Redis实战 - it_man的专栏 - 博客频道

2013-08-03 10:59 6632人阅读 评论(0) 收藏 举报

分类：

数据库/NoSql（43）

开源应用系统（11）

目录(?)[+]

大约一年多前，公司同事开始使用Redis，不清楚是配置，还是版本的问题，当时的Redis经常在使用一段时间后，连接爆满且不释放。印象中，Redis 2.4.8以下的版本由于设计上的主从库同步问题，就会导致整个问题，不知是否确为这个Bug所致。但从那以后，我就很少敢去尝试使用Redis。曾想转投MongoDB，但公司同事给我的回复是，由于MongoDB宕机，数据丢失，公司损失惨重。于是，我一直停留在Memcached使用范畴，且用的还比较一般。

相关链接：

由于前段时间使用**Kestrel**，同时要操作**Memcached**及时更新缓存，又要操作database，持久化数据。

貌似**Redis**既可以当**Cache**又可以当**Queue**！于是，今天开始研究**Redis**！

一、Redis简要介绍**Redis** —— **RE**mote **DI**ctionary**S**erver，可以直接理解为远程字典服务，也就是基于Key-Value模式Memcached+Database Persistence。

如果真要把Redis与Memcached进行对比，参考下图：

Items	Redis	Memcached
IO Model	Single Thread Model blocking	Multi Thread Model Nonblocking
Security		binary and sasl protocol
Memory Management	current time	Predistribution
Data Consistency	transaction	Provide Cas Command
Data Store	key-value store Data Types -> Strings(Integers)、list、hash、set、sorted set、KEYS support Enum	key-value store nonsupport Enum
Management tool	phpRedisAdmin， Admin UI	MemAdmin、statsproxy
Replication	Master-Slave Model	
expandability	DB Sharding or hash distribution	Hash distribution
Client for Java	JRedis、Jedis、Redis4j	Memcached Java Client2.6.1 Xmemcached spymemcached
Session store	tomcat-redis-session-manager-1.0.jar	Memcached Session Manager
Object size	<=1G [512M]	<=1M
port	6379	11211
others	support Persistence, binlog、scripting	

使用Memcached，让我感触颇深的是Object Size的问题，由于SQL未作优化直接映射对象，导致缓存对象大于1MB，Memcached就抛了异常。而Redis默认缓存对象512MB，最大支持1GB。至少在缓存对象时，可以有更大的伸缩空间了！

此外，是数据类型。Memcached比较简单，而Redis可以支持更多复杂的数据类型，如HASH、SET、SortedSet等等。

PS:**Memcached**是在**Server**端实现的**Sharding**，**Redis**没有对应的实现，据说**3.0**系列开始支持，不过这话貌似说了2年之久。

二、安装

Redis装起来，实在是过于简单，让我几乎“无从下手”。因为连“configure”文件都不需要，你只需要做个“make”就好。

在这里下载Redis最新版，这里用Redis 2.4.16

下载&解压：

Shell代码

```
1. wget http://redis.googlecode.com/files/redis-2.4.16.tar.gz
2. tar zxvf redis-2.4.16.tar.gz
```

Redis可以解压至任何目录，一个make安装即可获得执行、配置文件。
安装（这里将redis解压到/opt/目录下）：
Shell代码

make之后，我们会得到以下可执行文件：

- **redis-server**：Redis服务器的daemon启动程序
- **redis-cli**：Redis命令行操作工具。或者通过telnet进行纯文本协议操作
- **redis-benchmark**：Redis性能测试工具，测试Redis在你的系统及你的配置下的读写性能

上述文件位于**src**目录下。

我习惯性的执行了make install，貌似我需要的可执行文件，安装到了**/usr/local/bin**：
引用# make install
cd src && make install
make[1]: Entering directory `/opt/software/redis-2.4.16/src'
MAKE hiredis
make[2]: Entering directory `/opt/software/redis-2.4.16/deps/hiredis'
make[2]: Nothing to be done for `static'.
make[2]: Leaving directory `/opt/software/redis-2.4.16/deps/hiredis'
MAKE linenoise
make[2]: Entering directory `/opt/software/redis-2.4.16/deps/linenoise'
make[2]: “linenoise_example”是最新的。
make[2]: Leaving directory `/opt/software/redis-2.4.16/deps/linenoise'
MAKE hiredis
make[2]: Entering directory `/opt/software/redis-2.4.16/deps/hiredis'
make[2]: Nothing to be done for `static'.
make[2]: Leaving directory `/opt/software/redis-2.4.16/deps/hiredis'
LINK redis-benchmark

LINK redis-cli

Hint: To run 'make test' is a good idea ;)

mkdir -p /usr/local/bin
cp -pf redis-server /usr/local/bin
cp -pf redis-benchmark /usr/local/bin
cp -pf redis-cli /usr/local/bin
cp -pf redis-check-dump /usr/local/bin
cp -pf redis-check-aof /usr/local/bin
make[1]: Leaving directory `/opt/software/redis-2.4.16/src'
这样，就不用我拷贝文件了。

意外收获！

此外，还会得到一个默认的配置文件——**redis.conf**。
最好，把它拷贝到固定的目录下，例如：/etc/redis/目录下！
Shell代码

然后，我们就可以在任何路径下，直接启动Redis了！

三、运行运行Redis：引用[1958] 13 Aug 16:18:24 * Server started, Redis version 2.4.16
[1958] 13 Aug 16:18:24 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.
[1958] 13 Aug 16:18:24 * The server is now ready to accept connections on port 6379
[1958] 13 Aug 16:18:24 - 0 clients connected (0 slaves), 717544 bytes in use四、测试通过客户端命令**redis-cli**访问Redis
引用# redis-cli
redis> set name zlex
OK
redis> get name
"zlex"
进行数据测试：

这个测试会一直进行下去，直到你Ctrl+C：

====== PING (inline) ======
10000 requests completed in 0.12 seconds
50 parallel clients
3 bytes payload
keep alive: 1

99.31% <= 1 milliseconds
99.53% <= 2 milliseconds
99.64% <= 3 milliseconds
99.70% <= 4 milliseconds
99.74% <= 5 milliseconds
99.78% <= 6 milliseconds
99.82% <= 7 milliseconds
99.84% <= 8 milliseconds
99.86% <= 9 milliseconds
99.89% <= 10 milliseconds
99.91% <= 11 milliseconds
99.93% <= 12 milliseconds
99.96% <= 13 milliseconds
99.98% <= 14 milliseconds
100.00% <= 15 milliseconds
81300.81 requests per second

====== PING ======
10000 requests completed in 0.12 seconds
50 parallel clients
3 bytes payload
keep alive: 1

99.96% <= 1 milliseconds
100.00% <= 1 milliseconds
84033.61 requests per second

^CET (10 keys): 26200.00

五、关闭

也可通过客户端命令**redis-cli**完成**Redis**关闭操作:

Shell代码

```
1. redis-cli shutdown
```

引用[2639] 13 Aug 16:35:35 # User requested shutdown...
[2639] 13 Aug 16:35:35 * Saving the final RDB snapshot before exiting.
[2639] 13 Aug 16:36:49 * DB saved on disk
[2639] 13 Aug 16:36:49 # Redis is now ready to exit, bye bye...六、调优

1.**/etc/sysctl.conf**

前面启动Redis时，看到如下警告：

引用[1958] 13 Aug 16:18:24 # WARNING overcommit_memory is set to 0! Background save may fail under low memory condition. To fix this issue add 'vm.overcommit_memory = 1' to /etc/sysctl.conf and then reboot or run the command 'sysctl vm.overcommit_memory=1' for this to take effect.

需要修改**/etc/sysctl.conf**文件：

Shell代码

```
1. vim /etc/sysctl.conf
```

末尾追加**vm.overcommit_memory = 1**

然后执行**sysctl vm.overcommit_memory=1**，使之生效：

Shell代码

2.**/proc/sys/vm/overcommit_memory**

为了调整内存分配策略，需要配置**/proc/sys/vm/overcommit_memory**

- 0， 表示内核将检查是否有足够的可用内存供应用进程使用；如果有足够的可用内存，内存申请允许；否则，内存申请失败，并把错误返回给应用进程。
- 1， 表示内核允许分配所有的物理内存，而不管当前的内存状态如何。
- 2， 表示内核允许分配超过所有物理内存和交换空间总和的内存

默认为0，如果内存情况比较紧张的话，设为1：

Shell代码

1. echo 1 > /proc/sys/vm/overcommit_memory

3.**redis.conf**

前面启动Redis后，总是在命令行里不断跳着各种日志，很麻烦。即便通过“&”，领其后台运行，也无济于事。这就需要修改**redis.conf**，以Daemo模式运行！

redis.conf参数：

- daemonize：是否以后台daemon方式运行
- pidfile：pid文件位置
- port：监听的端口号
- timeout：请求超时时间
- loglevel：log信息级别
- logfile：log文件位置
- databases：开启数据库的数量
- save * *：保存快照的频率，第一个*表示多长时间（秒级），第三个*表示执行多少次写操作。在一定时间内执行一定数量的写操作时，自动保存快照。可设置多个条件。
- rdbcompression：是否使用压缩
- dbfilename：数据快照文件名（只是文件名，不包括目录）
- dir：数据快照的保存目录（这个是目录）
- appendonly：是否开启appendonlylog，开启的话每次写操作会记一条log，这会提高数据抗风险能力，但影响效率。
- appendfsync：appendonlylog如何同步到磁盘（三个选项，分别是每次写都强制调用fsync、每秒启用一次fsync、不调用fsync等待系统自己同步）
- slaveof <masterip> <masterport> ：主从配置，在redis-slave上配置master的ip port，即可。

例如，我们可以修改为如下方式：

引用daemonize yes #守护进程模式

save 60 1000 #当时间间隔超过60秒，或存储超过1000条记录时，进行持久化。

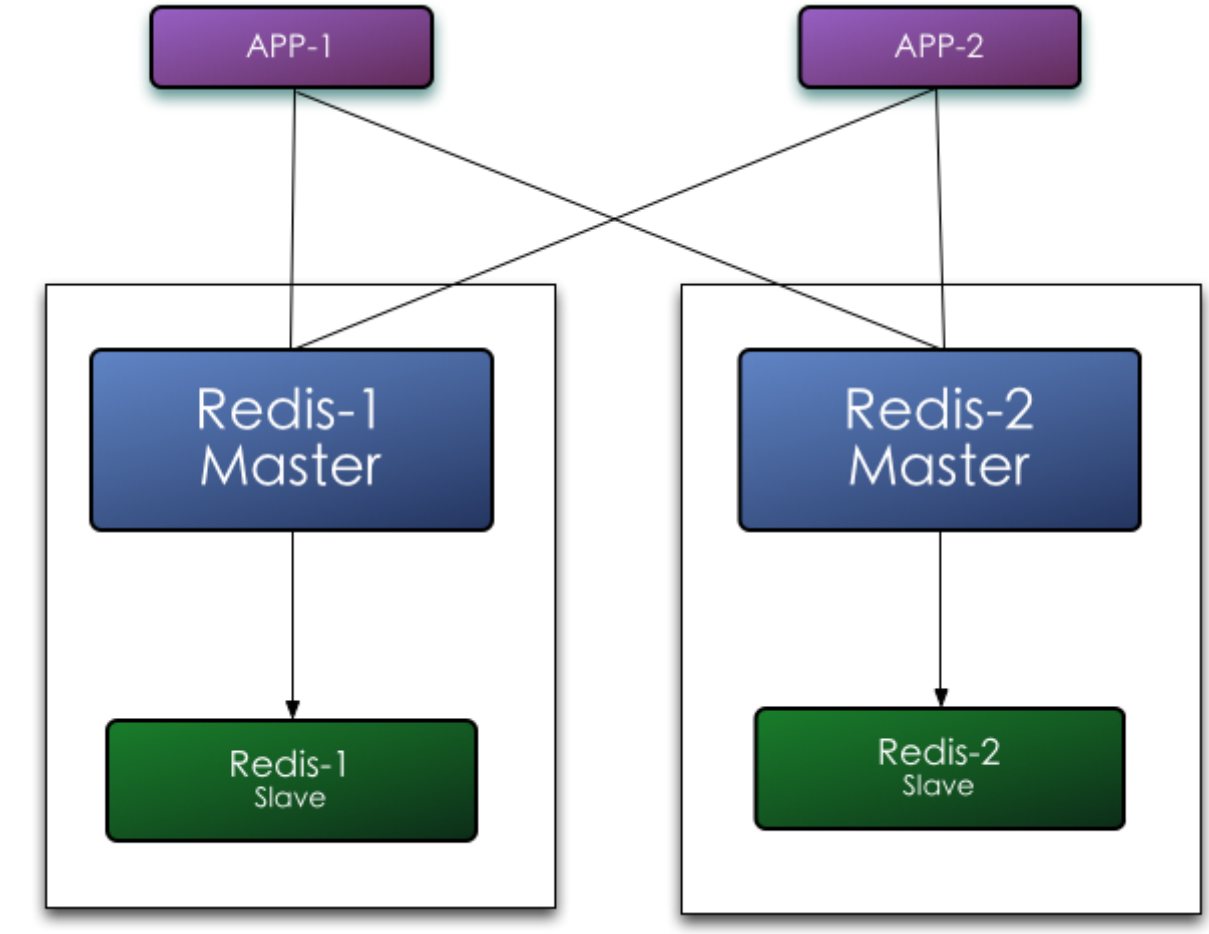
maxmemory 256mb #分配256MB内存

PS：切记，一定要设定**maxmemmory**，且配置大小要小于物理内存，留有足够的内存供系统使用。

公司一同学的Redis，某期间数据暴涨，导致内存吃紧，SWAP加剧，直接宕机。就是因为没有设置**maxmemmory**。

七、集群配置

把鸡蛋都放在一个篮子里是件危险的事情。首先，要做好主备。其次，如果可以做一致性哈希，可以起到负载均衡的作用。



配置Master-Slave，只需要在Slave上配置Master节点IP Port：

这里的Master IP 为192.168.133.139 端口位6379，配置redis.conf：

slaveof 192.168.133.139 6379

PS：为了两个Redis Server可以互访，需要注释掉**bind 127.0.0.1**

依次启动Master，Slave：

Master

[7651] 17 Aug 19:08:07 * Server started, Redis version 2.4.16

[7651] 17 Aug 19:08:07 * DB loaded from disk: 0 seconds

[7651] 17 Aug 19:08:07 * The server is now ready to accept connections on port 6379

[7651] 17 Aug 19:08:08 * Slave ask for synchronization

[7651] 17 Aug 19:08:08 * Starting BGSAVE for SYNC

[7651] 17 Aug 19:08:08 * Background saving started by pid 7652

[7652] 17 Aug 19:08:08 * DB saved on disk
[7651] 17 Aug 19:08:08 * Background saving terminated with success
[7651] 17 Aug 19:08:08 * Synchronization with slave succeeded
Slave
[7572] 17 Aug 19:07:39 * Server started, Redis version 2.4.16
[7572] 17 Aug 19:07:39 * DB loaded from disk: 0 seconds
[7572] 17 Aug 19:07:39 * The server is now ready to accept connections on port 6379
[7572] 17 Aug 19:07:39 * Connecting to MASTER...
[7572] 17 Aug 19:08:08 * MASTER <-> SLAVE sync started: SYNC sent
[7572] 17 Aug 19:08:08 * MASTER <-> SLAVE sync: receiving 10 bytes from master
[7572] 17 Aug 19:08:08 * MASTER <-> SLAVE sync: Loading DB in memory
[7572] 17 Aug 19:08:08 * MASTER <-> SLAVE sync: Finished with success
看到上述日志，就说明Master-Slave已经连通。

简单测试，Master写，Slave读：

Master写
telnet 192.168.133.139 6379
Trying 192.168.133.139...
Connected to 192.168.133.139.
Escape character is '^]'.
set name snowolf
+OK
Slave读
telnet 192.168.133.140 6379
Trying 192.168.133.140...
Connected to 192.168.133.140.
Escape character is '^]'.
get name
$7
snowolf

搞定！

八、主从备份在从服务器上执行下列命令：
Shell代码

编辑**/etc/init.d/redis-server**，键入如下内容：

Shell代码

```
#!/bin/bash
#
# redis    Startup script for redis processes
#
# author: snowolf
#
# processname: redis
redis_path="/usr/local/bin/redis-server"
redis_conf="/etc/redis/redis.conf"
redis_pid="/var/run/redis.pid"
# Source function library.
. /etc/rc.d/init.d/functions
[ -x $redis_path ] || exit 0
RETVAL=0
prog="redis"
# Start daemons.
start() {
    if [ -e $redis_pid -a ! -z $redis_pid ];then
        echo $prog" already running...."
        exit 1
    fi
    echo -n $"Starting $prog "
    # Single instance for all caches
    $redis_path $redis_conf
    RETVAL=$?
    [ $RETVAL -eq 0 ] && {
```

```
27.         touch /var/lock/subsys/$prog
28.         success $"$prog"
29.     }
30.     echo
31.     return $RETVAL
32. }
33. # Stop daemons.
34. stop() {
35.     echo -n $"Stopping $prog "
36.     killproc -d 10 $redis_path
37.     echo
38.     [ $RETVAL = 0 ] && rm -f $redis_pid /var/lock/subsys/$prog
39.     RETVAL=$?
40.     return $RETVAL
41. }
42. # See how we were called.
43. case "$1" in
44.         start)
45.             start
46.             ;;
47.         stop)
48.             stop
49.             ;;
50.         status)
51.             status $prog
52.             RETVAL=$?
53.             ;;
54.         restart)
55.             stop
56.             start
57.             ;;
58.         condrestart)
59.             if test "x`pidof redis`" != x; then
60.                 stop
61.                 start
62.             fi
```

```
63.             ;;
64.         *)
65.             echo $"Usage: $0 {start|stop|status|restart|condrestart}"
66.             exit 1
67. esac
68. exit $RETVAL
```

引用# service redis-server restart
Stopping redis [失败]
Starting redis [确定]
service redis-server status
redis (pid 14965) 正在运行...

非常方便！**相关链接：**