Redis设计规范

（1.0.0）

北京时代亿信科技股份有限公司

2021年08月

|  |  |  |  |
| --- | --- | --- | --- |
| **文档编号** | **V1.0.0** | **文档名称** | **Redis设计规范** |
| **编写人** | 王志鹏 | **编写日期** | 2021-08-25 |
| **审核人** |  | **审核日期** |  |
| **批准人** |  | **批准日期** |  |
| **修改人** | **修改日期** | **修改内容** | |
| **王志鹏** | 2021-08-25 | **编写规范初稿** | |
|  |  |  | |

目录

[第一章 引言 4](#_Toc10731)

[1.1. 目的 4](#_Toc16608)

[1.2. 使用范围 4](#_Toc10259)

[1.3. 相关术语 4](#_Toc31283)

[第二章 键/值设计 5](#_Toc2264)

[2.1. 键名设计 5](#_Toc14670)

[2.2. 键值设计 5](#_Toc4209)

[第三章 命令使用 7](#_Toc6249)

[第四章 字段设计规范 9](#_Toc17293)

[第五章 索引设计规范 10](#_Toc30570)

[第六章 SQL开发规范 11](#_Toc166)

[第七章 操作行为规范 11](#_Toc26727)

[第八章 参考资料 12](#_Toc4341)

# 引言

## 目的

。

## 使用范围

技术经理

基础平台研发工程师

产品研发工程师

项目研发工程师

质量保证人员

## **相关术语**

|  |  |
| --- | --- |
| **术语** | **解释** |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |
|  |  |

# 键/值设计

## 键名设计

以下所有规范按照**[高危]、[强制]、[建议]**三个级别进行标注，遵守从高到低的优先级。

1. **[建议] 可读性和可管理性**

以业务名（或数据库名）为前缀（防止Key名冲突），用冒号分隔，例如：*业务名:实体名:id*。

1. **[建议] 简洁性**

保证语义的前提下，控制Key名的长度，当Key名较长时，内存占用也不容忽视，例如：*user:{uid}:friends:messages:{mid}* -> *u:{uid}:fr:m:{mid}*。

1. **[强制] 不要包含特殊字符**

反例：包含空格、换行、单双引号以及其他转义字符。

## 键值设计

1. **[强制]** 拒绝bigkey（防止网卡流量、慢查询）

String类型控制在10KB以内，Hash、List、Set和ZSet等元素个数不要超过5000个。

反例：一个包含200万个元素的List。

非字符串的bigkey，不要使用del命令删除，使用hscan、sscan、zscan等命令渐进式删除，同时要注意防止bigkey过期时间自动删除问题。

1. **[推荐]** 选择适合的数据类型

合理控制和使用数据结构内存编码优化配置，如ziplist，同时注意节省内存和性能之间的平衡。

正例：*hmset user:1 name tom age 19 favor football*

1. **[推荐]** 控制键的生命周期，Redis不是垃圾桶

建议使用expire设置过期时间（条件允许可以使用过期时间+随机值方式，防止集中过期），不过期的数据重点关注idletime。

# 命令使用

1. **[推荐]** O(N)命令关注N的数量

对于时间复杂度为O(N)的命令，例如：*hgetall、lrange、smembers、zrange、sinter*等命令，尽量避免使用，如果使用也要明确N的值。

如有遍历键值的需求，可以使用*hscan、sscan、zscan*代替。

1. **[推荐]** 禁用命令

禁止线上使用keys、flushall、flushdb等命令，通过Redis的rename机制禁用命令，或者使用scan方式渐进处理。

1. **[推荐]** 合理使用select

Redis的多数据库较弱，使用数字进行区分，很多客户端支持较差，同时多业务使用多数据库还是单线程处理，会有干扰。

1. **[推荐]** 使用批量操作提高效率

使用原生命令（mget、mset等）或pipeline来批量操作数据，但要注意控制一次批量操作的元素个数不能过多。

1. **[建议]** Redis事务功能较弱，不建议过多使用

Redis的事务功能较弱（不支持回滚），而且集群版本要求一次事务操作的key必须在一个slot上（可以使用hashtag功能解决）。

1. **[建议]** Redis集群版本在使用Lua上有特殊要求：

* 所有Key都由KEYS数组来传递，redis.call/pcall里面调用的redis命令，key参数都必须是KEYS数组，否则直接返回错误（“-ERR bad lua script for redis cluster, all the keys that the script uses should be passed using the KEYS array”）；
* 所有Key必须在一个slot上，否则直接返回错误（“-ERR eval/evalsha command keys must in same slot”）。

1. **[建议]** 必要情况下可以使用monitor命令，注意不要长时间使用

# 客户端使用

1. **[推荐]** 避免多个应用使用同一个Redis实例

正例：不相干的业务拆分，公共数据做服务化。

1. **[推荐]** 使用带有连接池的客户端，有效控制连接和提高访问效率
2. **[建议]** 高并发下建议客户端添加熔断功能
3. **[推荐]** 设置合理的密码，如有必要可以使用SSL加密访问
4. **[建议]** 根据业务类型不同，选择最大内存淘汰策略，设置好过期时间

最大内存淘汰策略（maxmemory-policy）缺省为volatile-lru，即超过最大内存后，在过期键中使用LRU算法剔除Key，保证不过期的数据不被删除，但可能会出现OOM问题。

其他淘汰策略有：

* allkeys-lru：根据LRU算法删除键，不管数据是否设置超时属性，直至腾出足够空间为止；
* allkeys-random：随机删除所有键，直至腾出足够空间为止；
* volatile-random：随机删除过期键，直至腾出足够空间为止；
* volatile-ttl：根据键值对象的TTL属性，删除最近将要过期数据，如果没有则回退到noeviction策略；
* noeviction：不会剔除任何数据，拒绝所有写入操作并返回客户端错误信息（"(error) OOM command not allowed when used memory"），此时Redis只响应读取操作。

# Redis最佳实践

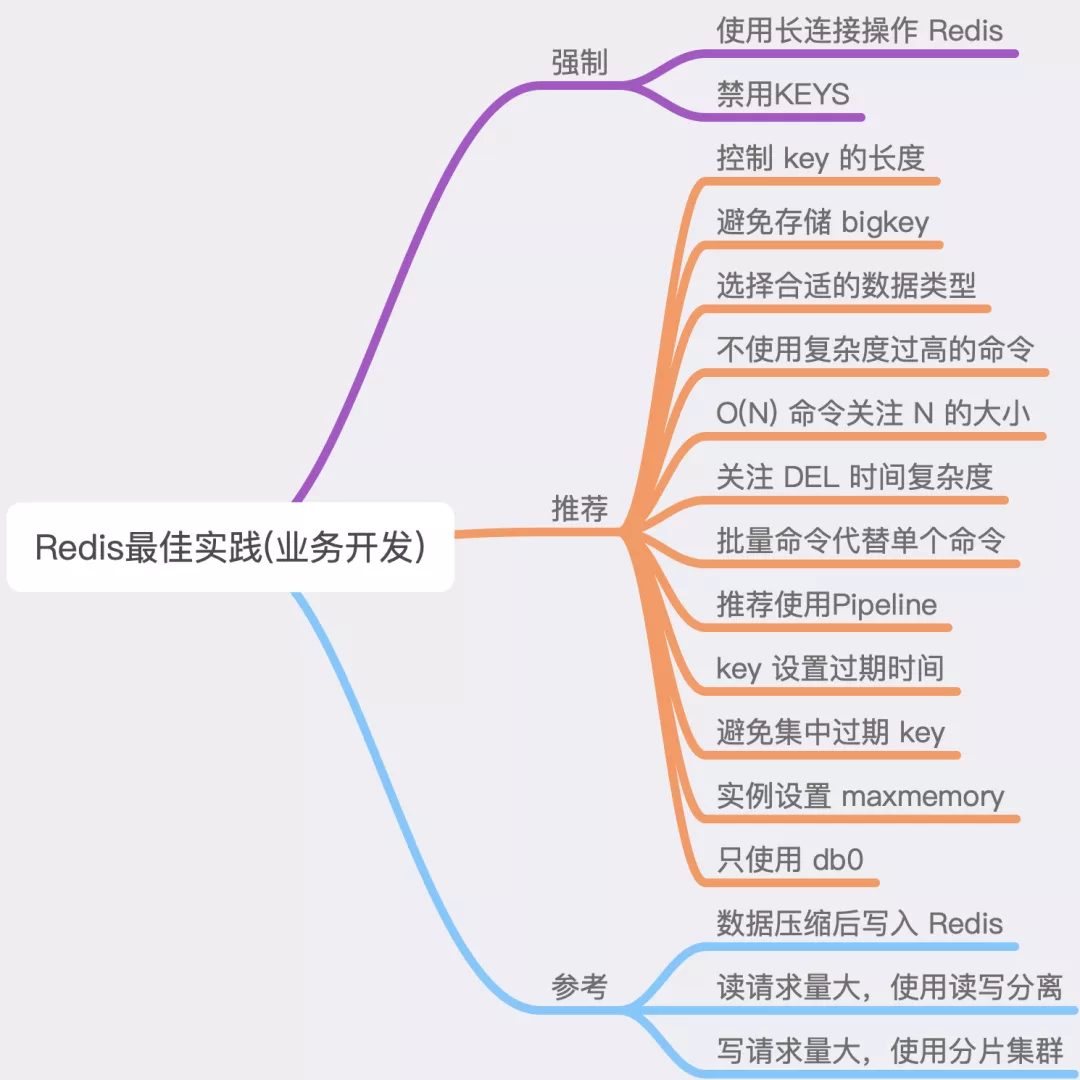
按照“业务开发”和“运维”2个维度，对于Redis最佳实践进行划分，并且分别以「强制」、「推荐」、「参考」3 个级别进行标注，指导相关人员的工作。

3个级别的实施规则分别为：

* 强制：需严格遵守，否则危害极大
* 推荐：尽量按推荐方式执行，可提升性能、降低内存、方便运维等
* 参考：根据业务特点采纳和参考

## 业务开发

业务人员需要了解Redis的运行机制（命令时间复杂度、数据过期策略、数据淘汰策略等），使用合理的命令，并结合业务场景进行优化。



## 系统运维

系统运维人员需要未雨绸缪，在资源规划、运维、监控和安全层面做到位。

