咕泡科技
GUPAO TECH

Redis 项目实战之微服务架构中的应用

GUPAO TECH

咕泡科技
GUPAO TECH

我是谁

Wilson

深耕技术的 Coder

经验丰富的 Architect

充满活力的 TecLeader

职业12余年

一半在阿里，从微服务到云原生平台，先后lead过多个系统的架构和落地

另一半有1/2在创业公司度过，作为技术总监负责整个技术部，为公司上新三板提供技术支撑

再往前，曾在甲骨文负责ATS产品“weblogic集群调度模块”的设计和研发

目前在一个电商公司负责核心电商framwork相关的设计和开发工作

咕泡科技
GUPAO TECH

内容提纲

- 实战准备：Redis & Spring
  - 实战中Redis的几种模式
  - Redis Client 的选择
  - Spring Redis
- 实战一： 分布式Session
  - 项目背景
  - 实战
- 实战二： 用户状态缓存
  - 项目背景
  - 实战
- 实战三： 分布式限流
  - 项目背景
  - 实战

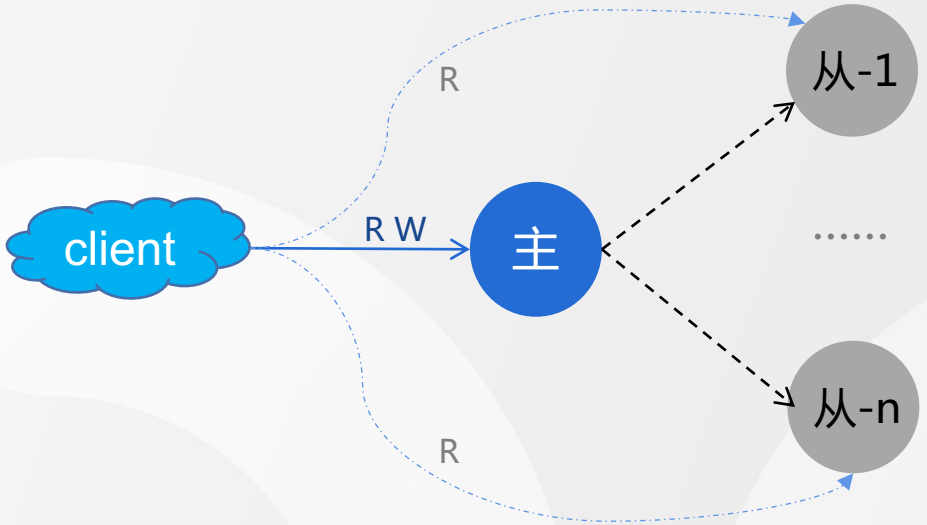
咕泡科技 GUPAO TECH

实战中Redis的几种模式

- Redis 主从模式
  - 1 主（服务）+ N 从（备用）
  - 手动切换
  - 适用于可用性要求不高的场景
  - Local 开发调试: 1主0从

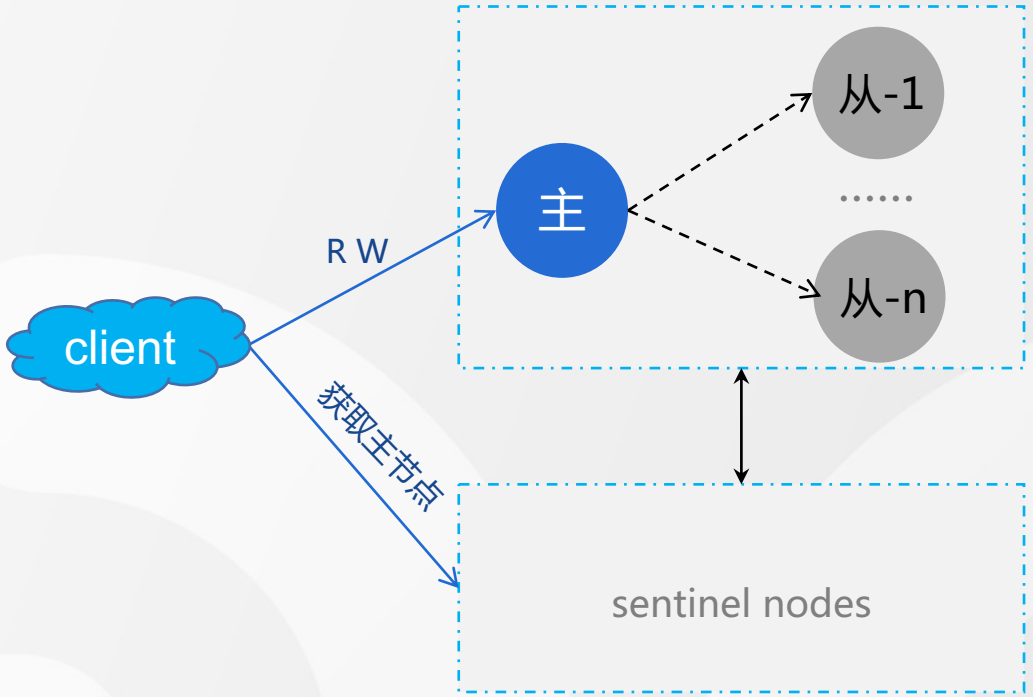
Docker - Local 的简单方式:

- docker pull redis
- docker run --name redis –d –p 6379:6379 redis
- docker exec --it redis redis-cli



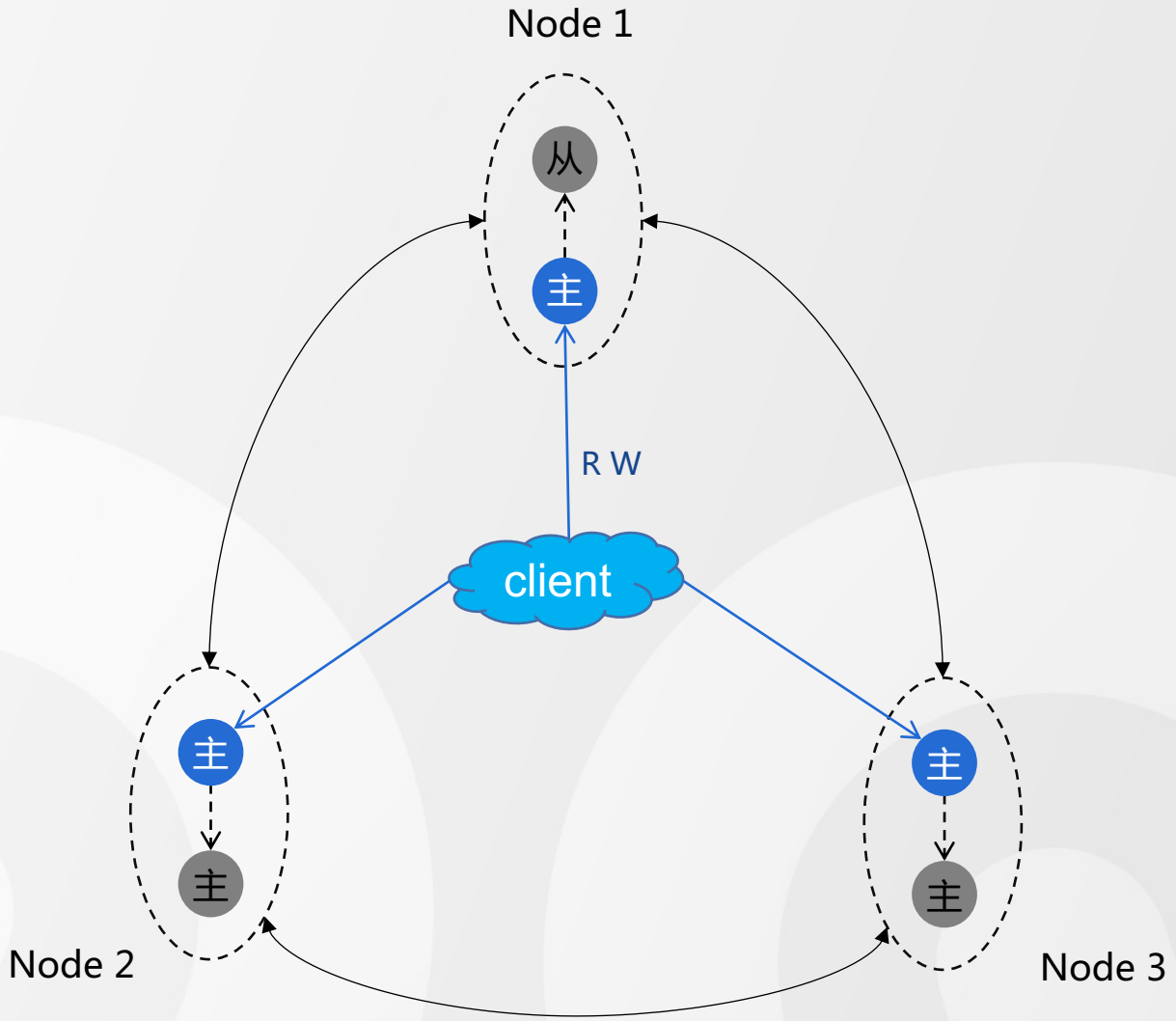
实战中Redis的几种模式

- Redis 哨兵模式
  - Redis的一种高可用方案
  - 自动切换
  - 监控、通知、failover、服务发现



实战中Redis的几种模式

- Redis 集群模式
  - 去中心化 (部分节点失效时有一定可用性)
  - 数据分片（16384个Hash Slot）
  - Redis节点自监控



咕泡科技
GUPAO TECH

Redis Client 的选择

- Jedis 介绍
  - 非线程安全，一般需要连接池
  - 同步阻塞API，使用相对简单
  - 通过 JedisPool 获得 Jedis 实例
  - 支持多种模式
    a) 主从模式： JedisPool → Jedis
    b) 集群模式： JedisPool + cluster nodes → JedisCluster
    c) 哨兵模式： JedisSentinelPool → Jedis

咕泡科技
GUPAO TECH

Redis Client 的选择

- Jedis Pool 及 配置

```xml
<dependency>
    <groupId>redis.clients</groupId>
    <artifactId>jedis</artifactId>
</dependency>

<dependency>
    <groupId>org.apache.commons</groupId>
    <artifactId>commons-pool2</artifactId>
</dependency>
```

```
@Bean
@ConfigurationProperties("redis")
public JedisPoolConfig jedisPoolConfig() {
    return new JedisPoolConfig();
}
```

```
### jedis pool config
redis.maxTotal=8
redis.maxIdle=8
redis.testOnBorrow=true
```

咕泡科技 GUPAO TECH

Redis Client 的选择

- Jedis 三种模式

```
//使用 redis.host 配置主节点地址
@Bean(destroyMethod = "close")
public JedisPool jedisPool(
        @Value("${redis.host:localhost}") String host,
        JedisPoolConfig poolConfig) {
    return new JedisPool(poolConfig, host);
}
```

```
//redis.sentinel.nodes=ip:port,ip:port,ip:port
//redis.sentinel.master_name=xxx
@Bean
public JedisSentinelPool jedisSentinelPool(
        @Value("${redis.sentinel.nodes}") String sentinelNodes,
        @Value("${redis.sentinel.master_name}") String masterName,
        JedisPoolConfig poolConfig ){
    Set<String> sentinelNodeSet = Arrays.stream(sentinelNodes.split( regex: ","))
            .collect(Collectors.toSet());
    return new JedisSentinelPool( masterName, sentinelNodeSet, poolConfig);
}
```

```
//redis.cluster.nodes=ip:port,ip:port,ip:port
@Bean
public JedisCluster jedisCluster(
        @Value("${redis.cluster.nodes}") String clusterNodesStr,
        JedisPoolConfig poolConfig){
    return new JedisCluster(createClusterNodeSet(clusterNodesStr), poolConfig);
}
```

咕泡科技
GUPAO TECH

Redis Client 的选择

- Jedis 三种模式演示

代码演示

redis-client-demo/jedis

咕泡科技 GUPAO TECH

Redis Client 的选择

- Lettuce 介绍
  - 线程安全
  - 支持同步、异步、以及 Reactive 编程模型
  - 支持读写分离
  - 支持多种模式
    a) 主从模式： RedisClient，StatefulRedisConnection
    b) 集群模式： RedisClusterClient，StatefulRedisCLusterConnection，
    c) 哨兵模式： RedisClient，StatefulRedisSentinelConnection，

咕泡科技
GUPAO TECH

Redis Client 的选择

- Lettuce 配 置

```
<dependency>
    <groupId>io.lettuce</groupId>
    <artifactId>lettuce-core</artifactId>
</dependency>
```

咕泡科技 GUPAO TECH

Redis Client 的选择

- Lettuce Standalone

```
@Bean(destroyMethod = "shutdown")
public RedisClient redisClient(@Value("${redis.host:localhost}") String host,
                               @Value("${redis.port:6379}") int port){
    return RedisClient.create(RedisURI.create(host, port));
}
```

```
private StatefulRedisConnection<String, String> sharedConnection;
public void init(){
    sharedConnection = redisClient.connect();
}
public void op(){
    RedisCommands<String,String> syncCommand = sharedConnection.sync();
    //... do your ops with sync command here

    RedisAsyncCommands<String,String> asyncCommand = sharedConnection.async();
    //... do your ops with async command here

    RedisReactiveCommands<String,String> reactiveCommand = sharedConnection.reactive();
    //... do your ops with reactive command here
}
```

咕泡科技 GUPAO TECH

Redis Client 的选择

- Lettuce Cluster

```
@Bean
public RedisClusterClient redisClusterClient(@Value("${redis.cluster.nodes}") String clusterNodesStr){
    List<RedisURI> clusterNodes = createClusterNodes(clusterNodesStr);
    return RedisClusterClient.create(clusterNodes);
}
```

```
private StatefulRedisClusterConnection<String, String> sharedClusterConnection;
public void init(){
    sharedClusterConnection = redisClusterClient.connect();
}
public void op(){
    RedisAdvancedClusterCommands<String,String> syncCommand = sharedClusterConnection.sync();
    //... do your ops with sync command here

    RedisAdvancedClusterAsyncCommands<String,String> asyncCommand = sharedClusterConnection.async();
    //... do your ops with async command here

    RedisAdvancedClusterReactiveCommands<String,String> reactiveCommand = sharedClusterConnection.reactive();
    //... do your ops with reactive command here
}
```

Redis Client 的选择

- Lettuce Sentinel

```
@ConditionalOnProperty(name = "lettuce.demonstration.sentinel", havingValue = "true")
public static class SentinelDemonstration{
    @Bean(destroyMethod = "shutdown")
    public RedisClient redisClient(@Value("${redis.sentinel_host}") String sentinelHost,
                                   @Value("${redis.master}") String master){
        RedisURI redisUri = RedisURI.Builder.sentinel(sentinelHost, master).build();
        return RedisClient.create(redisUri);
    }

}
```

Redis Client 的选择

- Lettuce 三种模式演示

代码演示

redis-client-demo/lettuce

Redis Client 的选择

- Redission
  - 分布式锁是其一大亮点
  - 高级功能：分布式对象、容器

咕泡科技
GUPAO TECH

Spring 对 Redis 的支持

- 支持 Jedis 和 Lettuce
  - 连接工厂解读
    - JedisConnectionFactory && LettuceConnectionFactory
      - RedisStandaloneConfiguration
      - RedisClusterConfiguration
      - RedisSentinelConfiguration
  - 自动配置
    - RedisAutoConfiguration
    - RedisReactiveAutoConfiguration

咕泡科技
GUPAO TECH

Spring 对 Redis 的支持

- RedisTemplate 统一API
  - RedisTempate<K, V>
    - opsForXxx()
  - StringRedisTemplate
  - RedisTemplate<Object, Object>

Spring 对 Redis 的支持

- RedisTemplate 使用演示

代码演示

spring-redis-demo/RedisTemplate

咕泡科技
GUPAO TECH

Spring 对 Redis 的支持

- Repository
  - @RedisHash
  - @Id
  - @Indexed

Spring 对 Redis 的支持

- Repository 使用演示

代码演示

spring-redis-demo/RedisRepository

咕泡科技
GUPAO TECH

Spring 对 Redis 的支持

- Spring 缓存抽象对Redis的支持
  - @EnableCaching
    - @Cacheable
    - @CacheEvict
    - @CachePut
    - @Caching
    - @CacheConfig

咕泡科技
GUPAO TECH

Spring 对 Redis 的支持

- Spring Redis Cache 使用演示

代码演示

spring-redis-demo/SpringRedisCache

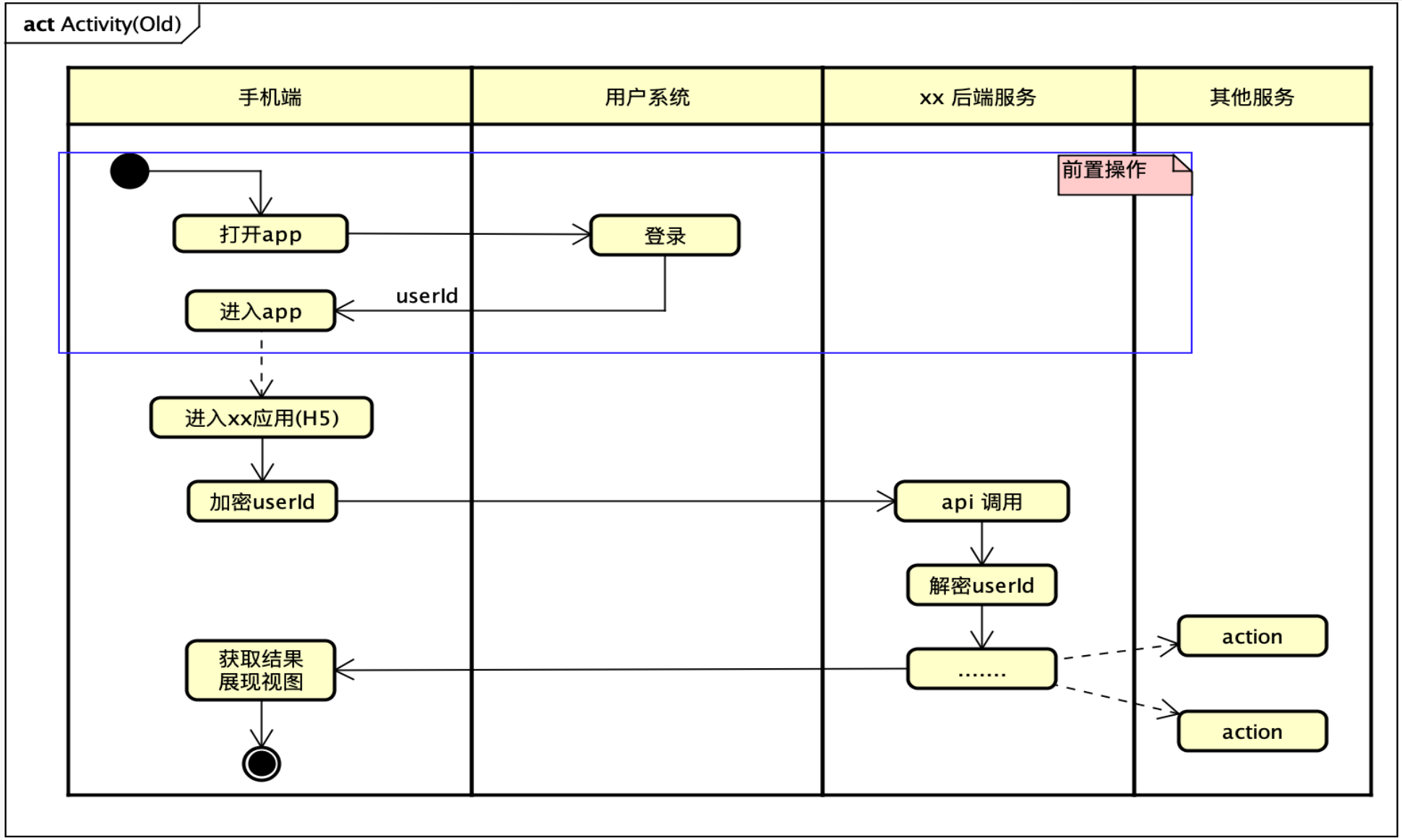
咕泡科技
GUPAO TECH

实战一： 分布式 session

- 背景
  - 对“收购”系统 改造为“自有”应用
  - 对“H5”应用 改造为 “小程序”生态并存
  - 对 UID 加解密 改造为“分布式”session

实战一： 分布式 session

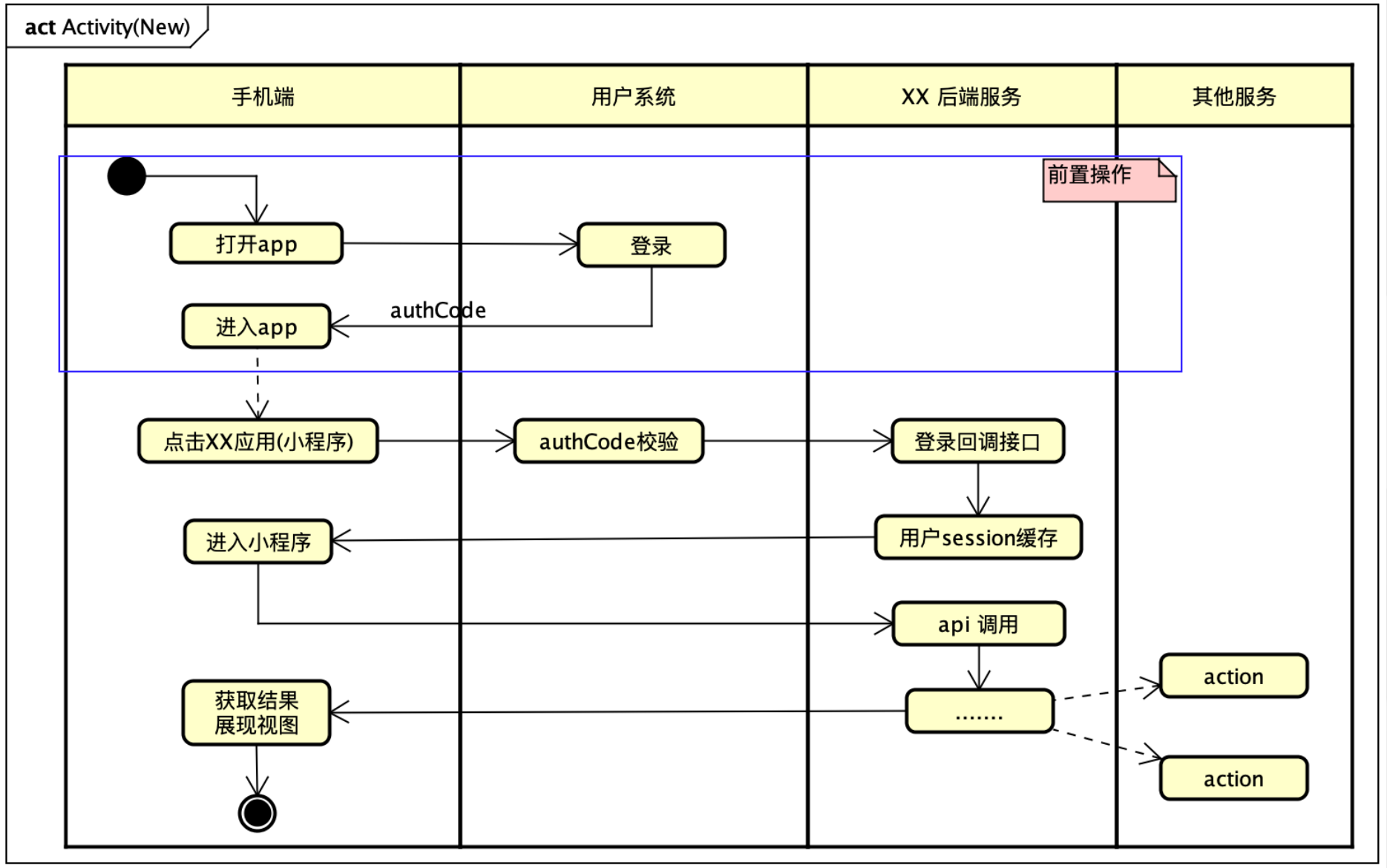
- 改造前



咕泡科技 GUPAO TECH

实战一： 分布式 session

- 改造后
  - 以小程序为主
  - 兼容老版本



咕泡科技
GUPAO TECH

实战一： 分布式 session

- 方案选择
  - 统一Session
  - Spring-session
  - 自己定制，够用就好

实战一： 分布式 session

- 设计实现
  - Redis Cluster(3主3从)
    - redis-cluster-conf/redis-8001.conf
    - redis-cluster-conf/redis-8002.conf
    - redis-cluster-conf/redis-8003.conf
    - redis-cluster-conf/redis-8004.conf
    - redis-cluster-conf/redis-8005.conf
    - redis-cluster-conf/redis-8006.conf
  - 数据结构选择
    - *String (prefix_sessionId)*
      *c_session_*27E00EF352F224207224353D27D262C3 => userId
    - *Hash (<key, sessionId>)*
      <*c_session*, 27E00EF352F224207224353D27D262C3> => userId
      <27E00EF352, F224207224353D27D262C3> => userId

  - *命令*
    启动节点： redis-server redis-xxx.conf
    创建集群： redis-cli create –cluster-replicas <所有节点>
    连接节点： redis-cli –c –h <host> -p <port>
    查看节点：cluster nodes

咕泡科技
GUPAO TECH

实战一： 分布式 session

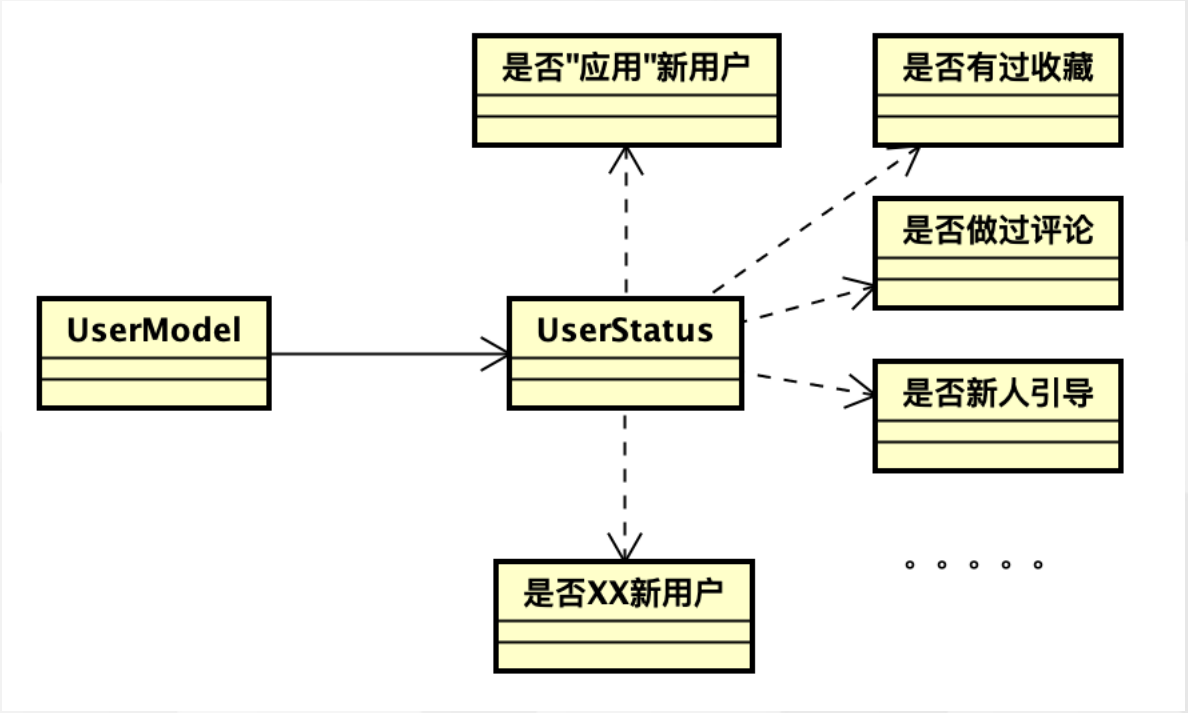
- 实战代码

代码演示

session-cache

实战二：用户状态缓存

- 背景
  - 随着业务功能增加，用户状态越来越多
  - 随着用户状态增加，对用户模型越来越依赖
  - 随着用户/流量增加，数据库压力越来越大



咕泡科技
GUPAO TECH

实战二：用户状态缓存

- 分析
  - 数据需要持久化(数据库)，缓存目的是提升性能
  - 用户状态不是一成不变，缓存需保持和数据库一致(防止缓存污染)
  - 万一出现不一致，后果可能是用户看到两次新人引导(不致命)
  - 用户在做某个操作的时候会触发状态更新(频率低，读多写少)
  - 状态位更新不可逆(从0变到1，反之不成立)
  - 以用户为粒度读写(userId 构建key)

实战二： 用户状态缓存

- 用户状态模型设计

➢ 数据库表

字段名	字段类型	说明
user_id	varchar(64) 实际是数字类型	主键，分表键(后三位分1000张)
status	BIGINT	64位，除符号位之外，后63位代表63个用户状态
last_update	Datetime	最后一次修改时间

➢ 状态位说明

(符号位)	前55位（第9位到第63位）	0000	0/1	0/1	0/1	0/1
-				首次领卡位	首次收藏位	应用新用户位

➢ 后三位说明

✓ 应用新用户位： 0 新用户，1 非新用户.

✓ 首次收藏位： 0 之前未有过收藏，1之前有过收藏.

✓ 首次领卡位： 0 之前没领过卡，1之前有过领卡.

✓ 判断： status & 1 ！= 1

✓ 判断： status & 2 ！= 2

✓ 判断： status & 4 ！= 4

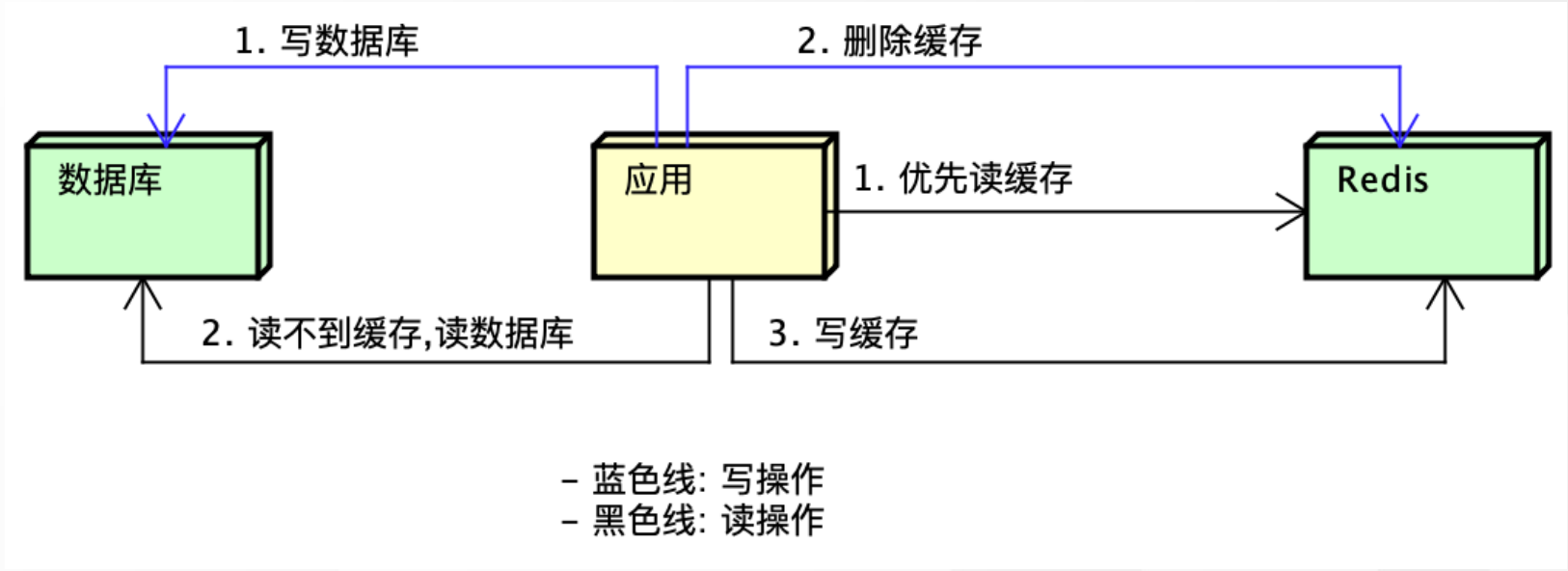
✓ 更新： status = status | 1

✓ 更新： status = status | 2

✓ 更新： status = status | 4

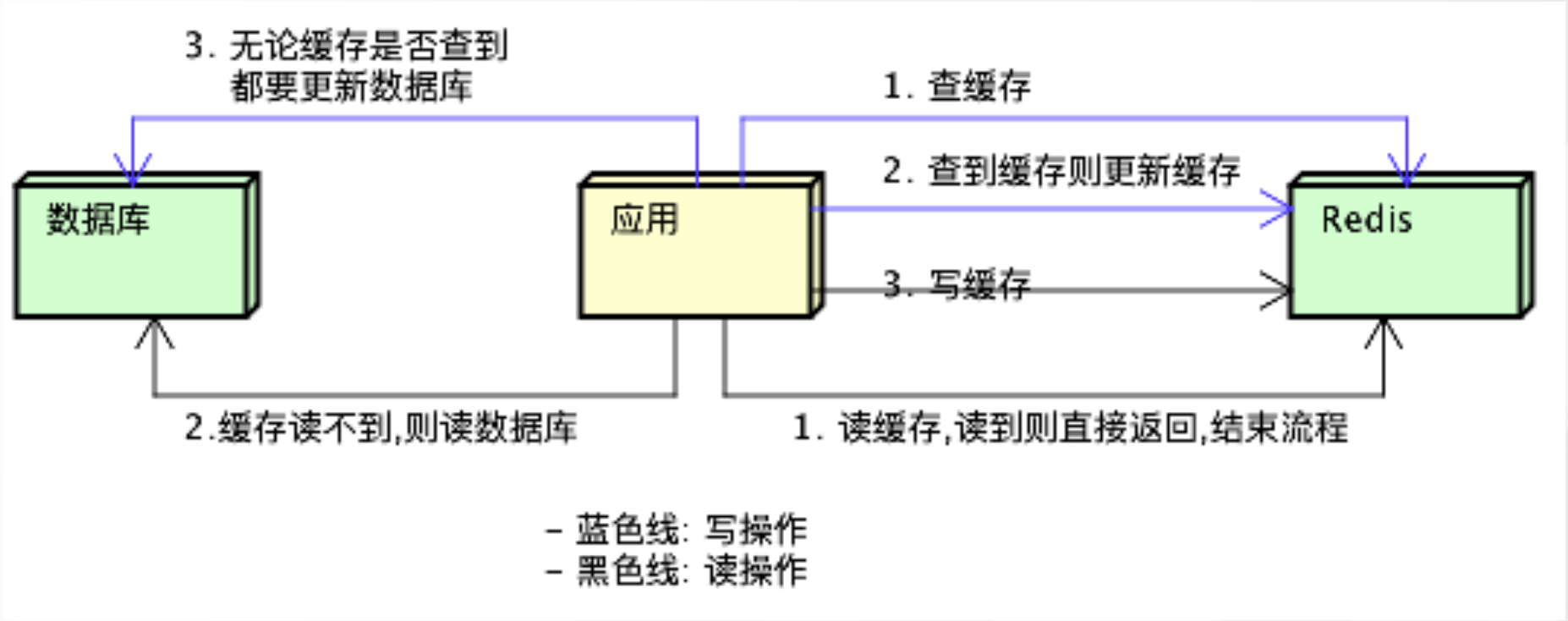
实战二：用户状态缓存

- 缓存模式选择
  - 旁路缓存模式（Cache Aside Pattern）



实战二：用户状态缓存

- 缓存模式选择
  - 读写穿透模式（Read/Write Through Pattern）



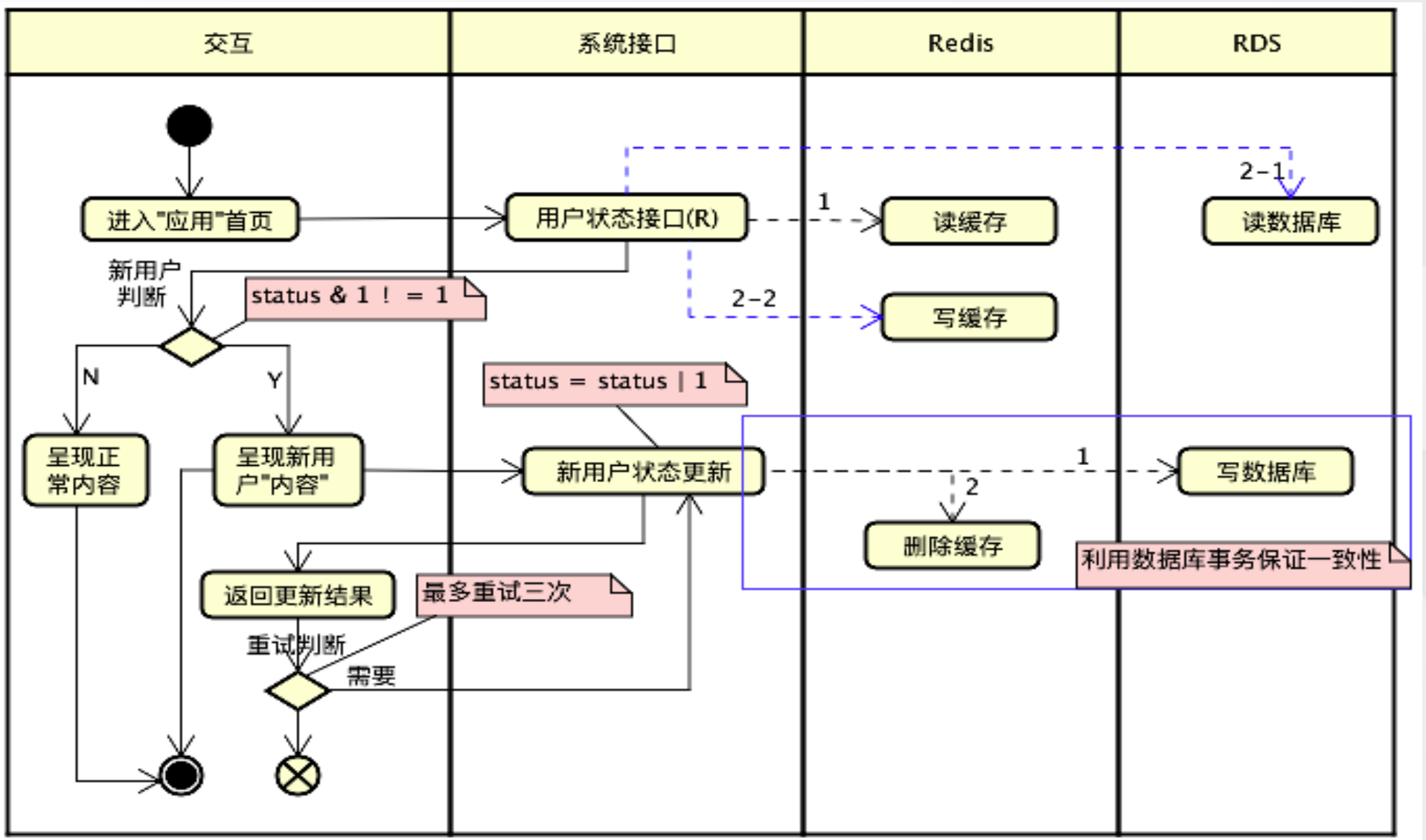
实战二：用户状态缓存

- 缓存模式选择
  - 异步写入模式（Write Behind Pattern）

    和读写穿透模式类似，不同点读写穿透模式是同步写入缓存和数据库，而异步写入模式只会更新cache，DB的更新是批量异步方式进行

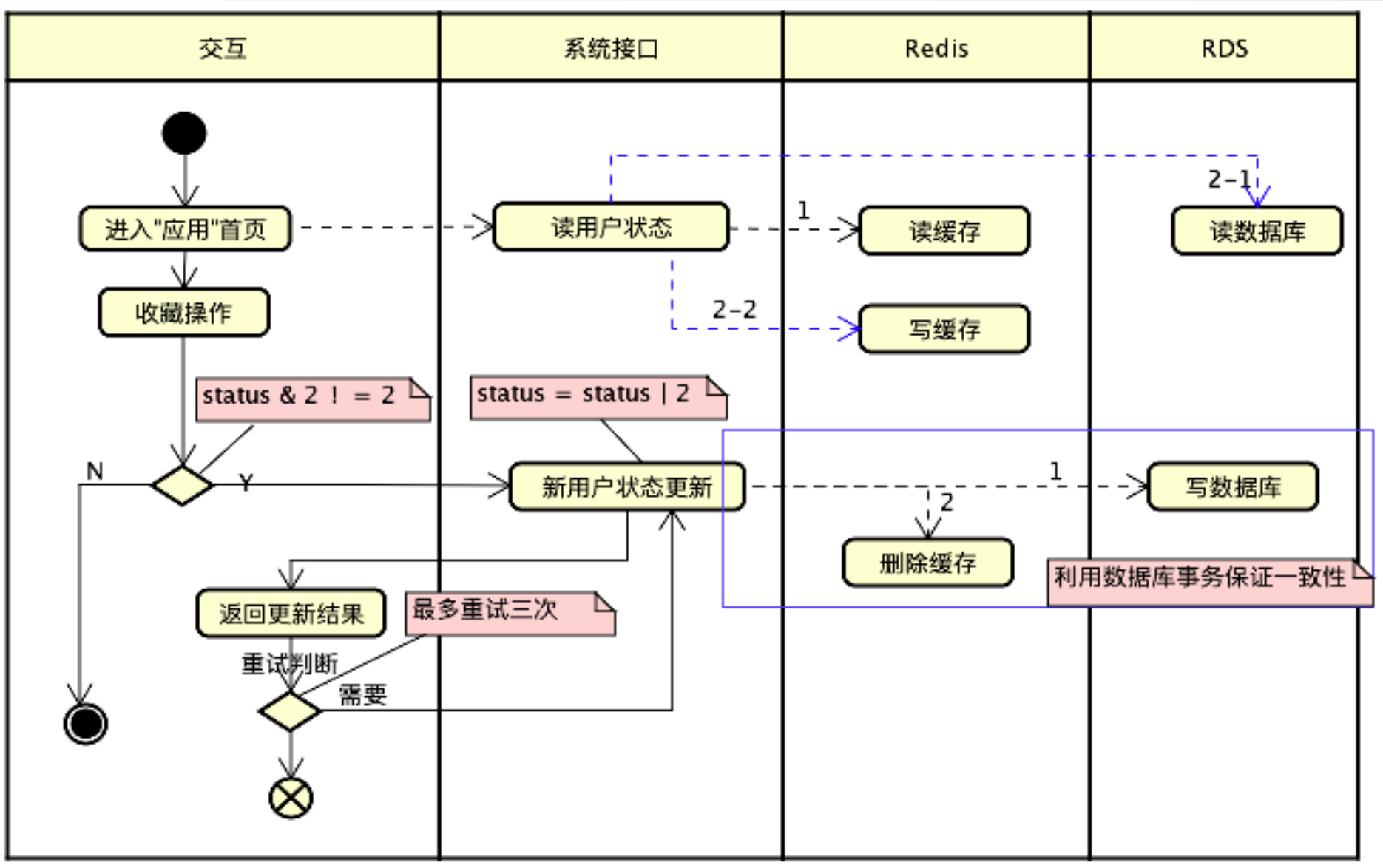
实战二：用户状态缓存

- 流程设计
  - 应用新用户状态
  - 旁路缓存模式



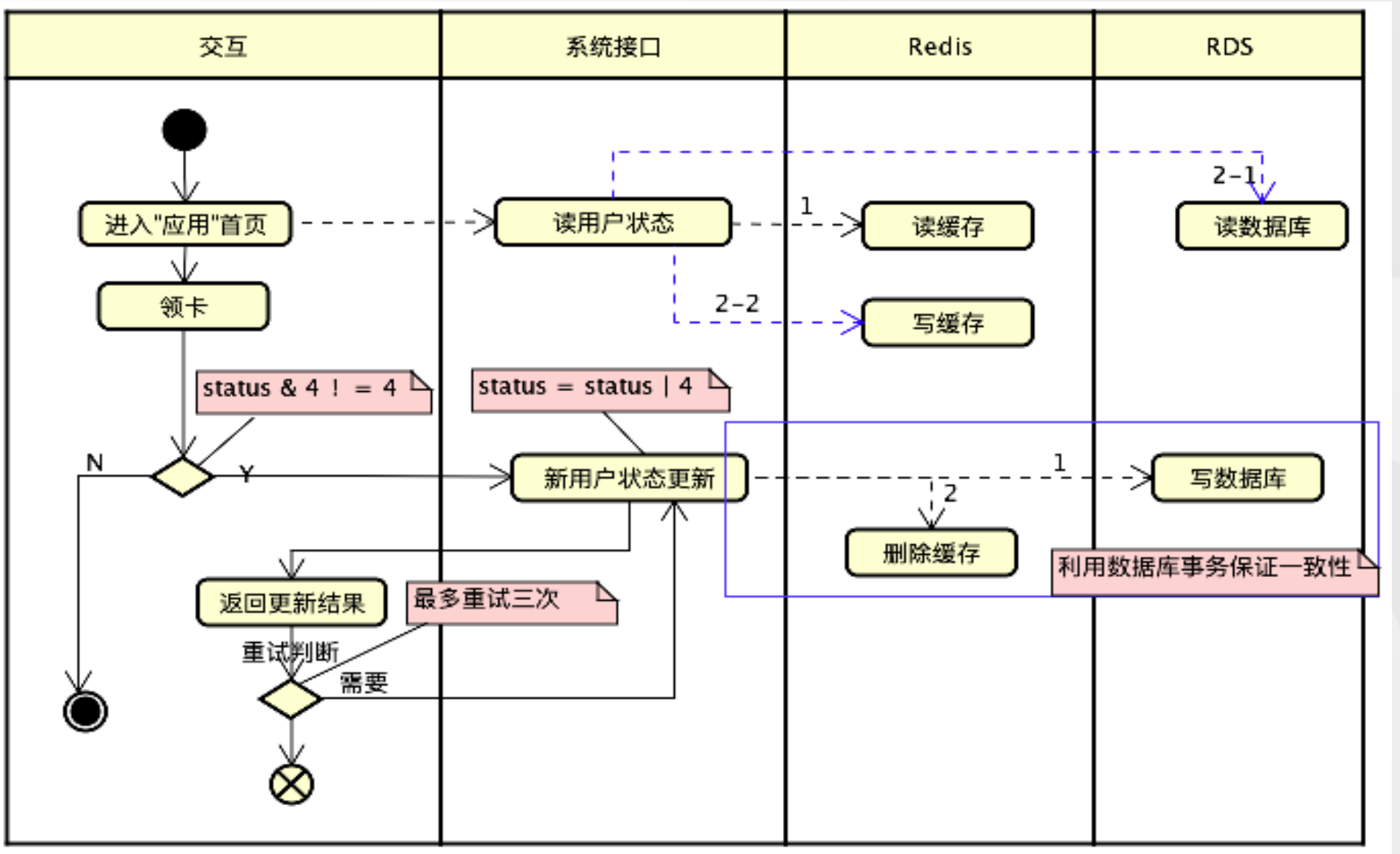
实战二：用户状态缓存

- 流程设计
  - 用户“收藏”状态
  - 旁路缓存模式



实战二：用户状态缓存

- 流程设计
  - 用户“领卡”状态
  - 旁路缓存模式



实战二： 用户状态缓存

- 实战代码

代码演示

xx-service

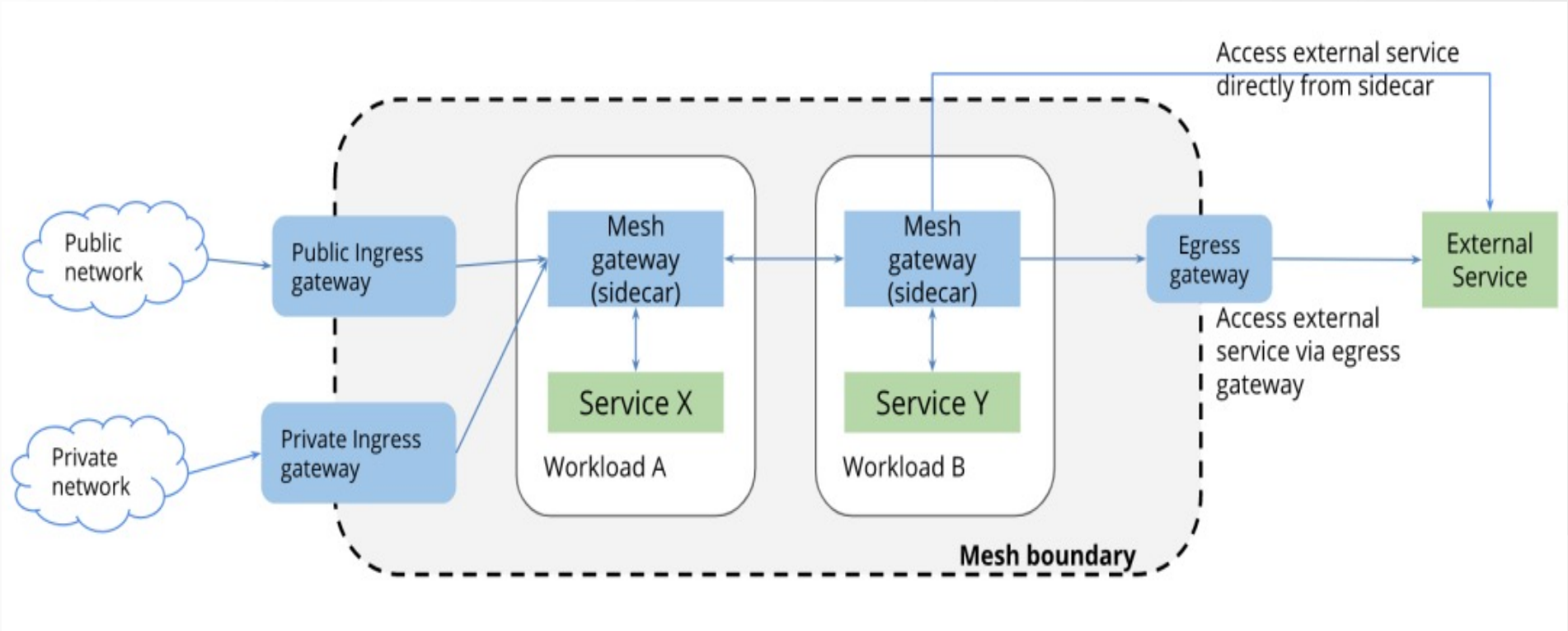
咕泡科技
GUPAO TECH

实战三：分布式限流

- 背 景
  - 核心链路高可用治理，“限流”成为不可或缺的组件之一
  - 各系统(团队)纷纷自建，百花怒放
  - Framework 组统一管控，提高效率
  - 边缘网关是“最佳”载体

实战三：分布式限流

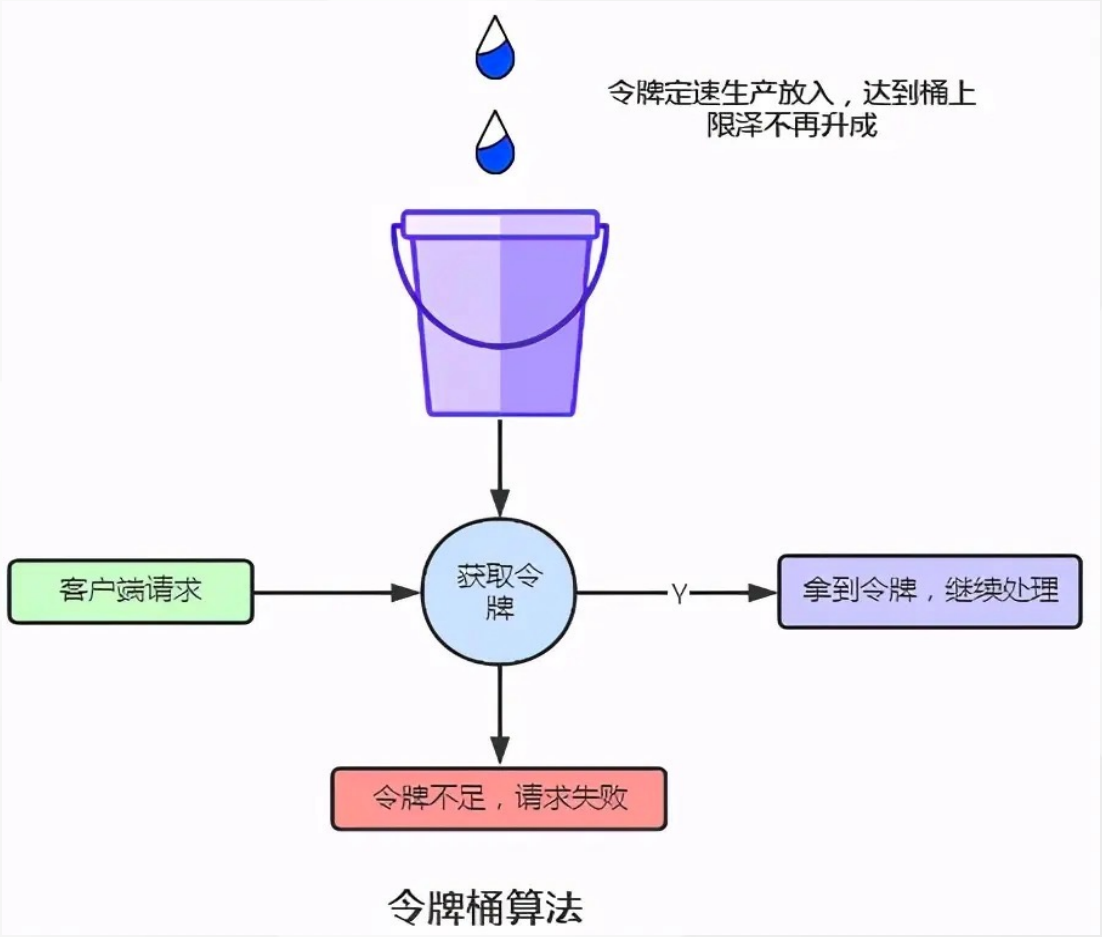
- 微服务高可用“四大利器”
  - 熔断
  - 限流
  - 降级
  - 重试



- 对自身的保护
- 对upstream的保护

实战三：分布式限流

- 限流的基本概念
  - 对访问频率的限制（QPS）
  - 对连接数的限制
  - 对客户端的限制（e.g 用户黑白名单，IP黑白名单）
  - 对速率的限制（e.g 文件的下载速度，视频流的加载速度）
- 限流的常用算法：
  - 时间窗口算法
  - 漏桶算法
  - 令牌桶算法

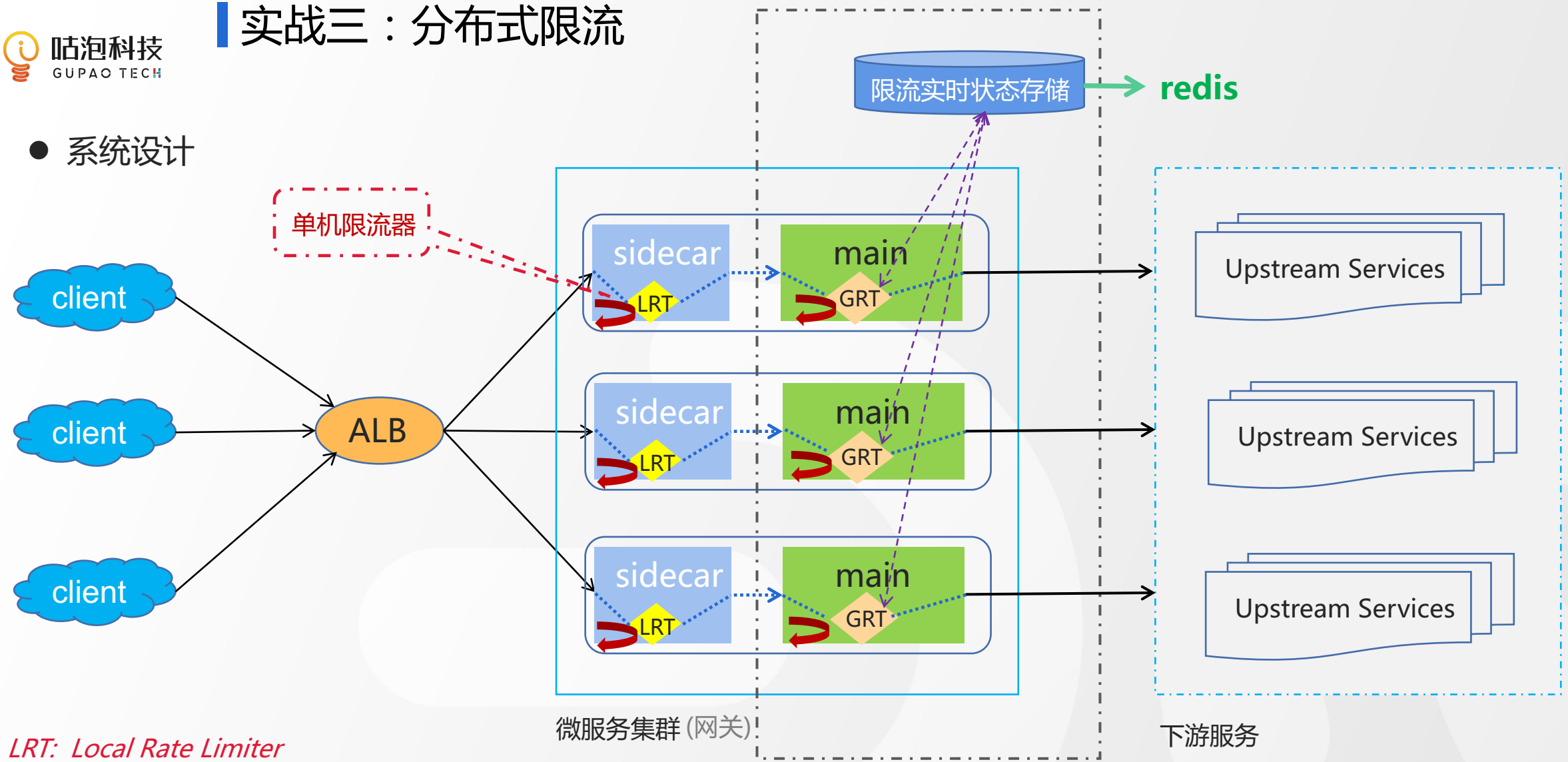


令牌桶算法

- ✓ 桶的容量
- ✓ 生成令牌的速率
- ✓ 请求令牌的速率

实战三：分布式限流

- 系统设计



LRT: Local Rate Limiter

GRT: Global Rate Limiter

咕泡科技
GUPAO TECH

实战三：分布式限流

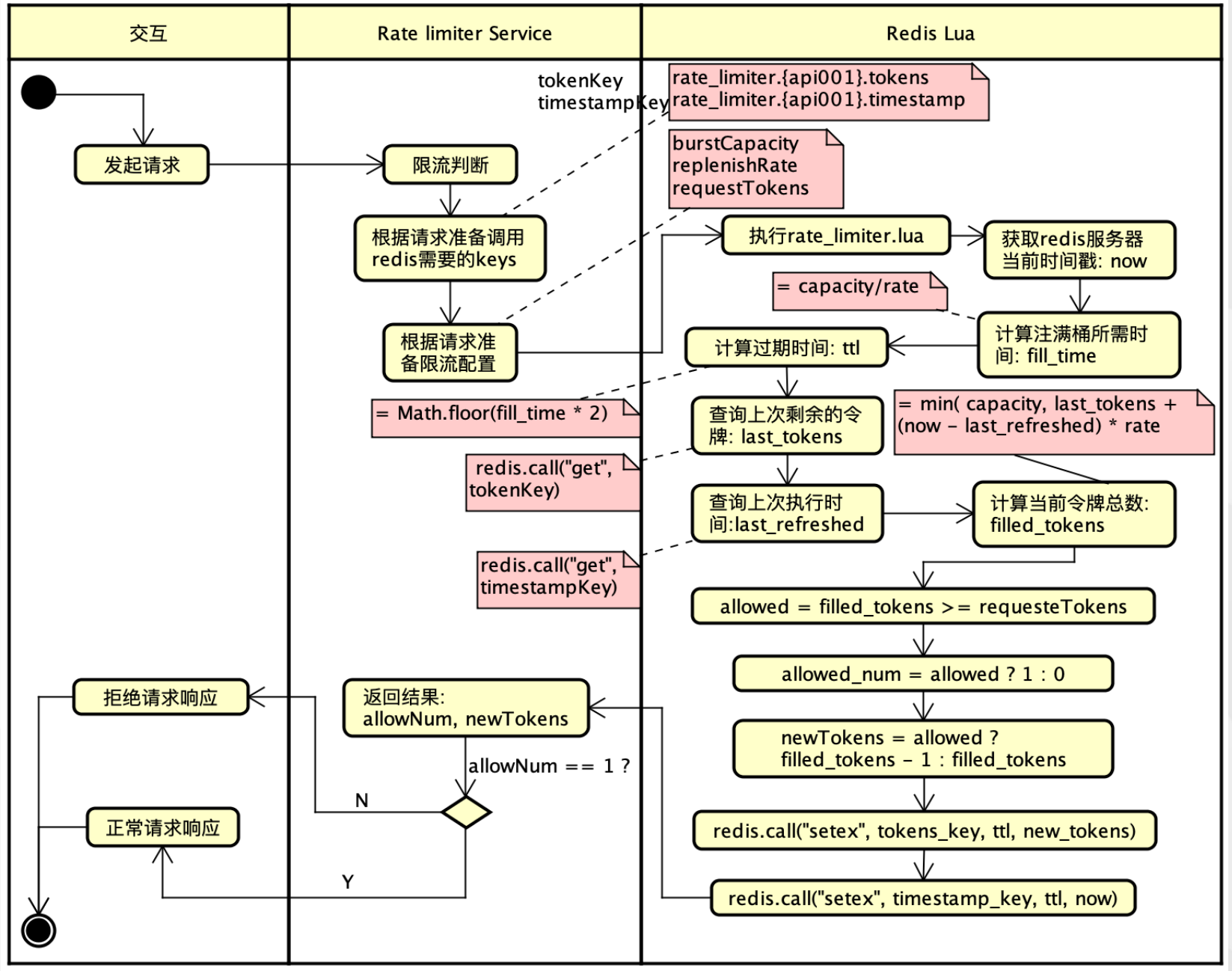
- 两大限流能力
  - 单机限流能力（LRT）
    - ✓ 保护自身的最大流量
    - ✓ sidecar(envoy) 的能力
  - 分布式限流能力（GRT）
    - ✓ 保护下流服务(upstream services)
    - ✓ 利用Redis作为限流实时状态存储
    - ✓ 基于令牌桶算法(*burstCapacity, replenishRate*)
    - ✓ 支持 api 粒度的限流

咕泡科技 GUPAO TECH

实战三：分布式限流

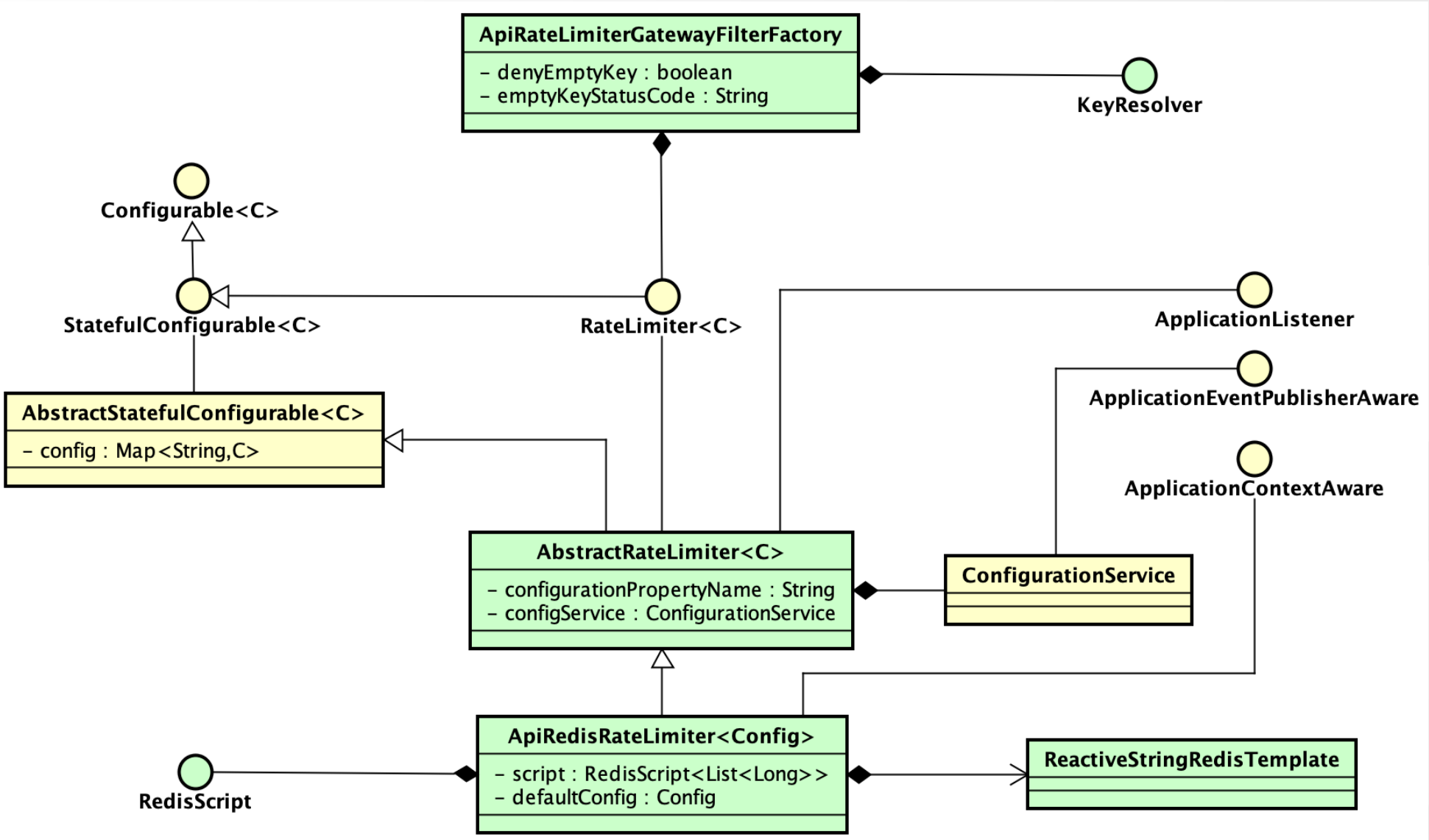
- 详细流程
  - 主从 v.s 集群
    - ✓ 主从要注意用哨兵模式提高可用性
    - ✓ ReadFrom.MASTER_PREFERRRED
  - key
    - ✓ rate_limiter.{key}.tokens
    - ✓ rate_limiter.{key}.timestamp
  - Lua

    rate_limiter.lua
  - 限流粒度: apiCode



实战三：分布式限流

- 类图设计



实战三：分布式限流

- 实战代码

代码演示

rate-limiter

咕泡科技
GUPAO TECH

谢谢观赏

GUPAO TECH

我们的愿景
推动每一次人才升级

我们的使命
让每个人的职业生涯不留遗憾