云数据库 Redis

最佳实践

产品文档

腾讯云

版权所有：腾讯云计算（北京）有限责任公司

【版权声明】

©2013-2024 腾讯云版权所有

本文档著作权归腾讯云单独所有，未经腾讯云事先书面许可，任何主体不得以任何形式复制、修改、抄袭、传播全部或部分本文档内容。

【商标声明】

及其它腾讯云服务相关的商标均为腾讯云计算（北京）有限责任公司及其关联公司所有。本文档涉及的第三方主体的商标，依法由权利人所有。

【服务声明】

本文档意在向客户介绍腾讯云全部或部分产品、服务的当时的整体概况，部分产品、服务的内容可能有所调整。您所购买的腾讯云产品、服务的种类、服务标准等应由您与腾讯云之间的商业合同约定，除非双方另有约定，否则，腾讯云对本文档内容不做任何明示或默示的承诺或保证。

文档目录

最佳实践
热 Key 与 大 key

版权所有：腾讯云计算（北京）有限责任公司

最佳实践

热 Key 与 大 key

最近更新时间：2024-01-26 15:22:24

日常业务运行过程中，Redis 实例经常因各种 Big keys（下文称为“大 Key”）或 Hotkeys（下文称为“热 Key”）的问题未及时处理，导致服务性能下降、访问超时、用户体验变差，甚至可能造成实例大范围故障。本文针对大 Key 与热 Key 产生的原因、排查方法及其优化解决方案进行详细说明。

基本定义

大 Key 定义

大 Key 具体表现为 Redis 中的 Key 对应的 Value 很大，占用 Redis 空间比较大，本质上是大 Value 问题。对于 Redis 中不同的数据结构类型，常见示例如下所示。
对于 String 类型的 Value 值，值超过 10MB（数据值太大）。
对于 Set 类型的 Value 值，含有的成员数量为 10000 个（成员数量多）。
对于 List 类型的 Value 值，含有的成员数量为 10000 个（成员数量多）。
对于 Hash 格式的 Value 值，含有的成员数量 1000 个，但所有成员变量的总 Value 值大小为 1000MB（成员总的体积过大）。

热 Key 定义

热 Key 指一段时间内某个 Key 的访问量远远超过其他的 Key，导致大量访问流量 QPS 集中在某一个 Redis 实例中的特定 Key，或者是带宽使用率集中在特定的 Key，又或者是 CPU 使用占比集中在特定的 Key。常见示例如下所示。
Redis 实例的总请求 QPS（每秒命令执行次数）为10,000，而其中一个 Key 的每秒访问量达到7,000。
一个包含 2000 个 Field 的 Hash 格式的 Key，每秒发送大量的 **hgetall** 操作请求
一个包含 10000 个 Field 的 Key 每秒发送大量的 **zrange** 操作请求。

现象与影响

大 Key

内存使用不均匀
在 Redis 集群架构中，某个数据分片的内存使用率远超其他数据分片，内存资源无法达到均衡。另外，Redis 内存可能达到 maxmemory 参数定义的上限，导致重要的 Key 被逐出，甚至引发内存溢出。
请求响应时间上升，超时阻塞
Redis 是单线程架构，操作大 Key 耗时较长，可能造成请求阻塞。

同步中断或主从切换
内存不足时，对大 Key 进行驱逐操作或者 **rename** 一个大 Key，容易长时间阻塞主库，进而可能引发同步中断或主从切换。
网络拥塞
一个大 Key 占用空间是1MB，每秒访问1000次，就有1000MB的流量，可能造成实例或局域网的带宽被占满，自身服务变慢，同时影响其他服务。

热Key

实例性能指标 CPU 使用率持续偏高，整体服务性能降低。
集群架构下，请求分配不均，存在热 key 的节点面临较大的访问压力，可能出现该数据分片的连接数被耗尽甚至宕机。（即使采取扩容也会对资源有很大的浪费）。
热点缓存流量高度集中，超出 Redis 的承受能力，易造成缓存与数据库被击穿，从而引发系统雪崩。

原因分析

大 Key 分析

Redis 中的 key-value 键值对设置不当，如使用 String 类型的 Key 存放大体积二进制文件型数据；造成 key 对应的 value 值特别大。
对于 list，set 这种类型的结构，无效的数据没有及时的清理。造成 Key 中的成员持续不断地增加。
业务上线前，对业务分析不准确，没有对 Key 中的成员进行合理的拆分，造成个别 Key 中的成员数量过多。

热 Key 分析

预期之外的访问量陡增，如突然出现的爆款商品、访问量暴涨的热点新闻、直播间某主播搞活动带来的大量刷屏点赞、游戏中某区域发生多个工会之间的战斗涉及大量玩家等。

排查方法

云数据库 Redis 接入了数据库智能管家（TencentDB for DBbrain，DBbrain）的诊断优化功能，可以辅助您快速查找数据库中存在的大 Key，及其热 Key。
大 Key 排查详细操作，请参见 内存分析。
热 Key 排查详细操作，请参见 热 Key 分析。

解决方法

大 Key

1. 清理无效的数据

主要针对 list 和 set 这种类型，在使用的过程中，list 和 set 中对应的内容不断增加，但是由于之前存储的已经是无效的了，需要定时的对 list 和 set 进行清理。

2. 压缩对应的大 Key 的 Value

通过序列化或者压缩的方法对 value 进行压缩，是其变为较小的 value，但是如果压缩之后如果对应的 value 还是特别大的话，就需要使用拆分的方法进行解决。

3. 针对大 Key 进行拆分

通过将 BigKey 拆分成多个小 Key 的键值对，并且拆分后的对应的 value 大小和拆分成的成员数量比较合理，然后进行存储即可，在获取的时候通过 get 不同的 key 或是用 mget 批量获取存储的键值对。

4. 实时监控 Redis 内存、带宽及 Key 增长变化趋势

通过监控系统，监控 Redis 中的内存占用大小和网络带宽的占用大小，以及固定时间内的内存占用增长率，当超过设定的阈值的时候，进行报警通知处理。监控指标的具体信息，请参见 监控功能（5秒粒度）。设置阈值告警，具体操作，可参见 配置告警。

热 Key

可以使用使用读写分离架构，如果热 Key 的产生来自于读请求，那么读写分离是一个很好的解决方案。在使用读写分离架构时，可以通过不断的增加从节点来降低每个 Redis 实例中的读请求压力。具体信息，请参见 开关读写分离。