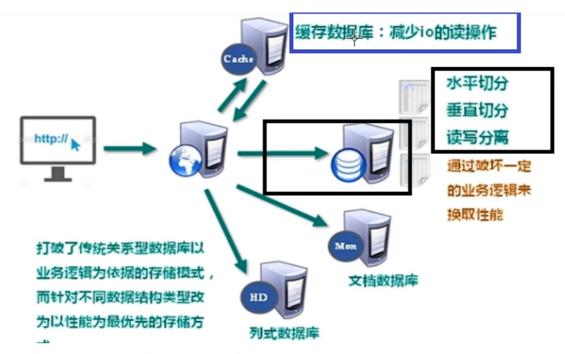
客户信息: session存储问题

存储到NOSQL安全和缓解了: CPU 和内存的压力 (能直接在内存读取)

解决了: IO压力



NoSQL数据库

_1.2.1. NoSQL 数据库概述。

NoSQL(NoSQL = Not Only SQL),意即"不仅仅是 SQL",泛指非关系型的数据库。
NoSQL 不依赖业务逻辑方式存储,而以简单的 key-value 模式存储。因此大大的增加了数据库的扩展能力。

- 不遵循 SQL 标准。』
- 不支持 ACID。
- 远超于 SQL 的性能。↓

nosql使用场景

- 对数据高并发的读写。
- 海量数据的读写。
- 对数据高可扩展性的。

redis官网: Redis

redis.io/

rides安装

终端:安装gcc

终端输入: make 编译成 c语言 (报错就输入 make distclean 清空编译文件)

安装: make install (默认安装目录 /usr/local/bin)

_**2.2.3**. 安装目录: /usr/local/bin-

查看默认安装目录: ₽

redis-benchmark:性能测试工具,可以在自己本子运行,看看自己本子性能如何。

redis-check-aof: 修复有问题的 AOF 文件, rdb和 aof 后面讲。

redis-check-dump:修复有问题的 dump.rdb 文件。I

redis-sentinel: Redis 集群使用。

redis-server: Redis 服务器启动命令4

redis-cli:客户端,操作入口。

后台启动

后台启动需要: redis.conf文件

复制文件:

cp(复制) 把redis.conf文件 复制到 etc目录

```
root@localhost bin]# cd /opt
[root@localhost opt]# ls
redis-6.2.1 redis-6.2.1.tar.gz
[root@localhost opt]# cd redis-6.2.1
[root@localhost redis-6.2.1]# ls
00-RELEASENOTES CONTRIBUTING INSTALL
                                         README.
                                                     runtest-cluster
                                                                        sentinel.conf
                                                                                       TLS.md
                                         redis.conf
BUGS
                COPYING
                              Makefile
                                                    runtest-moduleapi
                                                                                       utils
                                                                        SEC
CONDUCT
                              MANIFESTO
                                         runtest
                                                     runtest-sentinel
                                                                        tests
[root@localhost redis-6.2.1]# cp redis.conf[/etc/redis.conf
root@localhost redis-6.2.1]#
```

2.2.5.2.后台启动设置 daemonize no 改成 yes.

修改 redis.conf(128行)文件将里面的 daemonize no 改成 yes , 让服务在后台启动。

第一注意改的位置: 我们使用 vi命令 进入rider.cenf

```
depmod.d
                          ld.so.conf
                                                     protocols
                                                                            system-release-cpe
dhcp
DIR_COLORS
DIR_COLORS.256color
                                                                            terminfo
                          ld.so.conf.d
                                                     python
                          libaudit.conf
                                                     rc0.d
                                                                            tmpfiles.d
                          libnl
                                                     rc1.d
                                                                            tuned
DIR_COLORS.lightbgcolor
                          libuser.conf
                                                                            udev
                                                     rc2.d
dracut.conf
                                                                            vconsole.conf
                                                     rc3.d
                                                     rc4.d
dracut.conf.d
                          localtime
                                                                            vmware-tools
e2fsck.conf
                          login.defs
                                                     rc5.d
                                                                            wpa_supplicant
environment
                          logrotate.conf
                                                     rc6.d
                                                                            X11
ethertypes
                          logrotate.d
exports
                          lvm
                                                     rc.local
                                                                            xdg
favicon.png
                          machine-id
                                                     redhat-release
                                                                            xinetd.d
filesystems
                          magic
                                                                            yum
firewalld
                          makedumpfile.conf.sample
                                                     resolv.conf
fstab
                          man_db.conf
                                                                            yum.repos.d
                          mke2fs.conf
                                                     грш
                          modprobe.d
                                                     rsyslog.conf
gcrypt
 [root@localhost etc]# vi redis.conf
仅将文本发送到当前选项卡
```

搜索一下:

```
# 1mb => 1024*1024 bytes
# 1g => 10000000000 bytes
# 1gb => 1024*1024*1024 bytes
#
/daem

(仅将文本发送到当前选项卡
已连接192.168.44.168:22。
```

把 no --》yes

```
_ _ ×
                                        root@ni01:/etc
文件(F) 编辑(E) 查看(V) 搜索(S) 终端(T) 帮助(H)
# to unlimited size. The default size is 20480.
# tls-session-cache-size 5000
# Change the default timeout of cached TLS sessions. The default timeout is 300
# seconds
# tls-session-cache-timeout 60
# By default Redis does not run as a daemon. Use 'yes' if you need it.
# Note that Redis will write a pid file in /var/run/redis.pid when daemonized.
# When Redis is supervised by upstart or systemd, this parameter has no impact.
daemonize yes
# If you run Redis from upstart or systemd, Redis can interact with your
# supervision tree. Options:
                           - no supervision interaction
    supervised upstart - signal upstart by putting Redis into SIGSTOP mode requires "expect stop" in your upstart job config
    supervised systemd - signal systemd by writing READY = to $NOTIFY_SOCKET
                            on startup, and updating Redis status on a regular
```

把配置文件改好后: 就是启动配置文件

第一步: 先进入 cd /usr/local/bin

第二步: Is

```
[root@localhost etc]# /usr/local/bin
-bash: /usr/local/bin: 是一个目录
[root@localhost etc]# l

仅将文本发送到当前选项卡
```

启动命令: redis-server /etc/redis.conf

用命令查看启动状态: ps -ef | grep redis

测试连接客户端: redis-cli

```
[root@localhost bin]# redis-cli
127.0.0.1:6379> pint
(error) ERR unknown command `pint`, with args beginning with:
127.0.0.1:6379> ping
PONG
```

关闭redis: shutdown

2.2.5.7. Redis 关闭。

单实例关闭: redis-cli shutdown₽

```
[root@zy bin]# redis-server /myredis/redis.conf
[root@zy bin]# redis-cli shutdown
[root@zy bin]# ps -ef|grep redis
root 12722 6326 0 12:47 pts/2 00:00:00 grep --color=auto redis
```

也可以进入终端后再关闭。

```
127.0.0.1:6379> shutdown not connected>
```

Ι

多实例关闭,指定端口关闭: redis-cli -p 6379 shutdown

redis的优点

Redis 是单线程+多路 IO 复用技术。

多路复用是指使用一个线程来检查多个文件描述符(Socket)的就绪状态,比如调用 select 和 poll 函数,传入多个文件描述符,如果有一个文件描述符就绪,则返回,否则 阻塞直到超时。得到就绪状态后进行真正的操作可以在同一个线程里执行,也可以启 动线程执行(比如使用线程池)。

串行 vs 多线程+锁(memcached) vs 单线程+多路 IO 复用(Redis)。

(与 Memcache 三点不同: 支持多数据类型, 支持持久化, 单线程+多路 IO 复用)

常用的五大数据类型

redis 键 (key)

_3.1. Redis 键(key)。

keys *查看当前库所有 key (匹配: keys *1)-

exists key 判断某个 key 是否存在。

type key 查看你的 key 是什么类型。

del key 删除指定的 key 数据。

unlink key 根据 value 选择非阻塞删除。

仅将 keys 从 keyspace 元数据中删除,真正的删除会在后续异步操作。4

expire key 10 10秒钟:为给定的 key 设置过期时间。

ttl key 查看还有多少秒过期,-1表示永不过期,-2表示已过期。

select 命令切换数据库+

dbsize 查看当前数据库的 key 的数量。

flushdb 清空当前库。

flushall 通杀全部库。

redis 字符串 类型 (String)

_3.2. Redis 字符串(String)。

_3.2.1. 简介。

String 是 Redis 最基本的类型,你可以理解成与 Memcached 一模一样的类型,一个 key 对应一个 value。

String 类型是二进制安全的。意味着 Redis 的 string 可以包含任何数据。比如 jpg 图片或者序列化的对象。

String 类型是 Redis 最基本的数据类型 , 一个 Redis 中字符串 value 最多可以是 $512M_{\odot}$

常用命令

127.0.0.1:6379> set key value [EX seconds|PX milliseconds|KEEPTTL] [NX|XX]

- *NX: 当数据库中 key 不存在时,可以将 key-value 添加数据库+
- *XX: 当数据库中 key 存在时,可以将 key-value 添加数据库,与 NX 参数互斥+
- *EX: key 的超时秒数↓
- *PX: key 的超时毫秒数,与EX 互斥+

get <key>查询对应键值。

append <key><value>将给定的<value> 追加到原值的末尾。
strlen <key>获得值的长度。

Ι

setnx <key><value>只有在 key 不存在时 设置 key 的值。

incr <key>↓

将 key 中储存的数字值增 1₽

只能对数字值操作,如果为空,新增值为1%

decr <key>↓

将 key 中储存的数字值减 1~

可以同时设置多个

mset <key1><value1><key2><value2> 4

同时设置一个或多个 key-value 对 🖟

mget <key1><key2><key3>.....

同时获取一个或多个 value 4

msetnx <key1><value1><key2><value2> 4

同时设置一个或多个 key-value 对, 当且仅当所有给定 key 都不存在。

原子性,有一个失败则都失败。[

getrange <key><起始位置><结束位置>+

获得值的范围,类似java中的substring,前包,后包。

setrange <key><起始位置><value>+

用 <value> 覆写<key>所储存的字符串值,从<起始位置>开始(索引从 0 开始)。 4

setex <key><过期时间><value>

设置键值的同时,设置过期时间,单位秒。~

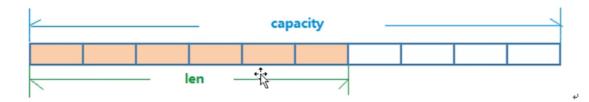
getset <key><value>

以新换旧,设置了新值同时获得旧值。4

string的数据结构

3.2.3. 数据结构。

String 的数据结构为简单动态字符串(Simple Dynamic String,缩写 SDS)。是可以修改的字符串,内部结构实现上类似于 Java 的 ArrayList 几采用预分配冗余空间的方式来减少内存的频繁分配。



如图中所示,内部为当前字符串实际分配的空间 capacity 一般要高于实际字符串长度 len。当字符串长度小于 1M 时,扩容都是加倍现有的空间,如果超过 1M,扩容时一次只会多扩 1M 的空间。需要注意的是字符串最大长度为 512M。4

列表List的 操作

简介

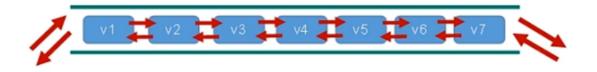
_3.3. Redis 列表(List)。

3.3.1. 简介:

单键多值。

Redis 列表是简单的字符串列表,按照插入顺序排序。你可以添加一个元素到列表的头部(左边)或者尾部(右边)。4

它的底层实际是个双向链表,对两端的操作性能很高,通过索引下标的操作中间的节点性能会较差。



list 常用命令

_3.3.2. 常用命令。

lpush/mush <key><value1><value2><value3> 从左边/右边插入一个或多个值。 lpop/rpop <key>从左边/右边吐出一个值。值在键在,值光键亡。

rpoplpush <key1><key2>从<key1>列表右边吐出一个值,插到<key2>列表左边。+

Irange <key><start><stop>-

按照索引下标获得元素(从左到右)。