1. 多线程

2. Client Side Cache

3. Acls

多线程：

1. redis 6.0 提供了多线程的支持，redis 6 以前的版本，严格来说也是多线程，只不过执行用户命令的请求时单线程模型，还有一些线程用来执行后台任务， 比如 unlink 删除 大key，rdb持久化等。

redis 6.0 提供了多线程的读写IO, 但是最终执行用户命令的线程依然是单线程的，这样，就没有多线程数据的竞争关系，依然很高效。

redis 6.0 以前线程执行模式，如下操作在一个线程中执行完成

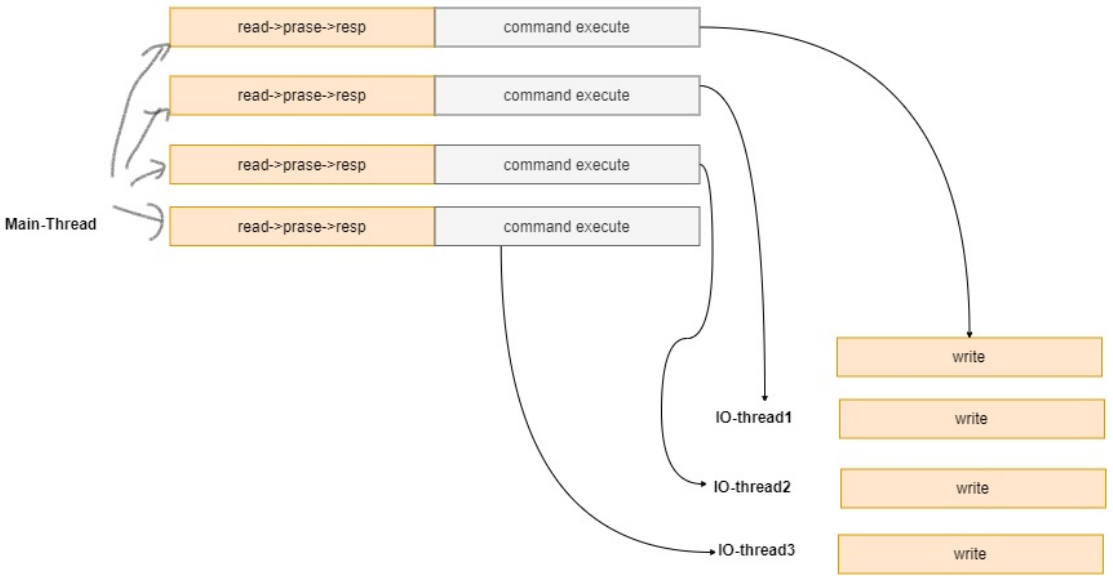
redis 6.0 线程执行模式：

可以通过如下参数配置多线程模型：

如：

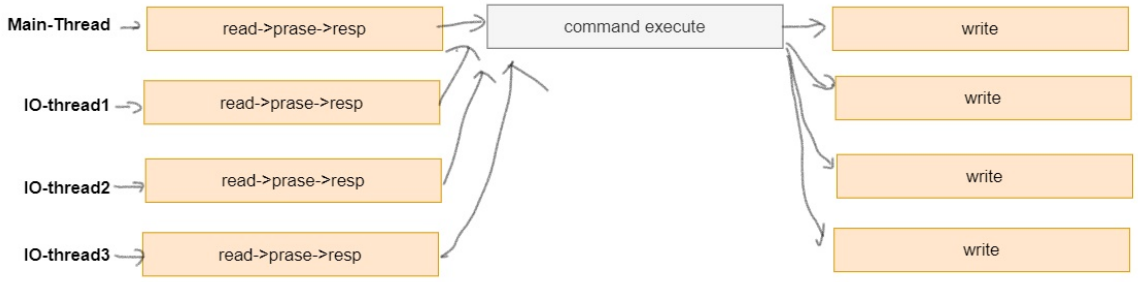
```
io-threads 4 // 这里说 有三个IO 线程，还有一个线程是main线程，main线程负责IO读写和命令执行操作
```

默认情况下，如上配置，有三个IO线程， 这三个IO线程只会执行 IO中的write 操作，也就是说，read 和 命令执行 都由main线程执行。最后多线程将数据写回到客户端。



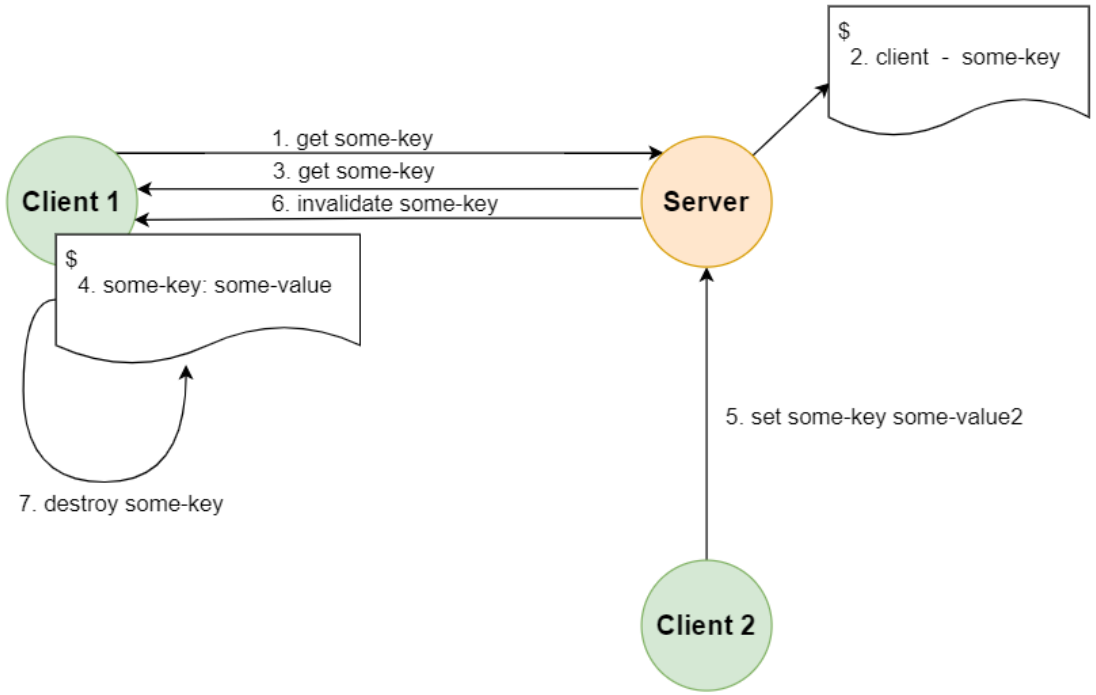
开启了如下参数：

```
io-threads-do-reads yes // 将支持IO线程执行 读写任务。
```



2. client side caching

客户端缓存：redis 6 提供了服务端追踪key的变化，客户端缓存数据的特性，这需要客户端实现



执行流程为， 当客户端访问某个key时，服务端将记录key 和 client ，客户端拿到数据后，进行客户端缓存，这时，当key再次被访问时，key将被直接返回，避免了与redis 服务器的再次交互，节省服务端资源，当数据被其他请求修改时，服务端将主动通知客户端失效的key，客户端进行本地失效，下次请求时，重新获取最新数据。

目前只有lettuce对其进行了支持：

```
<dependency>
 <groupId>io.lettuce</groupId>
 <artifactId>lettuce-core</artifactId>
 <version>6.0.0.RELEASE</version>
</dependency>
```

```
public static void main(String[] args) throws InterruptedException {
RedisClient redisClient = RedisClient.create("redis://192.168.109.200");

Map<String, String> clientCache = new ConcurrentHashMap<>();
```

```
5
6  StatefulRedisConnection<String, String> myself = redisClient.connect();
7
8  CacheFrontend<String, String> frontend =
9  ClientSideCaching.enable(CacheAccessor.forMap(clientCache),
10  myself,
11  TrackingArgs.Builder.enabled().noloop());
12
13  String key="csk";
14  int count = 0;
15  while (true){
16
17  System.out.println(frontend.get(key));
18  TimeUnit.SECONDS.sleep(3);
19  if (count++ == Integer.MAX_VALUE){
20  myself.close();
21  redisClient.shutdown();
22  }
23  }
24 }
```

3. ACL 是对于命令的访问和执行权限的控制，默认情况下，可以有执行任意的指令，兼容以前版本

ACL设置有两种方式：

1. 命令方式

   ACL SETUSER + 具体的权限规则， 通过 ACL SAVE 进行持久化

2. 对 ACL 配置文件进行编写，并且执行 ACL LOAD 进行加载

ACL存储有两种方式，但是两种方式不能同时配置，否则直接报错退出进程

1.redis 配置文件： redis.conf

2.ACL配置文件, 在redis.conf 中通过 aclfile /path 配置acl文件的路径

命令方式：

```
1 ACL SETUSER alice // 创建一个 用户名为 alice的用户
```

用如上的命令创建的用户语义为：

1. 处于 off 状态， 它是被禁用的，不能用auth进行认证
2. 不能访问任何命令
3. 不能访问任意的key
4. 没有密码

如上用户alice 没有任何意义。

创建一个对 cached: 前缀具有get命令执行权限的用户，并且设置密码：

```
acl setuser alice on >pass123 ~cached:* +get
```

```
auth alice pass123
set a a
(error) NOPERM this user has no permissions to run the 'set' command or its subcommand
get a a
(error) NOPERM this user has no permissions to access one of the keys used as arguments
get cached:name
vvv

```

如上，如果访问没有被授权的命令，或者key， 将报错，set 命令没有被授权， key a 没有被授权，

cached:name 可以通过验证。

更符合阅读习惯的格式

```
ACL GETUSER alice
```

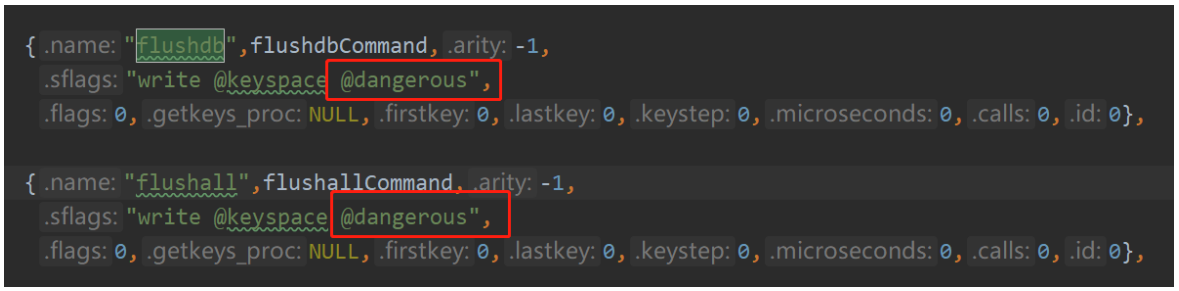
添加多个访问模式，空格分隔， 注意，切换其他用户进行登录，alice没有admin权限

```
ACL SETUSER alice ~objects:* ~items:* ~public:*
```

针对类型命令的约束

```
ACL SETUSER alice on +@all -@dangerous >密码 ~*
```

这里+@all: 包含所有得命令 然后用-@ 去除在redis command table 中定义的 dangerous 命令



可以通过如下命令进行查看具体有哪些命令属于某个类别

```
acl cat // 查看所有类别
acl cat dangerous // 查看所有的 dangerous 命令
```

开放子命令

```
ACL SETUSER myuser -client +client|setname +client|getname
```

禁用client 命令，但是开放 client 命令中的子命令 setname 和 getname ，只能是先禁用，后追加子命令，因为后续可能会有新的命令增加。

文档：Redis 6.0 新特性.note
链接：http://note.youdao.com/noteshare?id=2689d0dca5926b2169374b277d17fad5&sub=23BE542644D648DBB49D825347C38139