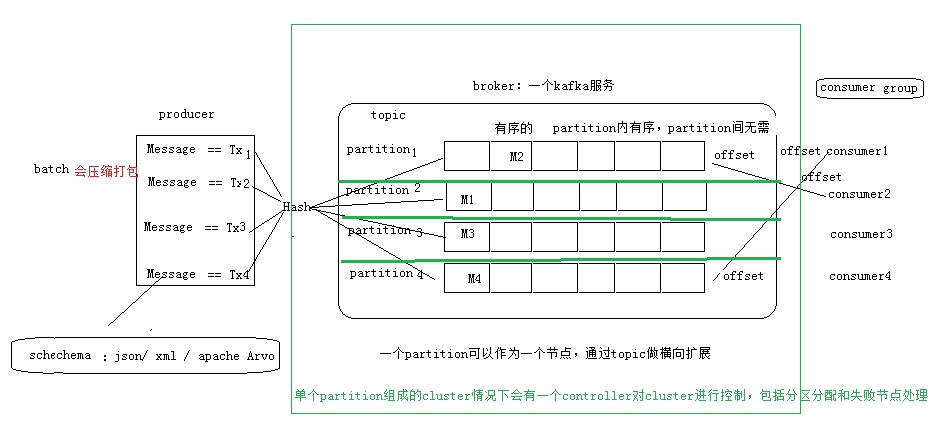
交易在经过batch压缩后，将消息发送给kafka集群，集群中对交易进行hash后，放入到partition中，partition中的顺序是一定的，partition间的顺序不固定，此时可以通过controller对顺序进行某种操作，确定顺序



kafka是一个基于分布式日志提交机制设计的发布订阅系统，数据在kafka中持久化，用户可以随时读取，由于数据是分布式存储的，所以可以提高容错性，可扩展。

Message和Batchs

kafka中最基本的消息单元是message，类似于数据库中的一条行或者一条记录。kafka并不关心message中存的是什么，索引消息的具体格式和kafka无关。消息可以有一个可选的key，这个key也是个字符数据，与消息一样，对kafka也是透明的。key用来确定消息写入时写入到那个partitions，最简单的方式是将key作为hash字符串，拥有相同key的消息，肯定会进入到同一个分区。

为提高效率，kafka以批量的方式写入数据，一个batch就是一个消息的集合，这一组消息会进入到同一个topic和partition的。【每个partition中会存储有own registry，其中包含有consumer的ID值】。每一个而消息进行一次网络传输都会消耗性能，因此将消息收集放到一起可以提高效率。当然这会引起更好的延迟和吞吐率的问题。batch越大，同一时间处理的消息就越多，当然网络延迟和吞吐量也会加大。batch通常会对消息进行压缩，以提升效率。

Message：kafka中的消息单元

Batch：将Message进行打包并压缩，提升吞吐率。

Schemas

对kafka来说，消息本身是透明的，这样就可以对消息定义更多容易理解的内容。根据个人的需求的不同，消息会有多种schema.可以时json或者xml，容易阅读。但缺点是在不同的模式版本中间缺乏一些处理鲁棒性和扩展性的机制。也可以使用apache Avro（hadoop序列化的），提供了紧凑的序列化格式，也不需要重新生成代码，具有很强的数据类型和模式，具有很好的向前扩展和向后兼容的能力。kafka中的数据是连续的，数据在进入的同时也可能被消费，这样数据的格式就很重要了，如果数据格式发生变化，消费的应用方面也需要做出调整。如果事先定义好了存储格式，那么读取数据的时候就不需要做特殊处理了。

Topic和Partition

Message都是以主题topic的方式组织在一起的，topic可以看作是一个数据库表中的表或者一个系统文件中的目录。一个topic由一个broker上的一个或多个partition组成，kafka中的数据是以log形式存储的，一个partition就是一个log。message以追加的方式加入到log中，消费的时候从头开始消费。partition内的message是有序的，但是partition间的顺序是无序的，如果要保证partition中的消息按照时间的先后顺序进行排序，那么需要设置partition的数目为1。partition的存在提高了kafka的扩展性，也就是一个partition可以当作一个服务节点，这个一个topic就可以在多个服务节点间进行横向扩展。kafka中的stream说的是以流的方式看待producer发送来的数据，不以分区的方式看待。在同类的如apache samza，storm等也是以这种方式看待的。

每个分区在物理上就是一个文件夹，以“主题名称——分区编号”来命名，每个分区有一个至多个副本(Replica)，分区的副本在逻辑上抽象为一个日志对象(log),即分区的副本与日志对象时一对一的关系。分区的副本分布在集群不同的代理上。每个主题对应的分区数可以在kafka启动时加载的配置文件中进行配置。

Leader副本和Follower副本

由于分区存在多个副本，为了保证多个副本间数据的一致性，kafka会选择一个副本作为leader副本，其他副本作为follower副本。只有leader副本才能进行读写请求，同时还需要保证副本之间的一致性。

offset偏移量

发布到分区消息会追加到日志文件的末尾，每一条消息在日志文件中的位置都会对应一个按需递增的偏移量。kafka不提供索引机制到存储偏移量，消费者可以通过控制偏移量来对消息进行消费。为了保证消息被顺序消费，消费者已消费的消息的偏移量也会保存下来。

日志段

一个日志又被分为许多日志段，日志段是kafka日志对象的最小单位，也是一个逻辑概念。一个日志段对应磁盘上具体日志文件和两个索引文件。日志文件存储实际的消息信息，索引文件分为.index存储偏移量和.timeindex存储消息的时间戳。

Producer和Consumer

producer也叫publisher或writer，默认情况下，producer不关心message进入那个partition，它会自动在多个partition之间进行负载均衡。message一般通过hash的方式确定进入那个partition，这意味者相同hash的message会进入到同一个partition。当然也可以指定message进入partition的方式。

Consumer也叫subscribe或reader，consumer订阅一个或多个topic，然后按照顺序读取topic中的message。consumer需要记录读取到的message的offset，每个message在partition中只有唯一一个固定的offset，通过存储最后消费的offset，消费者应用可以在重启或者停止后重新从之前的offset位置继续读取，实现的方式有zookeeper或者kafka自己。consumer一般 以consumer group的方式加入一个topic对message进行消费。

Broker和Cluster

单独的kafka也叫broker，broker从producer那里获取message，分配offset并保存到磁盘上。同时将offset提供给consumer，让consumer读取分区上的消息，同时将message提供给consumer。在kafka的集群中，有一个controller，由controller进行集群分区的分配和失败节点的处理。一个partition只能出现在一个broker上，并且这个partition也被叫做分区的leader。一个分区可以分配给多个broker，达到多备份的效果。这种多机备份在一个broker宕掉后，可以自动选出其他的broker选出服务。但是producer和consumer必须连接到leader上才能正常工作。

代理

kafka集群中的kafka实例，用ID来区分，也叫brokerid

ISR

kafka在zookeeper中动态维护了一个ISR（In-Sync-Replica）,即保存同步的副本列表，其中保存了与leader副本保持同步消息的所有副本的代理节点的ID，如果某个副本所在节点宕机或落后太多，那么该副本节点将会从列表中去除掉。

zookeeper

kafka利用zookeeper来保存元数据信息：代理节点信息，集群信息，旧消费信息和消费信息偏移量，主题信息，分区状态信息，分区副本的分配方案信息，动态配置信息等。kafka会在启动或者运行过程中向这些节点保存元数据信息，通过监听机制来注册监听器来监听这些信息的状态变化，从而有zookeeper来负责管理维护kafka集群，同时zookeeper能够很方便的对kafka集群进行水平扩展。

kafka高吞吐量的一个保证是：每条消息被追加到一个分区中，是顺序写磁盘。