Redis6.0新特性

从单线程处理网络请求到多线程处理

- Redis 的多 IO 线程只是用来处理网络请求的，对于读写命令，Redis 仍然使用单线程来处理
- Redis 处理请求时，网络处理经常是瓶颈，通过多个 IO 线程并行处理网络操作，可以提升实例的整体处理性能。
- 继续使用单线程执行命令操作，就不用为了保证 Lua 脚本、事务的原子性，额外开发多线程互斥机制了

实现服务端协助的客户端缓存（跟踪（Tracking）功能）

- 普通模式（RESP 3 协议）
  - 实例会在服务端记录客户端读取过的 key，并监测 key 是否有修改。一旦 key 的值发生变化，服务端会给客户端发送 invalidate 消息，通知客户端缓存失效了。
  - 服务端对于记录的 key 只会报告一次 invalidate 消息，也就是说，服务端在给客户端发送过一次 invalidate 消息后，如果 key 再被修改，此时，服务端就不会再次给客户端发送 invalidate 消息。只有当客户端再次执行读命令时，服务端才会再次监测被读取的 key，并在 key 修改时发送 invalidate 消息。这样设计的考虑是节省有限的内存空间。毕竟，如果客户端不再访问这个 key 了，而服务端仍然记录 key 的修改情况，就会浪费内存资源。
- 广播模式（RESP 3 协议）
  - 服务端会给客户端广播所有 key 的失效情况，不过，这样做了之后，如果 key 被频繁修改，服务端会发送大量的失效广播消息，这就会消耗大量的网络带宽资源。
  - 所以，在实际应用时，我们会让客户端注册希望跟踪的 key 的前缀，当带有注册前缀的 key 被修改时，服务端会把失效消息广播给所有注册的客户端
  - 和普通模式不同，在广播模式下，即使客户端还没有读取过 key，但只要它注册了要跟踪的 key，服务端都会把 key 失效消息通知给这个客户端。
- 重定向模式（RESP 2 协议）
  - 在重定向模式下，想要获得失效消息通知的客户端，就需要执行订阅命令 SUBSCRIBE，专门订阅用于发送失效消息的频道 _redis_:invalidate
  - 同时，再使用另外一个客户端，执行 CLIENT TRACKING 命令，设置服务端将失效消息转发给使用 RESP 2 协议的客户端。

从简单的基于密码访问到细粒度的权限控制

- 6.0 版本支持创建不同用户来使用 Redis
- 6.0 版本还支持以用户为粒度设置命令操作的访问权限

启用 RESP 3 协议

- 在 RESP 2 中，客户端和服务器端的通信内容都是以字节数组形式进行编码的，客户端需要根据操作的命令或是数据类型自行对传输的数据进行解码，增加了客户端开发复杂度。
- RESP 3 直接支持多种数据类型的区分编码，包括空值、浮点数、布尔值、有序的字典集合、无序的集合等。
- 区分编码，就是指直接通过不同的开头字符，区分不同的数据类型，这样一来，客户端就可以直接通过判断传递消息的开头字符，来实现数据转换操作了，提升了客户端的效率。
- RESP 3 协议还可以支持客户端以普通模式和广播模式实现客户端缓存。

新增特性	作用	注意事项	适用场景
多IO线程	使用多个IO线程并行读取网络请求、进行协议解析、回写Socket	多IO线程只负责处理网络请求，不执行命令操作	提升Redis吞吐量
客户端缓存	使用普通模式，监测客户端读取的key的修改情况	普通模式和广播模式需要启用RESP 3协议接收失效消息	加速业务应用访问
	使用广播模式，将key的失效消息发送给所有客户端		
	使用重定向模式，支持使用RESP 2协议的客户端		
访问权限控制	区分不同用户，支持以用户和key为粒度设置某个或某类命令的调用权限		支持多用户以不同权限访问Redis
RESP 3协议	使用不同开头字符表示多种数据类型，简化客户端开发复杂度		高效支持不同数据类型使用，支持客户端缓存