

北京航空航天大学

软件需求说明书

Redis

SY1406108 陈志伟 SY1406112 王珊珊

SY1406311 林 璐 SY1406117 王志鹏

2015/3/30

版本变更历史

版本	提交日期	编制人	说明
Version 1.0	2015-03-31	全体组员	初步完成需求规格说明书的第一版。
Version 1.1	2015-04-01	陈志伟	修改了服务器模块的部分用例规格，排版文档。
Version 1.2	2015-04-01	林璐	1. 改正编号错误，设置数据字典的自动编号；改正错误字，设置统一行距和格式； 2 用例图更新。
Version 1.3	2015-04-01	王志鹏	1. 修正版本提交日期 2. 客户端模块用例图更新，用例更新
Version 1.4	2015-04-02	王珊珊	1. 删减空白页 2 调整格式 3 更新目录和页码

目录

1 范围.....	1..
1.1 标识.....	1..
1.2 系统概述.....	1..
1.3 数据字典.....	2..
1.4 文档概述.....	5..
2 项目描述.....	5..
2.1 模块概述.....	5..
2.2 典型应用场景描述.....	7..
3 功能需求.....	8..
3.1 服务器模块.....	8..
3.1.1 启动服务.....	9..
3.1.2 自定义服务器配置.....	10
3.1.3 处理事件.....	11.
3.1.4 关闭服务器.....	12
3.2 客户端模块.....	12
3.2.1 发送命令请求.....	13
3.2.2 读取命令请求.....	13
3.2.3 执行命令请求.....	14
3.2.4 回复命令.....	15
3.3 RDB持久化模块.....	15
3.3.1 保存快照.....	16
3.3.2 同步回写 SAVE.....	17
3.3.3 异步回写 BGSAVE.....	18
3.3.4 载入数据.....	18
3.4 AOF持久化模块.....	20
3.4.1 命令同步.....	20
3.4.2 命令传播.....	21.
3.4.3 缓存追加.....	21.
3.4.4 文件写入和保存.....	22
3.4.5 AOF 文件读取和数据还原.....	23
3.4.6 AOF 后台重写.....	23
4 非功能性需求分析.....	24
4.1 鲁棒性.....	24.
4.2 安全性.....	24.
4.3 效率.....	25.
5 参考文献.....	25.

1 范围

1.1 标识

Redis 版本号： redis-3.0.0-rc5

模 块 名 称： Redis 服务器、客户端、RDB 持久化、AOF 持久化模块

需求报告版本：V1.0

1.2 系统概述

Redis 是 NoSQL 系列中的一种 key-value 类型的轻量级内存数据库，全名为远程字典服务(REmote DIctionary Server)。它是由 Salvatore Sanfilippo 使用 ANSI C 语言编写的，支持网络、可基于内存亦可持久化的开源日志型、Key-Value 数据库，并提供多种语言的 API。从 2013 年 3 月 15 日起，Redis 的开发工作由 VMware 主持。从 2013 年 5 月开始，Redis 的开发由 Pivotal 赞助。

Redis 定位于一个内存数据库，但事实是其并不是将所有的数据都存储在内存中，其持久性体现在在硬盘上进行写操作，它不仅仅是一种简单的 key-value 存储。Redis 支持存储的 value 类型有 string(字符串)、lists(链表)、sets (集合)、zsets(有序集合)和 hash 等。这些数据类型均支持 push、pop、add、remove，取交集并集和差集及更丰富的操作，且这些操作都是原子性的。它支持各种不同方式的排序，为保证效率，它的数据都保存在内存中但是 Redis 会周期性地把更新的数据写入磁盘或把修改操作写入追加的记录文件，并且在此基础上实现了 master-slave 同步。因为是纯内存操作，Redis 的性能非常出色，每秒可以处理超过 10 万次读写操作，是已知性能最快的 Key-Value 数据库。

Redis 具有许多优秀的特性，如支持多种数据结构、支持简单事务控制、支持持久化、支持主从复制、Virtual Memory 功能等。

1.3 数据字典

编号	术语	英文	说明
1	非关系型数据库	NoSQL	NoSQL 数据库是 Not Only SQL 的简称，即不仅仅是 SQL，是对 SQL 应用不足的一种补充，中文翻译过来也称为非关系型数据库。它是一类通常强调非关系性、分布式、开源和并行性的数据库。CAP 理论，BASE 思想(BASE 模型是反 ACID 模型的，该模型牺牲高一致性,获得可用性)和最终一致性三者为 NoSQL 的三大理论基础。
2	key-value 数据库	Key-value DataBase	key-value 是 NoSQL 数据库中最简单,也是发展最快的一种数据模型,其遵循的最基础的原理即 CAP 理论。每一个 key 值对应任意的数据值，这个任意的数据值不用特别去管它到底是什么。key-value 的数据是通过 key 值来进行相关存储和检索的,唯一键 key 可以识别一个条目和所有通过此键进行的删除、修改以及插入新key 的操作。它可被看作是一个大的、分布式的、持久的、容错的哈希表。
3	持久化	Persistence	数据持久化就是将内存中的数据模型转换为存储模型,以及将存储模型转换为内存中的数据模型的统称。数据模型可以是任何数据结构或对象模型,存储模型可以是关系模型、XML、二进制流等。Redis 是将数据存储在内存中的,或者使用虚拟内存，但是会定期将内存中的数据写入磁盘,或者用日志方式记录每次更新的日志。前者的性能较

			高,但是可能会引起一定程度的数据丢失,而后者相反。
4	主从同步	Master-slave Synchronization	Redis 支持将数据同步到多台从数据库上,这样可以显著提高数据读取性能。
5	事件	Event	<p>事件是 Redis 服务器的核心,它处理两项重要的任务:</p> <ol style="list-style-type: none"> 1. 处理文件事件:在多个客户端中实现多路复用,接受它们发来的命令请求,并将命令的执行结果返回给客户端。 2. 时间事件:实现服务器常规操作(server cron job)。
6	事务	Transaction	事务提供了一种“将多个命令打包,然后一次性、按顺序地执行”的机制,并且事务在执行的期间不会主动中断——服务器在执行完事务中的所有命令之后,才会继续处理其他客户端的其他命令。
7	虚拟内存	Virtual Memory	虚拟内存是计算机系统内存管理的一种技术。它使得应用程序认为它拥有连续的可用的内存(一个连续完整的地址空间),而实际上,它通常是被分隔成多个物理内存碎片,还有部分暂时存储在外部磁盘存储器上,在需要时进行数据交换。Redis 的 Virtual Memory(虚拟内存)通过配置可以让用户设置最大使用内存,当超出这个内存时,通过 LRU 类似算法,将一部分数据存入文件中,在内存中只保存使用频率高的数据来提高 Redis 性能。
8	AOF	Append Only File	AOF 将“操作 + 数据”以格式化指令的方

			式追加到操作日志文件的尾部，在 append 操作返回后(已经写入到文件或者即将写入)，才进行实际的数据变更，“日志文件”保存了历史所有的操作过程；当 server 需要数据恢复时，可以直接 replay 此日志文件，即可还原所有的操作过程。AOF 文件内容是字符串，非常容易阅读和解析，它和 redis-protocol 具有一样的格式约束
9	重写	Rewrite	重写即创建一个新的 AOF 文件来代替原有的 AOF 文件，新 AOF 文件和原有 AOF 文件保存的数据库状态完全一样，但新AOF 文件的体积小于等于原有 AOF 文件的体积。
10	数据还原	Data Reduction	数据还原即只要根据 AOF 文件里的协议，重新执行一遍里面指示的所有命令，就可以还原 Redis 的数据库状态。
11	I/O 多路复用	I/O Multiplexing	IO 多路复用是指一旦发现进程指定的一个或者多个描述符可进行无阻塞 IO 访问时，它就通知该进程。
12	命令执行器	Command Actuator	命令执行器是真正执行命令的程序，由 Redis 服务器来调用。
13	快照	Snapshot	快照是 Redis 默认的持久化方式。这种方式是就是将内存中数据以快照的方式写入到二进制文件中,默认的文件名为 dump.rdb。用户可以通过配置设置自动做快照持久化的方式，比如配置 redis 在 n 秒内如果超过 m 个 key 被修改就自动做快照。
14	过期键	Expire	Redis 的一大特性之一是可以为数据库中的每一个键关联一个过期时间，当到达指定时

			间后 Redis 就会把该键从数据库中删除。 Redis在 RedisDb中维护了一个 expires字典，其中的键就是那些已经设置了过期时间的键，而键值就是该键的过期时间（毫秒）。
15	同步 / 异步保存	Synchronization/ Asynchronization	Redis 在将内存中的数据库数据保存到磁盘时，有两种方式，同步或异步。同步保存直接在主进程中调用保存函数，阻塞 Redis 主进程，直到保存完成为止；而异步保存将 fork 出一个子进程，子进程调用保存函数，并在保存完成后向主进程发送信号通知。

1.4 文档概述

本文档是对 Linux 进程通信模块的需求分析和规格说明书，主要借助 RUCM 结构化模板，采用用例图等形式进行分析。

2 项目描述

2.1 模块概述

本次实验中所选择的四个模块为：

(1) 服务器模块：Redis 服务器模块主要实现了 Redis 从启动服务器，到服务器可以接受外来客户端的网络连接这段时间，需要执行的一系列初始化操作及初始化完成后，Redis 进入事件轮询处理来自客户端的请求。整个过程主要包括初始化服务器全局状态，程序创建一个 redisServer 结构的实例变量 server 用作服务器的全局状态，并将 server 的各个属性初始化为默认值。接着，程序进入服务器初始化的下一步读入配置文件，并创建 daemon 进程来运行 Redis，并创建相应的 pid 文件。然后，程序初始化服务器的功能模块，完成这一步之后，服务器打印出 Redis 的 ASCII LOGO、服务器版本等信息，表示所有功能模块已经就绪，但这时服务器的数据库还是一片空白，程序还需要将持久化在 RDB 或者 AOF 文件里

的数据,载入到服务器进程里面。最后程序打开事件循环,开始接受客户端连接,处理来自客户端的请求。

(2) 客户端模块: 每个客户端可以向服务器发送命令请求,而对于每个与服务器进行连接的客户端,都有相应的 redisClient 结构,这个结构保存了客户端当前的状态信息,以及执行相关功能时需要用到的数据结构。客户端状态包含许多属性,可以分为两类,一类是比较通用的属性,另一类是与特定功能相关的属性。多个客户端可以连接到同一个服务器,服务器会将新的客户端状态添加到服务器状态结构 clients 链表的末尾。

(3) RDB持久化模块: RDB 模块主要将当前内存中的数据库快照保存到磁盘文件中,在 Redis 重启动时,RDB 程序可以通过载入 RDB 文件来还原数据库的状态。RDB 持久化可以在指定的时间间隔内生成数据集的时间点快照,在配置文件中的格式是: save N M 表示在 N 秒之内,redis 至少发生 M 次修改则 redis 抓快照到磁盘。这种不定期的保存方式称为“半持久化模式”,包括同步回收 SAVE 或者异步回收 BGSAVE。SAVE 方式直接调用 rdbSave 函数将内存中的数据库数据以 RDB 格式保存到磁盘中,若 RDB 文件已存在,则新的 RDB 文件将替换已有的 RDB 文件,阻塞 Redis 主进程,直到保存完成为止,在主进程阻塞期间服务器不能处理客户端的任何请求;BGSAVE 方式则 fork 出一个子进程调用 rdbSave 函数,并在保存完成后向主进程发送信号,通知保存完成,此期间 Redis 服务器主程序仍可继续处理客户端请求。为了防止保存命令的冲突,在执行 SAVE 时服务器阻塞,因此新的 SAVE 或 BGSAVE 命令调用都不会产生作用直到上一个 SAVE 命令执行完,Redis 重新开始接受请求;在执行 BGSAVE 时,服务器会检查 BGSAVE 是否在执行当中,若是则返回出错信息告知。当 Redis 服务器启动,RDB 模块将读取 RDB 文件,并将文件中的数据库数据载入到内存。载入期间,服务器每载入 1000 个键就处理一次已到达的部分请求,其他命令一律返回错误,直到载入完成之后,服务器才会开始正常处理所有命令。

(4) AOF持久化模块: AOF 持久化模块实现了 AOF 以协议文本的方式,将所有对数据库进行过写入的命令(及其参数)记录到 AOF 文件,以此达到记录数据库状态的目的。整个模块过程包括 Redis 将所有对数据库进行过写入的命令(及其参数)记录到 AOF 文件,以此达到记录数据库状态的目的,为了方便起见,我

们称呼这种记录过程为同步；可以实现 AOF 文件的读取和数据还原，AOF 文件保存了 Redis 的数据库状态，而文件里面包含的都是符合 Redis 通讯协议格式的命令文本，根据 AOF 文件里的协议，重新执行一遍里面指示的所有命令，就可以还原 Redis 的数据库状态；AOF 文件通过同步 Redis 服务器所执行的命令，从而实现了数据库状态的记录，但是，这种同步方式会造成一个问题：随着运行时间的流逝，AOF 文件会变得越来越来大。为了解决以上的问题，Redis 需要对 AOF 文件进行重写（rewrite）：创建一个新的 AOF 文件来代替原有的 AOF 文件，新 AOF 文件和原有 AOF 文件保存的数据库状态完全一样，但新 AOF 文件的体积小于等于原有 AOF 文件的体积。

2.2 典型应用场景描述

Redis 的一个典型应用场景就是在高并发环境。

由于集群方案即采用分布式 Memcache 结合 Mysql 集群，随着用户人群的增加无法应付日益增大的访问压力。此时，把 DBMS 作为主库将数据全部存放在磁盘中，主要负责写操作；而 Redis 作为高速数据查询从库，主要负责查操作，最大限度的存放近期热点数据，在节约成本的同时尽最大量提高效率。基于 Redis 的信息存储图，如下：

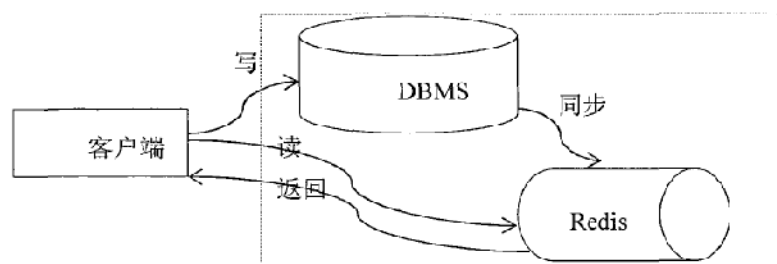


图 1

总体流程如下：用户发送http 请求，如果是写操作将直接操作关系型数据库如果是读操作则直接访问 Redis，并由 Redis 返回 http 请求结果。数据平台内部主要实现关系型数据库与 Redis 之间的数据同步一致性。

下面介绍一下 Redis 在实际中的使用情况，即新浪微博平台。每天超过 2200 亿命令操作、超过 5000 亿的读操作、500 亿的写操作、超过 18TB 内存、有 6 个

网络中心超过 500 服务器提供服务、同时运行超过 2000 个实例 ,是相当大的 Redis 使用平台。新浪微博采用的处理方式就是将 Memcache 和 MySql 全部替换为 Redis。Redis 作为存储替代 Mysql , 这样可以解决数据多份之间的一致性问题。同时可以通过修改 Redis 源码 , 一方面来满足自己的业务需求 ; 另一方面,完善 Redis 本身的缺陷。

3 功能需求

3.1 服务器模块

服务器模块主要实现了服务器启动时的初始化 ,接着进入事件轮询处理来自客户端的请求 ,最后关闭服务器。用例图如下 :

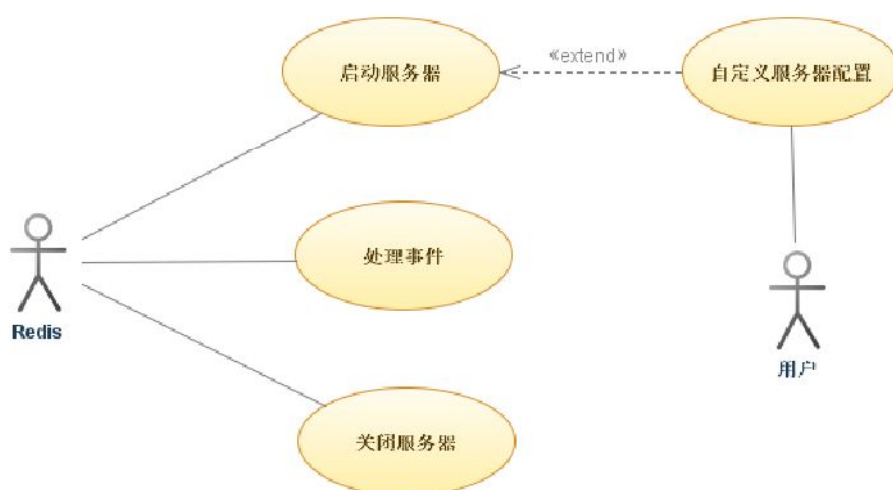


图 2

3.1.1 启动服务

Use Case Specification	
Use Case Name	启动服务器
Brief Description	Redis进行服务器的参数、启动模式等配置，并开启事件循环准备处理来自客户端的请求。
Precondition	None
Primary Actor	Redis
Secondary Actors	None
Dependency	EXTENDED BY USE CASE 自定义服务器配置
Generalization	None

Basic Flow	Steps
"main" ▼	1 Redis初始化服务器的运行库；
	2 Redis检查服务器的启动模式；
	3 Redis设置服务器的配置参数为默认值；
	4 IF <server.sentinel_mode> == true THEN
	5 Redis初始化Sentinel模式；
	6 服务器的配置参数被Sentinel模式的参数覆盖；
	7 ENDIF
	8 IF <argc> >= 2 THEN
	9 EXTENDED BY USE CASE 自定义服务器配置
	10 ENDIF
	11 Redis初始化服务器事件处理对象、数据库、socket等部分；
	12 IF <server.sentinel_mode> == false THEN
	13 Redis VALIDATES THAT AOF或RDB文件存在；
	14 Redis从AOF文件或者RDB文件中载入数据；
	15 ELSE
	16 Redis VALIDATES THAT sentinel mode的配置文件存在；
	17 Redis创建monitor事件；
	18 ENDIF
	19 Redis开启事件轮询；
Postcondition	服务器的初始化完成。 程序打开事件循环，开始接受客户端的连接。

Specific Alternative Flow	RFS 13
"abort" ▼	1 Redis提示未找到AOF或RDB文件；
	2 ABORT
Postcondition	服务器未启动。

Specific Alternative Flow	RFS 16
"abort" ▼	1 Redis提示未找到sentinel mode的配置文件文件；
	2 ABORT
Postcondition	服务器未启动。

3.1.2 自定义服务器配置

Use Case Specification	
Use Case Name	自定义服务器配置
Brief Description	用户通过输入配置指令修改服务器的配置参数。
Precondition	服务器使用默认值完成配置参数的初始化。
Primary Actor	用户
Secondary Actors	Redis
Dependency	EXTENDED BY USE CASE 自定义服务器配置
Generalization	None
Basic Flow	Steps
"main" ▼	1 用户输入配置指令；
	2 Redis VALIDATES THAT 配置指令；
	3 Redis处理配置指令中的特殊选项；
	4 Redis解析配置文件的相关信息；
	5 Redis重置保存服务器参数的条件；
	6 Redis获取配置文件的路径；
	7 Redis载入用户指定的配置文件；
	8 Redis把服务器中的配置参数设置为配置文件中对应参数的值；
Postcondition 服务器中的配置参数被修改。	
Specific Alternative Flow	RFS 2
"abort" ▼	1 Redis提示输入的配置命令有误；
	2 ABORT
Postcondition 服务器中的配置参数未被修改。	

3.1.3 处理事件

Use Case Specification	
Use Case Name	处理事件
Brief Description	服务器启动后，Redis进入事件轮询，处理来自客户端的请求。
Precondition	服务器已启动。
Primary Actor	Redis
Secondary Actors	None
Dependency	None
Generalization	None

Basic Flow	Steps
"main" ▼	1 Redis设置每次进行事件处理前会执行的回调函数beforeSleep;
	2 Redis进入事件轮询;
	3 Redis把eventLoop->stop设置为0;
	4 DO
	5 IF <eventLoop->beforesleep> != NULL THEN
	6 Redis执行回调函数beforeSleep;
	7 ENDIF
	8 IF<flags> == 0 THEN
	9 RESUME STEP 4
	10 ENDIF
	11 IF (<flags> & AE_TIME_EVENTS) && !(<flags> & AE_DONT_WAIT) THEN
	12 Redis获取最近的时间事件;
	13 IF 最近的时间事件存在 THEN
	14 Redis根据最近可执行的时间事件和当前时间的时间差来确定文件事件的阻塞时间;
	15 ELSE
	16 Redis根据AE_DONT_WAIT标识来确定文件事件的阻塞时间;
	17 ENDIF
	18 Redis处理文件事件;
	19 ENDIF
	20 IF <flags> & AE_TIME_EVENTS THEN
	21 Redis处理时间事件;
	22 ENDIF
	23 UNTIL eventLoop->stop != 0
	Postcondition Redis处理服务器的内部时间事件和客户端请求事件，并返回处理结果。

3.1.4 关闭服务器

Use Case Specification	
Use Case Name	关闭服务器
Brief Description	Redis退出事件轮询，删除事件轮询所占用的内存。服务器关闭。
Precondition	Redis收到关闭服务器指令或<event->stop>值为1。
Primary Actor	Redis
Secondary Actors	None
Dependency	None
Generalization	None
Basic Flow	Steps
"main" ▼	1 Redis退出事件轮询；
	2 Redis释放多路复用库的私有数据；
	3 Redis删除已注册的文件事件；
	4 Redis删除已就绪的文件事件；
	5 Redis删除事件轮询；
	6 服务器退出。
	Postcondition 服务器关闭。

3.2 客户端模块

Redis 服务器负责与客户端建立网络连接，处理客户端发送的命令请求。客户端首先发送命令请求给服务器端，服务器读取命令请求并调用命令执行器执行命令请求，最后将请求结果返回给客户端，完成与客户端的网络连接与通信。

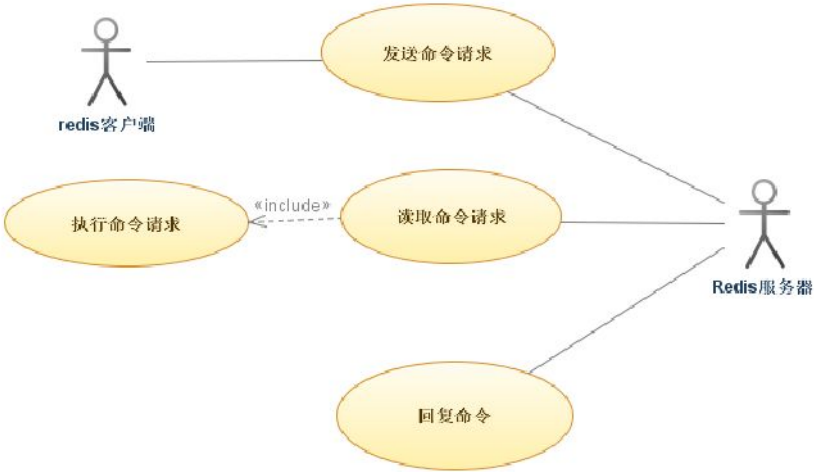


图 3

3.2.1 发送命令请求

Use Case Specification	
Use Case Name	发送命令请求
Brief Description	Redis 服务器的命令请求来自 Redis 客户端，客户端发送命令请求要转换成协议格式
Precondition	None
Primary Actor	Redis客户端 
Secondary Actors	Redis服务器
Dependency	None
Generalization	None
Basic Flow	Steps
"main" ▼	1 客户端收到来自用户的命令请求；
	2 客户端会将这个命令请求转换成协议格式；
	3 通过连接到服务器的套接字， 将协议格式的命令请求发送给服务器；
	Postcondition 命令协议格式转换完成， 协议格式发送到服务器

3.2.2 读取命令请求

Use Case Specification	
Use Case Name	读取命令请求
Brief Description	服务器读取来自客户端发送过来的协议格式
Precondition	当客户端与服务器之间的连接套接字因为客户端的写入而变得可读时
Primary Actor	Redis服务器
Secondary Actors	None
Dependency	INCLUDE USE CASE 执行命令请求
Generalization	None
Basic Flow	Steps
"main" ▼	1 读取套接字中协议格式的命令请求；
	2 将命令请求保存到客户端状态的输入缓冲区里面；
	3 提取出命令请求中包含的命令参数和参数个数；
	4 将参数和参数个数保存到客户端状态的 argv 属性和 argc 属性里；
	5 调用命令执行器执行客户端指定的命令；
	Postcondition 命令请求读取完毕，请求保存在各属性中

3.2.3 执行命令请求

Use Case Specification	
Use Case Name	执行命令请求
Brief Description	执行服务器端接收到的命令
Precondition	服务器已经将执行命令所需的命令实现函数、参数、参数个数都收集齐
Primary Actor	Redis服务器
Secondary Actors	None
Dependency	None
Generalization	None

Basic Flow "main" ▼	Steps	
	1	获取执行命令所需的命令实现函数、参数、参数个数；
	2	Redis VALIDATES THAT 客户端状态不为NULL；
	3	Redis VALIDATES THAT 参数个数正确；
	4	Redis VALIDATES THAT 客户端通过身份验证；
	5	Redis 查找命令对应的实现；
	6	Redis 调用命令的实现函数；
	7	IF 服务器开启了慢查询日志功能 THEN
	8	慢查询日志模块会检查是否需要为刚刚执行完的命令请求添加一条新的慢查询日志； ELSE
	9	ENDIF
	10	IF 服务器开启了 AOF 持久化功能 THEN
	11	AOF 持久化模块会将刚刚执行的命令请求写入到 AOF 缓冲区里；
	12	ENDIF
	13	IF 有其他从服务器正在复制当前这个服务器 THEN
	14	服务器会将刚刚执行的命令传播给所有从服务器
	15	ENDIF
	Postcondition	命令执行成功，服务器可以执行下一命令

Specific Alternative Flow "abort" ▼	RFS 2	
	1	向客户端返回一个错误
	Postcondition	服务器不再执行命令

Specific Alternative Flow "abort" ▼	RFS 3	
	1	向客户端返回一个错误
	Postcondition	服务器不再执行命令

Specific Alternative Flow "abort" ▼	RFS 4	
	1	向客户端返回一个错误
	Postcondition	服务器不再执行命令

3.2.4 回复命令

Use Case Specification	
Use Case Name	回复命令
Brief Description	服务器执行命令回复器，将保存在客户端输出缓冲区中的命令回复发送给客户端
Precondition	客户端套接字变为可写状态
Primary Actor	Redis服务器
Secondary Actors	None
Dependency	None
Generalization	None
Basic Flow	Steps
"main" ▾	1 命令回复处理器会将协议格式的命令回复发送给客户端；
	2 客户端接收协议格式的命令回复；
	3 将回复转换成人类可读的格式；
	Postcondition 命令回复完成

3.3 RDB 持久化模块

RDB 持久化主要负责，在 Redis 运行时，将当前内存中的数据库快照保存到磁盘文件中；在 Redis 重新启动时，RDB 程序可以通过载入 RDB 文件来还原数据库的状态。

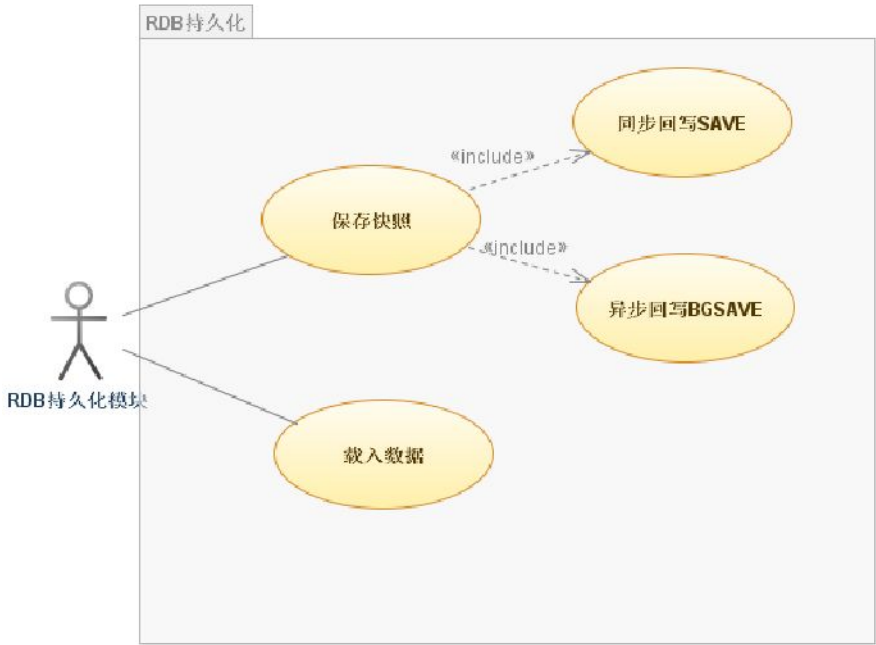


图 4

3.3.1 保存快照

快照是 Redis 默认的持久化方式。这种方式是就是将内存中数据以快照的方式写入到二进制文件中,默认的文件名为 dump.rdb。

Use Case Specification		
Use Case Name	保存快照	
Brief Description	RDB持久化模块将当前内存中的数据库快照保存到磁盘文件中,存为.rdb格式文件	
Precondition	Redis正在运行且未断电	
Primary Actor	RDB持久化模块	
Secondary Actors	None	
Dependency	INCLUDE USE CASE SAVE保存方式, INCLUDE USE CASE BGSAVE保存方式	
Generalization	None	

Basic Flow	Steps	
(Untitled) ▼	1	RDB模块读取配置文件;
	2	IF 配置为RDB持久化方式 THEN
	3	IF 成功打开.rdb文件 THEN
	4	服务器检查和保存校验码CHECK-SUM到.rdb文件末尾;
	5	读取文件所使用的RDB版本号,选择不同读写方式;
	6	DO
	7	IF 文件长度大于0 THEN
	8	读取键过期时间;
	9	读取值所使用编码;
	10	根据读取的指示,保存数据类型的键值对到临时文件中;
	11	ENDIF
	12	UNTIL 达到最大的server.dbnum
	13	ENDIF
	14	IF .rdb文件已存在 THEN
	15	用临时文件内容替换已有.rdb文件;
	16	ELSE
	17	通过原子性rename系统调用将临时文件重命名为.rdb文件;
	18	ENDIF
	19	ENDIF
Postcondition		内存数据被保存到磁盘中。

3.3.2 同步回写 SAVE

Use Case Specification		
Use Case Name	同步回写SAVE	
Brief Description	保存快照的一种方式，Redis主进程直接调用rdbSave，执行期间被阻塞	
Precondition	Redis配置文件开启RDB方式，手动调用SAVE命令，服务器未执行BGSAVE	
Primary Actor	RDB持久化模块	
Secondary Actors	None	
Dependency	None	
Generalization	None	

Basic Flow (Untitled) ▼	Steps	
	1	RDB模块收到SAVE命令；
	2	IF BGSAVE不在执行当中 THEN
	3	RDB模块读取save参数定义快照周期；
	4	IF 时间事件满足快照周期条件 THEN
	5	调用rdbSave进行保存快照，阻塞主进程；
	6	ENDIF
	7	ENDIF
	Postcondition	以快照周期进行.rdb文件保存

3.3.3 异步回写 BGSAVE

Use Case Specification	
Use Case Name	异步回写BGSAVE
Brief Description	保存快照的一种方式，利用子进程调用rdbSave，主进程不阻塞
Precondition	Redis配置文件开启RDB方式，默认或手动调用BGSAVE命令
Primary Actor	None
Secondary Actors	None
Dependency	None
Generalization	None

Basic Flow	Steps
(Untitled) ▼	1 RDB模块 VALIDATES THAT 当前没有SAVE或BGSAVE正在执行，Redis重新开始接受请求；
	2 IF BGSAVE正在被执行 THEN
	3 BGREWRITEAOF的重写请求被延迟；
	4 执行BGREWRITEAOF命令的客户端收到请求被延迟回复；
	5 ENDIF
	6 IF BGREWRITEAOF正在被执行 THEN
	7 BGSAVE客户端收到出错信息；
	8 ENDIF
	9 fork一个子进程，调用rdbSave；
	10 保存完毕向主进程发送通知；
	Postcondition .rdb文件被保存，通知Redis服务器仍可继续处理客户端请求

3.3.4 载入数据

当 Redis 服务器启动时，执行载入函数 rdbLoad，读取 RDB 文件，并将文件中的数据库数据载入到内存中。

Use Case Specification		
Use Case Name	载入数据	
Brief Description	读取.rdb文件，将文件中数据库数据载入到内存	
Precondition	Redis服务器启动	
Primary Actor	RDB持久化模块	
Secondary Actors	None	
Dependency	None	
Generalization	None	

Basic Flow	Steps	
(Untitled) ▼	1	打开.rdb文件；
	2	检查版本号；
	3	将服务器调整到开始载入状态；
	4	读入类型指示，决定该如何读入之后的数据；
	5	读入过期时间值；
	6	读入切换数据库指示；
	7	读入键；
	8	将键值对关联到数据库中；
	9	设置过期时间；
	10	关闭.edb文件；
	11	服务器从载入状态退出；
	Postcondition	将数据从.rdb文件载入内存。

3.4 AOF 持久化模块

AOF 持久化模块包括命令同步、AOF 文件读取和数据还原、AOF 后台重写等功能。而命令同步包含命令传播、缓存追加、文件写入和保存等过程。

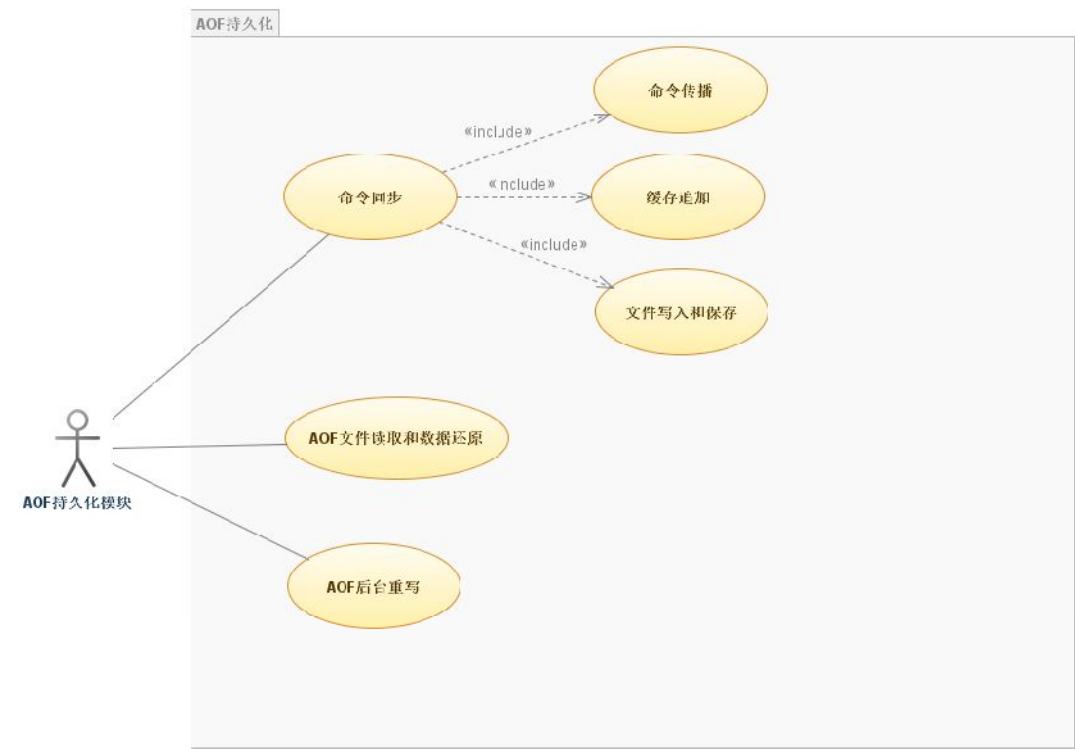


图 5

3.4.1 命令同步

Use Case Specification	
Use Case Name	命令同步
Brief Description	Redis 将所有对数据库进行写入的命令（及其参数）记录到 AOF 文件，以此达到记录数据库状态的目的。为了方便起见，我们称呼这种记录过程为命令同步。
Precondition	Redis 已经获取执行完的命令、命令的参数、命令的参数个数等信息
Primary Actor	AOF持久化模块
Secondary Actors	None
Dependency	INCLUDE USE CASE AOF持久化::命令传播, INCLUDE USE CASE AOF持久化::缓存追加, INCLUDE USE CASE AOF持久化::文件写入和保存
Generalization	None
Basic Flow	Steps
(Untitled) ▼	1 INCLUDE USE CASE AOF持久化::命令传播
	2 INCLUDE USE CASE AOF持久化::缓存追加
	3 INCLUDE USE CASE AOF持久化::文件写入和保存
	Postcondition 所有对数据库进行写入的命令（及其参数）记录到AOF文件中

3.4.2 命令传播

Use Case Specification																					
Use Case Name	命令传播																				
Brief Description	Redis 将执行完的命令、命令的参数、命令的参数个数等信息发送到 AOF 程序中。																				
Precondition	Redis 已经获取执行完的命令、命令的参数、命令的参数个数等信息																				
Primary Actor	ACF持久化模块																				
Secondary Actors	None																				
Dependency	None																				
Generalization	None																				
Basic Flow (Untitled) ▼	<table><tr><th colspan="2">Steps</th></tr><tr><td>1</td><td>命令执行成功</td></tr><tr><td>2</td><td>IF ACF功能已打开 THEN</td></tr><tr><td>3</td><td>传播命令到AOF 程序</td></tr><tr><td>4</td><td>ENDIF</td></tr><tr><td>5</td><td>IF REPLICATION功能已打开 THEN</td></tr><tr><td>6</td><td>传播命令到REPLICATION 程序</td></tr><tr><td>7</td><td>ENDIF</td></tr><tr><td>8</td><td>清理资源</td></tr><tr><td>Postcondition</td><td>命令、命令参数及其个数等信息被送到AOF程序</td></tr></table>	Steps		1	命令执行成功	2	IF ACF功能已打开 THEN	3	传播命令到AOF 程序	4	ENDIF	5	IF REPLICATION功能已打开 THEN	6	传播命令到REPLICATION 程序	7	ENDIF	8	清理资源	Postcondition	命令、命令参数及其个数等信息被送到AOF程序
Steps																					
1	命令执行成功																				
2	IF ACF功能已打开 THEN																				
3	传播命令到AOF 程序																				
4	ENDIF																				
5	IF REPLICATION功能已打开 THEN																				
6	传播命令到REPLICATION 程序																				
7	ENDIF																				
8	清理资源																				
Postcondition	命令、命令参数及其个数等信息被送到AOF程序																				

3.4.3 缓存追加

Use Case Specification											
Use Case Name	缓存追加										
Brief Description	AOF 程序根据接收到的命令数据，将命令转换为网络通讯协议的格式，然后将协议内容追加到服务器的 AOF 缓存中。										
Precondition	命令及其信息被传播到AOF程序										
Primary Actor	AOF持久化模块										
Secondary Actors	None										
Dependency	None										
Generalization	None										
Basic Flow (Untitled) ▼	<table><tr><th colspan="2">Steps</th></tr><tr><td>1</td><td>接受命令、命令的参数、以及参数的个数、所使用的数据库等信息</td></tr><tr><td>2</td><td>将命令还原成 Redis 网络通讯协议</td></tr><tr><td>3</td><td>将协议文本追加到 aof_buf 末尾</td></tr><tr><td>Postcondition</td><td>命令被转为网络通讯协议的格式追加到服务器的AOF缓存中</td></tr></table>	Steps		1	接受命令、命令的参数、以及参数的个数、所使用的数据库等信息	2	将命令还原成 Redis 网络通讯协议	3	将协议文本追加到 aof_buf 末尾	Postcondition	命令被转为网络通讯协议的格式追加到服务器的AOF缓存中
Steps											
1	接受命令、命令的参数、以及参数的个数、所使用的数据库等信息										
2	将命令还原成 Redis 网络通讯协议										
3	将协议文本追加到 aof_buf 末尾										
Postcondition	命令被转为网络通讯协议的格式追加到服务器的AOF缓存中										

3.4.4 文件写入和保存

Use Case Specification	
Use Case Name	文件写入和保存
Brief Description	AOF 缓存中的内容被写入到 AOF 文件末尾
Precondition	命令已被转为网络通讯协议的格式追加到服务器的AOF缓存中
Primary Actor	AOF持久化模块
Secondary Actors	None
Dependency	None
Generalization	None

Basic Flow (Untitled) ▼	Steps
	1 IF AOF保存模式为小保存 THEN
	2 IF 调用flushAppendOnlyFile 函数 THEN
	3 将 aof_buf 中的缓存写入到 AOF 文件
	4 END IF
	5 IF Redis 被关闭THEN
	6 调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中
	7 END IF
	8 IF AOF 功能被关闭THEN
	9 调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中
	10 END IF
	11 IF 系统的写缓存被刷新THEN
	12 调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中
	13 END IF

14 ELSEIF AOF保存模式为每一秒保存一次 THEN
15 IF 调用flushAppendOnlyFile 函数 THEN
16 IF 调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中正在执行THEN
17 IF 运行时间超过2秒THEN
18 执行将 aof_buf 中的缓存写入到 AOF 文件,但不执行新的调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中
19 ELSE
20 函数直接返回不执行将 aof_buf 中的缓存写入到 AOF 文件或新的调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中
21 END IF
22 ELSE
23 IF 距离上次调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中执行成功超过1 秒THEN
24 执行将 aof_buf 中的缓存写入到 AOF 文件和新调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中
25 ELSE
26 执行将 aof_buf 中的缓存写入到 AOF 文件但不执行新的调用 fsync 或 fdatasync 函数将 AOF 文件保存到磁盘中
27 END IF
28 END IF
29 ELSEIF AOF保存模式为每执行一个命令保存一次 THEN
30 执行将 aof_buf 中的缓存写入到 AOF 文件 和新的调用 fsync 或 fdatasync 函数, 将 AOF 文件保存到磁盘中
31 END IF
32 END IF
33

Postcondition	所有对数据库进行写入的命令（及其参数）记录到AOF文件中
---------------	------------------------------

3. 4. 5 AOF 文件读取和数据还原

Use Case Specification	
Use Case Name	AOF文件读取和数据还原
Brief Description	读取AOF 文件里的协议，重新执行一遍里面指示的所有命令，就可以还原Redis 的数据库状态
Precondition	AOF文件中已经保存了符合Redis 通讯协议格式的指令文本即是Redis数据库的状态
Primary Actor	AOF持久化模块
Secondary Actors	None
Dependency	None
Generalization	None
Basic Flow	Steps
(Untitled) ▼	1 打开并读取 AOF 文件
	2 DO
	3 读入一条协议文本格式的 Redis 命令
	4 根据文本命令，查找命令函数
	5 并创建参数和参数个数等对象
	6 执行命令
	7 UNTIL AOF 文件中的所有命令执行完毕
	8 关闭文件
Postcondition AOF 文件所保存的数据库就会被完整地还原出来	

3. 4. 6 AOF 后台重写

Use Case Specification	
Use Case Name	AOF后台重写
Brief Description	根据数据库键的类型，使用适当的写入命令来重现键的当前值，以节省AOF文件的存储空间
Precondition	AOF文件内容量过大
Primary Actor	AOF持久化模块
Secondary Actors	None
Dependency	None
Generalization	None
Basic Flow	Steps
(Untitled) ▼	1 子进程新建一个当前AOF文件的副本
	2 子进程执行新的副本AOF文件重写过程
	3 DO
	4 父进程处理命令请求
	5 父进程将写命令追加到现有的 AOF 文件中
	6 父进程将写命令追加到 AOF 重写缓存中
	7 UNTIL 子进程处理好AOF重写
	8 子进程向父进程发送一个完成信号
	9 父进程接收完成信号
	10 父进程将 AOF 重写缓存中的内容全部写入到新 AOF 文件中
	11 父进程对新的 ACF 文件进行改名，覆盖原有的 AOF 文件
Postcondition 重写后文件体积更小AOF文件	

4 非功能性需求分析

4.1 鲁棒性

鲁棒性就是系统的健壮性，指系统或组件在面对非法输入、相连接的系统或组件故障、或非预期的运行条件下能够持续正确运行的程度。

Redis 集群实现了分布式并且允许单点故障，当网络故障和节点发生故障时集群系统会尽力去保证数据的一致性和有效性。

Redis 主要使用主从同步的方式解决单点故障问题。此时，我们同时需要 masters 和 slaves。即使主节点(master)和从节点(slave)在功能上是一致的，甚至说他们部署在同一台服务器上，从节点也仅用以替代故障的主节点。实际上如果对从节点没有 read-after-write (写并立即读取数据以免在数据同步过程中无法获取数据) 的需求，那么从节点仅接受只读操作。

总之，在 Redis 集群系统中，节点有责任/义务保存数据和自身状态，这其中包括把数据 (key) 映射到正确的节点，所有节点都应该自动探测集群中的其他节点，并且在发现故障节点之后把故障节点的从节点更改为主节点。

4.2 安全性

Redis 通过如下方式保证系统的安全性：

1) 认证

Redis 提供了一个轻量级的认证方式，可以编辑 redis.conf 配置来启用认证。当授权方式启用时，redis 将会拒绝来自非认证用户的任何查询。用户可以通过发送 AUTH 命令并带上密码给 redis 服务器来给自己授权。系统管理员在 redis.conf 文件里以明文方式设置密码。而且密码必须足够长来抵御暴力破解密码方式的攻击。认证层的目标是提供多一层的保护。如果防火墙或者用来保护 redis 的系统防御外部攻击失败的话，外部用户如果没有通过密码认证还是无法访问 redis 的。

2) 禁用指定的命令

在 redis 你可以禁用某些命令或者将命令重命名成难以猜测的名字，这样一般的用户就只能使用部分命令了。

3) 代码安全

典型版本的 redis 中，客户端是可以使用全部命令的，但是访问 redis 实例时是没有能力控制运行着 redis 的系统的。Redis 使用好的编程方法来写代码，防止出现缓存溢出，格式错误和其他内存损坏问题。

4.3 效率

Redis 的高效率主要是因为以下几方面：

1) 自定义的数据结构

常用的数据结构和算法如字典、哈希算法、跳跃表和压缩链表，Redis 都提供了一套适合自己数据库的实现，提高了数据存储访问的效率

2) Redis 自己实现的事件分离器

Redis 的文件事件处理器以单线程的方式运行，避免了线程锁以及进程锁的开销，并且通过使用 I/O 复用程序来监听多个套接字，实现了高性能的网络通信模型。

3) 绝大部分请求是纯粹的内存操作

和大多 NoSQL 数据库一样，Redis 同样遵循了 Key/Value 数据存储模型。在有些情况下，Redis 会将 Keys/Values 保存在内存中以提高数据查询和数据修改的效率，然而这样的做法并非总是很好的选择。鉴于此，Redis 将之进一步优化，即尽量在内存中只保留 Keys 的数据，这样可以保证数据检索的效率，而 Values 数据在很少使用的时候则可以被换出到磁盘

5 参考文献

- [1] <http://redis.io/>
- [2] 黄健宏. Redis 设计与实现. 机械工业出版社[M]. 2014-06.
- [3] 张景云. 基于 Redis 的矢量数据组织研究[D]. 南京师范大学. 2013.
- [4] 白鑫. 基于 Redis 的信息存储优化技术研究与应用[D]. 北方工业大学. 2011.
- [5] 曾超宇，李金香. Redis 在高速缓存系统中的应用[J]. 微型机与应用，2013，32(12).