小象科技 ChinaHadoop.cn

Redis 从入门到精通

黄健宏（huangz）

扩展系统处理写请求的能力

小象科技 ChinaHadoop.cn

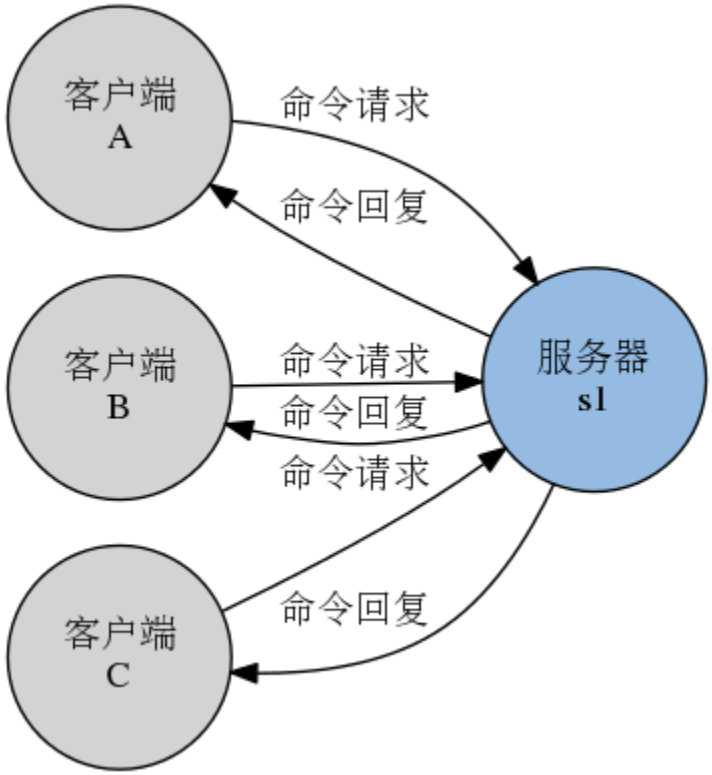
上一节课介绍了如何使用 Redis 提供的复制功能来扩展系统的读性能，但是在实际地使用 Redis 的时候，用户除了可能遇上读性能问题之外，还可能会遇上写性能问题，有时候，用户甚至需要同时扩展系统处理读请求和写请求的能力。

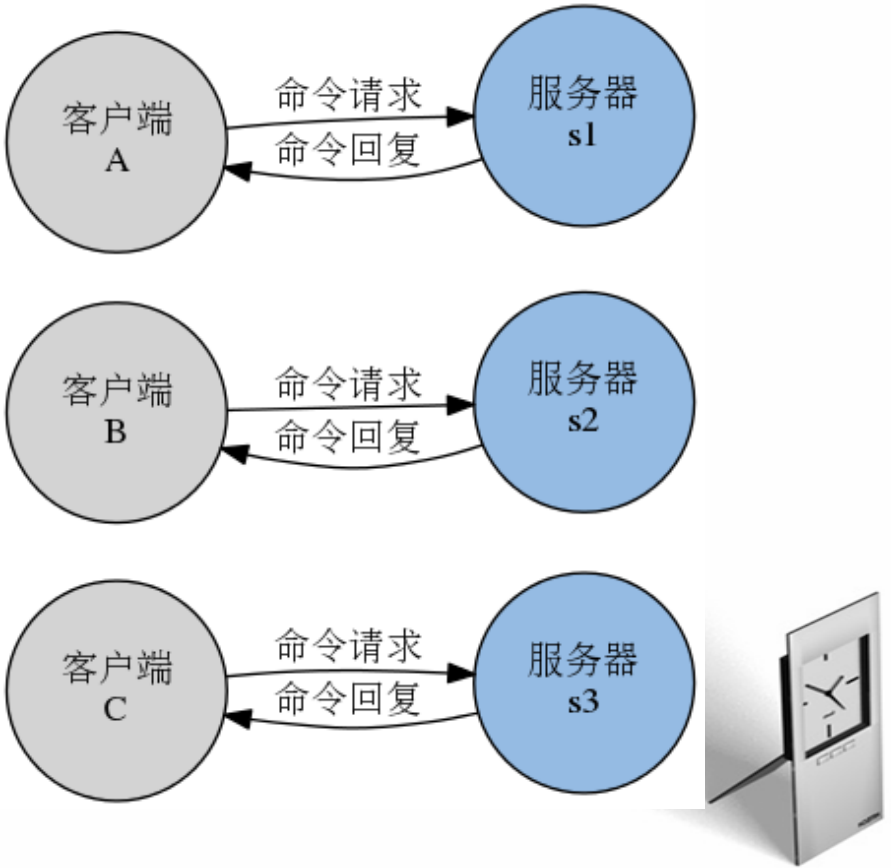
举个例子，如果用户只有一台服务器，该服务器每秒钟能够处理 10 万次命令请求，其中每秒 3 万次写、7 万次读，但是随着客户的数量增多，系统现在必须能够每秒钟处理一倍于现有数量的命令请求，也即是，每秒钟 6 万次写、14 万次读，总共 20 万次命令请求每秒。

为了解决这个问题，我们不仅需要像《复制》一节介绍的那样，使用多台服务器来处理客户端的读请求，还必须使用多台服务器来处理客户端的写请求。

达到这个目的的其中一种方法是使用分片（shard），这项技术的核心原理是将整个数据库分为多个部分，使用不同的服务器来储存不同部分，并负责处理相应部分的命令请求（包括读请求和写请求）。

使用分片来扩展性能





小象科技 ChinaHadoop.cn

twemproxy

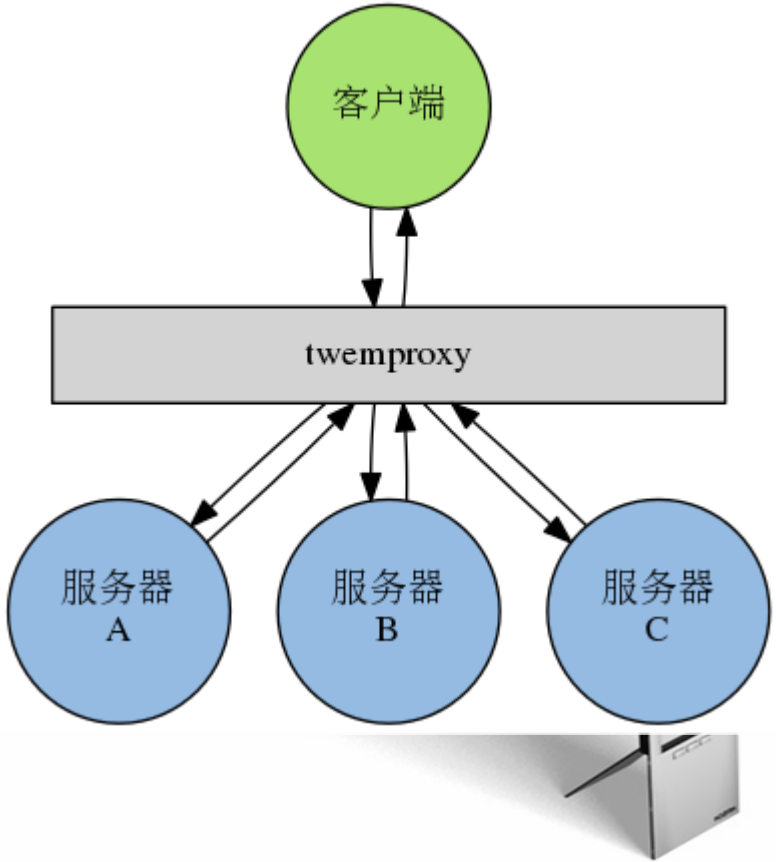
Redis 代理服务器

关于 twemproxy

twemproxy 是 twitter 开发的一个代理服务器，它兼容 Redis 和 Memcached，允许用户将多个 Redis 服务器添加到一个服务器池(pool)里面，并通过用户选择的散列函数和分布函数，将来自客户端的命令请求分发给服务器池中的各个服务器。

通过使用 twemproxy，我们可以将数据库分片到多台 Redis 服务器上面，并使用这些服务器来分担系统负责以及数据库容量:在服务器硬件条件相同的情况下，对于一个包含 N 台 Redis 服务器的池来说，池中每台服务器平均包含整个数据库数据的 1/N，并处理平均 1/N 的客户端命令请求。

向池里面添加更多服务器可以线性地扩展系统处理命令请求的能力，以及系统能够保存的数据量。



安装、配置和运行 twemproxy

1. 安装 twemproxy：
   下载并解压压缩包：http://code.google.com/p/twemproxy/downloads/list
   $ ./configure
   $ make
   $ sudo make install
2. 使用 yml 格式编写配置文件：

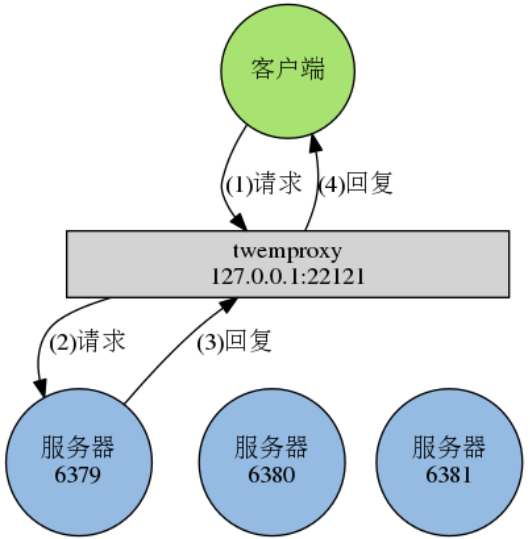
```
my_redis_server_group:    # 服务器池的名字，支持创建多个服务器池
  listen: 127.0.0.1:22121 # 这个服务器池的监听地址和端口号
  hash: fnv1a_64          # 键散列算法，用于将键映射为一个散列值
  distribution: ketama    # 键分布算法，决定键被分布到哪个服务器
  redis: true             # 代理 Redis 命令请求，不给定时默认代理 Memcached 请求
  servers:                # 池中各个服务器的地址和端口号，以及权重
    - 127.0.0.1:6379:1
    - 127.0.0.1:6380:1
    - 127.0.0.1:6381:1
```

3. $ nutcracker -c nutcracker.yml

使用 twemproxy

用户可以使用 Redis 客户端连接上 twemproxy，并向 twemproxy 发送命令请求，接收到命令请求的 twemproxy 会根据用户指定的散列函数和分布函数来决定将命令交 给哪个服务器来处理：

```
$ ./redis-cli -p 22121
127.0.0.1:22121> SET msg "hello world"
OK
127.0.0.1:22121> SADD numbers 1 3 5 7 9
(integer) 5
127.0.0.1:22121> RPUSH lst "a" "b" "c" "d" "e" "f" "g"
(integer) 7
127.0.0.1:22121> ZADD fruits-price 8.0 "apple" 3.5 "banana" 4.2 "cherry"
(integer) 3
127.0.0.1:22121> INCRBY counter 10086
(integer) 10086
```



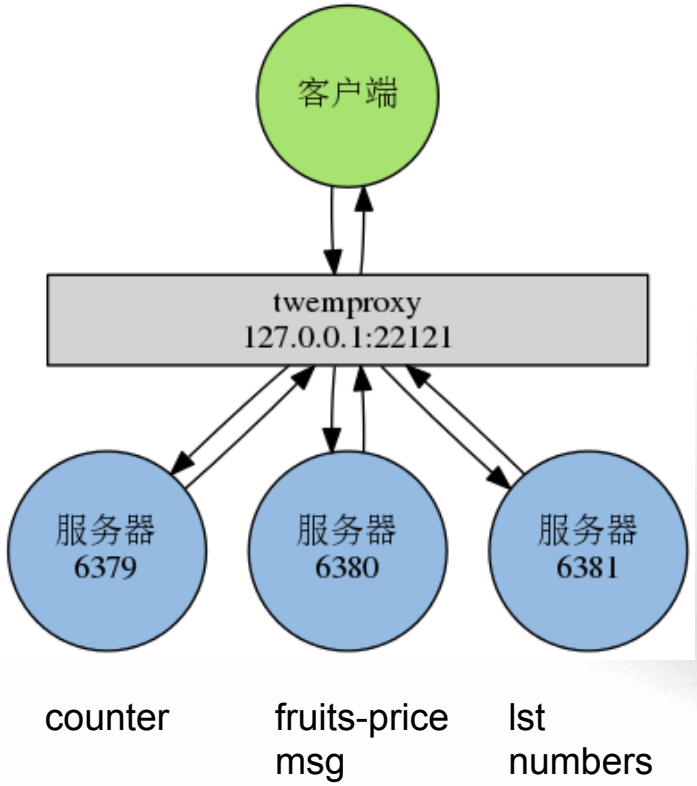
当 twemproxy 获取到命令请求的回复时，就会将命令回复返回 给发送命令请求的客户端，整个代理过程对于客户端来说是完全透明的。

数据库键分布


```
127.0.0.1:6379> KEYS *
1) "counter"

127.0.0.1:6380> KEYS *
1) "fruits-price"
2) "msg"

127.0.0.1:6381> KEYS *
1) "lst"
2) "numbers"
```



键分布

通过散列函数和分布函数来将数据 库的键放置到不同的服 务器里面

键分布

当客户端向 twemproxy 发送一个针对键 key 的命令请求时，twemproxy 会根据以下公式计算出应该由服务器池中的哪个服务器来处理这个命令请求：

```
# 计算键 key 的散列值
hash_value = hash_function(key)

# 根据键 key 的散列值，计算出键 key 应该被分布到哪个服务器里面
# 其中 all_servers_in_pool 是一个列表，它包含了池中的所有服务器的 IP 地址和端口号
# 比如 ['127.0.0.1:6379', '127.0.0.1:6380'. '127.0.0.1:6381']
server = distribution_funtion(hash_value, all_servers_in_pool)

# 将请求发送给服务器
forward_request(server, request)
```

散列函数

twemproxy 提供了多种散列函数供用 户选择，用户可以通过修改配置文件中 hash 选项的值来指定自己想要使用的散列函数：

- 包括 one at a time 算法在内的 Jenkins 散列函数
- 包括 crc16 、crc32 和 crc32a 在内的循环冗余校验和函数
- md5 散列算法
- 包括 fnv1_64 、fnv1a_64 、fnv1_32 和 fnv1a_32 在内的 Fowler Noll Vo 散列函数
- 由 Paul Hsieh 发明的 hsieh 散列算法
- murmur 散列算法

每个散列函数适用的数据 类型、它们的分布率以及 计算速度等情况都不尽相同，用 户可以先了解一下这些函数，然后再根据自己的情况来 选择合适的散列函数。

分布函数

用户可以通过修改 distribution 选项的值来改变 twemproxy 分布键的方式，选项的值可以是：

- ketama：一致性 hash，主要用于在实现缓存服务时，防止某个服务器下线而造成缓存大量失效 http://www.audioscrobbler.net/development/ketama/ 。
- modula：取模运算，相当于依靠散列函数本身来分布各个 键。
- random ：不考虑散列值，直接随机地将键分布到池中的某个服务器，在访问键时，也随机选择一个服务器来进行访问。

散列标签

在默认情况下，twemproxy 会根据键的整个键名来计算键的散列值，但如果用户通过 hash_tag 选项指定了散列标签（hash tag），那么 twemproxy 只会根据键名中被散列标签包围的部分来计算键的散列值。

举个例子，假设用户设置了配置 hash_tag: "{}"，那么以下两个键将计算出相同的散列值，因为它们散列标签中包含的内容相同 —— 都是 user：

bbs::{user}::123456

bbs::{user}::255255

而键 bbs::{topic}::98765 则可能会计算出与以上两个键不同的散列值，因为这个键在散列标签中包含的内容并不相同。

散列标签可以自由设置，比如对于键 bbs::$user$::123456 就可以使用 “$$” 作为标签、键 bbs::(user)::123456 就可以使用 “()” 作为标签、而键 bbs::*user*::123456 就可以使用 “**” 作为标签，诸如此类。

小象科技 ChinaHadoop.cn

服务器上下线管理

自动移除下线服务器和自动添加重新上线的服务器

服务器下线

在默认情况下，当服务器池中的某个服务器下线时，twemproxy 会向所有发送给该服务器的命令请求返回一个连接错误：

```
127.0.0.1:22121> INCR counter    # 该命令请求将被转发给 127.0.0.1:6379
(integer) 1
127.0.0.1:22121> GET counter
"1"
127.0.0.1:22121> GET counter     # 127.0.0.1:6379 已经下线
(error) ERR Connection refused   # 发送给该服务器的所有命令请求都将失败
127.0.0.1:22121> GET counter
(error) ERR Connection refused
127.0.0.1:22121> GET counter
(error) ERR Connection refused
```

自动移除下线服务器

为了解决服务器下线的问题，twemproxy 提供了 auto_eject_hosts <bool> 和 server_failure_limit <N> 这两个配置选项：当 auto_eject_hosts 选项的值为 true，并且 twemproxy 在连续 N 次向同一个服务器发送命令请求但都遇到错误时，twemproxy 就会将该服务器标记为下线，并将原本由这个服务器负责处理的键交给池中的其他在线的服务器来处理。

举个例子，对于包含以下配置的 twemproxy 来说：

```
my_redis_servers:
  # …
  auto_eject_hosts:  true
  server_failure_limit:  3
```

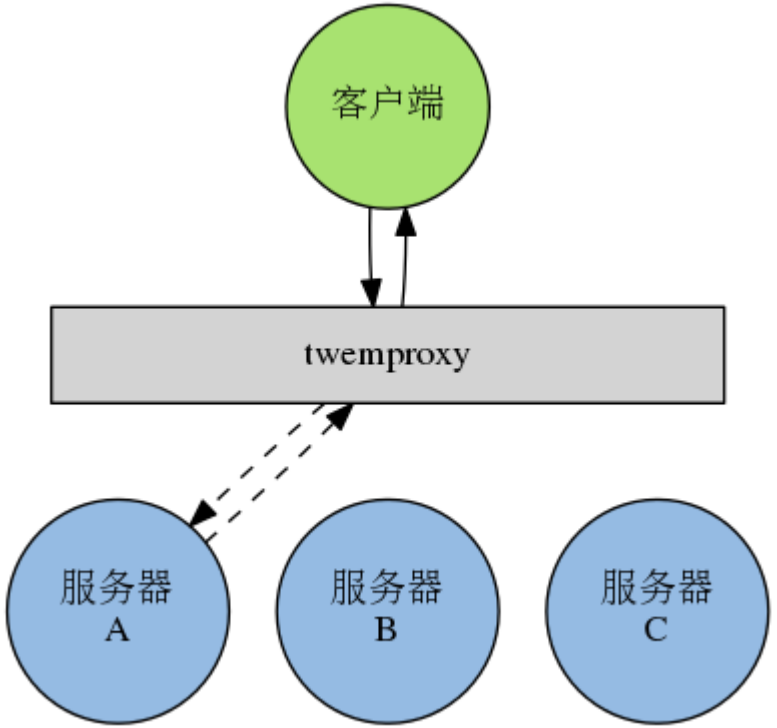
当 twemproxy 连续 3 次向同一个服务器发送命令请求都遇到错误时，twemproxy 就会将该服务器标记为下线，并将原本由这个服务器负责处理的键交给池中的其他在线的服务器来处理。

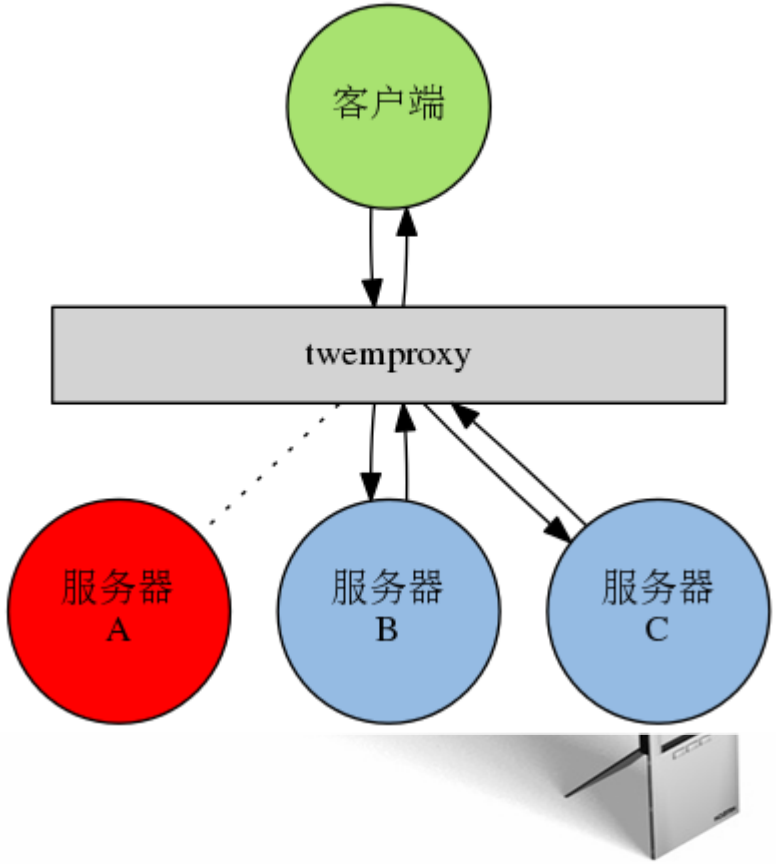
自动移除示例

小象科技 ChinaHadoop.cn

```
127.0.0.1:22121> SET counter 222    # 负责处理 counter 键的服务器正常上线
OK
127.0.0.1:22121> GET counter
"222"
127.0.0.1:22121> GET counter    # 负责处理 counter 键的服务器下线
(error) ERR Connection refused
127.0.0.1:22121> GET counter
(error) ERR Connection refused
127.0.0.1:22121> GET counter    # 第三次尝试失败，counter 键被转交给其他服务器处理
(nil)
```

图示过程





自动添加重新上线的服务器

twemproxy 提供了 server_retry_timeout <time> 选项，time 参数的格式为毫秒。

当一个服务器被 twemproxy 判断为下线之后，在 time 毫秒之内，twemproxy 不会再尝试向下线的服务器发送命令请求，但是在 time 毫秒之后，服务器会尝试重新向下线的服务器发送命令请求：

- 如果命令请求能够正常执行，那么 twemproxy 就会撤销对该服务器的下线判断，并再次将键交给那个服务器来处理。
- 但如果服务器还是不能正常处理命令请求，那么 twemproxy 就会继续将原本应该交给下线服务器的键转交给其他服务器来处理，并等待下一次重试的来临。

举个例子，对于以下配置来说，twemproxy 在将一个服务器标记为下线之后，会在三万毫秒之后，重新向下线的服务器发送命令请求，检测它是否重新上线：

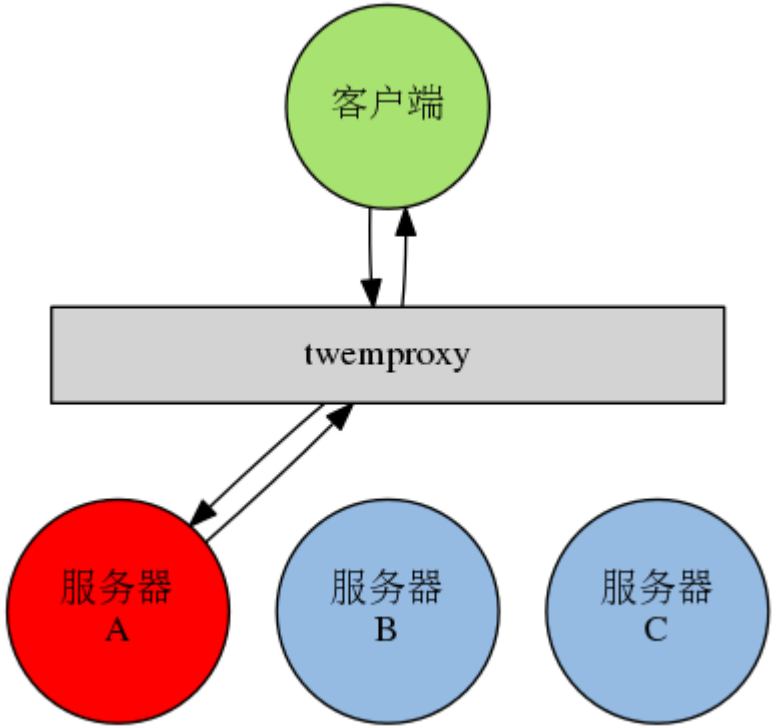
```
my_redis_servers:
  # ...
  server_retry_timeout: 30000
```

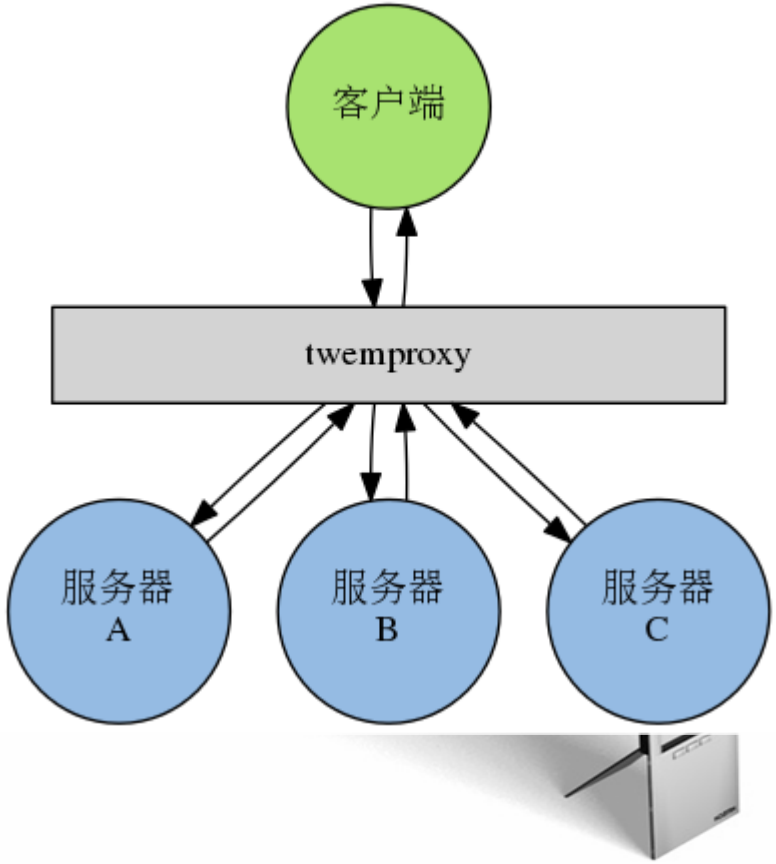
自动添加示例

小象科技 ChinaHadoop.cn

```
127.0.0.1:22121> SET counter 222    # 负责处理 counter 键的服务器正常上线
OK
127.0.0.1:22121> GET counter
"222"
127.0.0.1:22121> GET counter    # 负责处理 counter 键的服务器下线
(error) ERR Connection refused
127.0.0.1:22121> GET counter
(error) ERR Connection refused
127.0.0.1:22121> GET counter    # 第三次尝试失败，counter 键被转交给其他服务器处理
(nil)
127.0.0.1:22121> GET counter    # 原服务器重新上线，twemproxy 将 counter 键重新分配给它
"222"
```

图示过程





了解更多信息

前面的内容介绍了 twemproxy 提供的主要功能，以及最重要的几个配置 选项，想要了解更多 twemproxy 的相关信息，可以访问这个项目的主页：https://github.com/twitter/twemproxy

另外 redis-mgr 这个项目通过整合复制、Sentinel 以及 twemproxy 等组件，提供了一站式的 Redis 服务器部署、监控、迁移功能，有兴趣的可以去看看： https://github.com/idning/redis-mgr

复习

回顾本节学习的内容

本节重点

twemproxy 是一个代理服务器，它可以将多个 Redis 服务器添加到一个服务器池里面，并将客户端发送的命令请求转交给池中的各个服务器来处理，池中的每个服务器都会包含一部分数据，并处理一部分命令请求。

twemproxy 允许用户通过设置配置选项来指定散列函数、分布函数以及散列标签，从而将不同的数据库键分布到不同的服务器里面。

twemproxy 允许用户通过设置配置选项，让 twemproxy 在某个服务器下线时，自动地将原本由下线服务器负责处理的数据库键转交给池中的其他在线服务器来处理；并且用户还可以设置一个重试时间，让 twemproxy 在一段时间之后，重新向下线的服务器发送命令请求，检测该服务器是否已经重新上线，如果是的话，twemproxy 就会重新将数据库键交给该服务器来处理。