Redis集群演化的心路历程

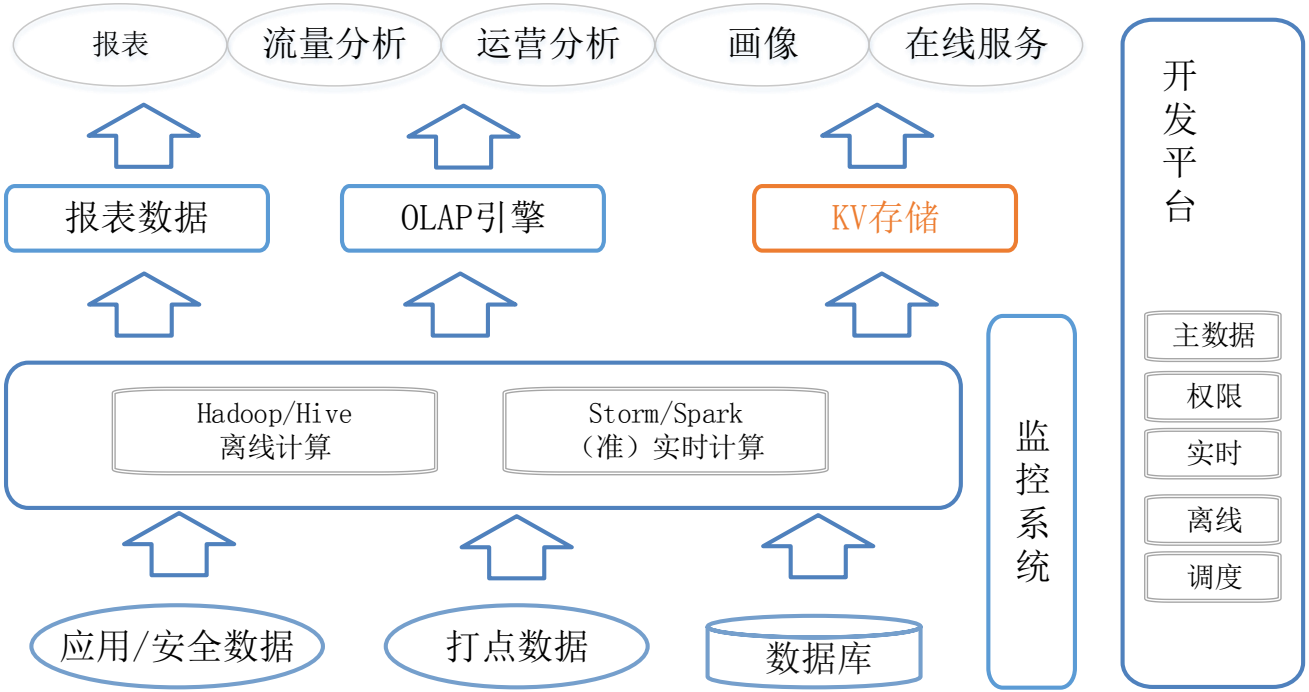
——从2.x到3.0时代

王新春
平安健康互联网 技术保障团队

目 录
content

扫码观看大会视频

大数据平台中的KV存储



KV存储的基本要求

1. 大容量，支持TB级别存储
2. 高性能
3. 功能丰富

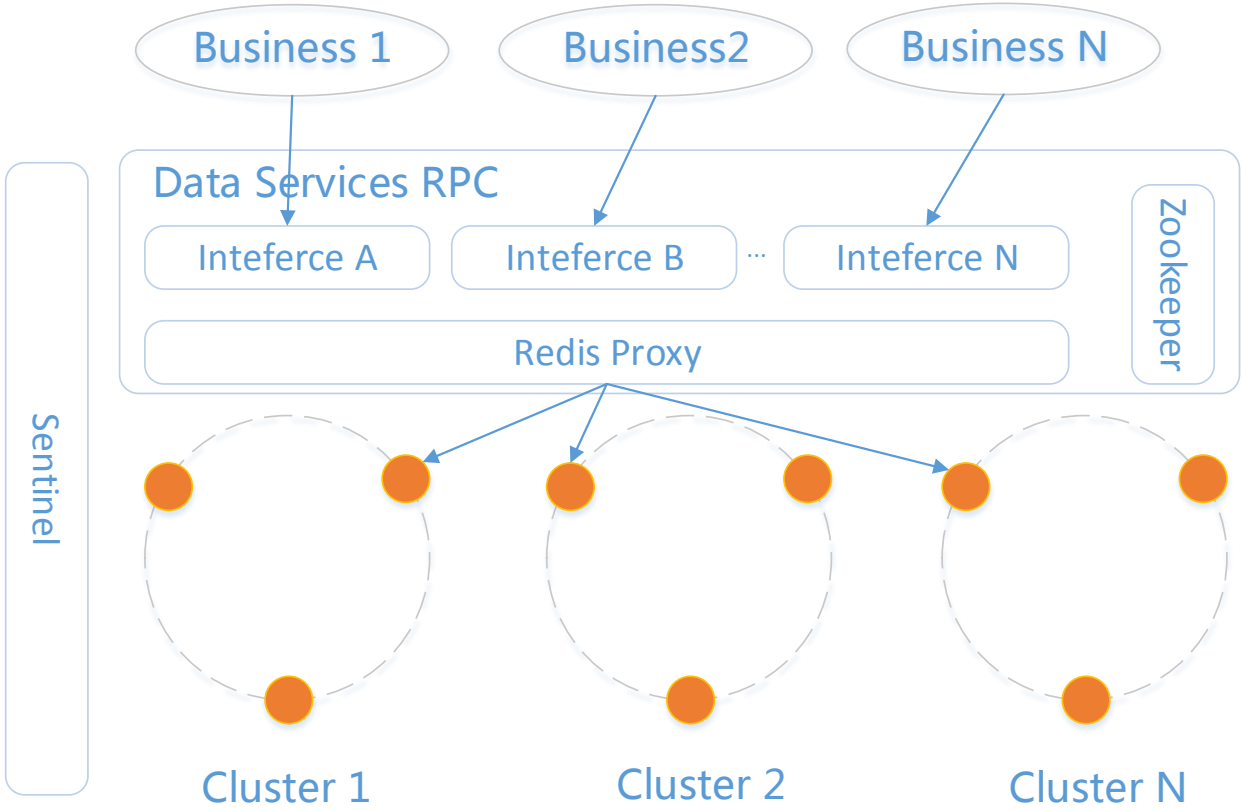
一、基于Redis 2.x的KV存储

优势：水平扩展

缺点：不满足在线核心业务SLA(99.9% 响应<25ms)

优势：高性能，API接口丰富

缺点：单节点存储能力有限


Business 1
Business2
Business N
Data Services RPC
Inteference A
Inteference B
...
Inteference N
Redis Proxy
Zookeeper
Sentinel
Cluster 1
Cluster 2
Cluster N

1

Cluster：每个集群32个节点

32个M/S共64个节点，由Sentinel保证主从可用性
集群中数据分布通过Key的一致性hash，支持读写分离

2

集群中数据分配规则存储在Zookeeper中

RedisKey key = new RedisKey(tableName, bKey)
tableName定位属于哪个集群和区分业务，

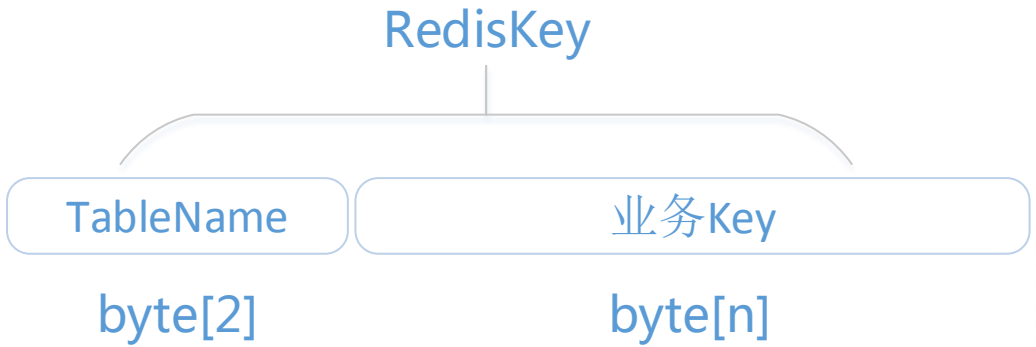
3

对外提供RPC服务

不同业务线有不同的接口名词和容量配置
集群维护和数据迁移业务无感知

Key设计：

1. 将业务的不同类型Key归一，减少Key的长度（每个Key减少1字节整个集群可以节约6G内存，超过60亿Key）
2. 增加Key的识别属性，根据Key前缀区分业务
3. 新Key的用前两字节表示表（表名的映射值），同一集群最大支持65535张表



整合业务监控系统：

1. 基于业务/Table的访问指标：QPS、延迟（平均/95%/99.9%）、ERROR
2. 无访问或者极少请求的数据下线

主要存在的问题：

1. 扩容困难，单个集群容量合理上限1TB（单节点30GB）
2. 共享集群，业务彼此之间存在影响
3. 数据导出到hive等复杂
4. RPC存在性能损耗

二、Redis 3.0 Cluster时代

1

根据业务线拆分集群

大业务独立集群，小业务共享集群

2

自动化部署集群

Redis Cluster On Docker

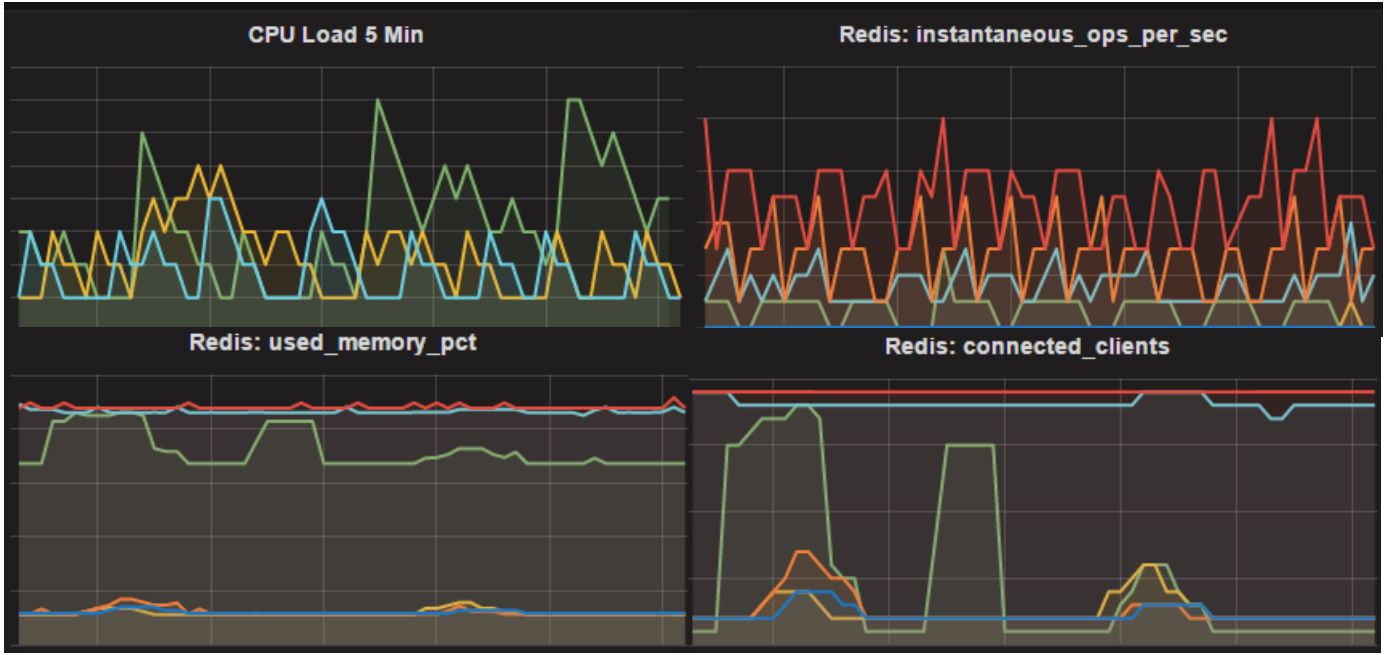
3

Server端信息对客户端透明

客户端根据Cluster名称即可访问

支持参数动态调整，配置信息集中维护

监控：每个集群独立dashboard



三、Redis Client的改进

基于Jedis的Client

```
RedisStoreClient storeClient = StoreClientFactory.getStoreClient("test-cluster");

StoreKey key = new StoreKey("business", "key");

String value = storeClient.get(key);
```

JedisCluster参数支持动态变化

```
private JedisCluster createClient() {
    JedisCluster client = new JedisCluster(
            redisClientConfig.getServers(),
            connTimeout,
            readTimeout,
            maxRedirects,
            getPoolConfig(redisClientConfig),
            clusterUpdateInterval,
            redisClientConfig.getPassword());
    return client;
}
```

SocketTimeoutException！ Which node ?

修改JedisConnectionException

```
public JedisConnectionException(String message, Throwable cause, HostAndPort targetNode) {
        super(message, cause);
        this.targetNode = targetNode;
    }
```

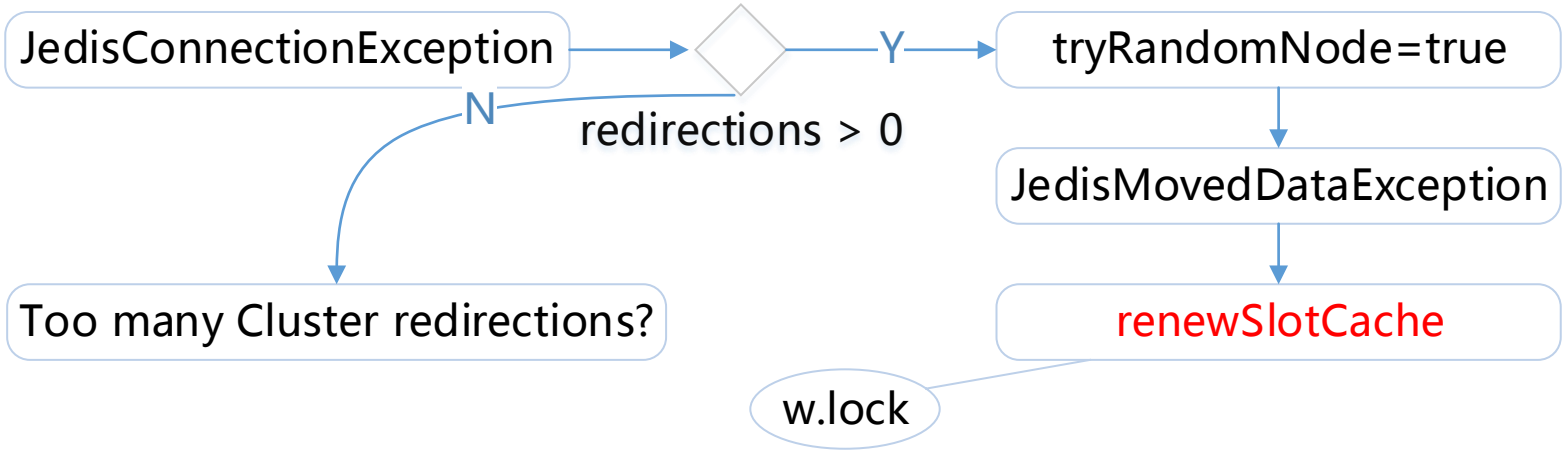
ConnectTimeout和ReadTimeout都打印HostAndPort

Too many Cluster redirections?

出现的时机：

1. 节点down机，同时集群性能急剧下降
2. 网络抖动或者慢操作

Root Cause：JedisClusterCommand：runWithRetries



Too many Cluster redirections?

Solution：

- redirections = 1
- 为解决集群迁移、扩容和节点主从切换的访问失败

1. 后台定期renewSlotCache
2. 增加SocketTimeoutException计数器，超过阈值，触发renewSlotCache
3. 当前同一slot的connection已经在renewSlotCache，则等待结果并返回

```
public void renewSlotCacheNow(boolean forceRenew, int waitMillis) {
  long currentTs = System.currentTimeMillis();
  if (isClusterInfoRenewing.compareAndSet(false, true)) {
    try {
      if (ready4Renew(forceRenew, currentTs)) {
        renewSlotCache();
        updateLastRenewTime();
      }
    } finally {
      isClusterInfoRenewing.set(false);
    }
  } else {
    if (forceRenew && waitMillis > 0) {
      waitRefreshFinish(waitMillis, currentTs);
    }
  }
}
```

基于node级别的multi 和pipeline

+pipelineWithNode

+mutiWithNode

开发使用建议

- Key设计
- TTL
- Hash VS JSON
- hash/list/set/sorted set的Item数量
- Value序列化
- 操作的时间复杂度o(1)？o(n)？

Redis Next ?

- SSD
- 非易失性存储

2016 The Computing Conference

THANKS

扫码观看大会视频