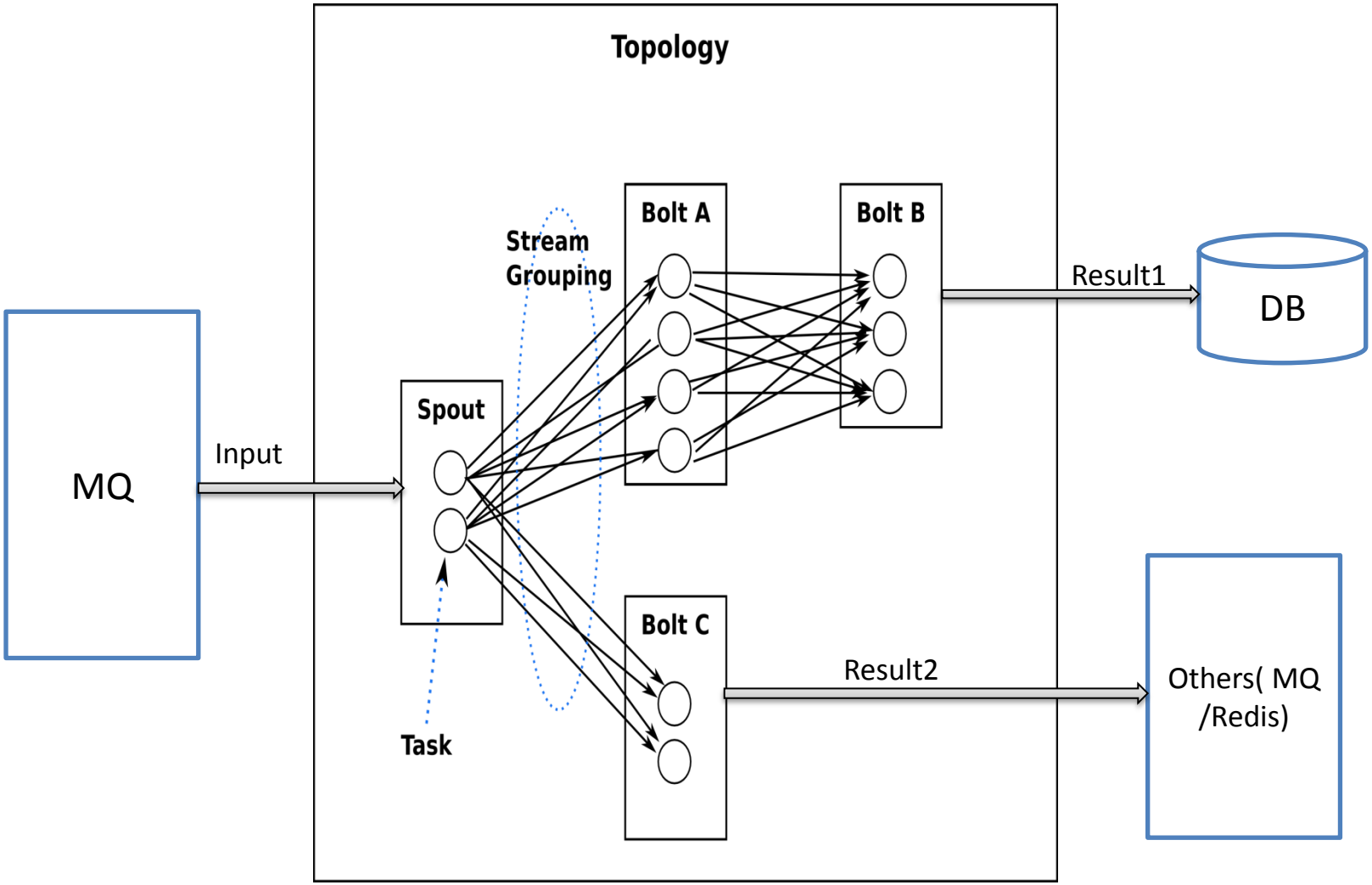
Storm系统架构


slaves
Supervisor
worker
worker
Task
Task
master
Nimbus
Supervisor
worker
worker
Task
Task
.....
Supervisor
worker
worker
Task
Task
Zookeeper
cluster

Storm组件

- NimBus: 责资源分配和任务调度
- Supervisor:负责接受nimbus分配的任务，启动和停止属于自己管理的worker进程
- Work:运行具体处理组件逻辑的进程
- Task: worker中每一个spout/bolt的线程称为一个task.

Storm Topology



Storm的关键概念

- 消息流中的Tuple
- 消息源: Spouts
- 消息处理者: Bolts
- Stream groupings： 消息分发策略

消息流中的Tuple

Tuple是一次消息传递的基本单元， tuple里的每个字段一个名字,并且不同tuple的对应字段的类型必须一样。tuple的字段类型可以是： integer, long, short, byte, string, double, float, boolean和byte array； 还可以自定义类型 — 只要实现对应的序列化器。每个消息流中包括若干个tuple。

消息源: Spout

消息源Spout是storm里面一个topology里面的消息生产者。一般来说消息源会从一个外部源(例如：MQ)读取数据并且向topology里面发出 tuple。消息源Spouts可以是可靠的也可以是不可靠的。如果一个tuple没有被storm成功的处理，一个可靠的Spout可以重新发射一个tuple， 但是一个不可靠的消息源Spout一旦发出一个tuple就不可能再发了。

消息处理者: Bolt

所有的消息处理逻辑被封装在Bolt里面。 Bolt可以做很多事情： 过滤，聚合， 查询数据库等；如果遇到复杂的处理流程也可能将tuple发送给另一个Bolt进行处理。

stream grouping消息分发策略

stream grouping用来定义一个Tuple应该如何分配给哪个Bolts；storm里面有6种类型的stream grouping:

- Shuffle Grouping: 随机分组， 随机派发stream里面的tuple， 保证每个bolt接收到的tuple数目相同。
- Fields Grouping：按字段分组， 比如按userid来分组， 具有同样userid的tuple会被分到相同的Bolts， 而不同的userid则会被分配到不同的Bolts。
- All Grouping： 广播发送， 对于每一个tuple， 所有的Bolts都会收到。
- Global Grouping: 全局分组， 所有的tuple被分配到一个bolt。

- Non Grouping: 不分组， 这个分组的意思是说stream不关心到底谁会收到它的tuple。目前这种分组和Shuffle grouping是一样的效果。
- Direct Grouping: 直接分组, 这是一种比较特别的分组方法，用这种分组意味着消息的发送者指定由消息接收者的哪个task处理这个消息。 只有被声明为Direct Stream的消息流可以声明这种分组方法。而且这种消息tuple必须使用emitDirect方法来发射。消息处理者可以通过TopologyContext来获取处理它的消息的taskid .

常用Topology模式

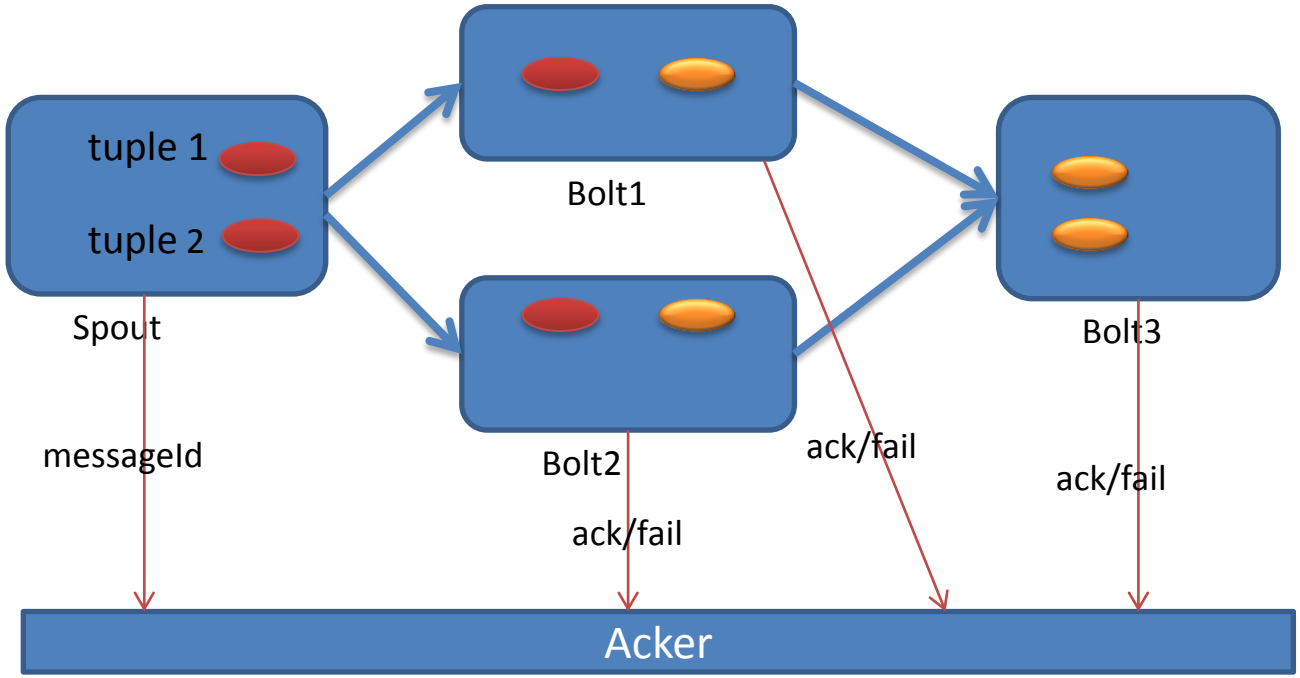
- 基本模式(BasicBolt)
- 流聚合(Stream join)
  - 根据某个公共字段，一个流式Job可以混合2个或2个以上的数据流到一个数据流,有点类似Sql中的join;但Sql中join的输入是有限的，并且join的语义是非常明确的，而流聚合的语义是不明确的并且输入流是无限的；参看BoltDeclarer.fieldsGrouping
- 批处理(Batching)
  - 把一组tuple一起处理， 而不是一个个单独处理。比如，批量更新数据库。如果想让数据处理具有可靠性，正确的方式是保存这些tuple对象的引用直到bolt批量处理这些tuple了。一旦这个批量操作结束， 可以批量的ack这些tuple

- 内存缓存+字段分组(In-memory caching + fields grouping combo)
  - 根据fields grouping规则在不同的bolt中可以缓存不同的内容。
- 计算Top N(Stream top N)
- 用TimeCacheMap来高效地保存一个最近被更新的对象的缓存
  - 在内存里面保存一些最近活跃的对象，以及让那些不再活跃的对象自动过期

- 分布式RPC: CoordinatedBolt和KeyedFairBolt
  - 用storm做分布式RPC应用的时候有两种比较常见的模式：它们被封装在 CoordinatedBolt andKeyedFairBolt 。
  - CoordinatedBolt包装bolt，并且确定什么时候bolt已经接收到所有的tuple,它主要用Direct Stream来实现
  - KeyedFairBolt包装bolt并且保证topology同时处理多个DRPC调用，而不是串行地一次只执行一个。
  - 更多有关分布式RPC的信息可以看这里

可靠性
(Guaranteeing Message Processing)

storm保证从spout发出的每个tuple都会被完全处理; 从spout发射的一个tuple，因它可以引起其它成千上万个tuple。在storm里面一个tuple被完全处理的意思是： 这个tuple以及由这个tuple所导致的所有的tuple都被成功处理。如果一个tuple在timeout所指定的时间内没有成功处理则这个tuple会被认为处理失败了。


tuple 1
tuple 2
Spout
Bolt1
Bolt2
Bolt3
messageId
ack/fail
ack/fail
ack/fail
Acker

事务性Topology

1. 强顺序性；如果一个tuple没有被完整的处理完，就不会处理下一个tuple，说简单一些，就是，采用同步方式
2. 可以批量处理
3. 因此将一个计算任务拆分为2个阶段： 1. processing 阶段：这个阶段可以并发 2. commit阶段：这个阶段必须强顺序性，因此，一个时刻，只有一个batch在被处理 任何一个阶段发生错误，都会完整重发batch

Trident

- 批量处理
- 聚合操作
- Tident的操作
  - Partition-local operations，对每个partition的局部操作，不产生网络传输
  - Repartitioning operations：对数据流的重新划分（仅仅是划分，但不改变内容），产生网络传输
  - Aggregation operations：聚合操作
  - Operations on grouped streams：作用在分组流上的操作
  - Merge、Join操作

遗留问题

- NimBus HA
  - nimbus ip 地址动态获取
  - Topology 代码存储方案可定制 (NimbusCloudStorage)
  - nimbus多节点选举，宕机自动切换
  - nimbus leader状态ui展示
  - https://issues.apache.org/jira/browse/STORM-166
- Topology online update
  - https://issues.apache.org/jira/browse/STORM-167

Refrence

- http://storm.apache.org/documentation/Home.html
- http://xumingming.sinaapp.com/category/storm