**zookeeper 相关介绍**

* zookeeper 是什么: ZooKeeper[?](http://twiki.dangdang.com/twiki/bin/edit.pl/Main/ZooKeeper?topicparent=Main.Realtime_redis_helper" \o "Create this topic)是一个分布式的，开放源码的分布式应用程序协调服务，ZooKeeper是以Fast Paxos算法为基础，实现同步服务，配置维护和命名服务等分布式应用。
* ZooKeeper[?](http://twiki.dangdang.com/twiki/bin/edit.pl/Main/ZooKeeper?topicparent=Main.Realtime_redis_helper" \o "Create this topic) 如何工作？
  + 1: 在编程上，ZooKeeper设计很简单，所使用的数据模型风格很像【文件系统的目录树结构】，简单来说，有点类似windows中注册表的结构，有名 称，有树节点，有【Key(键)/Value(值)对】的关系，可以看做一个树形结构的数据库，分布在不同的机器上做名称管理。
  + 2: Zookeeper分为2个部分：【服务器端和客户端】，客户端只连接到整个ZooKeeper服务的某个服务器上。客户端使用并维护一个【TCP连 接】，通过这个连接发送请求、接受响应、获取观察的事件以及发送心跳。如果这个TCP连接中断，客户端将尝试连接到另外的ZooKeeper服务器。客户 端第一次连接到ZooKeeper服务时，接受这个连接的 ZooKeeper[?](http://twiki.dangdang.com/twiki/bin/edit.pl/Main/ZooKeeper?topicparent=Main.Realtime_redis_helper" \o "Create this topic)服务器会为这个客户端建立一个会话。当这个客户端连接到另外的服务器时，这个会话会被新的服务器重新建立。
  + 3: 【启动】Zookeeper服务器集群环境后，多个Zookeeper服务器在工作前会【选举出一个Leader】，在接下来的工作中这个被选举出来的 【Leader死了】，而剩下的Zookeeper服务器会【知道这个Leader死掉了】，在【活着的Zookeeper集群】中会【继续选出一个 Leader】，选举出leader的目的是为了可以在分布式的环境中保证数据的一致性。如图：
  + 通过命令： echo stat | nc 172.16.252.201 2181 怎么知道现在 谁 是leader， 谁是follower
* 详细 入门 参见：
  + 1: <http://www.javabloger.com/article/apache-zookeeper-hadoop.html>
  + 2: <http://www.javabloger.com/article/apache-zookeeper-hadoop.html>
  + 3: <http://www.javabloger.com/article/apache-zookeeper-hbase-hadoop-3.html>

**kafka**

* 基本概念：
  + 1：Kafka是一个分布式的消息发布-订阅系统
  + 2： messages 是通信的基本单位。messages被一个producer发布给一个topic，也就是说它们被物理地传递给作为一个broker(kafka)的服务器
  + 3： 一个consumer订阅一个topic，每一个被发布的消息就被转交给这些consumer了。
  + 4： 每个consumer 属于一个group
  + 5： 一个group的consumers 【分摊】一个topic 的所有messages; 即 一个group中的两个consumer 【竞争】一个topic中的一个message
  + 6: 两个不同group的consumer 可以【共享】一个topic中的一个message
* 深入了解 Kafka存储策略
  + 1: 【kafka 架构图】
  + 2： kafka以topic来进行消息管理，每个topic包含多个part（ition），每个part对应一个逻辑log，有多个segment组成。
  + 3: 每个segment中存储多条消息（见下图），消息id由其逻辑位置决定，即从消息id可直接定位到消息的存储位置，避免id到位置的额外映射。
  + 4: 每个part在内存中对应一个index，记录每个segment中的第一条消息偏移。
  + 5: 发布者发到某个topic的消息会被均匀的分布到多个part上（随机或根据用户指定的回调函数进行分布），broker收到发布消息往对应part的最 后一个segment上添加该消息，当某个segment上的消息条数达到配置值或消息发布时间超过阈值时，segment上的消息会被flush到磁 盘，只有flush到磁盘上的消息订阅者才能订阅到，segment达到一定的大小后将不会再往该segment写数据，broker会创建新的 segment。

**storm程序介绍**

* Storm计算模型以Topology为单位， 一个Topology是由一系列Spout和Bolt构成的图
  + spout 从数据源 取数据(比如： 这里是从kafka取数据)， 然后将数据 分发给bolt
  + bolt 接受spout 发来的数据， 经过相应的处理， 存到一个地方(比如: redis)

**遇到的问题， 及解决方案**

* 1： 安装 kafka的python 驱动包 zookpeer， 参见： <http://justfansty.i.sohu.com/blog/view/217928676.htm>
* 2: 安装python版， zookeeper客户端程序: samsa 参见： <https://github.com/disqus/samsa>
* 3： 添加本地jar 包： 如下命令： mvn install:install-file -Dfile=lib/jedis-2.0.0.ddbeta.jar -DgroupId=redis.clients -DartifactId=jedis -Dversion=2.0.0 -Dpackaging=jar
* 4： zookeeper 怎么知道kafka 的， 怎么管理kafka的
  + 在文件 /d1/kafka\_data/kafka-0.7.2-incubating-src/config/server.properties【信息已经配 置好】启动kafka服务器， 如下： 配置zookeeper 机器$zk.connect=h252201:2181,h252202:2181,h252203:2181
  + kafka启动时 就会在相应的zookeeper 去注册信息， 告诉zookeeper 我在这， 你来管理我吧...
* Kafka在zookeeper上保存的信息 <http://blog.csdn.net/kevin_hx001/article/details/7579219>
* storm topology的一些 参数说明http://www.cnblogs.com/cnchenlx/archive/2012/03/07/2383368.html
* kakfa 中关键技术点： sendfile 原理
  + 一般流程： 当需要对一个文件进行传输的时候，其具体流程细节如下：
    - 1、调用read函数，文件数据被copy到内核缓冲区
    - 2、read函数返回，文件数据从内核缓冲区copy到用户缓冲区
    - 3、write函数调用，将文件数据从用户缓冲区copy到内核与socket相关的缓冲区。
    - 4、数据从socket缓冲区copy到相关协议引擎。
  + sendfile 运行流程如下：
    - 1、sendfile系统调用，文件数据被copy至内核缓冲区
    - 2、再从内核缓冲区copy至内核中socket相关的缓冲区
    - 3、最后再socket相关的缓冲区copy到协议引擎