



下载APP



19 | 波动的响应延迟：如何应对变慢的Redis？（下）

2020-09-21 蒋德钧

Redis核心技术与实战

[进入课程 >](#)



讲述：蒋德钧

时长 16:49 大小 15.41M



你好，我是蒋德钧。

上节课，我介绍了判断 Redis 变慢的两种方法，分别是响应延迟和基线性能。除此之外，我还给你分享了从 Redis 的自身命令操作层面排查和解决问题的两种方案。

但是，如果在排查时，你发现 Redis 没有执行大量的慢查询命令，也没有同时删除大量过期 keys，那么，我们是不是就束手无策了呢？

当然不是！我还有很多“锦囊妙计”，准备在这节课分享给你呢！

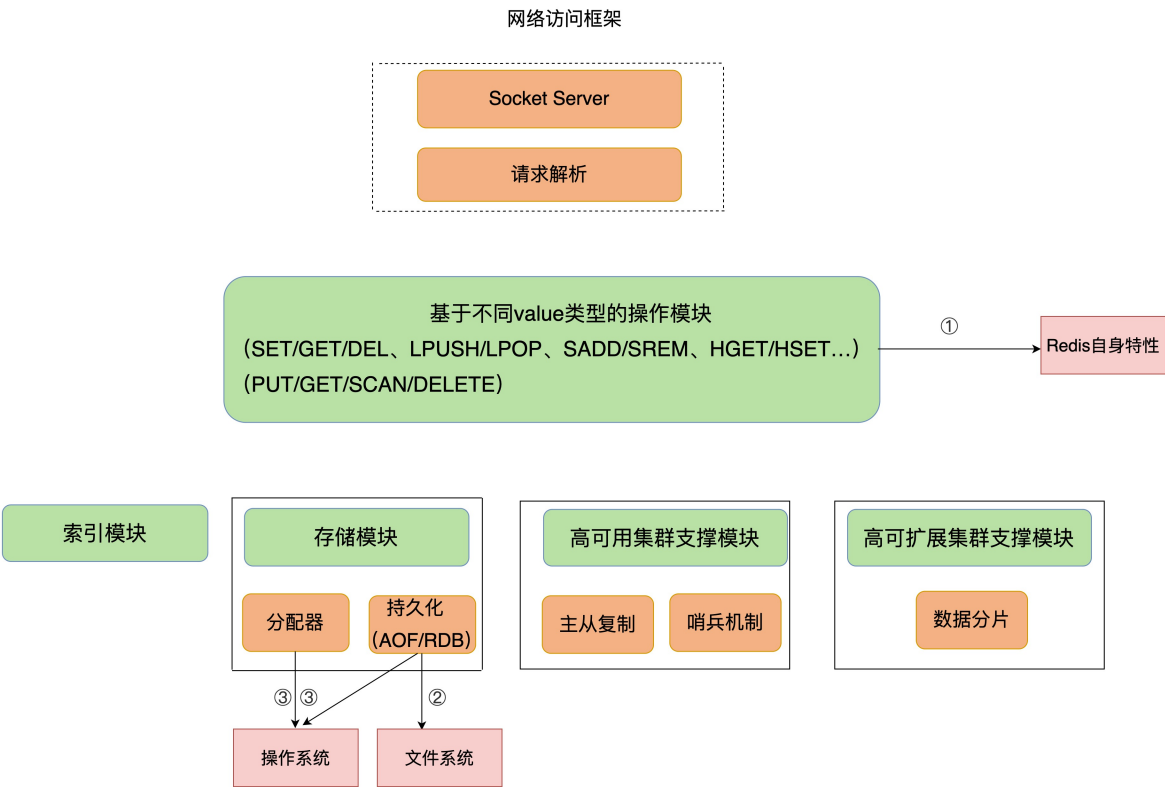


如果上节课的方法不管用，那就说明，你要关注影响性能的其他机制了，也就是文件系统和操作系统。

Redis 会持久化保存数据到磁盘，这个过程要依赖文件系统来完成，所以，文件系统将数据写回磁盘的机制，会直接影响到 Redis 持久化的效率。而且，在持久化的过程中，Redis 也还在接收其他请求，持久化的效率高低又会影响到 Redis 处理请求的性能。

另一方面，Redis 是内存数据库，内存操作非常频繁，所以，操作系统的内存机制会直接影响到 Redis 的处理效率。比如说，如果 Redis 的内存不够用了，操作系统会启动 swap 机制，这就会直接拖慢 Redis。

那么，接下来，我再从这两个层面，继续给你介绍，如何进一步解决 Redis 变慢的问题。



文件系统：AOF 模式

你可能会问，Redis 是个内存数据库，为什么它的性能还和文件系统有关呢？

我在前面讲过，为了保证数据可靠性，Redis 会采用 AOF 日志或 RDB 快照。其中，AOF 日志提供了三种日志写回策略：no、everysec、always。这三种写回策略依赖文件系统的两个系统调用完成，也就是 write 和 fsync。

write 只要把日志记录写到内核缓冲区，就可以返回了，并不需要等待日志实际写回到磁盘；而 fsync 需要把日志记录写回到磁盘后才能返回，时间较长。下面这张表展示了三种写回策略所执行的系统调用。

AOF写回策略	执行的系统调用
no	调用write写日志文件，由操作系统周期性地将日志写回磁盘
everysec	每秒调用一次fsync，将日志写回磁盘
always	每执行一个操作，就调用一次fsync将日志写回磁盘

当写回策略配置为 everysec 和 always 时，Redis 需要调用 fsync 把日志写回磁盘。但是，这两种写回策略的具体执行情况还不太一样。

在使用 everysec 时，Redis 允许丢失一秒的操作记录，所以，Redis 主线程并不需要确保每个操作记录日志都写回磁盘。而且，fsync 的执行时间很长，如果是在 Redis 主线程中执行 fsync，就容易阻塞主线程。所以，当写回策略配置为 everysec 时，Redis 会使用后台的子线程异步完成 fsync 的操作。

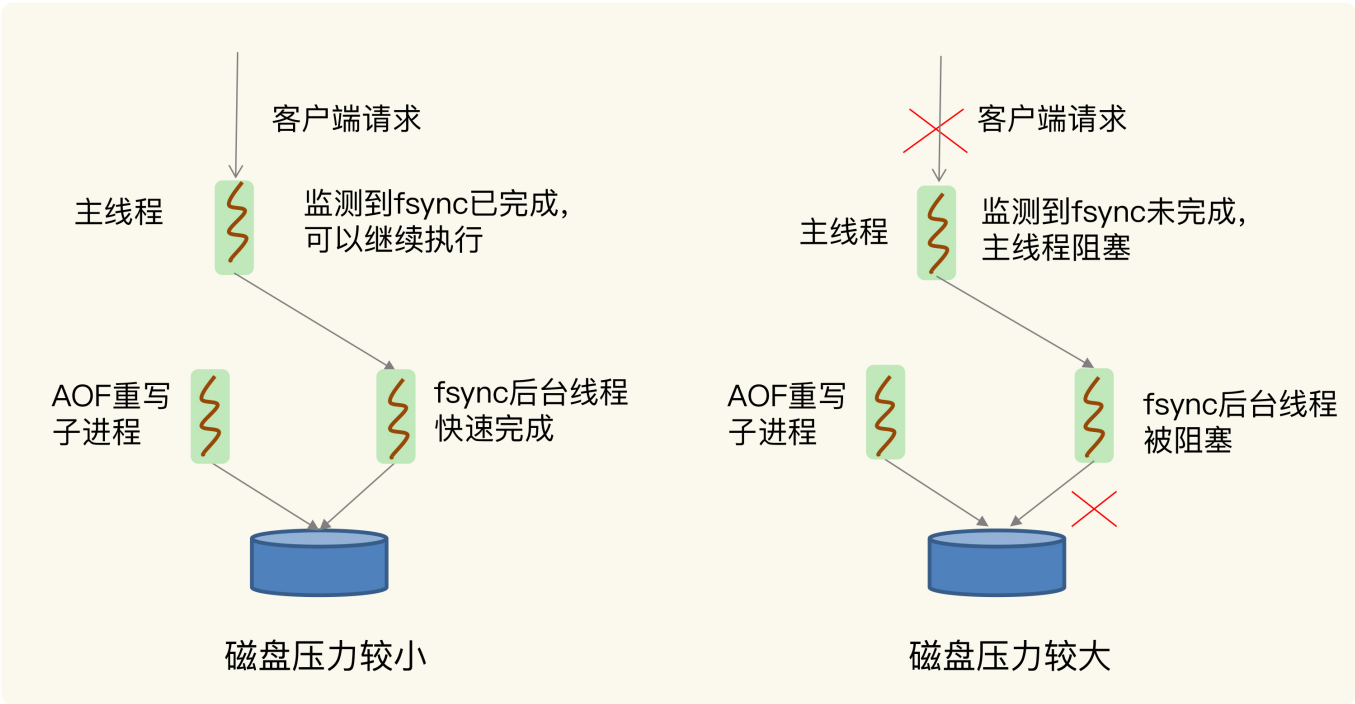
而对于 always 策略来说，Redis 需要确保每个操作记录日志都写回磁盘，如果用后台子线程异步完成，主线程就无法及时地知道每个操作是否已经完成了，这就不符合 always 策略的要求了。所以，always 策略并不使用后台子线程来执行。

另外，在使用 AOF 日志时，为了避免日志文件不断增大，Redis 会执行 AOF 重写，生成体量缩小的新的 AOF 日志文件。AOF 重写本身需要的时间很长，也容易阻塞 Redis 主线程，所以，Redis 使用子进程来进行 AOF 重写。

但是，这里有一个潜在的风险点：AOF 重写会对磁盘进行大量 IO 操作，同时，fsync 又需要等到数据写到磁盘后才能返回，所以，当 AOF 重写的压力比较大时，就会导致 fsync 被阻塞。虽然 fsync 是由后台子线程负责执行的，但是，主线程会监控 fsync 的执行进度。

当主线程使用后台子线程执行了一次 fsync，需要再次把新接收的操作记录写回磁盘时，如果主线程发现上一次的 fsync 还没有执行完，那么它就会阻塞。所以，如果后台子线程执行的 fsync 频繁阻塞的话（比如 AOF 重写占用了大量的磁盘 IO 带宽），主线程也会阻塞，导致 Redis 性能变慢。

为了帮助你理解，我再画一张图来展示下在磁盘压力小和压力大的时候，fsync 后台子线程和主线程受到的影响。



好了，说到这里，你已经了解了，由于 fsync 后台子线程和 AOF 重写子进程的存在，主 IO 线程一般不会被阻塞。但是，如果在重写日志时，AOF 重写子进程的写入量比较大，fsync 线程也会被阻塞，进而阻塞主线程，导致延迟增加。现在，我来给出排查和解决建议。

首先，你可以检查下 Redis 配置文件中的 appendfsync 配置项，该配置项的取值表明了 Redis 实例使用的是哪种 AOF 日志写回策略，如下所示：

写回策略	配置项
no	appendfsync no
everysec	appendfsync everysec
always	appendfsync always

如果 AOF 写回策略使用了 everysec 或 always 配置，请先确认下业务方对数据可靠性的要求，明确是否需要每一秒或每一个操作都记日志。有的业务方不了解 Redis AOF 机制，很可能就直接使用数据可靠性最高等级的 always 配置了。其实，在有些场景中（例如 Redis 用于缓存），数据丢了还可以从后端数据库中获取，并不需要很高的数据可靠性。

如果业务应用对延迟非常敏感，但同时允许一定量的数据丢失，那么，可以把配置项 `no-appendfsync-on-rewrite` 设置为 `yes`，如下所示：

[复制代码](#)

```
1 no-appendfsync-on-rewrite yes
```

这个配置项设置为 `yes` 时，表示在 AOF 重写时，不进行 `fsync` 操作。也就是说，Redis 实例把写命令写到内存后，不调用后台线程进行 `fsync` 操作，就可以直接返回了。当然，如果此时实例发生宕机，就会导致数据丢失。反之，如果这个配置项设置为 `no`（也是默认配置），在 AOF 重写时，Redis 实例仍然会调用后台线程进行 `fsync` 操作，这就会给实例带来阻塞。

如果的确需要高性能，同时也需要高可靠数据保证，我建议你考虑**采用高速的固态硬盘作为 AOF 日志的写入设备**。

高速固态盘的带宽和并发度比传统的机械硬盘的要高出 10 倍及以上。在 AOF 重写和 `fsync` 后台线程同时执行时，固态硬盘可以提供较为充足的磁盘 IO 资源，让 AOF 重写和 `fsync` 后台线程的磁盘 IO 资源竞争减少，从而降低对 Redis 的性能影响。

操作系统：swap

如果 Redis 的 AOF 日志配置只是 `no`，或者就没有采用 AOF 模式，那么，还会有什么问题导致性能变慢吗？

接下来，我就再说一个潜在的瓶颈：**操作系统的内存 swap**。

内存 swap 是操作系统里将内存数据在内存和磁盘间来回换入和换出的机制，涉及到磁盘的读写，所以，一旦触发 swap，无论是被换入数据的进程，还是被换出数据的进程，其性能都会受到慢速磁盘读写的影响。

Redis 是内存数据库，内存使用量大，如果没有控制好内存的使用量，或者和其他内存需求大的应用一起运行了，就可能受到 swap 的影响，而导致性能变慢。

这一点对于 Redis 内存数据库而言，显得更为重要：正常情况下，Redis 的操作是直接通过访问内存就能完成，一旦 swap 被触发了，Redis 的请求操作需要等到磁盘数据读写完成才行。而且，和我刚才说的 AOF 日志文件读写使用 fsync 线程不同，swap 触发后影响的是 Redis 主 IO 线程，这会极大地增加 Redis 的响应时间。

说到这儿，我想给你分享一个我曾经遇到过的因为 swap 而导致性能降低的例子。

在正常情况下，我们运行的一个实例完成 5000 万个 GET 请求时需要 300s，但是，有一次，这个实例完成 5000 万 GET 请求，花了将近 4 个小时的时间。经过问题复现，我们发现，当时 Redis 处理请求用了近 4 小时的情况下，该实例所在的机器已经发生了 swap。从 300s 到 4 个小时，延迟增加了将近 48 倍，可以看到 swap 对性能造成的严重影响。

那么，什么时候会触发 swap 呢？

通常，触发 swap 的原因主要是**物理机器内存不足**，对于 Redis 而言，有两种常见的情况：


Redis 实例自身使用了大量的内存，导致物理机器的可用内存不足；

和 Redis 实例在同一台机器上运行的其他进程，在进行大量的文件读写操作。文件读写本身会占用系统内存，这会导致分配给 Redis 实例的内存量变少，进而触发 Redis 发生 swap。


针对这个问题，我也给你提供一个解决思路：**增加机器的内存或者使用 Redis 集群**。

操作系统本身会在后台记录每个进程的 swap 使用情况，即有多少数据量发生了 swap。你可以先通过下面的命令查看 Redis 的进程号，这里是 5332。

```
1 $ redis-cli info | grep process_id
2 process_id: 5332
```

 复制代码

然后，进入 Redis 所在机器的 /proc 目录下的该进程目录中：

 复制代码

```
1 $ cd /proc/5332
```

最后，运行下面的命令，查看该 Redis 进程的使用情况。在这儿，我只截取了部分结果：

[复制代码](#)

```
1 $cat smaps | egrep '^(Swap|Size)'  
2 Size: 584 kB  
3 Swap: 0 kB  
4 Size: 4 kB  
5 Swap: 4 kB  
6 Size: 4 kB  
7 Swap: 0 kB  
8 Size: 462044 kB  
9 Swap: 462008 kB  
10 Size: 21392 kB  
11 Swap: 0 kB
```

每一行 Size 表示的是 Redis 实例所用的一块内存大小，而 Size 下方的 Swap 和它相对应，表示这块 Size 大小的内存区域有多少已经被换出到磁盘上了。如果这两个值相等，就表示这块内存区域已经完全被换出到磁盘了。

作为内存数据库，Redis 本身会使用很多大小不一的内存块，所以，你可以看到有很多 Size 行，有的很小，就是 4KB，而有的很大，例如 462044KB。**不同内存块被换出到磁盘上的大小也不一样**，例如刚刚的结果中的第一个 4KB 内存块，它下方的 Swap 也是 4KB，这表示这个内存块已经被换出了；另外，462044KB 这个内存块也被换出了 462008KB，差不多有 462MB。

这里有个重要的地方，我得提醒你一下，当出现百 MB，甚至 GB 级别的 swap 大小时，就表明，此时，Redis 实例的内存压力很大，很有可能会变慢。所以，swap 的大小是排查 Redis 性能变慢是否由 swap 引起的重要指标。

一旦发生内存 swap，最直接的解决方法就是**增加机器内存**。如果该实例在一个 Redis 切片集群中，可以增加 Redis 集群的实例个数，来分摊每个实例服务的数据量，进而减少每个实例所需的内存量。

当然，如果 Redis 实例和其他操作大量文件的程序（例如数据分析程序）共享机器，你可以将 Redis 实例迁移到单独的机器上运行，以满足它的内存需求量。如果该实例正好是

Redis 主从集群中的主库，而从库的内存很大，也可以考虑进行主从切换，把大内存的从库变成主库，由它来处理客户端请求。

操作系统：内存大页

除了内存 swap，还有一个和内存相关的因素，即内存大页机制（Transparent Huge Page, THP），也会影响 Redis 性能。

Linux 内核从 2.6.38 开始支持内存大页机制，该机制支持 2MB 大小的内存页分配，而常规的内存页分配是按 4KB 的粒度来执行的。

很多人都觉得：“Redis 是内存数据库，内存大页不正好可以满足 Redis 的需求吗？而且在分配相同的内存量时，内存大页还能减少分配次数，不也是对 Redis 友好吗？”

其实，系统的设计通常是一个取舍过程，我们称之为 trade-off。很多机制通常都是优势和劣势并存的。Redis 使用内存大页就是一个典型的例子。


虽然内存大页可以给 Redis 带来内存分配方面的收益，但是，不要忘了，Redis 为了提供数据可靠性保证，需要将数据做持久化保存。这个写入过程由额外的线程执行，所以，此时，Redis 主线程仍然可以接收客户端写请求。客户端的写请求可能会修改正在进行持久化的数据。在这一过程中，Redis 就会采用写时复制机制，也就是说，一旦有数据要被修改，Redis 并不会直接修改内存中的数据，而是将这些数据拷贝一份，然后再进行修改。

如果采用了内存大页，那么，即使客户端请求只修改 100B 的数据，Redis 也需要拷贝 2MB 的大页。相反，如果是常规内存页机制，只用拷贝 4KB。两者相比，你可以看到，当客户端请求修改或新写入数据较多时，内存大页机制将导致大量的拷贝，这就会影响 Redis 正常的访存操作，最终导致性能变慢。

那该怎么办呢？很简单，关闭内存大页，就行了。

首先，我们要先排查下内存大页。方法是：在 Redis 实例运行的机器上执行如下命令：

```
1 cat /sys/kernel/mm/transparent_hugepage/enabled
```

 复制代码

如果执行结果是 `always`，就表明内存大页机制被启动了；如果是 `never`，就表示，内存大页机制被禁止。

在实际生产环境中部署时，我建议你不要使用内存大页机制，操作也很简单，只需要执行下面的命令就可以了：

```
1 echo never /sys/kernel/mm/transparent_hugepage/enabled
```

[复制代码](#)

小结

这节课，我从文件系统和操作系统两个维度，给你介绍了应对 Redis 变慢的方法。

为了方便你应用，我给你梳理了一个包含 9 个检查点的 Checklist，希望你在遇到 Redis 性能变慢时，按照这些步骤逐一检查，高效地解决问题。

1. 获取 Redis 实例在当前环境下的基线性能。
2. 是否用了慢查询命令？如果是的话，就使用其他命令替代慢查询命令，或者把聚合计算命令放在客户端做。
3. 是否对过期 key 设置了相同的过期时间？对于批量删除的 key，可以在每个 key 的过期时间上加一个随机数，避免同时删除。
4. 是否存在 bigkey？对于 bigkey 的删除操作，如果你的 Redis 是 4.0 及以上的版本，可以直接利用异步线程机制减少主线程阻塞；如果是 Redis 4.0 以前的版本，可以使用 SCAN 命令迭代删除；对于 bigkey 的集合查询和聚合操作，可以使用 SCAN 命令在客户端完成。
5. Redis AOF 配置级别是什么？业务层面是否的确需要这一可靠性级别？如果我们需要高性能，同时也允许数据丢失，可以将配置项 `no-appendfsync-on-rewrite` 设置为 `yes`，避免 AOF 重写和 `fsync` 竞争磁盘 IO 资源，导致 Redis 延迟增加。当然，如果既需要高性能又需要高可靠性，最好使用高速固态盘作为 AOF 日志的写入盘。
6. Redis 实例的内存使用是否过大？发生 swap 了吗？如果是的话，就增加机器内存，或者是使用 Redis 集群，分摊单机 Redis 的键值对数量和内存压力。同时，要避免出现 Redis 和其他内存需求大的应用共享机器的情况。

7. 在 Redis 实例的运行环境中，是否启用了透明大页机制？如果是的话，直接关闭内存大页机制就行了。
8. 是否运行了 Redis 主从集群？如果是的话，把主库实例的数据量大小控制在 2~4GB，以免主从复制时，从库因加载大的 RDB 文件而阻塞。
9. 是否使用了多核 CPU 或 NUMA 架构的机器运行 Redis 实例？使用多核 CPU 时，可以给 Redis 实例绑定物理核；使用 NUMA 架构时，注意把 Redis 实例和网络中断处理程序运行在同一个 CPU Socket 上。

实际上，影响系统性能的因素还有很多，这两节课给你讲的都是应对最常见问题的解决方案。

如果你遇到了一些特殊情况，也不要慌，我再给你分享一个小技巧：仔细检查下有没有恼人的“邻居”，具体点说，就是 Redis 所在的机器上有没有一些其他占内存、磁盘 IO 和网络 IO 的程序，比如说数据库程序或者数据采集程序。如果有的话，我建议你将这些程序迁移到其他机器上运行。

为了保证 Redis 高性能，我们需要给 Redis 充足的计算、内存和 IO 资源，给它提供一个“安静”的环境。

每课一问

这两节课，我向你介绍了系统性定位、排查和解决 Redis 变慢的方法。所以，我想请你聊一聊，你遇到过 Redis 变慢的情况吗？如果有的话，你是怎么解决的呢？

欢迎你在留言区分享一下自己的经验，如果觉得今天的内容对你有所帮助，也欢迎分享给你的朋友或同事，我们下节课见。

提建议

更多课程推荐

数据结构与算法之美

为工程师量身打造的数据结构与算法私教课

王争

前 Google 工程师



立省¥40

破90000订阅特惠，到手价¥89

© 版权归极客邦科技所有，未经许可不得传播售卖。页面已增加防盗追踪，如有侵权极客邦将依法追究其法律责任。

上一篇 18 | 波动的响应延迟：如何应对变慢的Redis？（上）

下一篇 20 | 删除数据后，为什么内存占用率还是很高？

精选留言 (13)

写留言



Kaito

2020-09-21

关于如何分析、排查、解决Redis变慢问题，我总结的checklist如下：

1、使用复杂度过高的命令（例如SORT/SUION/ZUNIONSTORE/KEYS），或一次查询全量数据（例如LRANGE key 0 N，但N很大）

...

展开

4

55



子华宝宝萌萌哒

2020-09-21

```
echo never /sys/kernel/mm/transparent_hugepage/enabled
```

这个是不是得改成

```
echo never > /sys/kernel/mm/transparent_hugepage/enabled
```

这样？

展开 ∨



1



3



yeek

2020-09-21

如果redis是独立部署，那么当内存不足时，淘汰策略和操作系统的swap机制 哪个会优先执行？

曾遇到过线上触发内存淘汰的场景，并未观察当时的swap情况，感谢老师的建议

展开 ∨



1



2



王世艺

2020-09-23

看了下，貌似是这样

redis 有一个aof_buf缓存,用来缓存新的aof操作信息。

正常情况下主线程每次循环都是先将aof_buf write，然后aof_buf是清空，主线程会用去fsync，已经write的内容。

刷盘当时always的情况下，主线程去直接调用redis_fsync。...

展开 ∨



1



东

2020-09-21

“8. 是否运行了 Redis 主从集群？如果是的话，把主库实例的数据量大小控制在 2~4GB，以免主从复制时，从库因加载大的 RDB 文件而阻塞。” 这个2~4G的经验值和主库的内存大小无关吗？比如主从库内存都是64G，这个主库数据量依然是2~4G？

展开 ∨



1



1



小喵喵

2020-09-24

有时候需要内存大页怎么办？如何判断是否需要启用内存大页机制呢？



小喵喵



2020-09-24

请问下老师什么样的key叫bigkey呢？

展开 ▾

**Sunny**

2020-09-22

set/list 类型算bigkey 吗？ 考虑到主从库同步的时候，操作这个类型很耗时，求证下。

**王世艺**

2020-09-22

当主线程使用后台子线程执行了一次 fsync，需要再次把新接收的操作记录写回磁盘时，如果主线程发现上一次的 fsync 还没有执行完，那么它就会阻塞

老师这段代码逻辑在哪里

展开 ▾

**云学**

2020-09-21

除了绑定cpu也可以提升redis进程静态优先级，得到更多cpu调度

**土豆白菜**

2020-09-21

老师会讲布隆过滤器吗？

展开 ▾

**tt**

2020-09-21

可以再仔细分析一下，在虚拟机上部署REDIS实例时，由于虚拟化软件本身的内存管理算法导致的SWAP分析过程和物理机上部署时的不同么？

展开 ▾

**yeek**

2020-09-21

当主线程使用后台子线程执行了一次 fsync，需要再次把新接收的操作记录写回磁盘时，如果主线程发现上一次的 fsync 还没有执行完，那么它就会阻塞。所以，如果后台子线程执行的 fsync 频繁阻塞的话（比如 AOF 重写占用了大量的磁盘 IO 带宽），主线程也会阻塞，导致 Redis 性能变慢。

...

展开 ✓

