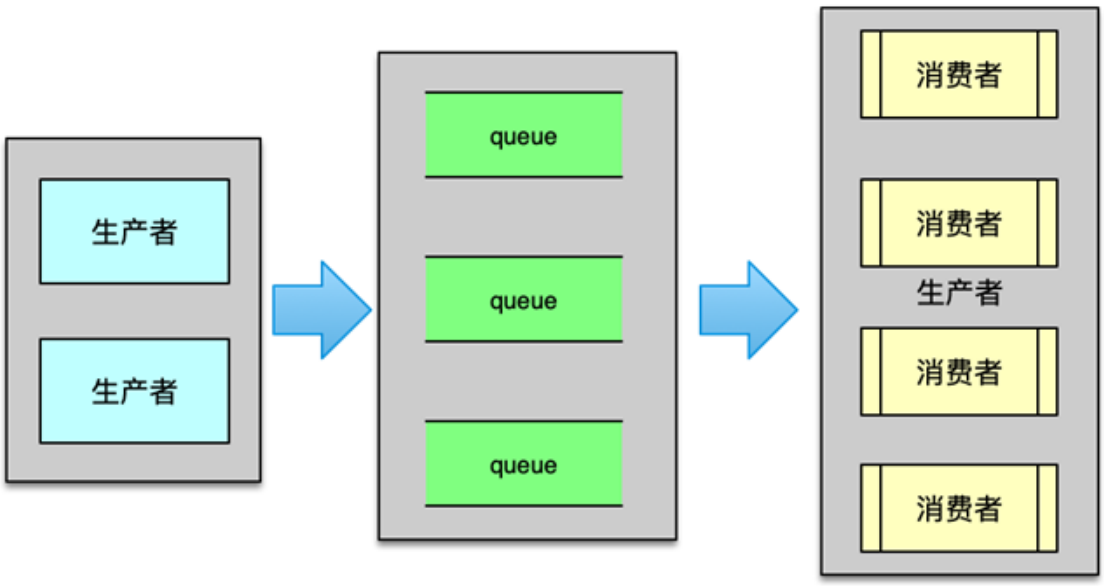
第25讲： Redis是如何处理容易超时的系统调用的?

本课时我们主要学习通过 BIO 线程解决处理容易超时的系统调用问题，以及 BIO 线程处理的任务与处理流程等内容。

BIO 线程简介

Redis 在运行过程中，不可避免的会产生一些运行慢的、容易引发阻塞的任务，如将内核中的文件缓冲同步到磁盘中、关闭文件，都会引发短时阻塞，还有一些大 key，如一些元素数高达万级或更多的聚合类元素，在删除时，由于所有元素需要逐一释放回收，整个过程耗时也会比较长。而 Redis 的核心处理线程是单进程单线程模型，所有命令的接受与处理、数据淘汰等都在主线程中进行，这些任务处理速度非常快。如果核心单线程还要处理那些慢任务，在处理期间，势必会阻塞用户的正常请求，导致服务卡顿。为此，Redis 引入了 BIO 后台线程，专门处理那些慢任务，从而保证和提升主线程的处理能力。



Redis 的 BIO 线程采用生产者-消费者模型。主线程是生产者，生产各种慢任务，然后存放到任务队列中。BIO 线程是消费者，从队列获取任务并进行处理。如果生产者生产任务过快，队列可用于缓冲这些任务，避免负荷过载或数据丢失。如果消费者处理速度很快，处理完毕后就可以安静的等待，不增加额外的性能开销。再次，有新任务时，主线程通过条件变量来通知 BIO 线程，这样 BIO 线程就可以再次执行任务。

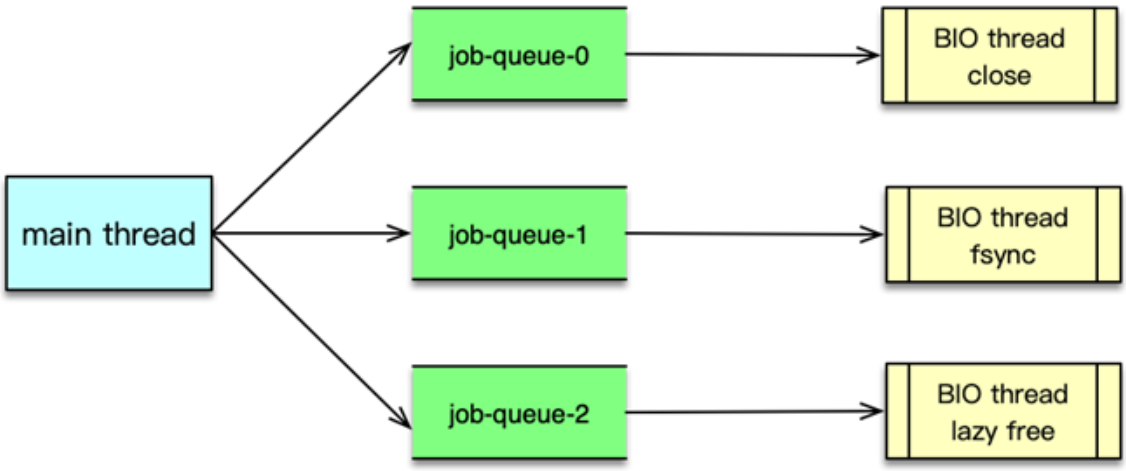
BIO 处理任务

Redis 启动时，会创建三个任务队列，并对应构建 3 个 BIO 线程，三个 BIO 线程与 3 个任务队列之间一一对应。BIO 线程分别处理如下 3 种任务。

1. close 关闭文件任务。rewriteaof 完成后，主线程需要关闭旧的 AOF 文件，就向 close 队列插入一个旧 AOF 文件的关闭任务。由 close 线程来处理。
2. fysnc 任务。Redis 将 AOF 数据缓冲写入文件内核缓冲后，需要定期将系统内核缓冲数据写入磁盘，此时可以向 fsync 队列写入一个同步文件缓冲的任务，由 fsync 线程来处理。
3. lazyfree 任务。Redis 在需要淘汰元素数大于 64 的聚合类数据类型时，如列表、集合、哈希等，就往延迟清理队列中写入待回收的对象，由 lazyfree 线程后续进行异步回收。

BIO 处理流程

BIO 线程的整个处理流程如图所示。当主线程有慢任务需要异步处理时。就会向对应的任务队列提交任务。提交任务时，首先申请内存空间，构建 BIO 任务。然后对队列锁进行加锁，在队列尾部追加新的 BIO 任务，最后尝试唤醒正在等待任务的 BIO 线程。



BIO 线程启动时或持续处理完所有任务，发现任务队列为空后，就会阻塞，并等待新任务的到来。当主线程有新任务后，主线程会提交任务，并唤醒 BIO 线程。BIO 线程随后开始轮询获取新任务，并进行处理。当处理完所有 BIO 任务后，则再次进入阻塞，等待下一轮唤醒。