Redis 事务：

相关命令：

multi（标志事务开启，将 multi 之后的 n 条命令放入到一个队列中，而不是立即执行）

exec（按顺序执行队列中的所有命令）
discard（放弃事务）

watch（监视一个或多个 key，若这些 key 在事务执行之前被改动过，则所有操作都将被放弃）

unwatch（取消对所有 key 的监控）

redis 的事务和关系型数据库中的事务相比，在 Redis 事务中如果有某一条命令执行失败，其后的命令仍然会被继续执行。

在事务中的所有命令都将会被串行化的顺序执行，事务执行期间，Redis 不会再为其它客户端的请求提供任何服务，从而保证了事物中的所有命令被原子的执行。

Redis 持久化：

快照、AOF 两种方式同时使用。（快照：按时将缓存中的数据进行备份；AOF：将 redis 的每次操作都记录在 AOF 文件，AOF 文件过大时会自动对文件进行部分重写）

都不要在 master 上进行，应该选择在 slave 上进行持久化操作。

Redis 的过期策略及内存淘汰机制：采用定期删除+惰性删除策略

定期删除，redis 默认每个 100ms 检查，是否有过期的 key,有过期 key 则删除。需要说明的是，redis 不是每个 100ms 将所有的 key 检查一次，而是随机抽取进行检查(如果每隔 100ms,全部 key 进行检查，redis 岂不是卡死)。因此，如果只采用定期删除策略，会导致很多 key 到时间没有删除。

惰性删除

redis 提供 6 种数据淘汰策略：
1）noeviction：当内存不足以容纳新写入数据时，新写入操作会报错。

2）allkeys-lru：当内存不足以容纳新写入数据时，在键空间中，移除最近最少使用的 key。

3）allkeys-random：当内存不足以容纳新写入数据时，在键空间中，随机移除某个 key。

4）volatile-lru：当内存不足以容纳新写入数据时，在设置了过期时间的键空间中，移除最近最少使用的 key。这种情况一般是把 redis 既当缓存，又做持久化存储的时候才用。

5）volatile-random：当内存不足以容纳新写入数据时，在设置了过期时间的键空间中，随机移除某个 key。

6）volatile-ttl：当内存不足以容纳新写入数据时，在设置了过期时间的键空间中，有更早过期时间的 key 优先移除。

**缓存穿透**，即黑客故意去请求缓存中不存在的数据，导致所有的请求都怼到数据库上，从而数据库连接异常。

解决方案:

(一) 如果一个查询返回的数据为空（不管是数 据不存在，还是系统故障），我们仍然把这个空结果进行缓存，但它的过期时间会很短，最长不超过五分钟。

(二)提供一个能迅速判断请求是否有效的拦截机制，比如，利用布隆过滤器，内部维护一系列合法有效的 key。迅速判断出，请求所携带的 Key 是否合法有效。如果不合法，则直接返回。

**缓存雪崩**，即缓存同一时间大面积的失效，这个时候又来了一波请求，结果请求都怼到数据库上，从而导致数据库连接异常。

解决方案:

(一)给缓存的失效时间，加上一个随机值，避免集体失效。

(二)双缓存。我们有两个缓存，缓存 A 和缓存 B。缓存 A 的失效时间为 20 分钟，缓存 B 不设失效时间。自己做缓存预热操作。然后细分以下几个小点

I 从缓存 A 读数据库，有则直接返回

II A 没有数据，直接从 B 读数据，直接返回，并且异步启动一个更新线程。

III 更新线程同时更新缓存 A 和缓存 B。

**缓存击穿**，即热点数据在失效的一瞬间，大量的访问请求涌入，导致数据库服务器出现问题

解决方案：

（一）利用互斥锁（setnx/redLock），缓存失效的时候，先去获得锁，得到锁了，再去请求数据库。没得到锁，则休眠一段时间重试。

（二）将热点数据设置为永远不过期

**并发竞争 key 问题**，这个问题大致就是，同时有多个子系统去 set 一个 key

对多个 set 没有顺序要求时：1、使用单个 redis 节点，可以通过使用 redis 的事务来解决。2、使用 redis 集群时，可以使用分布式锁（如 redis 提供的 redLock）来解决。

对多个 set 有顺序要求时：可以使用消息队列将多个请求变成串行方式执行。

应用：会话缓存，redis 全局锁，热点数据/构造时间长的数据缓存（区域树结构）