[中间件缺陷汇总](http://wiki.htzq.htsc.com.cn/pages/viewpage.action?pageId=32045802)

中间件包括Nginx、Lvs、Kafka、ZK、Redis。

|  |  |  |  |  |
| --- | --- | --- | --- | --- |
| **名称** | **缺陷项** | **复现手段** | **解决方式** | **备注** |
| Kafka | 极端网络故障下，发现Kafka集群在Zookeeper注册的临时节点消失。Kafka集群状态正常！基于kafka消费方式正常！但是基于Zookeeper的消费方式异常，消费者消费不了数据. | 仅出现过一次，跟微服务Zookeeper信息消失一起出现。暂时无法推出复现过程。 | 滚动重启集群，让Kafka重新注册Zookeeper。 |  |
| 极端网络情况下（多次网络抖动，每个Kafka节点都连不上Zookeeper集群），Kafka脑裂，每个Kafka节点认为的topic的partition 的leader不一致。 | 多次网络抖动，每个Kafka节点都连不上Zookeeper集群。 | 滚动重启集群。 |  |
| Kafka0.9版本，删除Kafka的topic时有业务在连此topic。造成该topic删除不了，残留在集群中，造成隐患。 | Kafka0.9版本，删除Kafka的topic时有业务在连此topic。 | 需要人工删除Zookeeper里面部分Kafka的topic信息。因为该问题对其他topic暂时无影响，故为了稳定起见，不再任何处理。 因为0.9不稳定性，我们将在今年把0.9逐步下线。 |  |
| 时间跳变： 如果Kafka时间源跳变，当跳变后的时间和Zookeeper的时间差超过[zookeeper.session.timeout.ms](http://zookeeper.session.timeout.ms/)（5s）后会出现Kafka主动关闭Zookeeper节点，退出集群。 如果跳变后时间和当前时间差超过[replica.socket.timeout.ms](http://replica.socket.timeout.ms/)（60s）后会出现该Kafka主动和其他Kafka节点断开连接。 | 上级时间源服务器出现跳变已经复现了该问题。 | 修改时钟同步时间，每天一次。 |  |
| Kafka集群内topic过多，存储的数据过大。如果出现整个集群不可用需要滚动重启时，恢复时间在10-30分钟级。 |  | 今年逐步迁移核心业务，做到分级保障。 |  |
| ZK | 磁盘性能： leader节点磁盘的io出现瓶颈，会影响都整个集群的事务处理，导致响应速度变慢甚至超时。 | 当leader选择磁盘性能比较差的机器的时候，响应速度明显降低。 | 重启leader节点，重新选举leader。 |  |
| 集群性能：  所有数据，都存储在内存中，导致不能存储太多的数据。扩充集群的数量无法提升性能，单机的性能决定了集群性能的上限。 |  | 提升单机cpu io 网络性能。 |  |
| 时间跳变： 如果leader服务器出现时间跳变，向后跳超过minSessionTimeout配置值，部分client就会强制下线。 如果向后跳变时间超过maxSessionTimeout，所有client都会强制下线。 ZK客户端向后跳变超过设置的SessionTimeout值，也会主动和服务器断链。 | 上级时间源服务器出现跳变已经复现了该问题。 | 修改时钟同步时间，每天一次； 需要客户端能够支持重连并能够容忍断链重建。 |  |
| Zookeeper的权限控制非常薄弱： Zookeeper上大部分znode目前没有有效的权限管控，没有设置acl的节点任意client都可以将其删除或者更新。 | 前期已经出现将ZK中的Kafka目录删除的事情。 | 目前集群支持acl，需要客户单设置acl权限控制。 每天两次备份zk的全部数据，如果出现问题可会恢复到最近时间点的状态。 缺点：没法针对每个目录进行恢复。 |  |
| Nginx/LVS | 发布版本无统一控制。 |  | 对接制品库, 面向平台支持的环境,做统一编译和发布 | Nginx/LVS作为负载均衡服务提供, 组件本身在现有环境未遇到重大缺陷问题, 主要的问题还是在集中管理/持续运营方面 |
| 配置信息散落,无集中管理。 |  | 建设CMDB, 对接配置中心 |
| 监控覆盖不全,组件特色指标未采集。 |  | 对接云平台监控系统, 收集组件相关的关键特色指标,实时上报,建立合理监控和报警 |
| 服务请求手工操作响应,效率低。 |  | 开发自动化配置脚本,对接salt通道, 向用户自服务改造 |
| 与挂载节点无状态交互, 无法做到高效均衡和”跛脚”服务的隔离。 |  | 需要深入该领域,改造成本较高, 暂无解决方案 |
| 集群方案依赖keepalived 框架, 无法做到集群内多活和水平扩展。 |  | 考虑借助DNS提供多活节点集群服务, 方案效果待验证 |
| LVS | 组件权限要求过高。 |  | 对LVS进行自服务改造 |
| 不支持主备模式。 |  | 对于主备模式的负载由Nginx承载实现 |
| Nginx | 负载策略单一 。 |  | 组件本身定位并不是纯的负载均衡服务提供, ,需要深入该领域,改造成本较高, 暂无解决方案 |
| Redis | 奥体和龙江master节点数不相等时，占多数master节点的机房失效后，无法自动failover。 |  | 1、 时刻监控保持奥体和龙江master节点数相等。 2、 手工提升少数派为新集群，但会出现两个孤立集群。 | 1、 Redis采用三机房部署，每个集群在游府西街存在一个节点，该节点用于仲裁，但仍有一个solt的数据（没有slot的master不参与选主）。这样实现的好处是，奥体（或龙江）机器或机房失效，都能在龙江（或奥体）继续提供服务。 2、 Redis Cluster选举需要多数master节点同意才能完成。 |
| 互为主备的两个节点A（主）、B，再奥体、龙江网络相继出现持续较长的抖动时，会出现A、B节点均为从节点的情况。 | 1、 A为主节点，B为从节点。 2、 A网络断开一段时间后，B成为主，此时A节点再重建内存数据。 3、 B网络断开，此时A节点即使对外网络连接无异常，但仍在加载数据。 4、 B网络恢复，但同集群断开较长时间，被认为是从节点，从本地加载数据。 | 1、 目前技术上无法解决，只能通过分级保障，减小数据量，加快内存重构速度。 2、 调研Redis 4.0，确认新的PSYNC 2方案是否可以缓解该问题。 |  |
| Redis 2.8.\*及3.\*.\*版本主从复制采取PSYNC，PSYNC在主从切换时）不能实现“断点续传”，退化为全量同步。 | 1、 网络抖动造成主从切换。 2、 手工提升从节点，从节点执行cluter failover。 | 调研Redis 4.0，确认新的PSYNC 2方案是否可以缓解该问题。 |  |
| Redis单节点不宜过大，从aof或rdb镜像文件恢复内存数据，每GB大概需要1分钟时间恢复，目前生产上单节点已经达到17GB左右。 | kill单个节点，观察内存重建时间，每GB大概需要1分钟。 | 1、通过分级保障，减小数据量，加快内存重构速度。 2、是否需要引进ssdb、pika之类redis兼容方案。 |  |
| 旧机器规划不合理，旧机器存下以下问题：使用512G内存，但确仅使用600GB磁盘；千兆网卡，机房间全量同步较慢；未采用ssd盘。 |  | 1、 逐步替换千兆网卡为万兆。 2、 添加磁盘（已采购）。 |  |
| 部分集群配置规划不合理，存在下列问题：root权限；集中度较高；flushdb这样的高危控制命令未禁用；3.2.0存在crash级bug。 |  | 逐步整改 | Redis已触发bug：  [https://github.com/antirez/redis/commit/3876d985a8ec2fa27e9b6fb1cda7e75de775f4e](https://github.com/antirez/redis/commit/3876d985a8ec2fa27e9b6fb1cda7e75de775f4eb) |