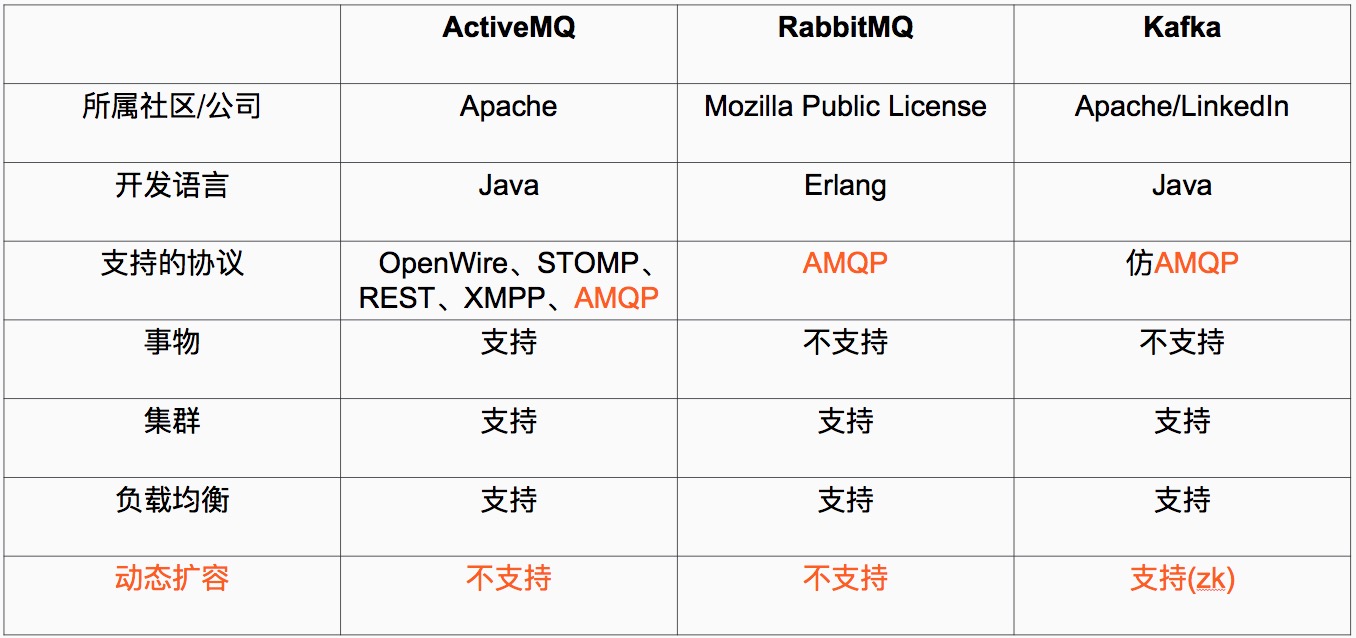
# 一，当前主流消息中间件产品各项主要指标对比

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
|  | ActiveMQ | RabbitMQ | RocketMQ | **Kafka** | ZeroMQ |
| 吞吐量 | 比RabbitMQ低 | 2.6w/s（消息做持久化） | 11.6w/s | **17.3w/s** | 29w/s |
| 开发语言 | Java | Erlang | Java | **Scala/Java** | C |
| 主要维护者 | Apache | Mozilla/Spring | Alibaba | **Apache** | iMatix，创始人已去世 |
| 成熟度 | 成熟 | 成熟 | 开源版本不够成熟 | **比较成熟** | 只有C、PHP等版本成熟 |
| 文档和注释 | 多 | 多 | 很少 | **较少** | 很少 |
| 订阅形式 | 点对点(p2p)、广播（发布-订阅） | 提供了4种：direct, topic ,Headers和fanout。fanout就是广播模式 | 基于topic/messageTag以及按照消息类型、属性进行正则匹配的发布订阅模式 | **基于topic以及按照topic进行正则匹配的发布订阅模式** | 点对点(p2p) |
| 持久化 | 支持少量堆积 | 支持少量堆积 | 支持大量堆积 | **支持大量堆积** | 不支持 |
| 顺序消息 | 不支持 | 不支持 | 支持 | **支持** | 不支持 |
| 消息回溯 | 不支持 | 不支持 | 支持指定时间点的回溯 | **支持指定分区offset位置的回溯** | 不支持 |
| 负载均衡 | 可以支持 | 可以支持 | 支持较好 | **支持很好** | 不支持 |
| 集群方式 | 支持简单集群模式，比如'主-备'，对高级集群模式支持不好。 | 支持简单集群，'复制'模式，对高级集群模式支持不好。 | 常用 多对'Master-Slave' 模式，开源版本需手动切换Slave变成Master | **天然的‘Leader-Slave’无状态集群，每台服务器既是Master也是Slave** | 不支持 |
| 管理界面 | 一般 | 较好 | 一般 | **无** | 无 |

对比二【ActiveMQ,RabbitMQ,Kafka】

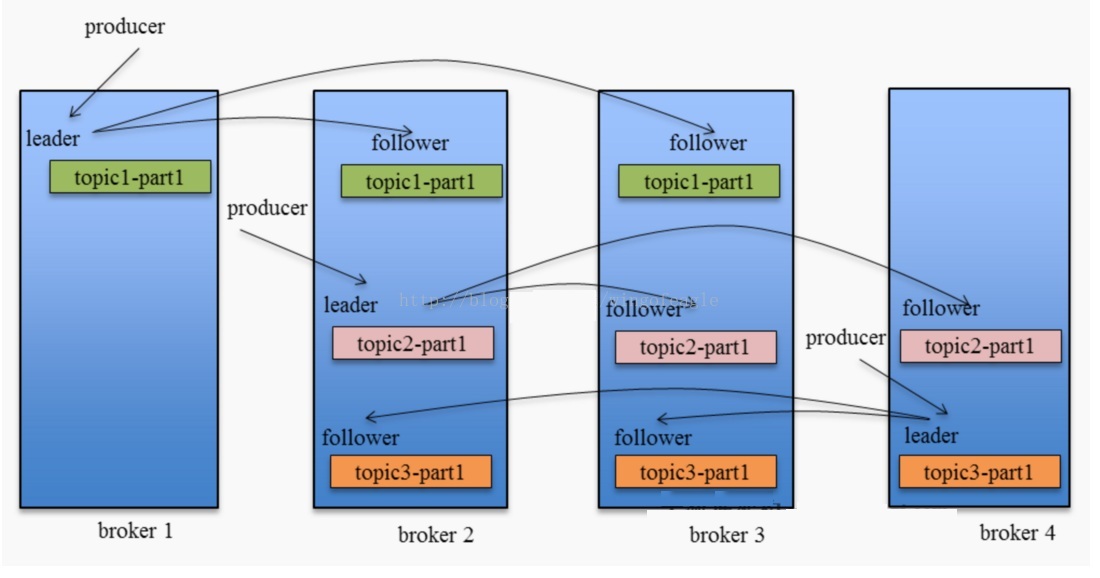


备注：AMQP（ Advanced Message Queuing Protocol 高级消息队列协议）。Zookeeper是一种在分布式系统中被广泛用来作为：分布式状态管理、分布式协调管理、分布式配置管理、和分布式锁服务的集群。kafka增加和减少服务器都会在Zookeeper节点上触发相应的事件kafka系统会捕获这些事件，进行新一轮的负载均衡，客户端也会捕获这些事件来进行新一轮的处理。

# 龙控采用的kafka集群方案

采用Zookeeper+kafka进行集群配置，支持动态扩容。其中集群物理部署遵循 2\*n+1 的原则进行搭建（原因是Zookeeper集群的工作是超过半数存活才能对外提供服务。3台中，允许1台挂掉 ，5台中，允许2台挂掉）。备注：zookeeper是用java写的所以他的需要JAVA环境【需要安装JDK,JRE】。为了达到高可用的目的，Zookeeper 自身也不能是单点，可以和kafka同机数量配置。

集群部署架构图

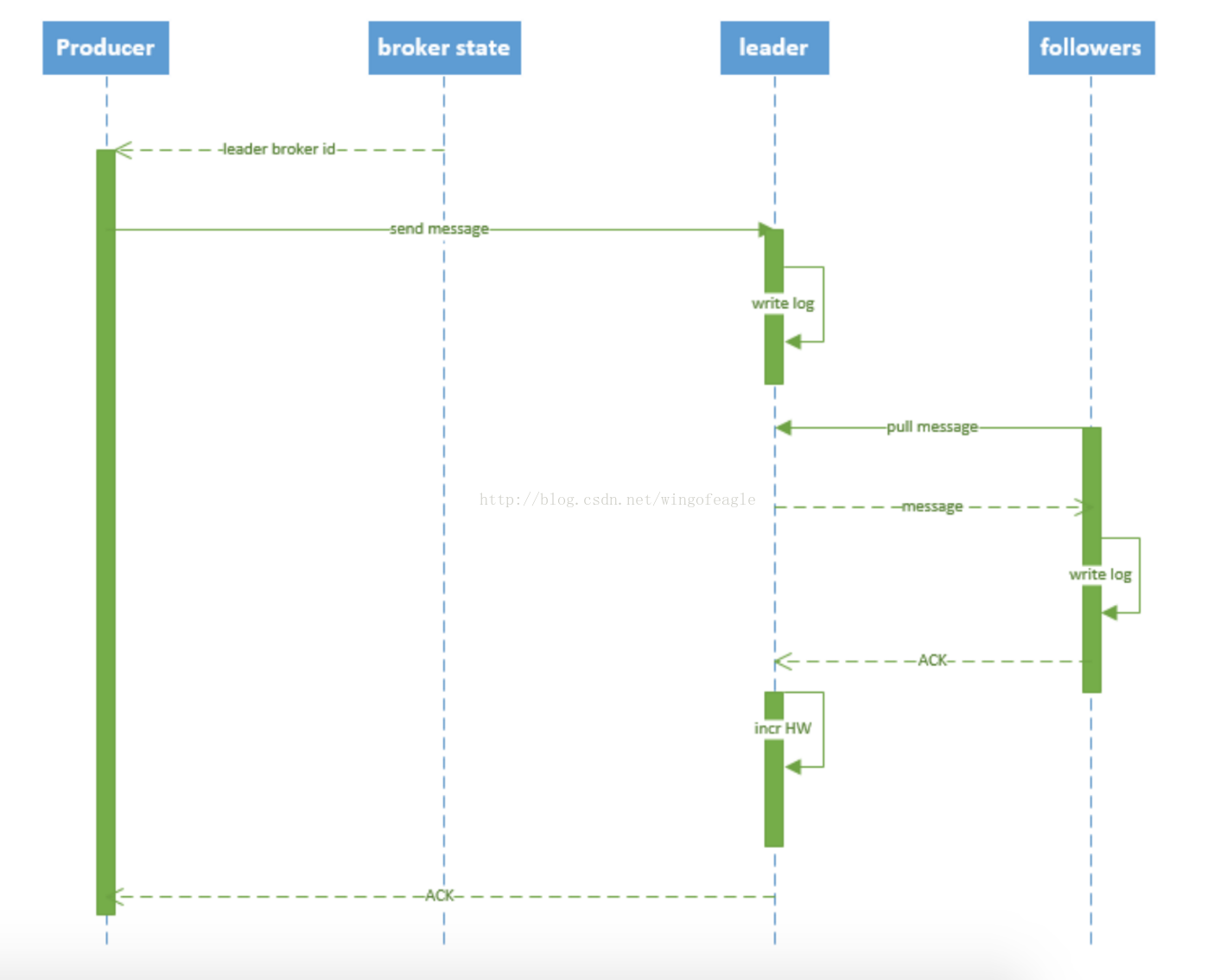


# 如何解决消息的顺序问题

Producer需采用同步发送数据，这样设置后对性能会有一定影响，但是能接受。

备注kafka的顺序做法【顺序写入：kafka producer同步发送数据，按orderid hash来区分队列存放数据。假设一个topic，有999个orderid分配到8个分区；顺序消费：kafka consumer都订阅所有topics，但是consumer线程受制于partition数量的限制，假设为8，那么最多8个线程并行消费数据，这8个线程并行的消费8个partition里面的数据】

Producer



# Kafka的一些基本概念

1、主题（Topic）：一个主题类似新闻中的体育、娱乐、教育等分类概念，在实际工程中通常一个业务一个主题。

2、分区（Partition）：一个Topic中的消息数据按照多个分区组织，分区是kafka消息队列组织的最小单位，一个分区可以看作是一个FIFO（ First Input First Output的缩写，先入先出队列）的队列。

kafka分区是提高kafka性能的关键所在，当你发现你的集群性能不高时，常用手段就是增加Topic的分区，分区里面的消息是按照从新到老的顺序进行组织，消费者从队列头订阅消息，生产者从队列尾添加消息。

# 常用的kafka操作命令

1. 查询liunx操作系统版本信息

输入"uname -a ",可显示电脑以及操作系统的相关信息；

输入"cat /proc/version",说明正在运行的内核版本；

输入"cat /etc/issue", 显示的是发行版本信息；

lsb\_release -a (适用于所有的linux，包括Redhat、SuSE、Debian等发行版，但是在debian下要安装lsb)

备注：如果不知道命令的意思.可以通过 "man 命令"可以查看它的使用方式.及详细信息.

2，启动kafka

bin/kafka-server-start.sh -daemon config/server.properties

3，创建topic

bin/kafka-topics.sh --create --zookeeper 192.168.6.104:2181,192.168.6.105:2181,192.168.6.106:2181 --replication-factor 2 --partitions 2 --topic loncom-test

4，查看topic

bin/kafka-topics.sh --describe --zookeeper 192.168.6.104:2181,192.168.6.105:2181,192.168.6.106:2181 --topic loncom-test

列出topic：bin/kafka-topics.sh --list --zookeeper 192.168.6.104:2181,192.168.6.105:2181,192.168.6.106:2181

5，防火墙相关命令

# 关闭防火墙

service iptables stop

# 启动防火墙

service iptables start

# 重启防火墙

service iptables restart

# 查看防火墙状态

service iptables status

# 永久关闭防火墙

chkconfig iptables off

# 永久关闭后启用

chkconfig iptables on

6，创建producer

bin/kafka-console-producer.sh --broker-list 192.168.6.104:9092 -topic loncom-test

7，创建consumer

bin/kafka-console-consumer.sh --zookeeper 192.168.6.104:2181,192.168.6.105:2181,192.168.6.106:2181 -topic loncom-test --from-beginning

8，删除topic和关闭服务

bin/kafka-topics.sh --delete --zookeeper 192.168.6.104:2181,192.168.6.105:2181,192.168.6.106:2181 --topic loncom-test

9，关闭kafka和zookeeper

先关闭kafka，然后再关闭zookeeper

10，增加topic分区数

bin/kafka-topics.sh --zookeeper zk\_host:port --alter --topic my\_topic\_name--partitions 401

11，增加topic配置

bin/kafka-topics.sh --zookeeper zk\_host:port --alter --topic my\_topic\_name --config flush.messages=11

12，删除topic配置

bin/kafka-topics.sh --zookeeper zk\_host:port --alter --topic my\_topic\_name --delete-config flush.messages=11