<https://www.cnblogs.com/likehua/p/3999538.html>

问题导读：

1.zookeeper在kafka的作用是什么？

2.kafka中几乎不允许对消息进行“随机读写”的原因是什么？

3.kafka集群consumer和producer状态信息是如何保存的？

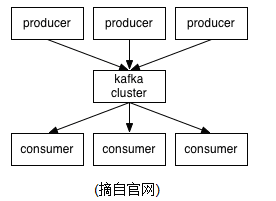
4.partitions设计的目的的根本原因是什么？

http://www.aboutyun.com/static/image/hrline/4.gif

一、入门

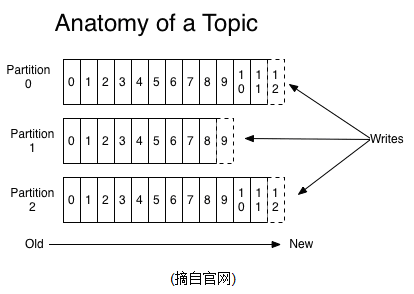
    1、简介

    Kafka is a distributed,partitioned,replicated commit logservice。它提供了类似于JMS的特性，但是在[设计](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%C9%E8%BC%C6&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=4&seller_id=1&di=128)实现上完全不同，此外它并不是JMS规范的实现。kafka对消息保存时根据Topic进行归类，发送消息者成为Producer,消息接受者成为Consumer,此外kafka集群有多个kafka实例组成，每个实例([server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128))成为broker。无论是kafka集群，还是producer和consumer都依赖于zookeeper来保证系统可用性集群保存一些meta信息。

<ignore\_js\_op>

   2、Topics/logs

    一个Topic可以认为是一类消息，每个topic将被分成多个partition(区),每个partition在存储层面是append log文件。任何发布到此partition的消息都会被直接追加到log文件的尾部，每条消息在文件中的位置称为offset（偏移量），offset为一个long型数字，它是唯一标记一条消息。它唯一的标记一条消息。kafka并没有提供其他额外的索引机制来存储offset，因为在kafka中几乎不允许对消息进行“随机读写”。

<ignore\_js\_op>

    kafka和JMS（Java Message Service）实现(activeMQ)不同的是:即使消息被消费,消息仍然不会被立即删除.日志文件将会根据broker中的配置要求,保留一定的时间之后删除;比如log文件保留2天,那么两天后,文件会被清除,无论其中的消息是否被消费.kafka通过这种简单的手段,来释放磁盘空间,以及减少消息消费之后对文件内容改动的磁盘IO开支.

    对于consumer而言,它需要保存消费消息的offset,对于offset的保存和使用,有consumer来控制;当consumer正常消费消息时,offset将会"线性"的向前驱动,即消息将依次顺序被消费.事实上consumer可以使用任意顺序消费消息,它只需要将offset重置为任意值..(offset将会保存在zookeeper中,参见下文)

    kafka集群几乎不需要维护任何consumer和producer状态信息,这些信息有zookeeper保存;因此producer和consumer的[客户端](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%BF%CD%BB%A7%B6%CB&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=3&seller_id=1&di=128)实现非常轻量级,它们可以随意离开,而不会对集群造成额外的影响.

    partitions的[设计](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%C9%E8%BC%C6&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=4&seller_id=1&di=128)目的有多个.最根本原因是kafka基于文件存储.通过分区,可以将日志内容分散到多个[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128)上,来避免文件尺寸达到单机磁盘的上限,每个partiton都会被当前server(kafka实例)保存;可以将一个topic切分多任意多个partitions,来消息保存/消费的效率.此外越多的partitions意味着可以容纳更多的consumer,有效提升并发消费的能力.(具体原理参见下文).

    3、Distribution

    一个Topic的多个partitions,被分布在kafka集群中的多个server上;每个server(kafka实例)负责partitions中消息的读写操作;此外kafka还可以配置partitions需要备份的个数(replicas),每个partition将会被备份到多台机器上,以提高可用性.

    基于replicated方案,那么就意味着需要对多个备份进行调度;每个partition都有一个[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128)为"leader";leader负责所有的读写操作,如果leader失效,那么将会有其他follower来接管(成为新的leader);follower只是单调的和leader跟进,同步消息即可..由此可见作为leader的server承载了全部的请求压力,因此从集群的整体考虑,有多少个partitions就意味着有多少个"leader",kafka会将"leader"均衡的分散在每个实例上,来确保整体的性能稳定.

    Producers

    Producer将消息发布到指定的Topic中,同时Producer也能决定将此消息归属于哪个partition;比如基于"round-robin"方式或者通过其他的一些算法等.

    Consumers

    本质上kafka只支持Topic.每个consumer属于一个consumer group;反过来说,每个group中可以有多个consumer.发送到Topic的消息,只会被订阅此Topic的每个group中的一个consumer消费.

    如果所有的consumer都具有相同的group,这种情况和queue模式很像;消息将会在consumers之间负载均衡.

    如果所有的consumer都具有不同的group,那这就是"发布-订阅";消息将会广播给所有的消费者.

    在kafka中,一个partition中的消息只会被group中的一个consumer消费;每个group中consumer消息消费互相独立;我们可以认为一个group是一个"订阅"者,一个Topic中的每个partions,只会被一个"订阅者"中的一个consumer消费,不过一个consumer可以消费多个partitions中的消息.kafka只能保证一个partition中的消息被某个consumer消费时,消息是顺序的.事实上,从Topic角度来说,消息仍不是有序的.

    kafka的[设计](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%C9%E8%BC%C6&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=4&seller_id=1&di=128)原理决定,对于一个topic,同一个group中不能有多于partitions个数的consumer同时消费,否则将意味着某些consumer将无法得到消息.

    Guarantees

    1) 发送到partitions中的消息将会按照它接收的顺序追加到日志中

    2) 对于消费者而言,它们消费消息的顺序和日志中消息顺序一致.

    3) 如果Topic的"replicationfactor"为N,那么允许N-1个kafka实例失效.

二、使用场景

    1、Messaging

    对于一些常规的消息系统,kafka是个不错的选择;partitons/replication和容错,可以使kafka具有良好的扩展性和性能优势.不过到目前为止,我们应该很清楚认识到,kafka并没有提供JMS中的"事务性""消息传输担保(消息确认机制)""消息分组"等企业级特性;kafka只能使用作为"常规"的消息系统,在一定程度上,尚未确保消息的发送与接收绝对可靠(比如,消息重发,消息发送丢失等)

    2、Websit activity tracking

    kafka可以作为"网站活性跟踪"的最佳工具;可以将网页/用户操作等信息发送到kafka中.并实时监控,或者离线统计分析等

    3、Log Aggregation

    kafka的特性决定它非常适合作为"日志收集中心";application可以将操作日志"批量""异步"的发送到kafka集群中,而不是保存在本地或者DB中;kafka可以批量提交消息/压缩消息等,这对producer端而言,几乎感觉不到性能的开支.此时consumer端可以使hadoop等其他系统化的存储和分析系统.

三、设计原理

    kafka的[设计](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%C9%E8%BC%C6&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=4&seller_id=1&di=128)初衷是希望作为一个统一的信息收集平台,能够实时的收集反馈信息,并需要能够支撑较大的数据量,且具备良好的容错能力.

    1、持久性

    kafka使用文件存储消息,这就直接决定kafka在性能上严重依赖文件系统的本身特性.且无论任何OS下,对文件系统本身的优化几乎没有可能.文件缓存/直接内存映射等是常用的手段.因为kafka是对日志文件进行append操作,因此磁盘检索的开支是较小的;同时为了减少磁盘写入的次数,broker会将消息暂时buffer起来,当消息的个数(或尺寸)达到一定阀值时,再flush到磁盘,这样减少了磁盘IO调用的次数.

2、性能

    需要考虑的影响性能点很多,除磁盘IO之外,我们还需要考虑网络IO,这直接关系到kafka的吞吐量问题.kafka并没有提供太多高超的技巧;对于producer端,可以将消息buffer起来,当消息的条数达到一定阀值时,批量发送给broker;对于consumer端也是一样,批量fetch多条消息.不过消息量的大小可以通过配置文件来指定.对于kafka broker端,似乎有个sendfile系统调用可以潜在的提升网络IO的性能:将文件的数据映射到系统内存中,socket直接读取相应的内存区域即可,而无需进程再次copy和交换. 其实对于producer/consumer/broker三者而言,CPU的开支应该都不大,因此启用消息压缩机制是一个良好的策略;压缩需要消耗少量的CPU资源,不过对于kafka而言,网络IO更应该需要考虑.可以将任何在网络上传输的消息都经过压缩.kafka支持gzip/snappy等多种压缩方式.

    3、生产者

    负载均衡: producer将会和Topic下所有partition leader保持socket连接;消息由producer直接通过socket发送到broker,中间不会经过任何"路由层".事实上,消息被路由到哪个partition上,有producer[客户端](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%BF%CD%BB%A7%B6%CB&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=3&seller_id=1&di=128)决定.比如可以采用"random""key-hash""轮询"等,如果一个topic中有多个partitions,那么在producer端实现"消息均衡分发"是必要的.

    其中partition leader的位置(host:port)注册在zookeeper中,producer作为zookeeper client,已经注册了watch用来监听partition leader的变更事件.

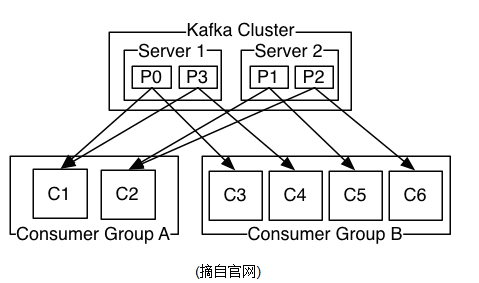
    异步发送：将多条消息暂且在客户端buffer起来，并将他们批量的发送到broker，小数据IO太多，会拖慢整体的网络延迟，批量延迟发送事实上提升了网络效率。不过这也有一定的隐患，比如说当producer失效时，那些尚未发送的消息将会丢失。

    4、消费者

    consumer端向broker发送"fetch"请求,并告知其获取消息的offset;此后consumer将会获得一定条数的消息;consumer端也可以重置offset来重新消费消息.

    在JMS实现中,Topic模型基于push方式,即broker将消息推送给consumer端.不过在kafka中,采用了pull方式,即consumer在和broker建立连接之后,主动去pull(或者说fetch)消息;这中模式有些优点,首先consumer端可以根据自己的消费能力适时的去fetch消息并处理,且可以控制消息消费的进度(offset);此外,消费者可以良好的控制消息消费的数量,batch fetch.

    其他JMS实现,消息消费的位置是有prodiver保留,以便避免重复发送消息或者将没有消费成功的消息重发等,同时还要控制消息的状态.这就要求JMS broker需要太多额外的工作.在kafka中,partition中的消息只有一个consumer在消费,且不存在消息状态的控制,也没有复杂的消息确认机制,可见kafka broker端是相当轻量级的.当消息被consumer接收之后,consumer可以在本地保存最后消息的offset,并间歇性的向zookeeper注册offset.由此可见,consumer[客户端](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%BF%CD%BB%A7%B6%CB&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=3&seller_id=1&di=128)也很轻量级.

<ignore\_js\_op>

    5、消息传送机制

    对于JMS实现,消息传输担保非常直接:有且只有一次(exactly once).在kafka中稍有不同:

    1) at most once: 最多一次,这个和JMS中"非持久化"消息类似.发送一次,无论成败,将不会重发.

    2) at least once: 消息至少发送一次,如果消息未能接受成功,可能会重发,直到接收成功.

    3) exactly once: 消息只会发送一次.

    at most once: 消费者fetch消息,然后保存offset,然后处理消息;当client保存offset之后,但是在消息处理过程中出现了异常,导致部分消息未能继续处理.那么此后"未处理"的消息将不能被fetch到,这就是"at most once".

    at least once: 消费者fetch消息,然后处理消息,然后保存offset.如果消息处理成功之后,但是在保存offset阶段zookeeper异常导致保存操作未能执行成功,这就导致接下来再次fetch时可能获得上次已经处理过的消息,这就是"at least once"，原因offset没有及时的提交给zookeeper，zookeeper恢复正常还是之前offset状态.

    exactly once: kafka中并没有严格的去实现(基于2阶段提交,事务),我们认为这种策略在kafka中是没有必要的.

    通常情况下"at-least-once"是我们搜选.(相比at most once而言,重复接收数据总比丢失数据要好).

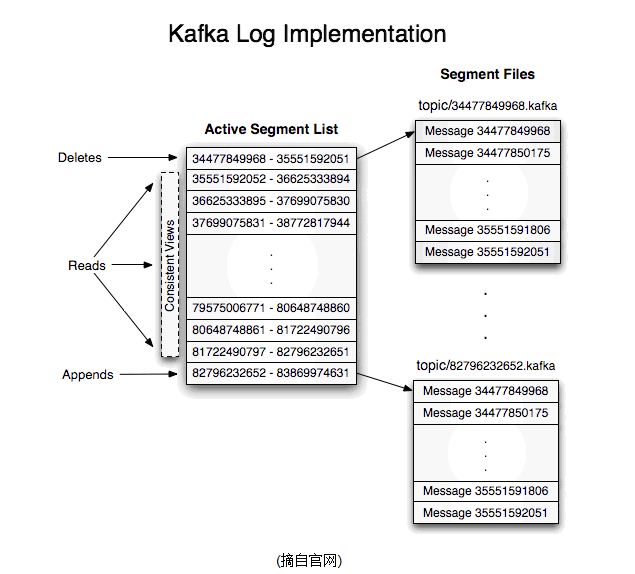
    6、复制备份

    kafka将每个partition数据复制到多个server上,任何一个partition有一个leader和多个follower(可以没有);备份的个数可以通过broker配置文件来设定.leader处理所有的read-write请求,follower需要和leader保持同步.Follower和consumer一样,消费消息并保存在本地日志中;leader负责跟踪所有的follower状态,如果follower"落后"太多或者失效,leader将会把它从replicas同步列表中删除.当所有的follower都将一条消息保存成功,此消息才被认为是"committed",那么此时consumer才能消费它.即使只有一个replicas实例存活,仍然可以保证消息的正常发送和接收,只要zookeeper集群存活即可.(不同于其他分布式存储,比如hbase需要"多数派"存活才行)

    当leader失效时,需在followers中选取出新的leader,可能此时follower落后于leader,因此需要选择一个"up-to-date"的follower.选择follower时需要兼顾一个问题,就是新leader[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128)上所已经承载的partition leader的个数,如果一个server上有过多的partition leader,意味着此server将承受着更多的IO压力.在选举新leader,需要考虑到"负载均衡".

    7.日志

    如果一个topic的名称为"my\_topic",它有2个partitions,那么日志将会保存在my\_topic\_0和my\_topic\_1两个目录中;日志文件中保存了一序列"log entries"(日志条目),每个log entry格式为"4个字节的数字N表示消息的长度" + "N个字节的消息内容";每个日志都有一个offset来唯一的标记一条消息,offset的值为8个字节的数字,表示此消息在此partition中所处的起始位置..每个partition在物理存储层面,有多个log file组成(称为segment).segmentfile的命名为"最小offset".kafka.例如"00000000000.kafka";其中"最小offset"表示此segment中起始消息的offset.

<ignore\_js\_op>

    其中每个partiton中所持有的segments列表信息会存储在zookeeper中.

    当segment文件尺寸达到一定阀值时(可以通过配置文件设定,默认1G),将会创建一个新的文件;当buffer中消息的条数达到阀值时将会触发日志信息flush到日志文件中,同时如果"距离最近一次flush的时间差"达到阀值时,也会触发flush到日志文件.如果broker失效,极有可能会丢失那些尚未flush到文件的消息.因为[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128)意外实现,仍然会导致log文件格式的破坏(文件尾部),那么就要求当server启东是需要检测最后一个segment的文件结构是否合法并进行必要的修复.

    获取消息时,需要指定offset和最大chunk尺寸,offset用来表示消息的起始位置,chunk size用来表示最大获取消息的总长度(间接的表示消息的条数).根据offset,可以找到此消息所在segment文件,然后根据segment的最小offset取差值,得到它在file中的相对位置,直接读取输出即可.

    日志文件的删除策略非常简单:启动一个后台线程定期扫描log file列表,把保存时间超过阀值的文件直接删除(根据文件的创建时间).为了避免删除文件时仍然有read操作(consumer消费),采取copy-on-write方式.

    8、分配

    kafka使用zookeeper来存储一些meta信息,并使用了zookeeper watch机制来发现meta信息的变更并作出相应的动作(比如consumer失效,触发负载均衡等)

    1) Broker node registry: 当一个kafkabroker启动后,首先会向zookeeper注册自己的节点信息(临时znode),同时当broker和zookeeper断开连接时,此znode也会被删除.

    格式: /broker/ids/[0...N]   -->host:port;其中[0..N]表示broker id,每个broker的配置文件中都需要指定一个数字类型的id(全局不可重复),znode的值为此broker的host:port信息.

    2) Broker Topic Registry: 当一个broker启动时,会向zookeeper注册自己持有的topic和partitions信息,仍然是一个临时znode.

    格式: /broker/topics/[topic]/[0...N]  其中[0..N]表示partition索引号.

    3) Consumer and Consumer group: 每个consumer[客户端](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%BF%CD%BB%A7%B6%CB&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=3&seller_id=1&di=128)被创建时,会向zookeeper注册自己的信息;此作用主要是为了"负载均衡".

    一个group中的多个consumer可以交错的消费一个topic的所有partitions;简而言之,保证此topic的所有partitions都能被此group所消费,且消费时为了性能考虑,让partition相对均衡的分散到每个consumer上.

    4) Consumer id Registry: 每个consumer都有一个唯一的ID(host:uuid,可以通过配置文件指定,也可以由系统生成),此id用来标记消费者信息.

    格式:/consumers/[group\_id]/ids/[consumer\_id]

    仍然是一个临时的znode,此节点的值为{"topic\_name":#streams...},即表示此consumer目前所消费的topic + partitions列表.

    5) Consumer offset Tracking: 用来跟踪每个consumer目前所消费的partition中最大的offset.

    格式:/consumers/[group\_id]/offsets/[topic]/[broker\_id-partition\_id]-->offset\_value

    此znode为持久节点,可以看出offset跟group\_id有关,以表明当group中一个消费者失效,其他consumer可以继续消费.

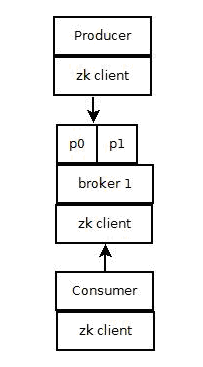
    6) Partition Owner registry: 用来标记partition被哪个consumer消费.临时znode

    格式:/consumers/[group\_id]/owners/[topic]/[broker\_id-partition\_id]-->consumer\_node\_id当consumer启动时,所触发的操作:

    A) 首先进行"Consumer id Registry";

    B) 然后在"Consumer id Registry"节点下注册一个watch用来监听当前group中其他consumer的"leave"和"join";只要此znode path下节点列表变更,都会触发此group下consumer的负载均衡.(比如一个consumer失效,那么其他consumer接管partitions).

    C) 在"Broker id registry"节点下,注册一个watch用来监听broker的存活情况;如果broker列表变更,将会触发所有的groups下的consumer重新balance.

<ignore\_js\_op>

    1) Producer端使用zookeeper用来"发现"broker列表,以及和Topic下每个partition leader建立socket连接并发送消息.

    2) Broker端使用zookeeper用来注册broker信息,已经监测partitionleader存活性.

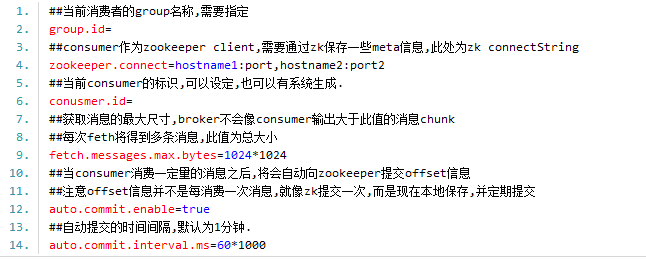
    3) Consumer端使用zookeeper用来注册consumer信息,其中包括consumer消费的partition列表等,同时也用来发现broker列表,并和partition leader建立socket连接,并获取消息.

四、主要配置

    1、Broker配置

<ignore\_js\_op>

    2.Consumer主要配置

<ignore\_js\_op>

3.Producer主要配置

<ignore\_js\_op>

以上是关于kafka一些基础说明，在其中我们知道如果要kafka正常运行，必须配置zookeeper，否则无论是kafka集群还是[客户端](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%BF%CD%BB%A7%B6%CB&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=3&seller_id=1&di=128)的生存者和消费者都无法正常的工作的，以下是对zookeeper进行一些简单的介绍：

五、zookeeper集群

    zookeeper是一个为分布式应用提供一致性服务的软件，它是开源的Hadoop项目的一个子项目，并根据google发表的一篇论文来实现的。zookeeper为分布式系统提供了高笑且易于使用的协同服务，它可以为分布式应用提供相当多的服务，诸如统一命名服务，配置管理，状态同步和组服务等。zookeeper接口简单，我们不必过多地纠结在分布式系统[编程](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%B1%E0%B3%CC&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=2&seller_id=1&di=128)难于处理的同步和一致性问题上，你可以使用zookeeper提供的现成(off-the-shelf)服务来实现来实现分布式系统额配置管理，组管理，Leader选举等功能。

    zookeeper集群的安装,准备三台服务器server1:192.168.0.1,server2:192.168.0.2,

    server3:192.168.0.3.

    1)下载zookeeper

    到<http://zookeeper.apache.org/releases.html>去下载最新版本Zookeeper-3.4.5的安装包zookeeper-3.4.5.tar.gz.将文件保存server1的~目录下

    2)安装zookeeper

    先在服务器[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128)分别执行a-c步骤

    a)解压

    tar -zxvf zookeeper-3.4.5.tar.gz

    解压完成后在目录~下会发现多出一个目录zookeeper-3.4.5,重新命令为zookeeper

    b）配置

    将conf/zoo\_sample.cfg拷贝一份命名为zoo.cfg，也放在conf目录下。然后按照如下值修改其中的配置：

    # The number of milliseconds of each tick

    tickTime=2000

    # The number of ticks that the initial

    # synchronization phase can take

    initLimit=10

    # The number of ticks that can pass between

    # sending a request and getting an acknowledgement

    syncLimit=5

    # the directory where the snapshot is stored.

    # do not use /tmp for storage, /tmp here is just

    # example sakes.

    dataDir=/home/wwb/zookeeper /data

    dataLogDir=/home/wwb/zookeeper/logs

    # the port at which the clients will connect

    clientPort=2181

    #

    # Be sure to read the maintenance section of the

    # administrator guide before turning on autopurge.

    #[http://zookeeper.apache.org/doc/ ... html#sc\_maintenance](http://zookeeper.apache.org/doc/current/zookeeperAdmin.html#sc_maintenance)

    #

    # The number of snapshots to retain in dataDir

    #autopurge.snapRetainCount=3

    # Purge task interval in hours

    # Set to "0" to disable auto purge feature

    #autopurge.purgeInterval=1

    server.1=192.168.0.1:3888:4888

    server.2=192.168.0.2:3888:4888

[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128).3=192.168.0.3:3888:4888

    tickTime：这个时间是作为 Zookeeper 服务器之间或客户端与服务器之间维持心跳的时间间隔，也就是每个 tickTime 时间就会发送一个心跳。

    dataDir：顾名思义就是 Zookeeper 保存数据的目录，默认情况下，Zookeeper 将写数据的日志文件也保存在这个目录里。

    clientPort：这个端口就是客户端连接 Zookeeper 服务器的端口，Zookeeper 会监听这个端口，接受客户端的访问请求。

    initLimit：这个配置项是用来配置 Zookeeper 接受[客户端](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%BF%CD%BB%A7%B6%CB&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=3&seller_id=1&di=128)（这里所说的客户端不是用户连接 Zookeeper 服务器的客户端，而是 Zookeeper 服务器集群中连接到 Leader 的 Follower 服务器）初始化连接时最长能忍受多少个心跳时间间隔数。当已经超过 5个心跳的时间（也就是 tickTime）长度后 Zookeeper 服务器还没有收到客户端的返回信息，那么表明这个客户端连接失败。总的时间长度就是 5\*2000=10 秒

    syncLimit：这个配置项标识 Leader 与Follower 之间发送消息，请求和应答时间长度，最长不能超过多少个 tickTime 的时间长度，总的时间长度就是2\*2000=4 秒

    server.A=B：C：D：其中 A 是一个数字，表示这个是第几号服务器；B 是这个服务器的 ip 地址；C 表示的是这个服务器与集群中的 Leader 服务器交换信息的端口；D 表示的是万一集群中的 Leader 服务器挂了，需要一个端口来重新进行选举，选出一个新的 Leader，而这个端口就是用来执行选举时服务器相互通信的端口。如果是伪集群的配置方式，由于 B 都是一样，所以不同的 Zookeeper 实例通信端口号不能一样，所以要给它们分配不同的端口号

注意:dataDir,dataLogDir中的wwb是当前登录用户名，data，logs目录开始是不存在，需要使用mkdir命令创建相应的目录。并且在该目录下创建文件myid,serve1,server2,server3该文件内容分别为1,2,3。

针对服务器server2,server3可以将server1复制到相应的目录，不过需要注意dataDir,dataLogDir目录,并且文件myid内容分别为2,3

    3)依次启动[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128)1，server2,server3的zookeeper.

    /home/wwb/zookeeper/bin/zkServer.sh start,出现类似以下内容

    JMX enabled by default

    Using config: /home/wwb/zookeeper/bin/../conf/zoo.cfg

    Starting zookeeper ... STARTED

   4) 测试zookeeper是否正常工作，在server1上执行以下命令

    /home/wwb/zookeeper/bin/zkCli.sh -server192.168.0.2:2181,出现类似以下内容

    JLine support is enabled

    2013-11-27 19:59:40,560 - INFO      [main-SendThread(localhost.localdomain:2181):ClientCnxn$SendThread@736]- Session   establishmentcomplete on [server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128) localhost.localdomain/127.0.0.1:2181, sessionid =    0x1429cdb49220000, negotiatedtimeout = 30000

    WATCHER::

    WatchedEvent state:SyncConnected type:None path:null

    [zk: 127.0.0.1:2181(CONNECTED) 0] [root@localhostzookeeper2]#

    即代表集群构建成功了,如果出现错误那应该是第三部时没有启动好集群，

运行，先利用

    ps aux | grep zookeeper查看是否有相应的进程的，没有话，说明集群启动出现问题，可以在每个服务器上使用

    ./home/wwb/zookeeper/bin/zkServer.sh stop。再依次使用./home/wwb/zookeeper/binzkServer.sh start，这时在执行4一般是没有问题，如果还是有问题，那么先stop再到bin的上级目录执行./bin/zkServer.shstart试试。

注意：zookeeper集群时，zookeeper要求半数以上的机器可用，zookeeper才能提供服务。

六、kafka集群

(利用上面server1,server2,server3,下面以server1为实例)

    1)下载kafka0.8(<http://kafka.apache.org/downloads.html>),保存到服务器/home/wwb目录下kafka-0.8.0-beta1-src.tgz(kafka\_2.8.0-0.8.0-beta1.tgz)

    2)解压 tar -zxvf kafka-0.8.0-beta1-src.tgz,产生文件夹kafka-0.8.0-beta1-src更改为kafka01

3)配置

    修改kafka01/config/[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128).properties,其中broker.id,log.dirs,zookeeper.connect必须根据实际情况进行修改，其他项根据需要自行斟酌。大致如下：

     broker.id=1

     port=9091

     num.network.threads=2

     num.io.threads=2

     socket.send.buffer.bytes=1048576

    socket.receive.buffer.bytes=1048576

     socket.request.max.bytes=104857600

    log.dir=./logs

    num.partitions=2

    log.flush.interval.messages=10000

    log.flush.interval.ms=1000

    log.retention.hours=168

    #log.retention.bytes=1073741824

    log.segment.bytes=536870912

    num.replica.fetchers=2

    log.cleanup.interval.mins=10

    zookeeper.connect=192.168.0.1:2181,192.168.0.2:2182,192.168.0.3:2183

    zookeeper.connection.timeout.ms=1000000

    kafka.metrics.polling.interval.secs=5

    kafka.metrics.reporters=kafka.metrics.KafkaCSVMetricsReporter

    kafka.csv.metrics.dir=/tmp/kafka\_metrics

    kafka.csv.metrics.reporter.enabled=false

4）初始化因为kafka用scala语言编写，因此运行kafka需要首先准备scala相关环境。

    > cd kafka01

    > ./sbt update

    > ./sbt package

    > ./sbt assembly-package-dependency

在第二个命令时可能需要一定时间，由于要下载更新一些依赖包。所以请大家 耐心点。

5) 启动kafka01

    >JMX\_PORT=9997 bin/kafka-[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128)-start.sh config/server.properties &

a)kafka02操作步骤与kafka01雷同，不同的地方如下

    修改kafka02/config/server.properties

    broker.id=2

    port=9092

    ##其他配置和kafka-0保持一致

    启动kafka02

    JMX\_PORT=9998 bin/kafka-server-start.shconfig/server.properties &

b)kafka03操作步骤与kafka01雷同，不同的地方如下

    修改kafka03/config/server.properties

    broker.id=3

    port=9093

    ##其他配置和kafka-0保持一致

    启动kafka02

    JMX\_PORT=9999 bin/kafka-[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128)-start.shconfig/server.properties &

6)创建Topic(包含一个分区，三个副本)

    >bin/kafka-create-topic.sh--zookeeper 192.168.0.1:2181 --replica 3 --partition 1 --topicmy-replicated-topic

7)查看topic情况

    >bin/kafka-list-top.sh --zookeeper 192.168.0.1:2181

    topic: my-replicated-topic  partition: 0 leader: 1  replicas: 1,2,0  isr: 1,2,0

8)创建发送者

   >bin/kafka-console-producer.sh--broker-list 192.168.0.1:9091 --topic my-replicated-topic

    my test message1

    my test message2

    ^C

9)创建消费者

    >bin/kafka-console-consumer.sh --zookeeper127.0.0.1:2181 --from-beginning --topic my-replicated-topic

    ...

    my test message1

    my test message2

^C

10)杀掉server1上的broker

  >pkill -9 -f config/[server](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=server&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=5&seller_id=1&di=128).properties

11)查看topic

  >bin/kafka-list-top.sh --zookeeper192.168.0.1:2181

  topic: my-replicated-topic  partition: 0 leader: 1  replicas: 1,2,0  isr: 1,2,0

发现topic还正常的存在

11）创建消费者，看是否能查询到消息

    >bin/kafka-console-consumer.sh --zookeeper192.168.0.1:2181 --from-beginning --topic my-replicated-topic

    ...

    my test message 1

    my test message 2

    ^C

说明一切都是正常的。

OK,以上就是对Kafka个人的理解，不对之处请大家及时指出。

补充说明：

1、public Map<String, List<KafkaStream<byte[], byte[]>>> createMessageStreams(Map<String, Integer> topicCountMap)，其中该方法的参数Map的key为topic名称，value为topic对应的分区数，譬如说如果在kafka中不存在相应的topic时，则会创建一个topic，分区数为value，如果存在的话，该处的value则不起什么作用

2、关于生产者向指定的分区发送数据，通过设置partitioner.class的属性来指定向那个分区发送数据，如果自己指定必须编写相应的程序，默认是kafka.producer.DefaultPartitioner,分区程序是基于散列的键。

3、在多个消费者读取同一个topic的数据，为了保证每个消费者读取数据的唯一性，必须将这些消费者group\_id定义为同一个值，这样就构建了一个类似队列的数据结构，如果定义不同，则类似一种广播结构的。

4、在consumerapi中，参数[设计](http://cpro.baidu.com/cpro/ui/uijs.php?rs=1&u=http%3A%2F%2Fwww%2Eaboutyun%2Ecom%2Fthread%2D9341%2D1%2D1%2Ehtml&p=baidu&c=news&n=10&t=tpclicked3_hc&q=92051019_cpr&k=%C9%E8%BC%C6&k0=java&kdi0=8&k1=%B1%E0%B3%CC&kdi1=8&k2=%BF%CD%BB%A7%B6%CB&kdi2=8&k3=%C9%E8%BC%C6&kdi3=8&k4=server&kdi4=1&sid=4ebca4a25f27e407&ch=0&tu=u1692056&jk=fb2f0911808fa875&cf=29&fv=14&stid=9&urlid=0&luki=4&seller_id=1&di=128)到数字部分，类似Map<String,Integer>,

numStream,指的都是在topic不存在的时，会创建一个topic，并且分区个数为Integer,numStream,注意如果数字大于broker的配置中num.partitions属性，会以num.partitions为依据创建分区个数的。

5、producerapi，调用send时，如果不存在topic，也会创建topic，在该方法中没有提供分区个数的参数，在这里分区个数是由服务端broker的配置中num.partitions属性决定的

关于kafka说明可以参考：<http://kafka.apache.org/documentation.html>

文章转自：http://www.aboutyun.com/thread-9341-1-1.html

标签: [分布式系统架构](http://www.cnblogs.com/likehua/tag/%E5%88%86%E5%B8%83%E5%BC%8F%E7%B3%BB%E7%BB%9F%E6%9E%B6%E6%9E%84/), [zookeeper](http://www.cnblogs.com/likehua/tag/zookeeper/)