redis 第一课

内容：

Nosql 数据库概述

sql vs Nosql

Redis 概述

redis 安装

```
https://www.runoob.com/redis/redis-tutorial.html  菜鸟教程
```

一、Nosql 数据库概述

NoSQL（Not Only SQL）是**非关系型数据库**的统称，设计目标是解决大规模数据管理、高并发访问、灵活数据模型等场景下的传统关系型数据库瓶颈问题。其核心特征包括：

1. **灵活数据模型**
   支持键值对、文档、列族、图结构等非结构化或半结构化数据，无需预定义固定表结构（如 JSON、XML 格式）。例如：
   - **文档型**：MongoDB、Couchbase（存储 JSON 文档）
   - **键值型**：Redis、DynamoDB（高效缓存场景）
   - **列族型**：Cassandra、HBase（时序数据、日志分析）
   - **图数据库**：Neo4j（社交网络关系分析）
2. **高扩展性**
   通过**水平扩展**（分片、分布式集群）支持海量数据存储和 PB 级吞吐量，适合云计算和大数据场景 1。
3. **高可用性与容错**
   采用 CAP 定理中的 AP 或 CP 模型，通过副本机制保障系统可用性。例如 Cassandra 提供最终一致性，MongoDB 支持副本集自动故障转移。
4. **弱化事务支持**
   多数 NoSQL 数据库遵循 **BASE 原则**（Basically Available, Soft-state, Eventually Consistent），牺牲强一致性（ACID）以换取高性能。

二、sql vs Nosql

对比维度	SQL 数据库	NoSQL 数据库
数据模型	结构化表结构，严格遵循关系模型（行与列）	灵活的非结构化模型（文档、键值等）
扩展方式	垂直扩展（升级硬件）	水平扩展（分布式集群）
事务支持	强 ACID 事务（如银行交易）	多数仅支持单文档事务，部分支持多文档弱事务
查询语言	标准化 SQL 语法	无统一标准，API 或特定查询语法（如 MongoDB 的 BSON）
适用场景	复杂关联查询、高一致性需求（如 ERP 系统）	高吞吐量、动态结构（如物联网、实时分析）
典型产品	MySQL、PostgreSQL、Oracle	MongoDB、Redis、Cassandra

2.1选择建议

1. 优先 SQL 的场景
    - 需要复杂事务（如金融系统）
    - 数据关系复杂（多表 JOIN 查询）
    - 强一致性要求（如库存管理）
2. 优先 NoSQL 的场景
    - 数据模型频繁变化（如敏捷开发）
    - 高写入吞吐量（如日志收集）
    - 分布式云原生架构（如微服务缓存）

三、Redis 概述

Redis（Remote Dictionary Server）是一个开源的内存数据库，遵守 BSD 协议，它提供了一个高性能的键值（key-value）存储系统，常用于缓存、消息队列、会话存储等应用场景。

性能极高：Redis 以其极高的性能而著称，能够支持每秒数十万次的读写操作。这使得Redis成为处理高并发请求的理想选择，尤其是在需要快速响应的场景中，如缓存、会话管理、排行榜等。

丰富的数据类型：Redis 不仅支持基本的键值存储，还提供了丰富的数据类型，包括字符串、列表、集合、哈希表、有序集合等。这些数据类型为开发者提供了灵活的数据操作能力，使得Redis可以适应各种不同的应用场景。

原子性操作：Redis 的所有操作都是原子性的，这意味着操作要么完全执行，要么完全不执行。这种特性对于确保数据的一致性和完整性至关重要，尤其是在高并发环境下处理事务时。

持久化：Redis 支持数据的持久化，可以将内存中的数据保存到磁盘中，以便在系统重启后恢复数据。这为 Redis 提供了数据安全性，确保数据不会因为系统故障而丢失。

支持发布/订阅模式：Redis 内置了发布/订阅模式（Pub/Sub），允许客户端之间通过消息传递进行通信。这使得 Redis 可以作为消息队列和实时数据传输的平台。

单线程模型：尽管 Redis 是单线程的，但它通过高效的事件驱动模型来处理并发请求，确保了高性能和低延迟。单线程模型也简化了并发控制的复杂性。

主从复制：Redis 支持主从复制，可以通过从节点来备份数据或分担读请求，提高数据的可用性和系统的伸缩性。

应用场景广泛：Redis 被广泛应用于各种场景，包括但不限于缓存系统、会话存储、排行榜、实时分析、地理空间数据索引等。

社区支持：Redis 拥有一个活跃的开发者社区，提供了大量的文档、教程和第三方库，这为开发者提供了强大的支持和丰富的资源。

跨平台兼容性：Redis 可以在多种操作系统上运行，包括 Linux、macOS 和 Windows，这使得它能够在不同的技术栈中灵活部署。

四、redis 安装

4.1、rocky linux 9.5 安装

dnf -y install redis
systemctl enable redis &&systemctl start redis

验证，能链接上证明数据库安装完成

redis-cli [-h ip:6379]

配置文件：

/etc/redis/redis.conf

4.2、 源码安装

https://www.runoob.com/redis/redis-install.html

```
yum -y install gcc*  #安装源码安装的环境

wget https://download.redis.io/redis-stable.tar.gz
tar -xzvf redis-stable.tar.gz
cd redis-stable
make
make PREFIX=/usr/local/redis install
ln -s /usr/local/redis/bin/* /usr/local/bin/
```

启动redis 服务

```
[root@localhost ~]# netstat -naptl |grep redis
tcp        0      0 0.0.0.0:6379            0.0.0.0:*               LISTEN      10956/redis-server
tcp6       0      0 :::6379                 :::*                    LISTEN      10956/redis-server
[root@localhost ~]#
```

停止服务： pkill -9 redis

4.3、docker-compose 安装（后续讲完docker，自行尝试）

```yaml
version: '3.1'
services:
  redis:
    image: daocloud.io/library/redis:5.0.7
    restart: always
    container_name: redis
    environment:
      - TZ=Asia/Shanghai
    ports:
      - 6379:6379
```

docker run -it -p 6379:6376 redis:latest /bin/bash

练习：

安装redis 服务

五、redis-cli 命令工具

5.1连接本地数据库

```
[root@localhost utils]# /usr/local/redis/bin/redis-cli
127.0.0.1:6379>
```

5.2连接远程数据库

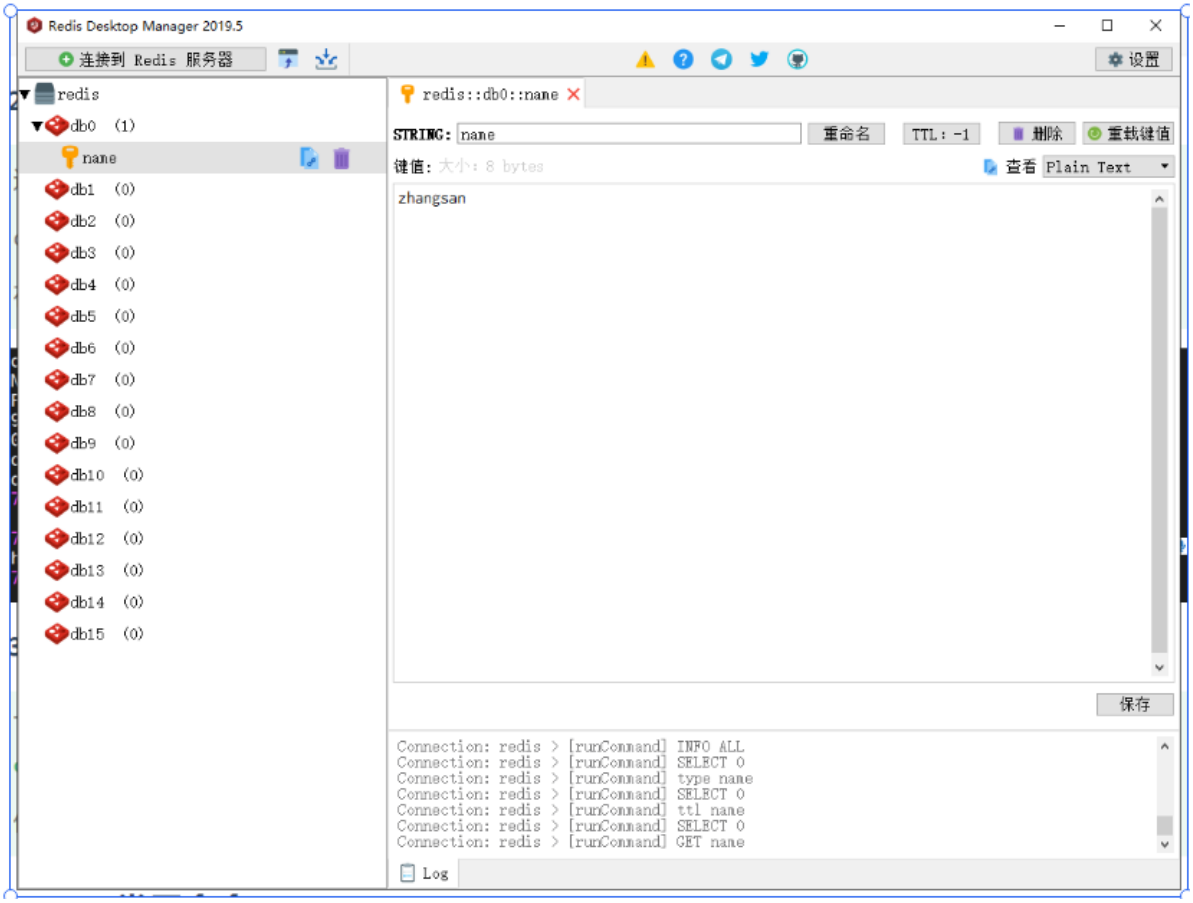
```
[root@localhost utils]# redis-cli -h 192.168.10.161 -p 6379
192.168.10.161:6379>
```

5.3使用图形化界面连接Redis

下载地址：https://github.com/lework/RedisDesktopManager-Windows/releases/download/2019.5/redis-desktop-manager-2019.5.zip

https://download.redisinsight.redis.com/latest/RedisInsight-v2-win-installer.exe

傻瓜式安装



第2次课

六、配置文件

Redis 的配置文件位于 Redis 安装目录下，文件名为 **redis.conf**(Windows 名为 redis.windows.conf)。

linux 中是 /etc/redis/redis.conf

你可以通过 **CONFIG** 命令查看或设置配置项。

语法

Redis CONFIG 命令格式如下：

`redis 127.0.0.1:6379> CONFIG GET CONFIG_SETTING_NAME`

实例

```
redis 127.0.0.1:6379> CONFIG GET loglevel

1) "loglevel"
2) "notice"
```

使用 * 号获取所有配置项：

实例

```
redis 127.0.0.1:6379> CONFIG GET *

  1) "dbfilename"
  2) "dump.rdb"
```

其中要知道的：

```
bind：监听的主机地址
port：端口
daemonize yes：启用守护进程
pidfile：指定PID文件
loglevel notice：日志级别
logfile：指定日志文件
```

https://www.runoob.com/redis/redis-conf.html 其他配置说明参考链接

序号	配置项	说明
1	`daemonize no`	Redis 默认不是以守护进程的方式运行，可以通过该配置项修改，使用 yes 启用守护进程（Windows 不支持守护线程的配置为 no ）
2	`pidfile /var/run/redis.pid`	当 Redis 以守护进程方式运行时，Redis 默认会把 pid 写入 /var/run/redis.pid 文件，可以通过 pidfile 指定
3	`port 6379`	指定 Redis 监听端口，默认端口为 6379，作者在自己的一篇博文中解释了为什么选用 6379 作为默认端口，因为 6379 在手机按键上 MERZ 对应的号码，而 MERZ 取自意大利歌女 Alessia Merz 的名字
4	`bind 127.0.0.1`	绑定的主机地址
5	`timeout 300`	当客户端闲置多长秒后关闭连接，如果指定为 0 ，表示关闭该功能
6	`loglevel notice`	指定日志记录级别，Redis 总共支持四个级别：debug、verbose、notice、warning，默认为 notice
7	`logfile stdout`	日志记录方式，默认为标准输出，如果配置 Redis 为守护进程方式运行，而这里又配置为日志记录方式为标准输出，则日志将会发送给 /dev/null
8	`databases 16`	设置数据库的数量，默认数据库为0，可以使用 SELECT 命令在连接上指定数据库id
9	`save <seconds> <changes>` Redis 默认配置文件中提供了三个条件：**save 900 1****save 300 10save 60 10000** 分别表示 900 秒（15 分钟）内有 1 个更改，300 秒（5 分钟）内有 10 个更改以及 60 秒内有 10000 个更改。	指定在多长时间内，有多少次更新操作，就将数据同步到数据文件，可以多个条件配合
10	`rdbcompression yes`	指定存储至本地数据库时是否压缩数据，默认为 yes，Redis 采用 LZF 压缩，如果为了节省 CPU 时间，可以关闭该选项，但会导致数据库文件变的巨大
11	`dbfilename dump.rdb`	指定本地数据库文件名，默认值为 dump.rdb
12	`dir ./`	指定本地数据库存放目录
13	`slaveof <masterip> <masterport>`	设置当本机为 slave 服务时，设置 master 服务的 IP 地址及端口，在 Redis 启动时，它会自动从 master 进行数据同步
14	`masterauth <master-password>`	当 master 服务设置了密码保护时，slave 服务连接 master 的密码

序号	配置项	说明
15	`requirepass foobared`	设置 Redis 连接密码，如果配置了连接密码，客户端在连接 Redis 时需要通过 AUTH 命令提供密码，默认关闭
16	`maxclients 128`	设置同一时间最大客户端连接数，默认无限制，Redis 可以同时打开的客户端连接数为 Redis 进程可以打开的最大文件描述符数，如果设置 maxclients 0，表示不作限制。当客户端连接数到达限制时，Redis 会关闭新的连接并向客户端返回 max number of clients reached 错误信息
17	`maxmemory <bytes>`	指定 Redis 最大内存限制，Redis 在启动时会把数据加载到内存中，达到最大内存后，Redis 会先尝试清除已到期或即将到期的 Key，当此方法处理后，仍然到达最大内存设置，将无法再进行写入操作，但仍然可以进行读取操作。Redis 新的 vm 机制，会把 Key 存放内存，Value 会存放在 swap 区
18	`appendonly no`	指定是否在每次更新操作后进行日志记录，Redis 在默认情况下是异步的把数据写入磁盘，如果不开启，可能会在断电时导致一段时间内的数据丢失。因为 redis 本身同步数据文件是按上面 save 条件来同步的，所以有的数据会在一段时间内只存在于内存中。默认为 no
19	`appendfilename appendonly.aof`	指定更新日志文件名，默认为 appendonly.aof
20	`appendfsync everysec`	指定更新日志条件，共有 3 个可选值：**no**：表示等操作系统进行数据缓存同步到磁盘（快）**always**：表示每次更新操作后手动调用 fsync() 将数据写到磁盘（慢，安全）**everysec**：表示每秒同步一次（折中，默认值）
21	`vm-enabled no`	指定是否启用虚拟内存机制，默认值为 no，简单的介绍一下，VM 机制将数据分页存放，由 Redis 将访问量较少的页即冷数据 swap 到磁盘上，访问多的页面由磁盘自动换出到内存中（在后面的文章我会仔细分析 Redis 的 VM 机制）
22	`vm-swap-file /tmp/redis.swap`	虚拟内存文件路径，默认值为 /tmp/redis.swap，不可多个 Redis 实例共享
23	`vm-max-memory 0`	将所有大于 vm-max-memory 的数据存入虚拟内存，无论 vm-max-memory 设置多小，所有索引数据都是内存存储的(Redis 的索引数据 就是 keys)，也就是说，当 vm-max-memory 设置为 0 的时候，其实是所有 value 都存在于磁盘。默认值为 0

序号	配置项	说明
24	`vm-page-size 32`	Redis swap 文件分成了很多的 page，一个对象可以保存在多个 page 上面，但一个 page 上不能被多个对象共享，vm-page-size 是要根据存储的 数据大小来设定的，作者建议如果存储很多小对象，page 大小最好设置为 32 或者 64bytes；如果存储很大大对象，则可以使用更大的 page，如果不确定，就使用默认值
25	`vm-pages 134217728`	设置 swap 文件中的 page 数量，由于页表（一种表示页面空闲或使用的 bitmap）是在放在内存中的，，在磁盘上每 8 个 pages 将消耗 1byte 的内存。
26	`vm-max-threads 4`	设置访问swap文件的线程数,最好不要超过机器的核数,如果设置为0,那么所有对swap文件的操作都是串行的，可能会造成比较长时间的延迟。默认值为4
27	`glueoutputbuf yes`	设置在向客户端应答时，是否把较小的包合并为一个包发送，默认为开启
28	`hash-max-zipmap-entries 64` `hash-max-zipmap-value 512`	指定在超过一定的数量或者最大的元素超过某一临界值时，采用一种特殊的哈希算法
29	`activerehashing yes`	指定是否激活重置哈希，默认为开启（后面在介绍 Redis 的哈希算法时具体介绍）
30	`include /path/to/local.conf`	指定包含其它的配置文件，可以在同一主机上多个Redis实例之间使用同一份配置文件，而同时各个实例又拥有自己的特定配置文件

6.1编辑配置

你可以通过修改 redis.conf 文件或使用 **CONFIG set** 命令来修改配置。

语法

CONFIG SET 命令基本语法：

```
redis 127.0.0.1:6379> CONFIG SET CONFIG_SETTING_NAME NEW_CONFIG_VALUE
```

实例

```
redis 127.0.0.1:6379> CONFIG SET loglevel "notice"
OK
redis 127.0.0.1:6379> CONFIG GET loglevel

1) "loglevel"
2) "notice"
```

注：上面的方式命令修改配置文件是临时的，重启会失效！

6.2将修改的配置保存到配置文件

Redis 提供了 `CONFIG REWRITE` 命令，可以将通过 `CONFIG SET` 修改的配置信息持久化到配置文件（`redis.conf`）。

具体步骤：

1. 使用redis-cli修改配置：

```
bash

redis-cli config set protected-mode  no
```

2. 将修改保存到配置文件：

```
bash

redis-cli config rewrite
```

   - 该命令会将当前配置写入到 `redis.conf` 文件中。
   - 如果配置文件路径不是默认的 `/etc/redis/redis.conf`，可以通过 `--config` 参数指定配置文件路径。
3. 验证配置文件： 打开redis.conf文件，检查修改的配置是否已生效。例如：

```
[root@localhost ~]# cat /etc/redis/redis.conf |grep protected
# When protected mode is on and if:
# By default protected mode is enabled. You should disable it only if
protected-mode no
# If the master is password protected (using the "requirepass" configuration
[root@localhost ~]#
```

6.3. 注意事项

- **权限问题**：确保 Redis 进程对配置文件有写权限，否则 `CONFIG REWRITE` 会失败。
- **配置文件格式**：`CONFIG REWRITE` 会保留原始配置文件的格式和注释。
- **重启 Redis**：虽然配置已保存到文件，但需要重启 Redis 服务才能使部分配置生效。

七、redis 工具介绍

7.1Redis 软件提供了多个命令工具。

安装 Redis 服务时，所包含的软件工具会同时被安装

到系统中，在系统中可以直接使用。这些命令工具的作用分别如下所示。

 redis-server：用于启动 Redis 的工具；

 redis-benchmark：用于检测 Redis 在本机的运行效率；

 redis-check-aof：修复 AOF 持久化文件；

 redis-check-rdb：修复 RDB 持久化文件；

 redis-cli：Redis 命令行工具。

本小节只介绍 redis-cli、redis-benchmark 命令工具的使用。

7.2Redis 服务的客户端软件就是其自带的 redis-cli 命令行工具。

使用 redis-cli 连接指定数据库，连接成功后会进入提示符为“远程主机 IP 地址：端口号>”的数据库操作环境，例如“127.0.0.1:6379>”。

用户可以输入各种操作语句对数据库进行管理。如执行 ping 命令可以检测 Redis 服务是否启动。

```
[root@localhost ~]# /usr/local/redis/bin/redis-cli //连接本机 Redis 数据库

127.0.0.1:6379>ping //检测 redis 服务是否启动

PONG

127.0.0.1:6379>
```

在进行数据库连接操作时，可以通过选项来指定远程主机上的 Redis 数据库。

命令语法为 redis-cli -h host -p port -a password，其中-h 指定远程主机、-p 指定 Redis 服务的端口号、-a 指定密码。若不添加任何选项表示，连接本机上的 Redis 数据库；若未设置数据库密码可以省略-a 选项。例如执行以下命令可连接到主机为 192.168.10.161，端口为 6379的 Redis 数据库，并查看 Redis 服务的统计信息。若要退出数据库操作环境，执行“exit”或“quit”命令即可返还原来的 Shell 环境。

```
[root@localhost ~]#redis-cli -h 192.168.10.161 -p 6379

192.168.10.161:6379>info

\# Server

redis_version:4.0.9

redis_git_sha1:00000000

redis_git_dirty:0

redis_build_id:7f55f2c1a630cbe5

……

//省略部分内容

192.168.10.161:6379>exit

[root@localhost ~]#
```

为了redis的安全，我们可以通过 redis 的配置文件设置密码参数，这样客户端连接到 redis 服务就需要密码验证，这样可以让你的 redis 服务更安全。

实例

我们可以通过以下命令查看是否设置了密码验证：

```
127.0.0.1:6379> CONFIG get requirepass
1) "requirepass"
2) ""
```

默认情况下 requirepass 参数是空的，这就意味着你无需通过密码验证就可以连接到 redis 服务。

你可以通过以下命令来修改该参数：

```
127.0.0.1:6379> CONFIG set requirepass "gyh"
OK
127.0.0.1:6379> CONFIG get requirepass
1) "requirepass"
2) "gyh"
```

设置密码后，客户端连接 redis 服务就需要密码验证，否则无法执行命令。

语法

AUTH 命令基本语法格式如下：

```
127.0.0.1:6379> AUTH password
```

实例

```
[root@localhost ~]# redis-cli
127.0.0.1:6379> get kk
(error) NOAUTH Authentication required.
127.0.0.1:6379> auth gyh
OK
127.0.0.1:6379> get kk
(nil)
127.0.0.1:6379> quit
[root@localhost ~]#
```

7.3在数据库操作环境中，使用 help 命令可以获取命令类型的帮助。

有三种获取命令帮助的方式。

```
□ help @<group>：获取<group>中的命令列表；

□ help <command>：获取某个命令的帮助；

□ help <tab>：获取可能帮助的主题列表。
```

具体操作方法如下所示。

```
[root@localhost ~]#redis-cli

127.0.0.1:6379>help @list //查看所有与 List 数据类型的相关命令

BLPOP key [key ...] timeout

summary: Remove and get the first element in a list, or block until one is
available

since: 2.0.0
```


```
[root@localhost ~]#redis-cli

127.0.0.1:6379>help @list //查看所有与 List 数据类型的相关命令

BLPOP key [key ...] timeout

summary: Remove and get the first element in a list, or block until one is
available

since: 2.0.0
```


7.4redis-benchmark 测试工具

redis-benchmark 是官方自带的 Redis 性能测试工具，可以有效的测试 Redis 服务的性能。基本的测试语法为 redis-benchmark [option] [option value]。常用选项如下所示。

```
□ -h：指定服务器主机名；

□ -p：指定服务器端口；

□ -s：指定服务器 socket；

□ -c：指定并发连接数；

□ -n：指定请求数；

□ -d：以字节的形式指定 SET/GET 值的数据大小；

□ -k：1=keep alive 0=reconnect；

□ -r：SET/GET/INCR 使用随机 key，SADD 使用随机值；

□ -P：通过管道传输<numreq>请求；

□ -q：强制退出 redis。仅显示 query/sec 值；

□ --csv：以 CSV 格式输出；□ -l：生成循环，永久执行测试；

□ -t：仅运行以逗号分隔的测试命令列表；

□ -I：Idle 模式。仅打开 N 个 idle 连接并等待。
```

结合上述选项，可以针对某台 Redis 服务器进行性能检测，如执行 redis-benchmark -h 192.168.10.161 -p 6379 -c 100 -n 100000 命令即可向 IP 地址为 192.168.10.161、端口为6379 的 Redis 服务器发送 100 个并发连接与 100000 个请求测试性能。

```
[root@localhost ~]#redis-benchmark -h 192.168.8.100 -p 6379 -c 100 -n 100000

..... //省略部分内容

8225.04 requests per second

====== MSET (10 keys) ======

100000 requests completed in 1.57 seconds

100 parallel clients

3 bytes payload

keep alive: 1

24.75% <= 1 milliseconds

99.02% <= 2 millisecon
```

执行 redis-benchmark -h 192.168.10.161 -p 6379 -q -d 100 命令的作用是测试存取大小为 100 字节的数据包的性能。

```
[root@localhost ~]#redis-benchmark -h 192.168.10.161 -p 6379 -q -d 100

PING_INLINE: 88261.25 requests per second

PING_BULK: 90991.81 requests per second

SET: 83612.04 requests per second

GET: 84961.77 requests per second

INCR: 83682.01 requests per second
```

还可以测试某些操作的性能，例如执行 redis-benchmark -t set,lpush -n 100000 -q 命令的作用是测试本机上 Redis 服务在进行 set 与 lpush 操作时的性能。

```
[root@localhost ~]#redis-benchmark -t set,lpush -n 100000 -q

SET: 85763.29 requests per second

LPUSH: 86580.09 requests per second
```

练习：

将讲的指令敲一遍

八、key相关命令

```
keys：获取符合规则的键值列表
exists：判断键值是否存在
del：删除当前数据库的指定key
type：获取key对应的value值类型
rename(覆盖）/ renamenx（不覆盖）：对已有的key进行重命名
dbsize：查看当前数据库中key的数目
```

在 Redis 数据库中，与 key 相关的命令主要包含以下几种。

（1）keys

使用 keys 命令可以取符合规则的键值列表，通常情况可以结合*、？等选项来使用。

```
127.0.0.1:6379>set k1 1
OK
127.0.0.1:6379>set k2 2
OK
127.0.0.1:6379>set k3 3
OK
127.0.0.1:6379>set v1 4
OK
127.0.0.1:6379>set v5 5
OK
127.0.0.1:6379>KEYS * //查看当前数据库中所有键

1) "teacher"

2) "k1"

3) "k2"

4) "k3"

5) "v1"

6) "v5"

127.0.0.1:6379>set v22 5

OK

127.0.0.1:6379>KEYS v* //查看当前数据库中以 v 开头的数据

1) "v1"

2) "v5"

3) "v22"

127.0.0.1:6379>KEYS v?  //查看当前数据库中以 v 开头后面包含任意一位的数据

1) "v1"

2) "v5"

127.0.0.1:6379>**KEYS v??  //查看当前数据库中以 v 开头 v 开头后面包含任意两位的数据
```

```
1) "v22"
```

(2) exists

exists 命令可以判断键值是否存在。

```
127.0.0.1:6379>exists teacher   //判断 teacher 键是否存在

(integer) 1  //表示 teacher 键是存在

127.0.0.1:6379>exists tea

(integer) 0   //表示 tea 键不存在
```

(3) del

del 命令可以删除当前数据库的指定 key。

```
127.0.0.1:6379>keys *

1) "teacher"

2) "v1"

3) "v22"

4) "k3"

5) "k1"

6) "k2"

7) "v5"

127.0.0.1:6379> del v5

(integer) 1

127.0.0.1:6379>get v5

(nil)
```

(4) type

使用 type 命令可以获取 key 对应的 value 值类型。

```
127.0.0.1:6379>type k1

string
```

（5）rename

rename 命令是对已有 key 进行重命名，其命令格式为：rename 源 key 目标 key。使用 rename 命令进行重命名时，无论目标 key 是否存在都进行重命名，且源 key 的值会覆盖目标 key 的值。在实际使用过程中，建议先用 exists 命令查看目标 key 是否存在，然后再决定是否执行 rename 命令，以避免覆盖重要数据。

```
127.0.0.1:6379>keys v*

1) "v1"

2) "v22"

127.0.0.1:6379>rename v22 v2

OK

127.0.0.1:6379>keys v*

1) "v1"

2) "v2"

127.0.0.1:6379>get v1

"4"

127.0.0.1:6379>get v2

"5"

127.0.0.1:6379>rename v1 v2

OK

127.0.0.1:6379>get v1

(nil)

127.0.0.1:6379>get v2

"4"
```

（6）renamenx

renamenx 命令的作用是对已有 key 进行重命名，并检测新名是否存在。其命令格式与rename 的命令格式除命令关键字不同外基本相同：

renamenx 源 key 目标 key。使用renamenx 命令进行重命名时，如果目标 key 存在则不进行重命名。

```
127.0.0.1:6379>keys *

1) "teacher"

2) "k3"
```

```
3) "k1"

4) "k2"

5) "v2"

127.0.0.1:6379>get teacher

"zhanglong"

127.0.0.1:6379>get v2

"4"

127.0.0.1:6379>renamenx v2 teacher

(integer) 0

127.0.0.1:6379>keys *

1) "teacher"

2) "k3"

3) "k1"

4) "k2"

5) "v2"

127.0.0.1:6379>get teacher

"zhanglong"

127.0.0.1:6379>get v2

"4"
```

（7）dbsize

dbsize 命令的作用是查看当前数据库中 key 的数目。

```
127.0.0.1:6379> dbsize

(integer) 5
```

九、多数据库常用命令

（1）多数据库间切换

Redis 支持多数据库，Redis 在没有任何改动的情况下默认包含 16 个数据库，数据库名称是用数字 0-15 来依次命名的。使用 select 命令可以进行 Redis 的多数据库之间的切换，命令格式为 select index,其中 index 表示数据库的序号。而使用 redis-cli 连接 Redis 数据库后，默认使用的是序号为 0 的数据库。

如下所示，使用 select 命令切换数据库后，会在前端的提示符中显示当前所在的数据库序号如“127.0.0.1:6379[10]>”表示当前使用的是序号为 10 的数据库。若当前使用的数据库是序号为 0 的数据库，提示符中则不显示序号，如“127.0.0.1:6379>”表示当前使用的是序号为 0 的数据库。

```
127.0.0.1:6379>select 10      //切换至序号为 10 的数据库

OK

127.0.0.1:6379[10]>select 15   //切换至序号为 15 的数据库

OK

127.0.0.1:6379[15]>select 0     //切换至序号为 0 的数据库

OK

127.0.0.1:6379>
```

（2）多数据库间移动数据

Redis 的多数据库在一定程度上是相对独立的，例如在数据库 0 上面存放 k1 的数据，在其它 1-15 的数据库上是无法查看到的。

```
127.0.0.1:6379>set k1 100

OK

127.0.0.1:6379>get k1

"100"

127.0.0.1:6379>select 1

OK

127.0.0.1:6379[1]>get k1

(nil)
```

Redis 数据库提供了一个 move 的命令，可以进行多数据库之间的数据移动。命令的基本语法格式为“move key dbindex”。

其中“key”表示当前数据库的目标键，“dbindex”表示目标数据库的序号，具体操作方法如下所示。

```
127.0.0.1:6379[1]>select 0  //切换至目标数据库 0

OK

127.0.0.1:6379>get k1   //查看目标数据是否存在
```

```
"100"

127.0.0.1:6379>move k1 1    //将数据库 0 中 k1 移动到数据库 1 中

(integer) 1

127.0.0.1:6379>select 1 //切换至目标数据库 1

OK

127.0.0.1:6379[1]>get k1    //查看被移动数据

"100"

127.0.0.1:6379[1]> select 0

OK

127.0.0.1:6379> get k1  //在数据库 0 中无法查看到 k1 的值

(nil)
```

（3）清除数据库内数据

Redis数据库的整库数据删除主要分为两个部分：

清空当前数据库数据，使用 FLUSHDB命令实现；

清空所有数据库的数据，使用 FLUSHALL 命令实现。

但是，数据清空操作比较危险，生产环境下一般不建议使用。

练习：上面的操作