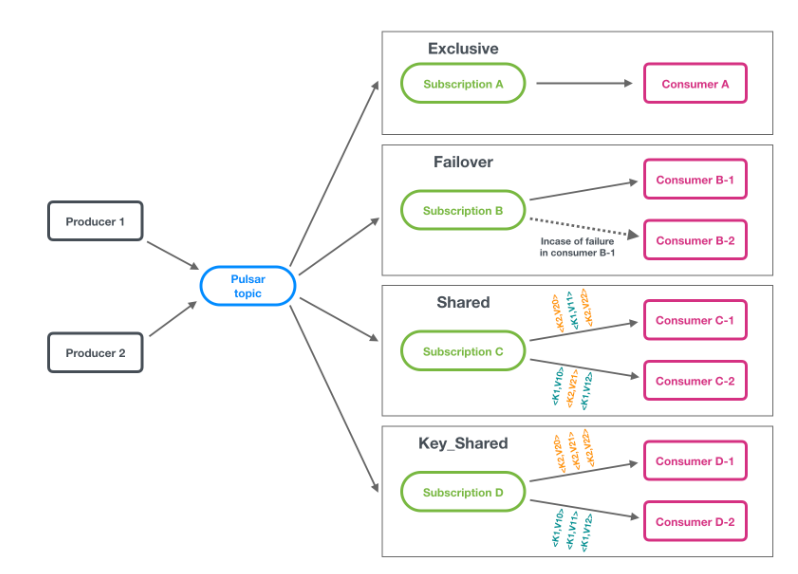
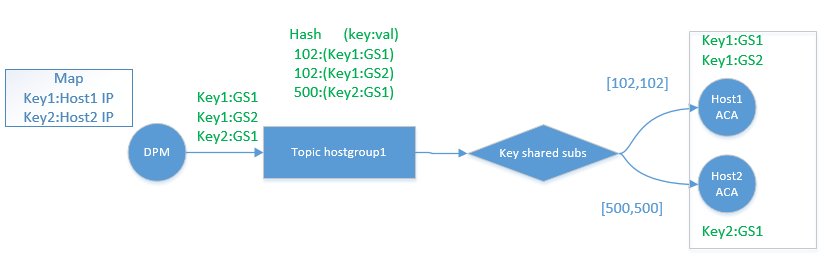
# Host Group topic



## Keyshared模式



public class PersistentStickyKeyDispatcherMultipleConsumers

public class HashRangeExclusiveStickyKeyConsumerSelector

Consumer select(int hash) {  
 if (rangeMap.size() > 0) {  
 int slot = hash % rangeSize;  
 Map.Entry<Integer, Consumer> ceilingEntry = rangeMap.ceilingEntry(slot);  
 Map.Entry<Integer, Consumer> floorEntry = rangeMap.floorEntry(slot);  
 Consumer ceilingConsumer = ceilingEntry != null ? ceilingEntry.getValue() : null;  
 Consumer floorConsumer = floorEntry != null ? floorEntry.getValue() : null;  
 if (floorConsumer != null && floorConsumer.equals(ceilingConsumer)) {  
 return ceilingConsumer;  
 } else {  
 return null; #空指针异常的原因  
 }  
 } else {  
 return null;  
 }  
}

修改源码select函数，使当key对应的consumer为空时，返回邻居consumer。从而在某个consumer宕机时，程序正常运行。

* + 1. Keyshared模式工作流程方案一（Host IP+字符为msg key，可适用于重新分组）：

1）对于每一个host group, map记录host group内各host ip选择的key，用一个set记录已经使用key的hash值；有新host加入host group时，查询set，计算该host IP与本group内已使用的key是否hash冲突，**冲突则使用IP+a为key，还是冲突则使用IP+aa，以此类推。**

2）Dpm发送GS时，查询目的host IP对应的host key附在消息上，发送给host group对应的topic中。

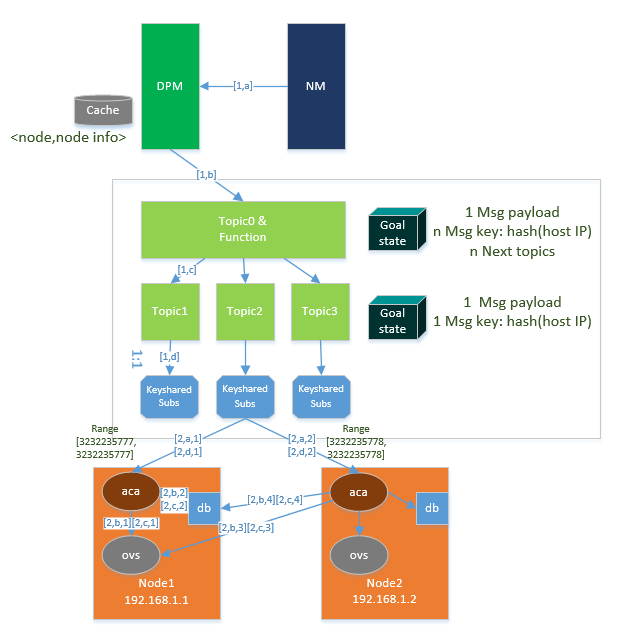
3）各host订阅本host group的topic，订阅模式为key shared，hash range为[hash（host key），hash（host key）]。各host ACA收消息时根据+的数量将host key转化为IP；host收到GS后查询GS的msg key，若与本host key相同，则处理；不相同则表明msg key对应的host ACA宕机（记作host2），根据IP代替其配置OVS。

* + 1. Keyshared模式工作流程方案二（Host IP为msg key，冲突host不订阅）：

1）对于每一个host group, 有一个set记录host group内各host ip；有新host加入host group时，查询set，计算该host IP与本group内已有host的IP是否存在hash冲突，**冲突的话则本host不订阅该topic。由冲突的host转发或代替本host配置**

2）Dpm发送GS时，将host IP作为host key附在消息上，发送给host group对应的topic中。

3）各host订阅本host group的topic，订阅模式为key shared，hash range为[hash（host IP），hash（host IP）]。各host ACA收消息时根据的msg key，若与本host IP相同，则处理；不相同则表明msg key对应的host ACA宕机或需要本host转发（记作host2），根据IP代替其配置OVS。

* + 1. Keyshared模式工作流程方案三（更换hash函数）：
    2. 

Dpm：

[1,a] Node manager为每个host分配一个topic，并存储在node manager本地node info中，同时缓存在dpm中。

[1,b] Dpm发送配置时，从目的host的node info中提取host IP以及next topic等信息，将gs（msg payload，msg key=host IP，next topic）发到一个pulsar MQ的单播或组播topic（topic0）。

[1,c] MQ通过pulsar function将msg发送到对应的next topic。

ACA：

[1,d] 每个aca根据node manager中node info查找对应topic，并订阅该topic唯一的subscription，该subscription的类型为key shared。

[2,a,1] Aca收到gs后，提取key值，比较key与本host的IP是否相同，相同则根据gs类型进行相应处理，包括[2,b,1]下发流表；[2,b,2]更新 aca db等。

[2,c,1][2,c,2]当处理成功上述步骤后，[2,d,1] aca将ack该gs；

Rescue path：

方案一：

[2,a,2]若key与本host的IP不同，区分gs的类型（ovs配置、ondemand配置）;

[2,b,3] 若为ovs配置，则根据key中的host IP直接对ovs进行代理配置;

[2,b,4] 若为ondemand配置则提取key中IP值，并将配置信息写入相应aca的db内

[2,c,1][2,c,2] 当上述配置成功后，[2,d,2] aca将ack该gs。

方案二：

[2,a,2]若key与本host的IP不同，aca不ack该消息，待该消息time out后broker会重新发送该gs，

[2,d,2] 当相应aca重启后gs才能被正确配置； 此时aca会ack该gs。

Pulsar MQ：

1）hash冲突概率: Pulsar的keyshared模式下使用key的hash值低16位进行msg调度，如果每个topic覆盖100个host，100个host都会映射到一个16位的hash值，假设使用的hash函数可以将一个输入随机分配到65535个结果中去，则冲突的概率为。

2）由于pulsar使用的hash函数会存在hash冲突的可能，因此为了提供一种无状态的防止hash冲突的topic分配方案，我们可以修改源码，替换keyshared模式使用的hash函数，保证node IP映射的hash值没有冲突，我们使用 32位IP地址对应的int值作为hash函数输出。这样就可以保证每个host使用host IP为key时没有冲突，这种方式在分组改变的时候也不需要有任何调整，且逻辑简单。pulsar使用的两种备选hash函数及我们自定义的hash函数都能输出32位的结果。但是代码中只使用了hash值低16位用于区分不同key，因此范围只有0-65535。这在大规模场景及以VPC为topic的场景下可能会出现冲突。我们可以修改源码，直接使用全部32位hash结果。这36亿的hash值范围完全可以覆盖100万host的规模。

public interface StickyKeyConsumerSelector {  
 int DEFAULT\_RANGE\_SIZE = 2 << 15;

3）IPv4可以转换得到IPv6地址，如果全网的IPv6地址都是通过如下方法计算，那么可以用IPv6地址低32位作为hash函数的输入，就能很方便的将该方法应用到IPv6上。

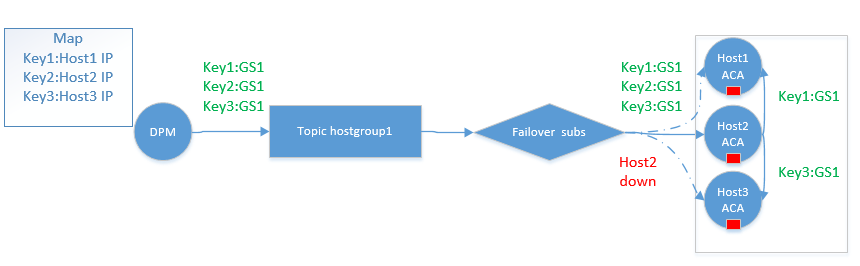


1.1.3 多集群管理

1）类似现网，将host node分为一个个cell，每个cell使用一个MQ集群。Node manager为每个新加入集群的host分配一个topic，并存入node info中。

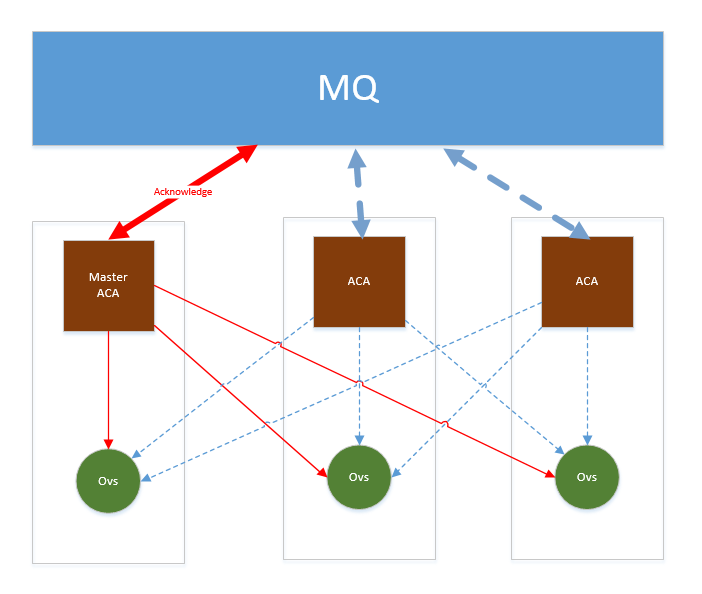
2）dpm记录每个cell对应的Pulsar MQ集群，发送gs时根据目的node发往相应集群的单播或组播topic。

## 1.2 Failover模式

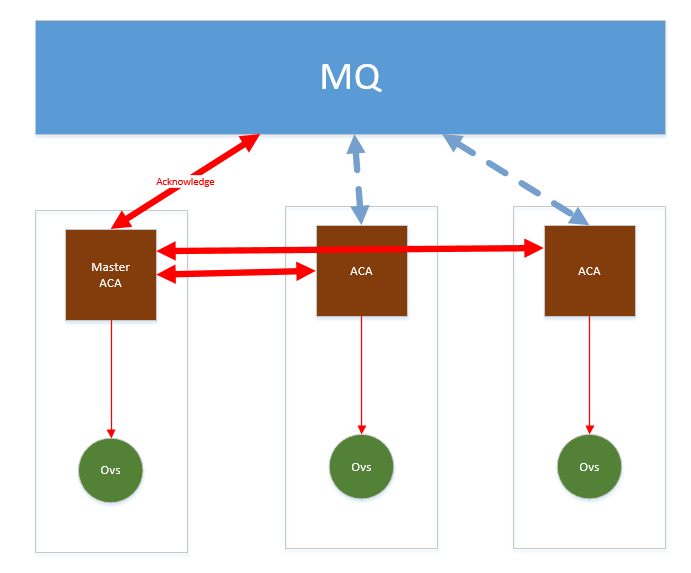


1.2.1 工作流程：

1. Dpm发送GS时，将目的host IP作为msg key附在消息上，发送给host group对应的topic中。
2. 各host订阅本host group的topic，订阅模式为Failover。Group内只有master aca能收到topic内的所有GS。Master aca收到GS后查询GS的msg key，若为本host IP，则处理；不相同则表明msg key对应的host ACA宕机（记作host2），则根据msg key对应的IP地址，master host的aca为host2的OVS下发表项。
3. 方案（1）：ACA配置完属于本host ovs的GS后，直接acknowledge该GS；对于host2的GS，ACA配置完host2 ovs后，也需要acknowledge。收到确认，才能acknowledge该GS。没有acknowledge的GS会在一段时间（time out）后重新发送。该方案下host group内所有host上的ovs需要将本group内所有aca加入controller列表，还要检测到group内aca的重启，动态添加新aca。方案（1）中master ACA负责host group内所有OVS的配置下发，可能会成为瓶颈。



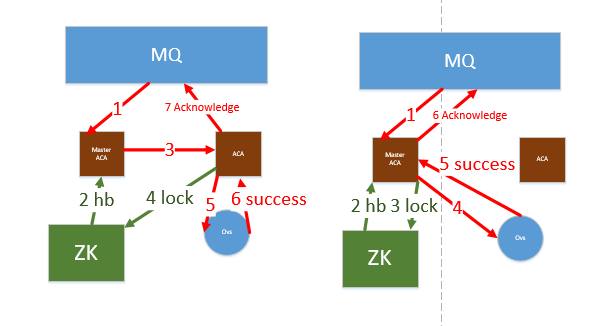
1. 方案（2）：ACA配置完属于本host ovs的GS后，直接acknowledge该GS；对于host2的GS，ACA将GS转发给host2的aca，host2的aca配置完成后需要确认，此时master host aca才能acknowledge该GS。这是一种链式确认，没有acknowledge的GS会在一段时间（time out）后重新发送。该方案下所有host只需要将本host下的aca加入controller列表。方案（2）无法处理aca失效的场景。



1. 方案（3）结合了方案（1）与方案（2），ACA配置完属于本host ovs的GS后，直接acknowledge该GS；对于host2的GS，ACA将GS转发给host2的aca，host2的aca配置完成后需要确认，此时master host aca才能acknowledge该GS，每个aca需要与本group内所有aca保活，如果其他aca失效，本aca代替失效的aca配置其ovs。该方案下host group内所有host上的ovs需要将本group内所有aca加入controller列表，还要检测到group内aca的重启，动态添加新aca。

多写问题的解决

1. aca配置消息保证幂等
2. 利用zookeeper分配分布式锁，从而避免重写，20K tps；保证ovs配置成功后master aca才acknowledge该msg，从而防止漏写。
3. 通过发msg前ping或者利用zookeeper来做host aca的心跳检查，如可以通过创建临时节点来建立心跳检测机制。如果分布式系统的某个服务节点宕机了，则其持有的会话会超时，此时该临时节点会被删除，相应的监听事件就会被触发。



ETCD、ZK集群过多， on demand rule如何解决

某个ACA成为master时，从pulsar中拉取本group的所有ondemand配置信息，并将自己VIP设置为本group的控制器IP

每个OVS使用两个控制器，一个是本host上aca，另一个是本group的master（vip）

Master看到zookeeper上某个aca失效后，才会回复该host的请求

健康检查

使用周期性检测心跳机制：server每隔s秒向各个node发送检测请求，设定一个超时时间，如果超过超时时间，则进入死亡列表。

累计失效检测机制：在1 的基础之上，统计一定周期内节点的返回情况，以此来计算节点的死亡概率（超过超时次数/总检测次数）。对于死亡列表中的节点发起有限次数的重试，来做进一步判断。

对于设定的概率进行比对如果达到设定的概率可以进行一个真实踢出局的操作

使用consul做健康检查：定时http、tcp、grpc请求，或者命令脚本

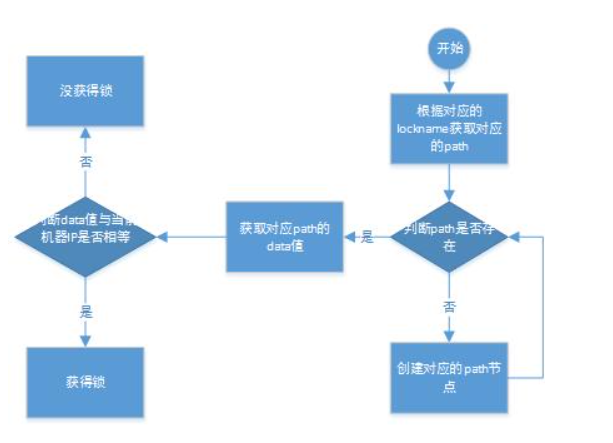
<https://www.tizi365.com/archives/514.html>

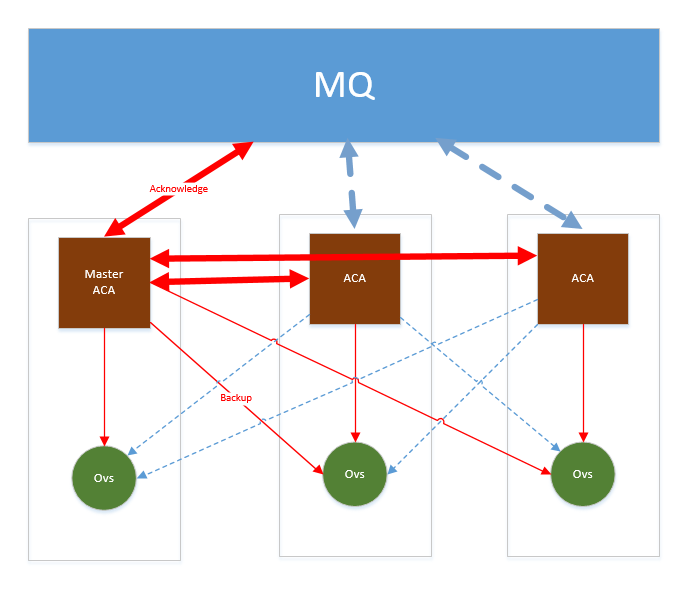
consul有两个级别的健康检查，一是节点级别，另一是app级别。节点级别就是监控consul agent是否存活；app级别可以对整个节点的service检查，也可以针对某一个service来检查。检查可以通过定义在配置文件中，也可以通过consul的HTTP接口来动态定义。consul的app健康检查有多种类型，比如shell脚本，HTTP query，TCP connection， TTL和使用docker容器来运行健康检查。

Zookepper健康检查

具体就是通过每次启动一个ACA的时候都对应启动一个client到zookeeper的连接（长连接、通过心跳包检测连接是否正常），并且创建一个临时的znode临时节点(EPHEMERAL类型)，这种类型的节点有一个特点就是如果clinet与zookeeper的连接一旦超时，就会删除该节点。

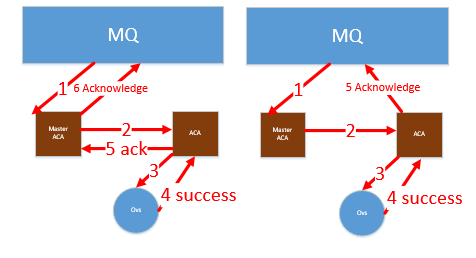
client可以通过监控该节点，配置回调函数，来获取变化，从而得到该server是否正常或者已经宕机。

1. 



1.2.2 存在问题：

1. 方案（2）（3）的转发模式下，是否可以使用三角形acknowledge，而不是链式acknowledge？consumer.acknowledge(msg);



1. Key shared模式下如果一个group内所有consumer全部宕机了，producer往topic内发消息该select谁？
2. 用sendasync（）方法发送时，ACA acknowledge部分GS；一段时间后Pulsar会重发所有GS，包括已经acknowledge的GS。用send（）方法发送不会出现该问题。

1.2.3 多集群管理

i、类似现网，将host node分为一个个cell，每个cell使用一个MQ集群。以rack group划分topic。

ii、新增host1时，根据host1所属rack确定其cell及候选host group；候选group中有同rack的host，指定一个host group包含的最大host数量，如果某个候选host group（topic）拥有的host数少于该最大值，则将新增的host1加入该host group，否则创建一个新的host group。dpm为该host1分配一个msg key，该host1以[hash(msg key), hash(msg key))为range订阅。Dpm发送给该host1的msg发往对应host group，msg key从映射表中查找。

## 1.3 增量更新（讨论）

疑问：

I、重启后的aca是否需要重新拿已经配置过的GS？

II、是否需要区分消息类别，对不同类别消息做不同处理，之前配置过的一次性配置（如create port）不用重新拉去，之前配置过的on demand类配置（如 alcor的arp responder）需要重新拉。

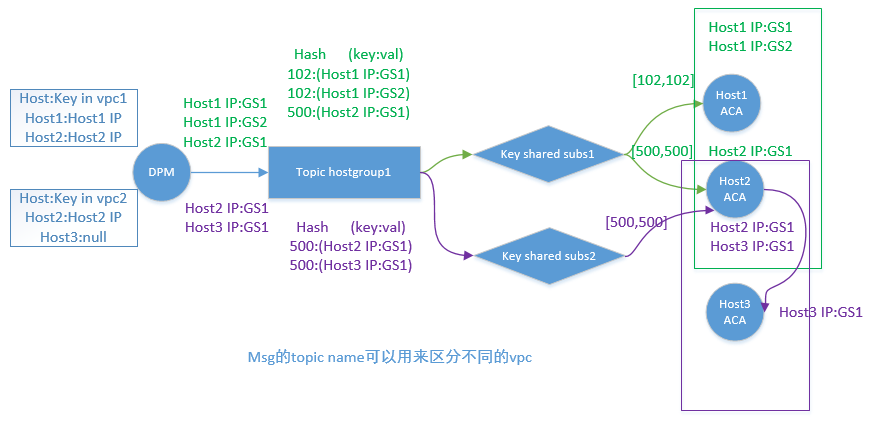
想法：

I、ACA本地持久化+未acknowledge的msg定时重发

II、msg带上时间戳、aca记录最后一个msg的时间戳,重启后从bookkeeper上查询该时间戳之后的on demand类配置。

# 二、VPC Topic

## 2.1 Keyshared模式



2.1.1 Hash冲突：类似于1.1的方案。以vpc做为topic，与该vpc相关的host采用keyshared模式订阅相应topic。每个msg附带的key为host的IP。

由于有些大的vpc内包含的vm数可能会超过65535，必然会造成hash冲突。因此我们将vpc做切分或者参考1.1.2中方案避免Hash冲突。

2.1.2 多集群管理

i、创建新VPC时，根据已有MQ集群的负载，为其指定MQ集群或开辟新集群。

ii、新增host1时，不为其指定MQ和topic，当host1与某个vpc有关联时，host1订阅相关的topic；dpm为各vpc下host1的IP作为msg key，将消息发往对应的vpc topic。

## 2.2 Failover模式（由于同一vpc关联的host不一定在同rack，不建议使用）

参考：

<https://blog.csdn.net/u013970991/article/details/52722680>

<http://pulsar.apache.org/docs/zh-CN/io-use/>

<https://blog.csdn.net/qiangcuo6087/article/details/79042035>