数据库两地三中心实践

由 路希（luxi）-智能平台部创建, 最后修改于三月 12, 2020

随着公司业务发展，对各业务系统的可用性、稳定性以及容灾能力要求越来越高。机房故障的容灾甚至地区级别的容灾需求越来越迫切，因此两地三中心的部署方式就显得尤为重要。

1. 两地三中心

1.1 什么是两地三中心

两地三中心中的“两地”是指同城和异地，“三中心”是指“生产中心”、“同城容灾中心”以及“异地容灾中心”。正常情况下两个容灾中心可以提供查询服务，分担流量。灾难情况下有能力快速切换到容灾中心，提供正常生产服务。两地三中心的部署可以提升业务系统容灾能力，提高系统可用性，保障业务的持续可用。

1.2 两地三中心规划

在介绍两地三中心规划前，我们先来了解三个名词：数据中心（IDC）、可用区（AZ）、区域（Region）：

- 数据中心（IDC）：指用于安置计算机系统及相关部件的设施，一般它包含冗余和备用电源，冗余数据通信连接，环境控制和各种安全设备，是承载公司各类业务，为业务提供专业化的服务器托管以及高可用基础网络资源的机房，核心机房之间通过专线互联互通。
- 可用区（AZ）：可由单个大型数据中心组成，也可以由地理位置相近的几个数据中心组成，同一个可用区（AZ）内的数据中心间，网络时延≤1ms。
- 区域（Region）：指可以在其中托管资源的特定地理位置。同一个区域（Region）可以有多个可用区（AZ），可用区（AZ）之间利用光纤网络互联，可用区（AZ）之间的时延视物理距离而定，一般在2~5ms之间。

目前爱奇艺已经建设了华北region（北京地区）、华中region（武汉地区）。华北region有多个AZ，华中region一个AZ，因此我们两地三中心的部署方案中的两地就是指华北region和华中region，三中心是指华北region的两个AZ和武汉region的一个AZ。

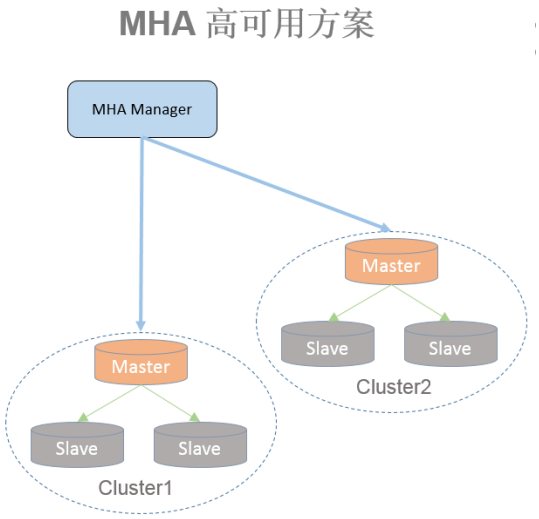
2. 数据库的高可用

2.1 常见的高可用方案

首先介绍一下常用数据库（MySQL、Redis、Couchbase）的常见高可用方案，主要目的是和我们提供的高可用方案做一个对比。

2.1.1 MySQL的常见高可用

MySQL有很多高可用方案，比如共享存储的方案SAN、基于磁盘复制的方案DRBD、基于ZK的、基于proxy的等等。比较常见的还是MHA，MHA是一个日本公司用Perl编写的开源实现，能做到30秒内自动完成故障切换，并且能最大程度的保障数据库的一致性。是目前MySQL高可用方面一个比较成熟解决方案。它的部署架构如下图：



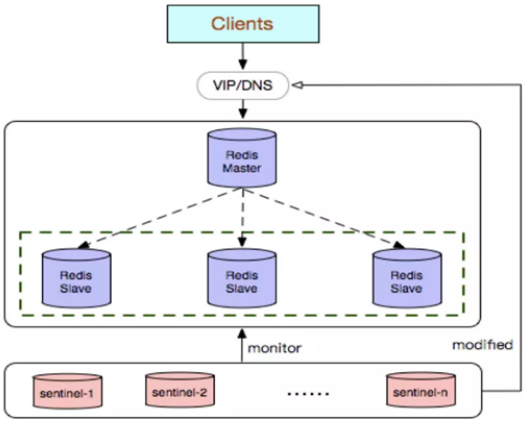
MHA整体主要有两个角色：MHA Manager和 MHA Node，Manager节点部署在一台独立的机器上，其主要作用是定时探测master节点，当master发生故障是，可以将最新的slave提升为新的master，然后再将所有slave重新指向master同步数据。node节点运行在每台MySQL机器上，其作用是复制主节点的binlog数据、对比从节点的中继日志文以及定时删除中继日志。MHA的一次切换流程是这样的：

- 从宕机崩溃的master保存二进制日志事件
- 识别含有最新更新的slave

- 应用差异的中继日志(relay log)到其它slave
- 应用从master保存的二进制日志事件
- 提升一个slave为新master并记录binlog file和position。
- 使其它的slave连接新的master进行复制
- 完成切换manager主进程OFFLINE

2.1.2 Redis常见高可用

Redis的高可用是使用官方提供的sentinel来做监控的。一个集群有至少3个以上的奇数个sentinel一起监控，sentinel的主要功能就是对实例监控，发故障通知以及故障自动转移。通过操作域名，实现故障的透明转移。



每个sentinel每隔1秒会向它所知的Master，Slave以及其他Sentinel发送一个PING命令，如果在一定时间内没有返回pong。这个实例就被sentinel标记为主观下线，当master实例被多个sentinel标记为主观下线后，这里的多个可以通过设置quorum来指定个数，我们线上设置的quorum=2，也就是说当有2个以上的sentinel标记了master主观下线，则状态会变成变成客观下线。此时sentinel会根据raft协议选举出一个leader，由leader对集群发起切换，将其中一个slave作为master，当然我们可以设置slave的优先级，提升预先指定的slave为新master，旧的master恢复后会作为slave加回集群。通过操作DNS，实现故障的透明转移。从Redis3.0开始，官方正式提供了分布式的cluster集群方案，有效地解决了redis分布式的需求，当一个redis节点挂了可以快速的切换到另一个节点。解决了单机内存、并发受限等瓶颈。

2.1.3 Couchbase常见高可用

Couchbase也是一种分布式的存储解决方案，集群里所有节点都是对等的，提供相同的功能，没有层次结构和主从之分，每个节点负责对一部分数据进行相应，集群自身就带了高可用。Couchbase我们主要是用作KV存储，基于成本考虑，我们只配置了2个副本，所以一个集群在同一时刻只能允许故障一个节点，如果同时故障2个以上节点，整个集群将处于不可用状态。因此Couchbase的高可用方案我们考虑的主要是集群级别的高可用。

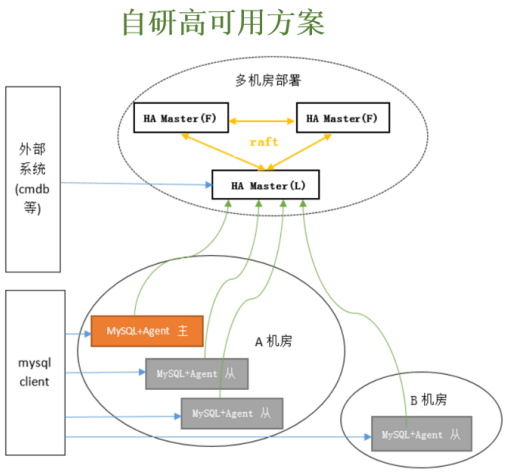
2.2 爱奇艺高可用方案

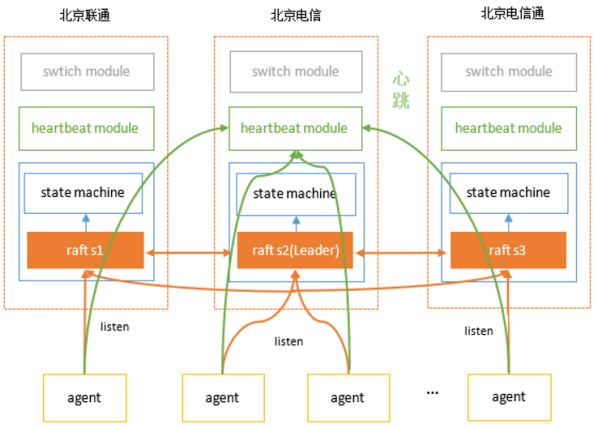
2.2.1 爱奇艺MySQL高可用方案

从上面的介绍来看，MHA已经可以实现MySQL的高可用，为什么我们还要选择自主开发一套HA系统呢？这主要是因为MHA存在如下缺点：

- 通过配置文件来管理主备关系，配置管理，不能重复切换，实例增减需要重启manager
- manager是单点，虽然有standby，但不能自动切换
- 使用Perl编写，二次开发困难
- 我们的MySQL部署环境复杂，存在跨DC跨地域的部署，新主的选举需要更多的规则
- 我们的集群数太多，MHA需要部署多套，增加管理的复杂性

因此，我们在借鉴了MHA的思想基础上自主开发了一套MySQL-HA方案。



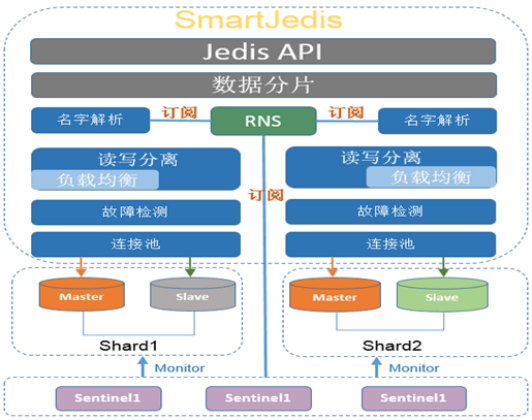


系统主要由HA-Master和Ha-Agent两部分组成。三个HA-Master组成一个最小集群单元，HA-Masterr有三个重要的模块，状态机、心跳模块和切换模块。状态机保存了当前raft leader的地址，心跳模块负责和agent保持没10秒1次的心跳检查。同时更新心跳时间戳和实例状态。切换模块负责根据规则轮训badinstance集合，对符合条件的instance发起切换。HA-Agent部署在MySQL服务器上，每台MySQL服务器部署一个agent，负责发送心跳，解析binlog、relay log purge、group replication failover等。HA-Master集群间通过raft协议实现高可用，每个HA-Master集群会选举出一个Leader，负责心跳和故障切换等工作。

Ha Master集群解决了MHA manager单节点的问题，通过和CMDB的交互，消除了MHA配置文件的管理方式，实现多次切换。同时一个HA Master集群可以监控多个MySQL集群，极大的简化了运维的复杂度。目前我们按照地域部署，只部署了北京地域、济阳地域、上海地域、武汉地域、重庆地域、香港地域以及新加坡一共7套Ha Master集群，这7个集群就覆盖了我们线上所有MySQL。

2.2.2 Redis客户端分片高可用方案

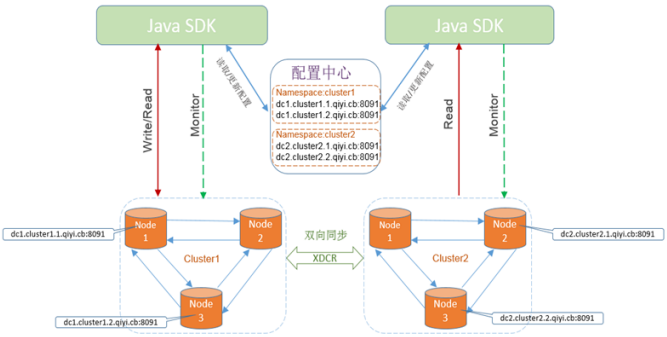
Jedis是Redis官网推荐的一个面向java的客户端，在公司使用广泛。通常情况下我们使用jedis主要是单实例的Redis应用或者使用JedisPool满足高并发环境的需求。但有时候单实例Redis不足以支持我们的业务，那么我们就需要引入多个实例进行数据分片。Jedis 支持通过 ShardedJedis 和 ShardedJedisPool 这两个类来实现客户端分片，但这种方式会存在两个问题： 1. 大多数情况下，我们是用同步的方式调用 redis 命令。命令发出去之后，我们需要等待结果返回，才会做后续的操作。由于 ShardedJedis 到每个分片都有一个连接，我们在访问一个分片的时候，到其它分片的连接都是闲置着的。2. 如果 ShardedJedis 访问了一个不可用的分片，这个 ShardedJedis 实例会被销毁，它上面所有的连接都会被关闭。后面再要访问一个正常的分片，还得重新创建一个 ShardedJedis。这样以来，由于一个分片的故障，整个集群的访问方式都成了短连接了。而且 ShardedJedis 在创建的时候需要连接到所有的分片，本身的创建成本就很高。因此当集群中有故障分片的时候，ShardedJedis 的性能下降得非常厉害。除此之外，jedis 不会跟踪 redis server 的状态。即使 redis 挂了，jedis 还是照样发送请求。在访问量大的情况下，这会造成很大浪费：客户端需要不停的创建 Jedis 实例、尝试建立 TCP 连接、处理异常。同时使用域名的方案其实也存在域名操作生效的TTL时间限制，为了解决这些问题，我们开发了smartjedis：



SmartJedis向外提供Jedis API访问一个分片集群，内部通过订阅sentinel消息，提供RNS（Redis Name Service）注册和订阅服务，实现自动切换故障节点。解决了单分片故障不影响其他分片的访问的问题、屏蔽DNS直接使用IP访问，同时支持在线地址变更、在线添加密码等功能。关于SmartJedis的更多特性，可以参考文档。

2.2.3 爱奇艺CB高可用方案

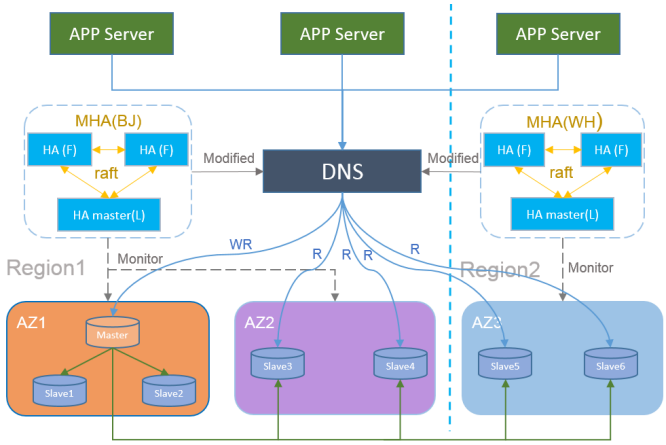
Couchbase我们提供了JAVA SDK，集群通过域名方式访问，每个集群提供2个域名，域名信息注册到配置中心，JAVA SDK通过读取配置中心配置，获取地址后访问集群。当集群出现问题时可以实现一键迁移流量



3. 两地三中心的部署方案

3.1 MySQL

MySQL两地三中心部署方案依托自研的MySQL-HA来实现，MySQL-HA支持自动切换和手动切换两种模式，在自动切换模式下，切换策略有4种：1. none 关闭自动切换，2. same_idc 同机房切换，3. same_location 同地域切换， 4. cross_location 跨地域切换。可在CMDB的集群配置里设置自动切换的策略。具体部署架构如下图所示：



1. AZ1部署master，同AZ部署slave1、slave2
2. AZ2部署slave3、slave4
3. AZ3部署slave5、slave6
4. 两个不同Region的HA-Master集群监控本地域的MySQL实例。
5. 华北Region的HA-Master集群多AZ部署，避免AZ故障导致全部HA-Master故障
6. CMDB调整集群切换策略为cross_location

在这个架构中，我们测试了5中场景，测试结果如下：

场景	切换结果	耗时	结论
1. 主库故障	主库切换正常（优先同机房切换）	6秒	符合预期
2. 主库机房故障	主库切换正常（优先同地域切换），从库不切换	10秒	符合预期
3. 机房地域性故障HA-Master正常	主库切换正常（优先同地域切换），从库不切换	10秒	符合预期
4. 机房地域性故障HA-Master单个节点故障	主库切换正常（优先同地域切换），从库不切换	10秒	符合预期
5. 地域故障（同地域HA-Master都故障）	切换失败（需要人工干预）	N/A	符合预期

从测试结果来看，同一Region需要保证至少一个HA-Master存活，主库才可以完成切换，在这种情况下选主的优先级策略是同DC>同地域>异地，但是从库流量不做切换，如果从库任然还有流量访问，需要SDK适配。

3.2 Couchbase

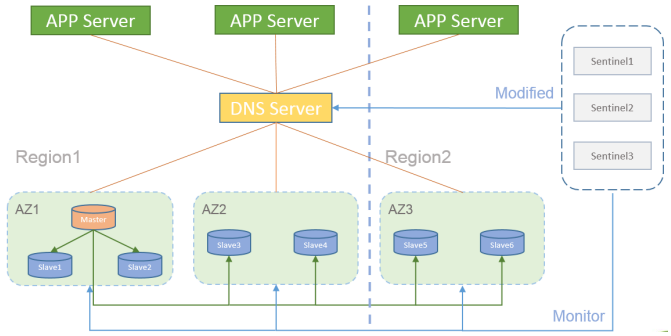
Couchbase的跨DC容灾主要是利用它本身所带的功能（XDCR）实现数据同步，XDCR主要是针对多个集群间的数据复制，复制以异步的方式通过DCP协议同步数据到其它集群中。同时我们借鉴了Redis Sentinel思想开发了一个监控服务：CB-HA Service。它是基于两个开源仓库：hashicorp/raft、hashicorp/memberlist实现一个与服务解耦的高可用监控方案。由3个以上奇数个CB-HA组成一个监控集群，集群间通过Gossip协议完成故障判断，通过raft协议选举leader执行故障转移和CB-HA集群的高可用。因CB-HA Service与具体服务解耦，不但可以监控CB，还可以监控比如Redis Cluster。CB的两地三中心部署架构如下图所示：

1. AZ1、AZ2分别部署cluster1、cluster2。cluster1与cluster2之间建立双向同步
2. AZ3部署cluster3，cluster1与cluster3之间建立单项同步，cluster2与cluster3之间建立单向同步
3. 业务通过SDK连接访问cluster1读写
4. cluster2，cluster3提供只读服务
5. CB-HA Server集群跨地域部署

CB-HA Leader通过http接口接收被监控集群的信息，信息以key/value的形式写入，Leader会将key/value转换成log entry，并持久化存储下来，同时通过一致性模块把log同步到各节点，这意味着可以安全的停止和从起CB-HA节点。当多个CB-HA节点认为cluster1故障，这里“多个”可以通过quorum参数进行配置，会调用notification script对配置中心进行修改，把域名指向cluster2，SDK会从配置中心重新读取配置，把对cluster1的请求转移到对cluster2之上，这样就完成了一次集群级别甚至机房级别的故障转移。地域级别的故障同理。

3.3 Redis

Redis的高可用是用Sentinel+DNS来实现的，Sentinel是Redis提供的原生高可用解决方案。Redis Sentinel集群是由若干Sentinel节点组成的分布式集群，可以实现故障发现、故障自动转移、配置中心和客户端通知。Redis Sentinel的节点数量要满足2n+1（n>=1）的奇数个。通过设置Slave priority优先级（值越小优先级越高，默认是100），当故障发生时，根据优先级的高低提升优先级高的slave为新的master，新的master产生后，其余的slave会主动向新master发起同步数据请求，完成故障自动转移。再把原master上的域名绑到新master恢复业务读写，整个切换过程40s内完成。但slave需要向新master做一次全量同步，在同步数据过程中有一段时间slave会不支持访问，这个的时间长短和数据量大小以及网络延迟有关。



1. AZ1部署master，同AZ部署slave1、slave2从master同步数据，设置slave1的优先级为10作为HA_Slave1，其余为100
2. AZ2部署slave3、slave4从AZ1的master同步数据，设置slave3的优先级为50作为AZ2的HA_Slave2，其余为100
3. AZ3部署slave5、slave6从AZ1的master同步数据，优先级为100
4. sentinel跨Region部署

当master宕机时，sentinel发送的ping命令在down-after-milliseconds时间后没有收到响应，就会把master标记为主观下线。当超过2个以上的sentinel都标记了主观下线，状态变成客观下线，这时通过raft协议选举出一个leader对master进行切换，提升优先级为10的ha_slave1作为新master，其他的slave向新master请求同步数据。如果ha_slave1也不可用，sentinel会提升ha_slave2为新master。切完完成后会触发client-reconfig-script配置的脚本，把旧master上的域名重新绑定到新的master上，同时发出报警，完成故障转移。地域级别故障同理。

3.4 MongoDB

MongoDB 本身就拥有高可用及数据分片的解决方案，分别为副本集(Replica Set)和分片(sharding)，我们今天只介绍副本集集群的方案，分片集群暂不支持。

3.4.1 MongoDB副本集

MongoDB 副本集具有有故障自动切换(Failover)、多机房部署以及读写行为控制等特性。主要有三类角色成员组成：Primary、Secondary、Arbiter。其中Primary是唯一具有写权限的节点，它会将所有的插入和更新操作记录到oplog中，默认情况下Primary也是所有读请求的节点。Secondary复制Primary的oplog记录并在本地回放，确保与Primary数据一致，默认情况下Secondary不允许读，但可设置slaveok参数允许客户端读。Arbiter是一种特殊角色，作为仲裁节点，不存数据，只参与选举，通常在拥有偶数个节点的复制集中添加（且仅能添加）一个Arbiter，这样可以使一次选举中达到大多数而避免选举分裂。一个复制集中仅有一个Primary，但在某些特殊场景下，可能没有Primary。Arbiter在集群中不是必须的。所以集群中最普通的角色是Secondary，一般不少于2个。不同的Secondary可以有不同的属性，处于不同的状态中。Secondary属性有如下几类：

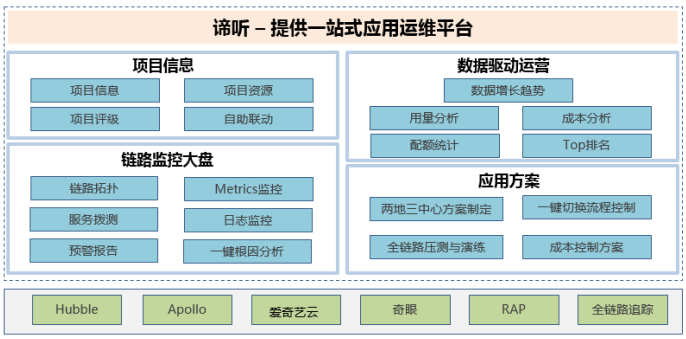
- Hidden：对客户端不可见，可以投票，但不参选，一般只用于备份节点，离线计算的任务，不处理客户端的读请求
- Prority： Prority属性是用来决定节点选主优先级的高低，取值范围0~100，值越大，优先级越高。如果值为0，那么节点将不会被选为Primary，但可以参与投票
- Delayed：延迟节点，通过slaveDelay 来设置。主要用来回滚人为误删除数据库或集合等，需要隐藏，prority为0
- Votes：决定是否具有投票权，MongoDB复制集最大是50个成员，但只允许7个成员具有投票权

1. AZ1部署primary，同AZ部署多个Secondary节点，设置其中一个Secondary的prority为3,其余节点为0
2. AZ2部署多个Secondary节点，设置其中一个Secondary的prority为2，其余为0
3. AZ3部署多个Secondary节点，设置Secondary的prority为0
4. 确保三个AZ的可投票节点数小于等于7

4. 两地三中心方案实施

4.1 谛听平台

谛听（Diting）平台是由云服务数据库组开发的一个面向业务的一站式应用运维平台，结合内部的业务监控系统奇眼、实时分析平台RAP、服务监控平台hubble、全链路追踪以及Apollo等系统，提供一站式服务拨测、集群巡检、两地三中心实施、资源使用分析、调用链追踪等功能。支持项目和服务维度的更全面系统的基础资源（数据库、中间件..）信息展示，促进资源的节约高效使用。



谛听主要分为4大模块：

- 项目信息：从项目维度展示项目所用到的各种数据库和中间件的集群信息（后期可以包含更多的基础服务资源：虚机、QAE、QLB、大数据等等），结合DBS用户可以提供更多的自助功能
- 资源（数据驱动运营）：分别从项目维度和集群维度展示基础服务的配额总量，利用率，数据增长趋势，成本分析，预算配额、征信等信息
- 链路监控：rover提供基于Skywalking的无侵入埋点，可将跨应用的调用链性能信息集中展现，度量整体和局部性能，方便找到故障的源头，缩短故障排查时间
- 应用方案：两地三中心方案制定、一键切换流程控制，成本控制等方案

4.2 故障切换

谛听向业务提供两地三中心方案制定，切换流程控制等功能，业务通过谛听平台提供的流程管理模块，制定应用每个层级的切换方案，每个层级的切换提供一个回调接口，谛听的流程管理就可以把整个应用的切换流程串联起来，形成一个应用的整套切换方案。当然还可以支持某一层级的独立切换，比如数据层的切换。每个切换步骤的成功或是失败都可以发出通知或告警，失败的支持重试机制。业务还可以制定不同的切换方案：plan A/plan B，并进行定期的实战演练，谛听记录每次演练的结果，优化改进的管理等等。最终是希望在出现机房甚至是地域级别故障是，可以快速的一键切换。

5. 两地三中心的落地

5.1 两地三中心在会员的演练

会员业务是爱奇艺最为重要的业务之一，从公司公布的2019年第三季度财报显示，会员服务收入37亿元人民币，占整体收入的50%左右。在第三季度末，订阅会员规模达到1.058 亿，因此保障会员业务的稳定成伟重中之重。2019年Q4以来我们配合会员业务的同学进行了测试集群演练2次10个场景，线上集群演练3次6个集群。在演练过程中我们也遇到一些问题：1. 前期HA系统文档不全面，因此在多次两地三中心高可用演练过程对HA逻辑重新进行了梳理和验证，并补充实例异常和agent异常两类测试用例。2. 配置不合理问题：在验证过程中，遇到HA配置从设计之初不符合两地三中心要求2处并做出调整。3. 多agent超时无法切换的问题，功能调整2处：1)新增网络抖动判断和处理逻辑("8分钟"检测)，并做到可配置，以满足业务对高可用级别(RTO)控制；2)去除连续切换检测对从库的限制。同时对高可用组件可用性提高，增加对agent的异常监控和报警，并完成自愈功能。

5.2 两地三中心在会员的落地

会员本季度总共上线部署同地域高可用集群74个，高可用开启同地域跨机房切换策略，并可满足跨地域主库高可用切换。其中涉及到的项目有会员院线、营销、第三方合作、奖励礼品卡系统，以及权益和交易核心集群28个。通过和业务配合，不断打磨优化，最终形成高效稳定的高可用方案。

无标签